Improved Cloud-Based N-Primes Model for Symmetric-Based Fully Homomorphic Encryption Using Residue Number System

K. J. Muhammed, R. M. Isiaka, A. W. Asaju-Gbolagade, K. S. Adewole, and K. A. Gbolagade

Abstract Encryption schemes that allow computation to be performed on an encrypted data are required in modern real-world applications. This technology is a necessity for cloud computing, processing resources and share storage, in order to preserve integrity and privacy of data. The existing partial and fully homomorphic encryption (PHE and FHE) for both asymmetric and symmetric approaches are still suffering from efficiency in terms of encryption execution time and very large ciphertext file produced at the end of encryption. In this paper, we consider symmetric approaches and focus on overcoming the drawbacks of N-prime model using residue number system (RNS). The experimental results obtained show that the proposed RNS-based N-prime model for symmetric-based FHE improves the system latency and reduces the ciphertext file expansion by approximately 72% as compared to the existing N-prime model. The proposed improved N-prime model is guaranteed to provide optimum performance and reliable solution for securing integrity and privacy of user's data in the cloud.

Keywords Homomorphic encryption · Fully homomorphic encryption · Residue number system · Cloud-based · N-prime model · RNS-based N-prime model · Symmetric encryption

K. J. Muhammed (✉)
Department of Educational Technology, University of Ilorin, Ilorin, Nigeria
e-mail: jimoh.km@unilorin.edu.ng

R. M. Isiaka · K. A. Gbolagade
Department of Computer Science, Kwara State University, Malete, Nigeria
e-mail: abdulrafiu.isiaka@kwasu.edu.ng

K. A. Gbolagade
e-mail: kazeem.gbolagade@kwasu.edu.ng

A. W. Asaju-Gbolagade · K. S. Adewole
Department of Computer Science, University of Ilorin, Ilorin, Nigeria
e-mail: ayisatwuraola@gmail.com

K. S. Adewole
e-mail: adewole.ks@unilorin.edu.ng

1 Introduction

Nowadays, there is a growing interest on cloud technology globally, which informs the efforts by many companies toward implementation of cloud technologies in their business processes. Cloud technology with its own unique feature comprises cloud services, storage, computing, grid computing and distributed computing, among others. Since the idea of cloud technology was projected, there is perpetual attention for research across the world. It has become the hottest research area in the domain of information technology which has been realized as unital of the technology that postures the next-generation computing revolution [1]. Cloud computing offers many opportunities to enterprises for their customers to access and operate computing services as “pay-as-you-go”. Customers saddle with responsibilities of batch processing and web applications explored cloud computing model as an efficient alternative way to acquiring and managing private data centers [2].

Cloud computing as described by US National Institute of Standards and Technology (NIST) as a model that allows and provides convenient ways of accessing unlimited pool of shared resources such as on-demand network access, servers, applications, storage and services categorized as software and hardware that can be rapidly made available to the client as soon as requested through the internet, these resources can be reviewed upward and downward automatically based on client’s demand. Thus, cloud computing has five essential characteristics, three service models and four deployment models such as on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service, software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS) and private cloud, public cloud, community cloud and hybrid cloud, respectively, as shown in Fig. 1 [1, 3].

The quick move toward adoption of cloud computing has raised critical fears on an ultimate fact for the success of information systems, communication, virtualization, data availability and integrity, public auditing, scientific application and information security. Despite the demand of businesses and individual organizations to transfer their data to the cloud, security in terms of data integrity and confidentiality, trust in

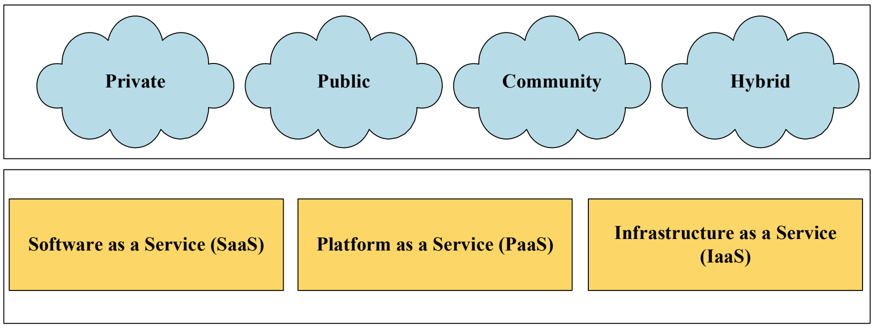
Private
Public
Community
Hybrid
Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)

Fig. 1 Cloud solutions based on the system’s deployment and service model [1]

terms of data accessibility and availability, and privacy issue have remained as the challenges for embracing cloud computing [4]. Thus, security anxieties arise when transforming data from locally owned storage to third parties cloud service provider's (CSP) storage. It is expected of CSP to encrypt user's data before storing it on their cloud server. Several cryptographic schemes such as encryption and decryption, key exchange and steganography have been used to address the security challenges facing the adoption of cloud computing. Encryption means conversion of plain text to ciphertext, and it can be used in various applications for moving data from one location to another. Encryption algorithms such as DES, AES, RSA, Blow fish and many more are meant for secured storage of data and operations on cloud computing server [5, 6]. The traditional encryption schemes seem to be weak for securing the integrity and confidentiality of stored data in the cloud computing server as it requires secret key to be provided by the users so as to decrypt the data prior to the computations, and this threatens the privacy, confidentiality and integrity of stored data.

Homomorphic encryption (HE) was introduced to overcome this concern. HE is the process of delegating processing of an encrypted data without having access to the raw data, and produces encrypted result when decrypt result matches the result of the same operations carried out on the raw data. It is considered as one of the most efficient solutions for protecting the confidentiality and user's information privacy in cloud-based services. Also, HE is grouped as partial homomorphic encryption (PHE) and fully homomorphic encryption (FHE) where PHE ropes either to additive or multiplicative operation, whereas FHE supports both operations and seems to be more secured and efficient than PHE as it poses properties of both operations [7].

The concept of FHE scheme was first introduced by Gentry and Boneh [8] and this was used to address cryptography central open issues. The scheme allows arbitrary functions to be computed over an encrypted data with no reference to decryption key [9]. The development of FHE is a revolutionary advancement, greatly extending the scope of the computations, which can be applied to process encrypted data homomorphically. Gentry [10] presented the first version of FHE scheme using ideal lattices; however, his construction was very complex and impracticable. Since then, FHE has been an interesting research area and several developments that centre on the application, implementation and improvement of FHE schemes have also been studied [11]. There are numerous practical applications of FHE in the real world such as buyer privacy in publicity, medical applications (genomics and health), national security, private prediction, private learning, education, monetary privacy and forensic image recognition, among others [12–14].

There are three major problems associated with the design and practical implementation of FHE scheme such as security problem, time efficiency problem and cipher expansion problem. However, several researchers [15, 16] have contributed to the improvement of FHE in terms of security that is based on hard mathematical problems, and the cost of breaking such scheme at a given security parameter λ should be at least 2λ. Also, in the last 11 years, progress has been made on time efficiency problem by different researchers such as [17–21], and the proposed schemes were categorized into three generations: FHE based on ideal lattices; the second generation is based on learning with error (LWE) and ring learning with error (RLWE) problems and the third generation is GSW which is also based on RLWE. However, each of these proposed schemes still needs further improvement for it to be efficiently useful in real-world applications.

It is highlighted in [7, 17] that cipher expansion for both asymmetric and symmetric FHE schemes are producing very large ciphertext size between hundreds and tens of thousands as compared to original plaintext size. For instance, ciphertext produced in [17] is till 400,000 times that of the original plaintext and likewise in [7]. These show the growth rate of ciphertext size in an existing FHE schemes which call for serious concern and constitute a major problem of the schemes because it will require large bandwidth to transfer such ciphertext through the network. We proposed scheme conversion concept for encryption scheme originally mentioned in [7] as symmetric-based FHE in order to address the issue. In addition, we provide efficient parallelization in encryption process which produces small ciphertext sizes. However, this study focuses on tackling cipher expansion problem on symmetric-based FHE scheme as proposed by [7] using residue number system.

The organization of this paper is as follows. Section 2 describes the related studies on homomorphic encryption; Sect. 3 discusses the research methodology of the proposed framework. Section 4 presents results findings and discussion. Section 5 is the conclusion to the study.

2 Related Works

HE schemes are classified into three major categories, as shown in Fig. 2 based on supported homomorphism operations and properties which are partial homomorphic encryption (PHE), somewhat homomorphic encryption (SWHE) and fully homomorphic encryption (FHE). This scheme is called PHE when it allows any number of either addition as in [22, 23] or multiplication operation as described

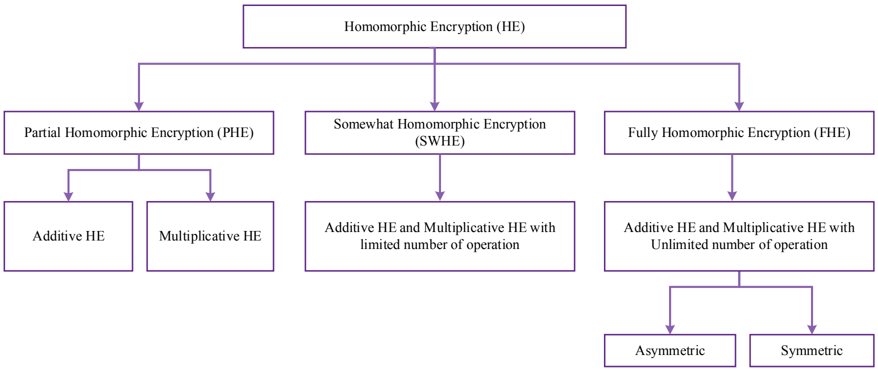
Homomorphic Encryption (HE)
Partial Homomorphic Encryption (PHE)
Somewhat Homomorphic Encryption (SWHE)
Fully Homomorphic Encryption (FHE)
Additive HE
Multiplicative HE
Additive HE and Multiplicative HE with limited number of operation
Additive HE and Multiplicative HE with Unlimited number of operation
Asymmetric
Symmetric

Fig. 2 Categorization of homomorphic encryption

in [24, 25]. However, SWHE supports both operations but with limited number of times allowed to evaluate the decryption function because of noise growth as illustrated in [8, 26]. Likewise in [27], the authors presented technique based on SWHE that efficiently secures pre-processing model for dishonest majority in multiparty computation over the ring $\mathbb{Z}_{2_k}$. Finally, improvement over SWHE is FHE which allows unlimited number of operations for both additive and multiplicative homomorphism to be performed on an encrypted data without increase in noise during the computation. This type of schemes is extensively discussed in [8, 10, 21, 28].

Parallel to conventional cryptography, FHE schemes are also categorized into asymmetric encryption and symmetric encryption. Asymmetric FHE schemes protect the privacy and confidentiality of data by using different pair of keys (public key and private key) to encrypt the data. Recent research on asymmetric-based FHE has been proposed by [10, 17, 19, 20, 23, 28–33]. However, these schemes suffer from very high computational overhead, cipher expansion and time efficiency which make it difficult to practice in real-world application. However, researchers such as [7, 27, 34–41] have proposed symmetric-based FHE schemes due to its strength in low computational overhead and practical consideration for real-world deployments. Moreover, some of these symmetric-based FHE schemes [7, 41–44] still suffer from large cipher expansion and low immunity against attack. This demanded that a robust security resistant and cipher expansion reduction are highly needed for these symmetric-based FHE schemes [40, 45].

3 Research Method

This section discusses the research methodology in relation to homomorphic encryption, N-prime model, residue number system and proposed RNS-based N-prime model for symmetric FHE.

3.1 Homomorphic Encryption

Just after discovery of RSA cryptosystem, HE was introduced by [24] as privacy homomorphism that protects the privacy, integrity and confidentiality of data. Generally, HE scheme allows computations on an encrypted variable without the need for decryption prior to the computation or the use of secret key [14, 46]. HE scheme contains four algorithms such as key generation (KeyGen), encryption (Enc), evaluation function (Eval(f)) and decryption algorithm (Dec) which can be described as follows:

KeyGen: *(pk, sk)* $\leftarrow$ *KeyGen(k)*, choose parameter k and the algorithm will generate *pk* as public key and *sk* as private key.

Enc: *c ← Enc(m, pk)*, where m is the plaintext-message, *pk* is public key and *c* is the ciphertext.
Dec: *m ← Dec(c, sk)*, where c is the ciphertext, *sk* is the private key and *m* is the plaintext message.
Eval(f): *c* ← Evaluate (f, Enc(m1), Enc(m2), …… Enc(mn))*

However, for the scheme to be homomorphic in nature, it must obey the following properties:

Dec(c, sk) = f(m1, m2, ……… mn)*
Dec(C1 op C2, sk) = M1 op M2

where *C1* and *C2* are *Enc(M1)* and *Enc(M2)*, respectively, *op* is the group operation (addition or multiplication) in the ciphertext and plaintext space, respectively.

3.2 N-Primes Model

In [7], an encryption scheme called N-primes model is investigated for constructing a symmetric-based FHE scheme which converts each plaintext character into ASCII code and passes it to the encryption algorithm $c = m + r * l * n$ where ct is the ciphertext, m is the plaintext message and $m \in [0, L-1]$, r is the noise added to the ciphertext, l is a prime big integer and $n = p1 * p2 * p3 * \ldots pi$ is the multiplication of numerous prime numbers resulting in one ciphertext for each character in the plaintext message. Thus, encryption and decryption are as follows:

Key generation

$$\text{Generate}\, n$$

$$\text{Generate}\, l \in Z_n$$

$$\text{where}\, n \text{ is a multiplication of various prime numbers } (n = P_1 * P_2 * P_3 * \ldots P_i) \text{ and}\, l \text{ is a big prime integer value}$$

Encryption algorithm

$c = m + r * l * n$ where c is the ciphertext, $m \in [0, L - 1]$ and r is random number

Decryption algorithm

$$m = \mathrm{c \bmod l}$$

Proof of homomorphism

Additive property

$$\begin{aligned}m_3 &= c_1 + c_2 = (m_1 + r_1 * l * n) + (m_2 + r_2 * l * n)\\ &= (m_1 + m_2) + l * n(r_1 + r_2)\\ &since\, l * n(r_1 + r_2) mod\, l = 0\, then\\ &m_3 = (m_1 + m_2)\end{aligned}$$

Multiplicative Property

$$m_3 = c_1 * c_2 = (m_1 + r_1 * l * n) * (m_2 + r_2 * l * n)$$

$$m_3 = [(m_1 * m_2 + m_1 * r_2 * l * n) + \{(m_2 * r_1 * l * n) + (r_1 * l * n)(r_2 * l * n)\}]$$

$$m_3 = \left[(m_1 * m_2 + m_1 * r_2 * l * n) + \left\{(m_2 * r_1 * l * n) + \left(r_1 * r_2 * l^2 * n^2\right)\right\}\right]$$

$$m_3 = \left(m_1 * m_2 + m_1 * l * \left\{r_2 * n + r_1 * n * m_2 + r_1 * r_2 * l * n^2\right\}\right)$$

$$\text{Since}\left[l * \left\{r_2 * n + r_1 * n * m_2 + r_1 * r_2 * l * n^2\right\}\right] \bmod l = 0$$
$$\text{and multiple of } l \bmod l = 0 \text{ then}$$

$$m_3 = m_1 * m_2 + m_1 * 0$$

$$m_3 = m_1 * m_2$$

3.3 Residue Number System

The residue number system (RNS) is a non-weighted and carry-free number system that speeds up arithmetic operations by dividing them into smaller parallel operations.

It is characterized based on its unique properties which include fault tolerance, modularity, parallelism and carry-free operations, among others. RNS is very efficient in terms of handling arithmetic operations (addition, subtraction and multiplication) [47–50]. RNS can as well be described as moduli set of positive pairwise relatively prime numbers $(m_1, m_2 \ldots m_n)$ and the product of all the modulo ($M = m_1 \times m_2 \times \ldots \times m_n$) that produces the dynamic range (DR) of such moduli set. Any integer X in the range [0, M – 1] can be uniquely represented by an ordered set of residues ($x_1, x_2, \ldots, x_n$). Each residue x_i is represented by $x_i = X \bmod m_i$.

To perform residue arithmetic operations such as addition or multiplication can be achieved by adding or multiplying corresponding digits relative to the modulus for their position [50]. For a given moduli set $\{m_1, m_2, \ldots, m_N\}$,

$X = \{x_1, x_2, \ldots, x_N\}$ and $Y = \{y_1, y_2, \ldots, y_N\}$ such that $x_i = |X|m_i$ and $y_i = |Y|m_i$ where *i = 1, 2, ..., N,* then we may define the residue arithmetic operations as follows:

$$X \otimes Y = Z$$

$$X \otimes Y = \{x_1, x_2, \ldots, x_N\} \otimes \{y_1, y_2, \ldots, y_N\} = \{z_1, z_2, \ldots, z_N\} = Z$$

where $z_i = |x_i \otimes y_i|m_i$ and $\otimes$ is residue arithmetic operator (+, *, and −). As an example, with the moduli-set {3, 4, 5}, the representation of fifteen is {0, 3, 0}, that of twenty-three is {2, 3, 3}, and adding the two residue numbers yield {2, 2, 3} which is the representation for thirty-eight and their product is {0, 1, 0} which is three hundred and forty-five in that system.

3.4 Proposed RNS-Based N-Primes Model for Symmetric FHE

Based on the N-prime model, RNS-based N-primes model is built without altering its homomorphic properties. The steps are illustrated in Fig. 3 and will be explained below.

The plaintext in a ring Z_n will be checked if it is numeric and pass it to RNS forward conversion module, otherwise, generate ASCII codes for each of the text characters before passing it to RNS forward conversion module.

Key Generation: This produces very big prime integer value as the secret key for the entire scheme and only to be known by owner of data.

Noise Generation: This generates a set of random integer r in a ring Zn as the noise to the entire scheme.

N-Primes Generation: This also generates product of prime numbers $\left(np_i = \prod_{i=1}^{n} np_1 * np_2 \ldots np_n\right)$

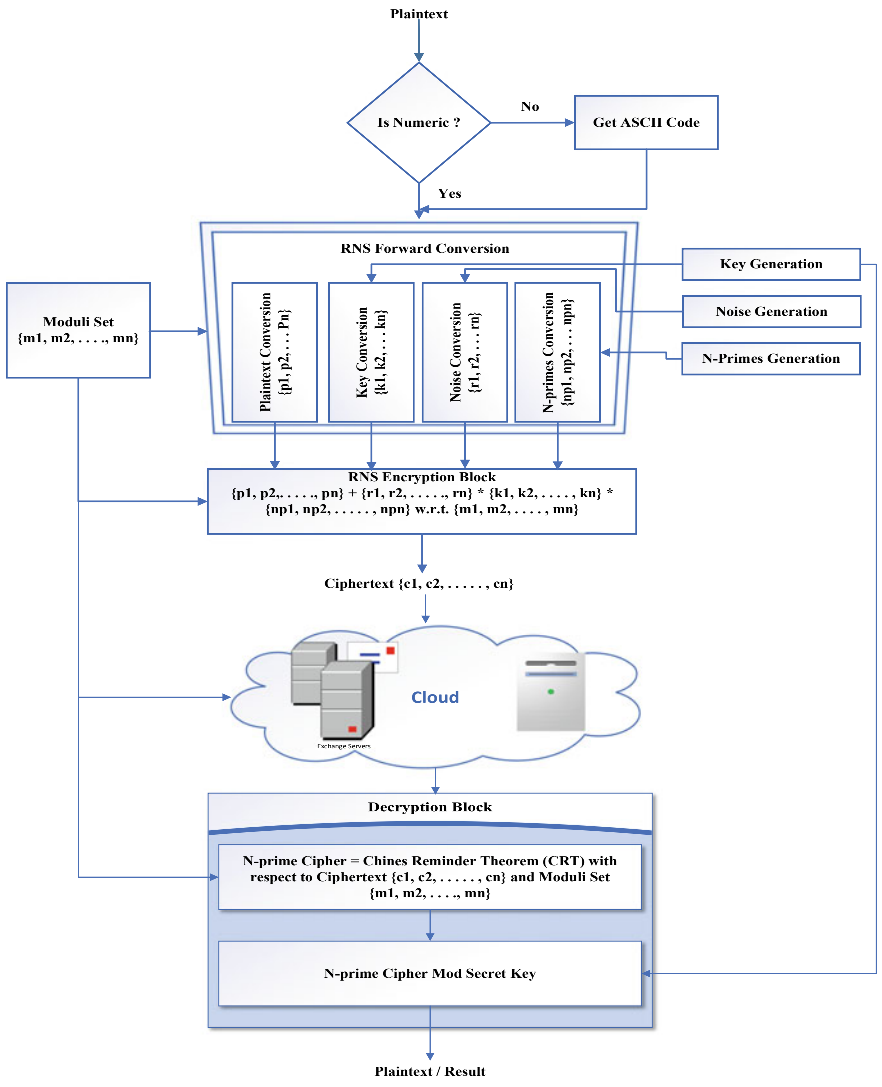
Plaintext
Is Numeric ?
No
Get ASCII Code
Yes
RNS Forward Conversion
Key Generation
Noise Generation
N-Primes Generation
Moduli Set {m1, m2, ..., mn}
Plaintext Conversion {p1, p2, . . . Pn}
Key Conversion {k1, k2, . . . kn}
Noise Conversion {r1, r2, . . . rn}
N-primes Conversion {np1, np2, . . . npn}
RNS Encryption Block
{p1, p2,. . . ., pn} + {r1, r2, ..., rn} * {k1, k2, ... , kn} * {np1, np2, ... , npn} w.r.t. {m1, m2, ... , mn}
Ciphertext {c1, c2, ... , cn}
Cloud
Exchange Servers
Decryption Block
N-prime Cipher = Chines Reminder Theorem (CRT) with respect to Ciphertext {c1, c2, ... , cn} and Moduli Set {m1, m2, ..., mn}
N-prime Cipher Mod Secret Key
Plaintext / Result

Fig. 3 Proposed RNS-based N-prime model

Moduli Set: It takes in n as the input and generates set of pairwise relatively prime integers and moduli set $\{3^n - 2, 3^n - 1, 3^n\}$ as proposed in [45] which will be used throughout this study, due to its large dynamic range.

RNS Forward Conversion: It requires five (5) parameters as input argument (plaintext (p_i), secret key *(k_i)*, noise *(r_i)*, n-prime numbers *(np_i)* and the moduli set *(m_i)*, where $i = 1, 2,\ldots, n$), convert p_i, k_i, r_i and np_i into residues with respect to m_i and send out four outputs into RNS encryption block.

RNS Encryption Block: Based on output received from RNS forward conversion block, the encryption block will perform RNS arithmetic operation in parallel with respect to moduli set m_i and produce the ciphertext.

$$Ciphertext\{c_1, c_2, ..., c_n\} = \{p_1, p_2, \ldots, p_n\} + \{r_1, r_2, \ldots, r_n\} * \{k_1, k_2, \ldots, k_n\} \\ * \{np_1, np_2, \ldots, np_n\} w.r.t. \{m_1, m_2, \ldots, m_n\}$$

However, the ciphertext can then be transmitted to the Cloud as shown in Fig. 3 and possibly, homomorphic bit operations can be carried out at the cloud server using RNS arithmetic operation based on ciphertext residues cx_i and cy_i with respect to moduli set m_i and the result will be sent to decryption block.

Decryption Block: The decryption block adopts Chinese Remainder Theorem as a backward conversion algorithm with the use of ciphertext residues (czi) and moduli set mi to obtain N-prime cipher.

$$X = \left| \sum_{i=1}^{k} \left| M_i^{-1} \right|_{mi} x_i \right|_M$$

where $M = \Pi_{i=1}^{k} m_i$, $M_i = \frac{M}{m_i} and\, M_i^{-1}$ is the multiplicative inverse of M_i with respect to m_i.

Finally, user will now use the generated secret key together with N-prime ciphertext to get the result of the operation.

The following experiments demonstrate the generation of the moduli set, secret key, other corresponding values and to show how these values are used for encryption and ciphertext reduction without loosen homomorphic behavior of the encryption and decryption block.

Experiment 1

Let the big prime number sk = 524287 be the secret key, random numbers r = 687529 as the noise, product of n-primes np = 23 * 73 * 101 = 69579 and moduli set $\{3^n - 2, 3^n - 1, 3^n\}$. When n = 23, then moduli set mi = [94143178825, 94143178826, 94143178827], let message m1 = 65 and message m2 = 66, the RNS-forward conversion for each of these parameters with respect to mi are:

$$\begin{aligned} \text{m1} &= [65, 65, 65] \\ \text{m2} &= [66, 66, 66] \\ \text{sk} &= [524287, 524287, 524287] \\ \text{r} &= [687529, 687529, 687529] \\ \text{np} &= [69579, 69579, 69579] \end{aligned}$$

To calculate the ciphertexts ct1 and ct2.

$$\begin{aligned}\mathrm{ct1} &= |\mathrm{m1} + \mathrm{r} * \mathrm{sk} * \mathrm{np}|_{\mathrm{mi}} \\ &= |[65, 65, 65] + [687529, 687529, 687529] \\ &\quad * [524287, 524287, 524287] \\ &\quad * [69579, 69579, 69579]|_{[94143178825, 94143178826, 94143178827]}\end{aligned}$$

$$\mathrm{ct1} = [31330438157, 31330171748, 31329905339]$$

$$\begin{aligned}\mathrm{ct2} =&|\mathrm{m2} + \mathrm{r} * \mathrm{sk} * \mathrm{np}|_{\mathrm{mi}} = |[66, 66, 66] + [687529, 687529, 687529] \\ &* [524287, 524287, 524287] \\ &* [69579, 69579, 69579]|_{[94143178825, 94143178826, 94143178827]}\end{aligned}$$

$$\mathrm{ct2} = [31330438158, 31330171749, 31329905340]$$

Proof of Homomorphism Properties

Additive Property

$$\begin{aligned}\mathrm{ct3} &= |\ \mathrm{ct1} + \mathrm{ct2}\ |_{\mathrm{mi}} \\ &= |[31330438157, 31330171748, 31329905339] \\ &\quad + [31330438158, 31330171749, 31329905340]|_{[94143178825, 94143178826, 94143178827]}\end{aligned}$$

$$\mathrm{ct3} = [62660876315, 62660343497, 62659810679]$$

To decrypt ct3,

$$\mathrm{m3} = \mathrm{chinese\_remainder}(\mathrm{m_i},\ \mathrm{ct3}) \bmod \mathrm{chinese\_remainder}(\mathrm{m_i}, \mathrm{sk})$$

m3 = chinese_remainder([94143178825, 94143178826, 94143178827], [62660876315, 62660343497, 62659810679]) mod chinese_remainder([94143178825, 94143178826, 94143178827], [524287, 524287, 524287])

$$\mathrm{m3} = 131\mathrm{m3} = 131$$

$$m3 = m1 + m2 = 65 + 66 = 131$$

Multiplicative Property

$$\begin{aligned} ct3 &= |\, ct1 * ct2 \,|_{mi} \\ &= |[31330438157, 31330171748, 31329905339] \\ &\quad * [31330438158, 31330171749, 31329905340]|_{[94143178825, 94143178826, 94143178827]} \end{aligned}$$

$$ct3 = [37639680706, 51114847858, 18251522556]$$

To decrypt ct3,

$$m3 = \text{chinese\_remainder}(m_i,\ ct3) \bmod \text{chinese\_remainder}(m_i, sk)$$

m3=chinese_remainder([94143178825, 94143178826, 94143178827], [37639680706, 51114847858, 18251522556]) mod chinese_remainder([94143178825, 94143178826, 94143178827], [524287, 524287, 524287])

$$m3 = 4290$$

$$m3 = m1 * m2 = 65 * 66 = 4290$$

Therefore, the proof shows that the proposed model supports both homomorphic addition and multiplication.

Experiment 2

The experiment is also carried out on a string plaintext that contains message “RNS based techniques for encryption of CAP in 2020” with secret key k = 6700417, random number r = 345957466 and product of n-prime np = 31 * 53 * 71 = 116653 and moduli set $\{3^n - 2, 3^n - 1, 3^n\}$. When n = 15, then moduli set m_i = [14348905, 14348906, 14348907]. Thus, the encryption is as follows:

[[13889688, 9226958, 7190942], [13889684, 9226954, 7190938], [13889689, 9226959, 7190943], [13889638, 9226908, 7190892], [13889704, 9226974, 7190958], [13889703, 9226973, 7190957], [13889721, 9226991, 7190975], [13889707, 9226977, 7190961], [13889706, 9226976, 7190960], [13889638, 9226908, 7190892], [13889722, 9226992, 7190976], [13889707, 9226977, 7190961], [13889705, 9226975, 7190959], [13889710, 9226980, 7190964], [13889716, 9226986, 7190970], [13889711, 9226981, 7190965], [13889719, 9226989, 7190973], [13889723, 9226993, 7190977], [13889707, 9226977, 7190961], [13889721, 9226991, 7190975], [13889638, 9226908, 7190892], [13889708, 9226978, 7190962], [13889717, 9226987, 7190971], [13889720,

Table 1 RNS-based N-prime model vs N-prime model

File size	Proposed RNS based N-prime model		N-prime model [7]	
	Encryption time (MS)	Ciphertext size (Bytes)	Encryption time (MS)	Ciphertext size (Bytes)
2 KB	0.99	5341	1.01	18789
4 KB	1.99	10633	2.99	37529
7 KB	4.98	21217	6.97	75009
14 KB	9.97	42385	10.01	149969
28 KB	18.94	84721	39.89	299889
55 KB	30.43	169393	34.90	599729
110 KB	75.76	338737	78.79	1199409
220 KB	161.54	677425	186.53	2398769
440 KB	319.18	1354801	348.02	4797489
880 KB	741.02	2709553	838.71	9594929
2 MB	3390.83	5419057	5289.16	19189809

9226990, 7190974], [13889638, 9226908, 7190892], [13889707, 9226977, 7190961], [13889716, 9226986, 7190970], [13889705, 9226975, 7190959], [13889720, 9226990, 7190974], [13889727, 9226997, 7190981], [13889718, 9226988, 7190972], [13889722, 9226992, 7190976], [13889711, 9226981, 7190965], [13889717, 9226987, 7190971], [13889716, 9226986, 7190970], [13889638, 9226908, 7190892], [13889717, 9226987, 7190971], [13889708, 9226978, 7190962], [13889638, 9226908, 7190892], [13889673, 9226943, 7190927], [13889671, 9226941, 7190925], [13889686, 9226956, 7190940], [13889638, 9226908, 7190892], [13889711, 9226981, 7190965], [13889716, 9226986, 7190970], [13889638, 9226908, 7190892], [13889656, 9226926, 7190910], [13889654, 9226924, 7190908], [13889656, 9226926, 7190910], [13889654, 9226924, 7190908], [13889652, 9226922, 7190906]].

Experiment 3

In experiment 3, small numbers were used such as 3 for secret key, 5 for random number, 2 * 3 * 5 = 30 for product of n-primes and n = 3 to obtain moduli set = [25–27]. These parameters were used to encrypt message of file size 931 Bytes in 0.47 MS and the ciphertext file is shown in Appendix A. The proposed model has also been tested for 11 different file sizes; encryption execution time and ciphertext file size were recorded and compared with original N-prime model as shown in Table 1.

4 Results and Discussion

The proposed RNS-based N-prime model and N-prime model implementations are done under Python using Dell Laptop having the following specifications: 64 bits

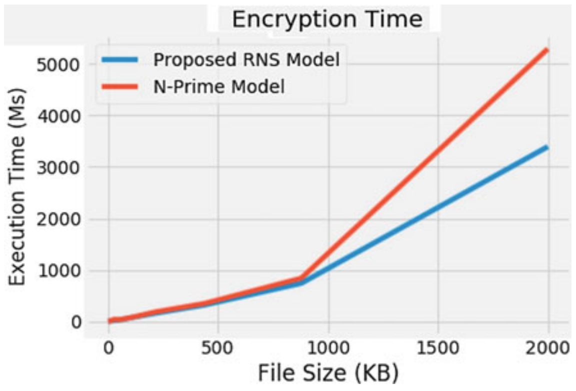

Fig. 4 Encryption time

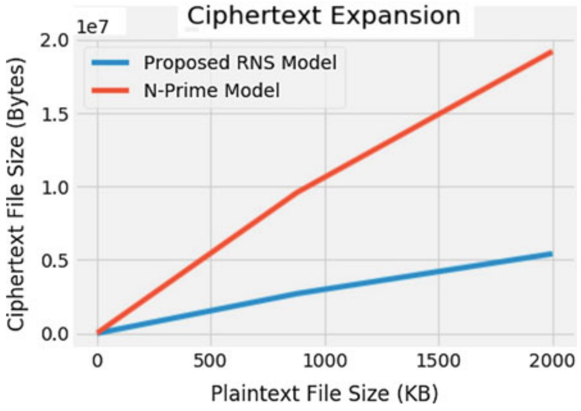

Fig. 5 Ciphertext expansion

Window 10 operating system, Intel® Core™ i9 processor with 2.90 GHz and 32 GB RAM. The execution time and ciphertext expansion are studied for different plaintext size, and the results are shown in Figs. 4 and 5, respectively. Based on the execution time, the proposed RNS-based N-prime model is taking the lowest execution time, while the N-prime model is taking the highest, and this can be interpreted that during proposed RNS-based N-prime model the encryption is done based on residues number system and also all the computations are performed in parallel without carry propagation which makes the proposed model more efficient and practical. Table 2 shows the improvement of proposed RNS-based N-prime model on the N-prime model in terms of execution time.

Table 2 Execution time RATIO (RNS-based N-prime model/N-prime model)

Plaintext size in kilobytes	2	4	7	14	28	55	110	220	440	880	2000
Execution time ratio	0.98	0.67	0.72	1.00	0.48	0.87	0.96	0.87	0.92	0.88	0.64

Table 3 Ciphertext percentage reduction in N-prime model and RNS-based N-prime model

Plaintext size in kilobytes	2	4	7	14	28	55	110	220	440	880	2000
Percentage reduction (%)	71.57	71.67	71.71	71.74	71.75	71.76	71.76	71.76	71.76	71.76	71.76

Likewise, in Fig. 5, shows the ciphertext expansion between proposed RNS-based N-prime model and N-prime model where the proposed model is able to reduce the growth of ciphertext file size by approximately 72% in comparison to N-prime model as shown in Table 3. This was achieved due to encryption process that was done based on residue number system with respect to moduli set.

Finally, the security of the proposed model still depends on problem of factorization of integers to their primary numbers as mentioned in [7], such integers are product of multiple N-prime numbers and big prime number that serve as secret key that is unknown to attacker.

5 Conclusion

In this paper, we proposed RNS-based N-prime model as an improvement over N-prime model for real-world applications of symmetric-based FHE scheme that requires lower latency and reduced ciphertext file size. The proposed model is based on residue number system approach and ensures parallel computation at encryption stage which brings about efficiency and robustness in addition to the homomorphic properties. The proposed model also deals explicitly with text after converting it to ASCII value instead of binary system, pass it to RNS forward conversion module in conjunction with moduli set and then pass it to the encryption block. The results show that the proposed model outperform N-prime model in terms of encryption execution time and ciphertext expansion at 72% reduction in ciphertext file size.

Further research can be investigated to gain better decryption execution time, as it is taking larger execution time to decrypting ciphertext file back to the original plaintext or result of the computation than encrypting its corresponding plaintext.

Appendix A: Ciphertext

[[0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [24, 5, 13], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13], [0, 6, 14], [24, 5, 13], [24,

5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [24, 5,
13], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [0,
6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4,
12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5, 13], [24, 5, 13],
[23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6,
14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14],
[23, 4, 12], [23, 4, 12], [24, 5, 13], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24,
5, 13], [0, 6, 14], [24, 5, 13], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [0,
6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24, 5, 13],
[23, 4, 12], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6,
14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14],
[23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23,
4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13], [24, 5, 13], [23, 4,
12], [0, 6, 14], [24, 5, 13], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13],
[24, 5, 13], [24, 5, 13], [0, 6, 14], [24, 5, 13], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6,
14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23,
4, 12], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5,
13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [24, 5, 13], [24, 5, 13], [0, 6, 14], [23, 4, 12],
[23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4,
12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24,
5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12],
[0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6,
14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0,
6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4, 12],
[24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24, 5,
13], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12],
[0, 6, 14], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4,
12], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24,
5, 13], [23, 4, 12], [24, 5, 13], [24, 5, 13], [23, 4, 12], [24, 5, 13], [24, 5, 13], [0, 6,
14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [0, 6, 14],
[0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6, 14], [24, 5,
13], [23, 4, 12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13],
[23, 4, 12], [24, 5, 13], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [23,
4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6, 14],
[23, 4, 12], [0, 6, 14], [0, 6, 14], [24, 5, 13], [24, 5, 13], [0, 6, 14], [23, 4, 12], [24, 5,
13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14],
[0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [23,
4, 12], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6,
14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [0,
6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14],
[23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [0,
6, 14], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5,
13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [24, 5, 13], [24, 5, 13],
[0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13], [24, 5, 13], [23, 4, 12], [24,
5, 13], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13],

[23, 4, 12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5, 13], [24,
5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6, 14],
[23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24, 5, 13], [24, 5, 13], [23, 4,
12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12],
[23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6,
14], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12],
[0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24, 5,
13], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13], [24, 5, 13],
[23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6,
14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [24, 5, 13],
[0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [23,
4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5,
13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24, 5, 13],
[23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24,
5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [24, 5, 13], [0, 6, 14], [23, 4, 12],
[0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24,
5, 13], [0, 6, 14], [24, 5, 13], [24, 5, 13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5,
13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14],
[23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4,
12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12],
[0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5,
13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14],
[0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4,
12], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13],
[23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [23,
4, 12], [0, 6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [0, 6, 14], [24, 5, 13], [24, 5, 13],
[0, 6, 14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24, 5,
13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0,
6, 14], [24, 5, 13], [24, 5, 13], [23, 4, 12], [24, 5, 13], [23, 4, 12], [24, 5, 13], [23, 4,
12], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12],
[0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4,
12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13],
[0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24,
5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14],
[0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [24, 5,
13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12],
[23, 4, 12], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24,
5, 13], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5,
13], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13], [24, 5, 13],
[23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0,
6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4,
12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12],
[24, 5, 13], [23, 4, 12], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [23, 4,
12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14],
[24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6,

14], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [0, 6, 14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [23, 4, 12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [24, 5, 13], [23, 4, 12], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [24, 5, 13], [23, 4, 12], [23, 4, 12], [24, 5, 13], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [0, 6, 14], [0, 6, 14], [23, 4, 12], [24, 5, 13], [23, 4, 12], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12], [24, 5, 13], [0, 6, 14], [0, 6, 14], [0, 6, 14], [0, 6, 14], [23, 4, 12], [23, 4, 12], [0, 6, 14], [23, 4, 12]].

References

1. Puthal, D., Sahoo, B., Mishra, S., Swain, S.: Cloud computing features, issues, and challenges: a big picture. In: 2015 International Conference on Computational Intelligence and Networks, pp. 116–123 (2015)
2. Stojmenovic, I., Wen, S.: The fog computing paradigm: Scenarios and security issues. In: 2014 Federated Conference on Computer Science and Information Systems, pp. 1–8 (2014)
3. Bokhari, M.U., Makki, Q., Tamandani, Y.K.: A survey on cloud computing. In: Big Data Analytics, pp. 149–164. Springer (2018)
4. Chang, V., Kuo, Y.-H., Ramachandran, M.: Cloud computing adoption framework: a security framework for business clouds. Future Gener. Comput. Syst. **57**, 24–41 (2016)
5. Patel, N., Oza, P., Agrawal, P.: Homomorphic cryptography and its applications in various domains. In: International Conference on Innovative Computing and Communications, pp. 269–278 (2019)
6. Prasad, J.P., Ramulu, B.S.: Homomorphic encryption security for cloud computing. Global J. Comput. Sci. Technol. (2019)
7. Mohammed, M.A., Abed, F.S.: An improved Fully Homomorphic Encryption model based on N-Primes. Kurdistan J. Appl. Res. **4**, 40–49 (2019)
8. Gentry, C., Boneh, D.: A fully homomorphic encryption scheme vol. 20: Stanford University Stanford (2009)
9. Ogburn, M., Turner, C., Dahal, P.: Homomorphic encryption. Procedia Compu. Sci. **20**, 502–509 (2013)

10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Stoc, pp. 169–178 (2009)
11. Archer, D., Chen, L., Cheon, J.H., Gilad-Bachrach, R., Hallman, R.A., Huang, Z. et al.: Applications of homomorphic encryption. Technical report, HomomorphicEncryption. org, Redmond, WA (2017)
12. Brenner, M., Perl, H., Smith, M.: Practical applications of homomorphic encryption. In: SECRYPT, pp. 5–14 (2012)
13. Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C.A., et al.: A guide to fully homomorphic encryption. IACR Cryptology ePrint Archive **2015**, 1192 (2015)
14. Laine, K.: Homomorphic encryption. In: Responsible Genomic Data Sharing, pp. 97–122. Elsevier (2020)
15. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 575–584 (2013)
16. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Annual Cryptology Conference, pp. 21–39 (2013)
17. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 3–33 (2016)
18. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Improving TFHE: faster packed homomorphic operations and efficient circuit bootstrapping (2017)
19. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 617–640 (2015)
20. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Annual Cryptology Conference, pp. 75–92 (2013)
21. Gao, S., MathCrypt, S.B.: Efficient fully homomorphic encryption scheme. IACR Cryptology ePrint Archive **2018**, 637 (2018)
22. Dhakar, R.S.,Gupta, A.K., Sharma, P.: Modified RSA encryption algorithm (MREA). In: 2012 Second International Conference on Advanced Computing & Communication Technologies, pp. 426–429 (2012)
23. Hu, Y.: Improving the efficiency of homomorphic encryption schemes (2013)
24. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations Secure Comput. **4**, 169–180 (1978)
25. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**, 469–472 (1985)
26. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: International Workshop on Public Key Cryptography, pp. 420–443 (2010)
27. Orsini, E., Smart, N.P., Vercauteren, F.: Overdrive2k: efficient secure MPC over **Z**_2k from Somewhat Homomorphic Encryption. In: Cryptographers' Track at the RSA Conference, pp. 254–283 (2020)
28. Gentry, C., Halevi, S.: Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, pp. 107–109 (2011)
29. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Annual Cryptology Conference, pp. 505–524 (2011)
30. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**, 831–871 (2014)
31. Chen, Y., Gu, B., Zhang, C., Shu, H.: Reliable fully homomorphic disguising matrix computation outsourcing scheme. In: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 482–487 (2018)
32. Van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 24–43 (2010)

33. Brakerski, Z.: Fundamentals of fully homomorphic encryption. In: Providing Sound Foundations for Cryptography, pp. 543–563 (2019)
34. Burtyka, P., Makarevich, O.: Symmetric fully homomorphic encryption using decidable matrix equations. In: Proceedings of the 7th International Conference on Security of Information and Networks, p. 186 (2014)
35. Gupta, C., Sharma, I.: A fully homomorphic encryption scheme with symmetric keys with application to private data processing in clouds. In: 2013 Fourth International Conference on the Network of the Future (NoF), pp. 1–4 (2013)
36. Li, J., Wang, L.: Noise-free Symmetric Fully Homomorphic Encryption based on noncommutative rings. IACR Cryptology ePrint Arch. **2015**, 641 (2015)
37. Liang, M.: Symmetric quantum fully homomorphic encryption with perfect security. Quantum Inf. Process. **12**, 3675–3687 (2013)
38. Umadevi, C., Gopalan, N.: Outsourcing private cloud using symmetric fully homomorphic encryption using Q^n_p matrices with enhanced access control. In: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 328–332 (2018)
39. Qu, Q., Wang, B., Ping, Y., Zhang, Z.: Improved cryptanalysis of a fully homomorphic symmetric encryption scheme. Secur. Commun. Netw. **2019** (2019)
40. Vizár, D., Vaudenay, S.: Cryptanalysis of Enhanced MORE (2018)
41. Hariss, K., Noura, H., Samhat, A.E.: Fully Enhanced Homomorphic Encryption algorithm of MORE approach for real world applications. J. Inform. Secur. Appl. **34**, 233–242 (2017)
42. Hariss, K., Noura, H., Samhat, A.E., Chamoun, M.: An efficient solution towards secure homomorphic symmetric encryption algorithms. In: ITM Web of Conferences, p. 05002 (2019)
43. Hariss, K., Noura, H., Samhat, A.E., Chamoun, M.: Design and realization of a fully homomorphic encryption algorithm for cloud applications. In: International Conference on Risks and Security of Internet and Systems, pp. 127–139 (2018)
44. Kipnis, A., Hibshoosh, E.: Efficient methods for practical fully homomorphic symmetric-key encrypton, randomization and verification. IACR Cryptology ePrint Arch. **2012**, 637 (2012)
45. Vizár, D., Vaudenay, S.: Cryptanalysis of chosen symmetric homomorphic schemes. Studia Scientiarum Mathematicarum Hungarica **52**, 288–306 (2015)
46. Kim, J., Shim, H., Han, K.: Comprehensive Introduction to fully homomorphic encryption for dynamic feedback controller via LWE-based cryptosystem. In: Privacy in Dynamical Systems, pp. 209–230. Springer (2020)
47. Hosseinzadeh, M., Navi, K.: A new moduli set for residue number system in ternary valued logic. J. Appl. Sci. **7**, 3729–3735 (2007)
48. Bankas, E.K., Gbolagade, K.A.: A New efficient RNS reverse converter for the 4-moduli set world academy of science, engineering and technology. Int. J. Comput. Electr. Autom. Control Inform. Eng. **8**, 328–332 (2014)
49. Younes, D., Steffan, P.: A comparative study on different moduli sets in residue number system. In: 2012 International Conference on Computer Systems and Industrial Informatics, pp. 1–6 (2012)
50. Omondi, A.R., Premkumar, B.: Residue number systems: theory and implementation vol. 2: World Scientific (2007)