Research Article

Ensuring the Confidentiality of Nuclear Information at Cloud Using Modular Encryption Standard

Aysha Shabbir, Maryam Shabbir, Muhammad Rizwan , and Fahad Ahmad

Department of Computer Science, Kinnaird College for Women, Lahore, Pakistan

Correspondence should be addressed to Muhammad Rizwan; mrizwan.hust@yahoo.com

Received 5 August 2019; Revised 7 November 2019; Accepted 21 November 2019; Published 18 December 2019

Academic Editor: Leandros Maglaras

Copyright © 2019 Aysha Shabbir et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Lifblood of every organization is its confidential information. The accentuation on cybersecurity has expanded considerably in the course of the last few years because of the expanded number in attacks at the individual and organization and even at the state level. One specific zone of consideration is the assurance of the security of nuclear information. This may relate to both Instrumentation and Control (I&C) and Information Technology (IT). The present security measures are insufficient for nuclear information because of their lack of identification, classification, and securing measures (because of their multifaceted nature). With the increasing trends of data storage and management with the assistance of cloud, data confidentiality threats are immensely increasing. As there is no such safeguard that can make our systems a hundred percent secure, the best approach is to provide security at distinct layers. The basic purpose of layered security is to have the benefit that if one layer fails or compromised, the other layer compensates or maintains that confidentiality with the access control in the owner's hand. In this paper, we have proposed a multilevel approach with protection-based computing by using Modular Encryption Standard (MES). We proposed a cloud framework as well to further enhance its security by utilizing a multicloud and modular approach. By performing simulations, the obtained results depicted that our proposed scheme works efficiently than other commonly used schemes.

1. Introduction

The IAEA (International Atomic Energy Agency) has provided suggestions on nuclear security against radioactive material and its associated resources. The general goal of the nuclear security regime is to ensure the security of people, society, and the environment from malevolent acts [1]. A “nuclear security plan” is created as a component of the nuclear security regime of a state. The confidentiality of nuclear information is a critical constituent of this plan. The demand for security of confidential information is not a new experience [1]. Cryptography has been utilized for a huge number of years in distinct domains, for example, in battles and political and judicial issues. It is the investigation of systems for secure transmission, which can be dated back to old Greece and Egypt.

Confidential nuclear information is highly vulnerable these days because of the rise of cyber warfare, cyberterrorism, and hacking. One quite certain domain of information confidentiality with expanding risks is nuclear information security. Information confidentiality is a vital

piece of nuclear security. The security of nuclear systems is considerate of the security of nuclear information from the threats explicit to nuclear frameworks. The aims of security of nuclear information constitute security against the larceny of assets (both information and physical); security against cyberterrorist acts; and security against a joined cyber assault, guaranteeing business progression, nuclear protection, and assurance against loss of nuclear confidential or additionally characterized information [1, 2]. The following section will explain the cloud-computing backdrop.

Cloud computing (CC) refers to network-based computing which provides shared computing power rather than relying on the local server to personal devices for high-level computation. The generic model for cloud computing is shown in Figure 1.

But when we use CC for nuclear information monitoring, confidentiality is the first and foremost issue to consider [5]. So, this paper is concerned about nuclear information security against cloud computing using cryptography techniques.

FIGURE 1: Cloud computing.

The structure for the rest of the paper is as follows. Section 2 presents the literature review. Section 3 presents an overview of the proposed framework. Section 4 demonstrates complete MES working. Section 5 presents performance analysis. Section 6 provides the application domain of the proposed scheme, and finally, the paper concludes with Section 7.

1.1. Problem Statement. The general target of nuclear security administration is to ensure the security of people, society, property, and the environment from harmful utilization of nuclear information. Now, with the immense increase in cyberattacks, nuclear information security has accumulated worldwide attention. Individuals and groups aiming to devise any spiteful action including radioactive material or nuclear material may be benefited by access to confidential information. Information monitoring and outsourcing at cloud ought to be managed in order to guarantee that it is not unintentionally imparted to or uncovered to the attackers or other intruders. Such information maintenance ought to be done by means of identification, classification, and securing with the adequate measures according to the user-specified preferences.

2. Literature Review

Many researchers have proposed different approaches for the security of confidential information.

The Data Encryption Standard is the US Government affirmed enciphering protocol. Its production has embraced open review, as it is exceptionally acclaimed and broadly acknowledged as a symmetric ciphering scheme in organizations and business places. ANSI (American National Standard Institute) has acknowledged it as US National Standard. It comprises Feistel cipher-based 16 rounds. This algorithm takes 64-bit plaintext as the input and by using the 64-bit key; it results in the 64-bit cipher text. The real change is performed by 56 bits of the key, which are enrolled as autonomous bits, and the remainder 8 bits are used for the purpose of error detection [7].

Triple DES proposed after the DES protocol, introduced in the mid 1970s, utilizes a 56-bit key. The viable security 3DES relies on the 112 bits against the meet-in-the-middle attacks. Triple DES runs multiple times slower than DES yet is much progressively secure whenever utilized appropriately. The method for deciphering is equivalent to the method for enciphering, aside from it running in the reverse manner [8].

The US Government discipline and NIST ("National Institute of Standards and Technologies") in 1997 stepped up and discovered a DES alternative. Because of mechanical, processing, and power progresses, DES expressed as less secure. The point behind this crusade by the US and NIST was the recognizable proof of DES alternative as a security, ensuing effort for nonmilitary applications. It would be useful for commercial as well as non-governmental purposes. Among all the submissions, AES was specified as the most recent and most secure protocol as a block cipher with three distinct kinds of keys ranging from 128 bits to 256 bits [9].

Blowfish is a Feistel structure-originated protocol and has block cipher classification. As input, it takes a 64-bit data chunk. Its key size changes between 32 and 448 bits. Its execution encompasses 16 rounds. Two significant functions of this protocol are the key extension and data enciphering. There is no reliance on keys and s-boxes. It requests for more execution time because of its variety of key sizes. As the composition of the sub-key sets uses extra time for execution, in case of brute force attack, it results in extraordinary trouble. Long-term security is given by Blowfish, with no concealed weaknesses. Its unwavering quality is incredibly influenced because of the broad utilization of less-secure keys. Four initial rounds are uncovered to differential attacks [9].

RC5 is a block cipher symmetric key algorithm, prominent for its clarity. Structured by Ronald Rivest in 1994, RC means "Rivest Cipher." A new highlight of RC5 is the overwhelming utilization of rotation that is data-dependent. It possesses varying sized words, varying round numbers, and a variable-length private key. It comprises three parts: encryption, decryption, and key expansion algorithm [10].

Data coloring and software watermarking techniques are used for protecting shared data objects with multiway authentications which tighten access control of confidential information in any type of cloud, private or public. In [11], they have proposed that each file uploaded by the data owner is encrypted into TTP-based hash code and the user gets the secure e-mail with the complete file. Other key details in secure cloud data sharing using a trusted third party is discussed here.

The privacy of confidential information remains the most significant concern while utilizing public cloud storage. One of the rising techniques for tending to this problem is cryptography. Bentajer et al. [12] presented the ID-based encryption design, i.e., CS-IBE, for using public cloud storage that targets to secure the confidentiality of sensitive information. According to CS-IBE, configuration files are associated with single file accession practice with the client ID which will be utilized as the key to encipher. User ID is used to encrypt the files before uploading them to third-

party cloud providers, so for the outsourced information, it will add a layer of security. Moreover, CS-IBE operates like an overlay framework against cloud storage.

Dorairaj et al. proposed a system that depends on a multispect/multilevel scheme for secure data storage in the cloud. Initially, the data are evaluated, characterized, and divided appropriately if necessary. Furthermore, data are secured from intruders by utilizing enciphering techniques depending on the sensitivity and criticality. The confidential information is secured from unapproved access by the consideration of Mandatory Access Control that incorporates multifaceted verification that relies on categorization. Consequently, in the log register, all types of access, i.e., authorized or unauthorized, are recorded which can be utilized later for foreseeing the attacks, taking control estimates dependent on the attacks, attempted by reclassifying or updating the measures of security to adapt the data changing sensitivity as indicated by the business requirements [13].

Different security attacks were identified by classifying security concerns at multiple levels. A new dimension was provided in [14] by highlighting threats at each cloud layer. These security layers can also have access as low, medium, and high. The security requirements, e.g., data privacy, multitenancy, and data encryption, were mapped to different cloud security issues for achieving confidentiality and integrity in the cloud environment.

To enable complete security transparency analysis at a cloud, a framework was proposed in [15]. In the cloud, security transparency is probably going to turn into the key concern that supports a proper revelation of security practices and designs that intensify client confidence towards cloud services. In this paper, they presented a structure that empowers an investigation of security transparency for cloud-enabled frameworks. Specifically, they considered security transparency from three distinct degrees of deliberation, i.e., organizational, technical, and conceptual levels, and from the perspective of these levels, they identified some important facets as well.

Data are protective if they satisfy the three conditions, specifically CIA, i.e., Confidentiality, Integrity, and Availability. Confidentiality is attained with the assistance of cryptography in cloud computing. Symmetric enciphering algorithms, eminently the Blowfish scheme, showed remarkable achievement. In [16], a distinct technique of the Blowfish algorithm utilizing the Shuffle algorithm was proposed against the assurance of data confidentiality at the cloud.

Remote information storage presents difficulties such as unsystematic use of assets and threat from insider's attack to information at rest in the distributed storage. In [17], an architecture is proposed for distributed allocation of storage for reasonable use of resources and likewise an incorporated end-to-end protective system for information at rest to take out insider's risks in the cloud storage.

For directing security requirements, HGKA-OA (i.e., hierarchical group key agreement protocol using orientable attribute) is presented in [18]. Using this scheme, diverse confidential data can be shared among many individuals who have distinct levels of authority. Expecting distinct levels of authorization, where one individual has confidential

data, he can replace data with certain individuals (i.e., those who have the specific degree of security consents) instead of all individuals of the group.

When a client enciphers his information by deterministic enciphering technique, a frequency attack is a critical issue to consider. Moreover, clients' information security is probably going to be uncovered to servers at the cloud when their encoded information is upgraded. In order to tackle these issues, in [19], the author proposed an efficient enciphering query technique over the data (that is outsourced to the third-party provider). Using this protocol, clients' information is enciphered based on every single feasible query, to fulfill clients' requests. Moreover, a twofold AES enciphering strategy is proposed to tackle the deterministic encryption-based frequency attacks.

In [21], the author proposed a security module dependent on a "Field Programmable Gate Array (FPGA)" to alleviate man-in-the-middle attack in nuclear power plants. It additionally provides support for applications that demand cybersecurity with embedded computing (i.e., a model-based engineering technique is provided). This FPGA-relied security module is proposed to tackle the attacks that are specifically intended to gain access to confidential information by gaining the system's physical access.

A steganography tool dependent on DCT is actualized in [2] to secure the nuclear reactor's confidential and sensitive data, utilizing the middle frequency-based "sequential embedding strategy." It was indicated that the proposed scheme improves the accuracy and security of the information and supplies a large capacity of embeddedness without distorting the resulting visual image.

3. Overview of the Proposed System

This scheme starts with the identification and classification of the intended information. This identification and classification would be from the perspective of the degree of security required. First of all, at the user side (at first level), an auto-generated key will be provided through the auto-generated key module (i.e., entropy-based key generation/randomly generated bits, i.e., to make it hintless). The data owner has the choice of keys. He/she can choose any of the keys according to his/her requirements, e.g., as mentioned in the classification step. At the next level, data are encrypted to some extent through the extender/contractor module (this module will accept 56 bits and extend those to 64 bits by adding 4 random bits at the beginning and 4 random bits at the last). After passing through the extender/contractor module, it is passed to the intermediary cloud. So, data are not handed over to the 3rd party cloud (i.e., intermediary cloud) in actual form but instead in the extended form (i.e., the data transmitted to the intermediary cloud are not the actual form of data but extended form). Here, the intermediary cloud is responsible for cryptography (for performing step 3, i.e., securing), and here, data would be split into various blocks and these blocks will be stored at different clouds (in encrypted form).

As it is a modular encryption algorithm, one module (for autogenerated key) is implemented on the user side, a

second module for extension/contraction before uploading to cloud, and the rest of the modules on the intermediary cloud; hence, this modular approach at distinct levels leads us to multilevel security. Thus, even CSP cannot have access to our confidential data because each CSP has access to just a block of data (in encrypted form) and not to the entire data, i.e., protection not just from outsiders but from insider's access as well. When a user requests for data, this request will proceed to the intermediary cloud as well as to the data owner. Requested data would be sent to the user through the intermediary cloud; secondly, the encryption scheme and extender/contractor module scheme would be shared by the data owner to the user through a secure message. The multilevel modeling of our proposed algorithm can be seen in Figure 2. Our proposed framework does not specify any particular type of cloud (i.e., private, public, community, or hybrid), and it can be used for any cloud environment, but we have proposed a new cloud storage framework (utilizing multicloud and modular paradigm).

Our proposed algorithm lies in the category "c" identified in [6]. Hence, this approach does not just protect us against insider attacks but also from outsider unauthorized access, and it can work against any other cryptography-based symmetric block cipher requirements.

4. Methodology

A multilevel security approach through MES at cloud against nuclear information security has been proposed as shown in Figure 3. Our proposed scheme comprises three major steps, i.e., Identification, Classification, and Securing. This solution can work against any type of cloud.

4.1. Identification. The need for protecting nuclear information is regulated by performing identification and classification according to the degree of risk factor. Here, we have to identify the sensitivity and criticality of the data. The identification of nuclear information is based upon user-specified requirements. It generally has two broad categories (with further subcategories): confidential information, which requires protection, and public information, which does not require protection.

Public Information

(i) Category 1: publicized information

Confidential Information

- (ii) Category 2: the information which is selected to be kept confidential, but whose disclosure would not result in any risk
- (iii) Category 3: the information whose disclosure would result in the threat of damage to nuclear warheads
- (iv) Category 4: the information if disclosed would almost certainly result in a serious threat to nuclear weapons or the people
- (v) Category 5: the information if disclosed would almost certainly result in a severe threat to nuclear warfare

Accordingly, there would be five categories based on the confidentiality level of information. The user would specify his preferences to secure his information such as category 1, category 2, etc., based upon the sensitivity of the information.

4.2. Classification. Nuclear information classification scheme decides the level of confidentiality of the information. This is helpful in choosing the information that actually should be protected, and it ultimately diminishes the cost of security. We have classified these two categories into five different subcategories (based upon the sensitivity level). These five different subcategories are mentioned below. In the securing step, we have proposed five different types of keys for the five subcategories mentioned as follows (based upon the sensitive level of nuclear information):

Public information

(i) Nonconfidential Information

Confidential information

- (ii) Less confidential information
- (iii) Moderately confidential information
- (iv) Highly confidential information
- (v) Extremely high confidential information

4.3. Securing. Here, we are going to explain the entire securing step thoroughly after classifying the type of information as shown in Figure 4. This algorithm is a symmetric block cipher; it will accept 56-bit plaintext as an input (block size). Our proposed algorithm categorization can be seen in Figure 3. Secondly, a simplified encryption model is illustrated in Figure 5.

4.3.1. Extension/Contraction. The extension (at encryption side) is the addition of random 4 bits at the beginning of the 56-bit chunk and at the end of the chunk. Similarly, decryption side contraction is the removal of 4 bits from the left and right side. Now, it will pass through the key whitening step. The main reason behind this extension is that data are not handed over to the third-party cloud provider in actual form but in extended form.

4.3.2. Key Whitening. This step constitutes three substeps, i.e., expansion, key addition, and contraction. In this step, we will first expand our data from 64 bits to 128 bits. The expansion process is shown in Figure 6. This step performs processing on data before the 1st round by performing key addition (for merging of data with key portions). We used here a large-size key to increase security, and for key addition, the data size should be equal to the key size. Therefore, we performed extension, and after key whitening, we performed the contraction on data (to get the actual size).

One data block would have eight bytes, and it would be converted to sixteen bytes by generating two bytes from a single one.

FIGURE 2: Layered modeling.

FIGURE 3: MES (block cipher categorization).

For this, we will perform XOR for every two bits and will get the 3rd bit after every two bits. Now, we have converted every eight bits to 12 bits. Moreover, XOR every three bits and we will get the fourth bit after every three bits. Our data have been converted to 16 bits from eight 8 bits, for every chunk (1 byte). This step will be performed on 8 bytes, and consequently, we will get 16 bytes (expansion).

Now, we will perform key whitening by performing XNOR (it checks the logical equality of input bits) on key 0 with our expanded 128-bit data. After key addition, we will contract our data by discarding every two bits following two bits, like 3rd, 4th, 7th, 8th, and so forth. Now, we would have 64-bit data at the end of the key-whitening step. After this, data would pass to round 1.

4.3.3. Encryption

(1) Round 1

(i) Permutation (odd round): now, we would have 64-bit data in the 8×8 matrix by taking the first two rows (16 bits) from the left to right manner. Here, these numbers represent the location of the bits. The 8×8 matrix is shown in Figure 7. We would tackle this by taking two rows at a time. Likewise,

we would take the first two rows from the left to right manner and place them in the 4×4 matrix as shown in Figure 8. The second time, we would take the next two rows of this 8×8 matrix from the right to left manner according to Figure 7 and so on. The 4×4 -matrix permutation process is shown in Figure 9.

Now, place the first two rows of Figure 7 (i.e., first 16 bits) in a 4×4 matrix.

Permutation would be performed in a way elaborated as follows (Figure 9).

1st location would replace 16th location bit, 4th would replace 13th, 6th would replace 11th bit, and 7th would be replaced 10th bit, just like a small matrix within a large one. Figure 10 demonstrates the output 4×4 matrix, i.e., after permutation.

(ii) Shifting: shifting of 64-bit matrix would be performed in manner like "X."

1st location bit would replace 8th location bit, 10th would replace 15th, 19th would replace 22nd, 28th would replace 29th and so on. Triangles below, up, left, and right of the cross would be replaced as follows: the left triangle would replace the right triangle and the upper triangle would replace the lower one. This step constitutes diffusion. Figure 11 shows preshifting step and Figure 12 shows the postshifting step, i.e., the 8×8 matrix.

(iii) Substitution: substitution would be performed by looking at the values through the s-box (lookup table), i.e., Table 1. As input is the 64-bit matrix with each row containing 8-bits, we divide each row into two nibbles (i.e., the first four bits in a row represent one nibble and the second four bit represent the second nibble). We would have to substitute each nibble with the s-box value (first two digits represent a row of s-box and second two columns represent a column of s-box) which indicates a certain cell of s-box. This particular value is the substituted value. This step involves confusion (which makes it hintless).

(iv) Key addition: now, we would XOR our 64-bit data with the key 1's first 64 bits

(v) Key subtraction: after this, we would XNOR our data with the next 64 bits of key 1.

FIGURE 4: The proposed algorithm.

FIGURE 5: Simplified encryption model.

01001101
 011000110011
 0110000011000110

Example of expansion of a single byte to two bytes

FIGURE 6: Expansion.

(2) Round 2

(i) Permutation (even round). In round 2, the permutation process is changed from round 1. Figure 13 shows the 64-bit matrix. Extract 1st sixteen bits and write them in a 4×4 matrix (as depicted in Figure 14) and each row of

that matrix (nibble) will be written in a 2×2 matrix and permute as diagonal bits would replace by itself (2nd and 3rd) as shown in Figure 15. After processing 4 rows in this way and writing them in a 4×4 matrix, we would again permute this 4×4 matrix by replacing 1st and last bits (1st

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

FIGURE 7: 8 × 8-matrix permutation.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

FIGURE 8: 4 × 4 matrix (first two rows).

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

FIGURE 9: First two rows (before permutation).

and 16th) and replace the diagonal digits with others in the same diagonal as shown in Figure 16.

The 2nd location bit would replace the 5th location bit, 3rd would replace 9th, 6th would remain unshifted, 7th would replace 10th, 4th would replace 13th, 8th would replace 14th, 11th would remain unshifted, and 12th would replace 15th. In this way, just 16 bits will be permuted. We will repeat these

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

FIGURE 10: First two rows (after permutation).

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

FIGURE 11: Before shifting.

8	58	59	60	61	62	63	1
16	15	51	52	53	54	10	9
24	23	22	44	45	19	18	17
32	31	30	29	28	27	26	25
40	39	38	37	36	35	34	33
48	47	46	20	21	43	42	41
56	55	11	12	13	14	50	49
64	2	3	4	5	6	7	57

FIGURE 12: After shifting.

steps for complete 64 bits by taking 16 bits of data at a time. We will perform the same permutation in the 3rd round and in the 4th round, and the 1st round's permutation will be used. Therefore, in the 1st, 4th, and 7th rounds, the same permutation will be used and all other rounds will have the same permutation. Figure 15 shows the 2 × 2-matrix

TABLE 1: S-box.

	00	01	10	11
00	0101	1101	0100	1100
01	0111	1111	0110	1110
10	0001	1001	0000	1000
11	0011	1011	0010	1010

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

FIGURE 13: 64-bit matrix.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

FIGURE 14: 16-bit matrix.

1	2
3	4

FIGURE 15: 2×2 matrix permutation.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

FIGURE 16: 4×4 matrix permutation.

permutation, and the 4×4 -matrix permutation is shown in Figure 16. The rest of the steps will be the same as round 1.

4.3.4. Decryption

(1) Round 1

(i) Key subtraction: the ciphertext of 64 bits will XOR with 64 bits of the key 9

(ii) Key addition: data will XNOR with the next 64 bits of key 9

(iii) Inverse substitution: inverse substitution by looking at data in inverse s-boxes

(iv) Inverse shifting: inverse shifting will be the same as shifting in encryption

(v) Inverse permutation: inverse permutation will be the same as encryption but opposite (arrows) in direction

(2) Round 2

In the second round, all other steps will be the same as round 1, and just inverse permutation will differ (as defined in encryption). Inverse permutation will be as like in encryption in round 2.

4.3.5. Key Transformation (5 Types of Keys)

(1) For 128-Bit Key. The key is divided into 4 chunks (32 bits or 4 bytes for each chunk).

Step 1: first, multiplications will be performed on 1st and 3rd chunk and 2nd and 4th chunk (byte-by-byte multiplication). From this, we will get two chunks; each chunk consists of four parts.

Step 2: from the highest common factor (HCF) between 1st part of first chunk and 1st part of second chunk, 2nd part of first chunk and 2nd part of second chunk, 3rd part of first chunk and 3rd part of second chunk and 4th part of first chunk and 4th part of second chunk, we will get a single chunk with four parts.

Step 3: each part may exceed 8 bits; we will reduce it to 8 bits by dividing each bit with fix prime polynomial of " $x^8 + x^6 + x^3 + x^2 + x$." Now, the chunk obtained with four parts, and each part will be equal to 1 byte (total 32 bits).

Step 4: for each byte, we will get first nibble's XOR result and second nibble's XOR result. As a result, we will get two bits from each byte, these two bits will XOR with each other, and we will get a single bit from each byte. Since the whole chunk has four bytes, we will get 4 bits after processing each byte.

By performing XNOR on first two bits and 3rd and 4th bit, we will get 2 bits. XOR these two bits again and now the whole chunk with four bytes has been reduced to a single bit.

Step 5: now, this single bit will process each byte of the chunk obtained after step 3 (derived from mathematical synthetic division concept). We named it "Synthesion." Synthesion process is shown in Figure 17.

Step 6: after this, we will get a new chunk with four bytes. This single chunk (bytes) will XNOR each of the

FIGURE 17: Synthesis.

four chunks obtained at the beginning before step 1 (on key 0). After this, the new chunk obtained is our key 1. The same steps will be repeated for key 2, key 3, key 4 to key 9. Thus, 10 keys will be generated by a nine-time transformation of key 0.

(2) *For 160-Bit Key.* The key is divided into 5 chunks (each of 32 bits). The fifth chunk will not be used at the beginning, but instead, we will XNOR that chunk with the chunk obtained after step 3. After this, the same steps will be performed in the same sequence except step 6 where we will just utilize the last four chunks of key 0 to XNOR (instead of complete key 0).

(3) *For 192-Bit Key.* The key is divided into 6 chunks.

Step 1: modulo-2 multiplications will be performed on 1st and 4th chunk, 2nd with 5th chunk, and 3rd with 6th chunk. It will result in three chunks; then, we will again multiply (modulo-2 multiplication) first and third chunk. Place this output chunk first and then the second chunk is placed. Finally, we have two chunks.

Step 2: HCF of 1st part of each chunk, 2nd part of each chunk, 3rd part of each chunk, and 4th part of each chunk; from this, we will get a single chunk and next steps will remain the same as for 128-bit key except step 6 where we will just utilize middle four chunks of key 0 to XNOR (instead of complete key 0).

(4) *For 224-Bit Key.* The key is divided into 7 chunks. The seventh chunk will only be used for XNOR with the chunk obtained after step 3. Therefore, we have the first six chunks. The same step sequence with the same method will be used for the rest of the key transformation as like the 192-bit key.

(5) *For 256-Bit Key.* For the 256-bit key, we will have 8 chunks.

Step 1: modulo-2 multiplications will be performed on 1st and 5th chunk, 2nd and 6th chunk, 3rd and 7th chunk, and 4th and 8th chunk. Now, we will get four chunks. Multiply (modulo-2) first and fourth chunk and second and third chunk. Now, we will get two chunks.

A = 1	B = 2	C = 3	D = 4	E = 5	F = 6
G = 7	H = 8	I = 9	J = 10	K = 11	L = 12
M = 13	N = 14	O = 15	P = 16	Q = 17	R = 18
S = 19	T = 20	U = 21	V = 22	W = 23	X = 24
		Y = 25	Z = 26		

FIGURE 18: Key encryption-decryption scheme.

Step 2: HCF of 1st part of each chunk, 2nd part of each chunk, 3rd part of each chunk, and 4th part of each chunk will be taken. From this, we will get a single chunk. The rest of the steps are the same except step 6 where we will make two parts of key 0 (one part of the first two chunks and last two chunks (each chunk with 4 sub-parts) and the second part of the middle four chunks). First, we will XNOR these two chunks and it will result in a single chunk (with four subparts). Now, this chunk will be XNORed with step 5 results. The rest of the key transformation is the same as explained for 128 bits. Next keys (i.e., key 2, key 3 to key 9) will also use 128-bit key transformation steps. No matter whether the key is of 128 bits, 160 bits, 192 bits, 224 bits, or 256 bits, the key whitening step will take only 128 bits and key addition and key subtraction will take 64 bits at a time.

4.3.6. Key Encryption and Decryption

(1) *Selection of Forward (Encryption) and Backward (Decryption) Key No.* After encrypting the data and converting them to ciphertext, we will encrypt the key as well. For example, for the 128-bit key (4 chunks of 32 bits), each chunk has 4 bytes or 8 nibbles. After performing XOR on each nibble, we will get 8 bits and after performing XOR on the 8 bits, we will get one single bit. Therefore, from each chunk (32 bits or 4 bytes or 8 nibbles), we will get a single bit. From the 4 chunks, we will have 4 bits. Convert these four bits to decimal and get a remainder with 26. The number obtained as remainder will be used for forward and backward intervals (just like Caesar cipher, but Caesar cipher has fixed key of 3 but we will use the number obtained as remainder).

(2) *Key Encryption.* First, each nibble will be converted to decimal and this number is matched in the alphabetic table. Then, the matched number will be forwarded to the times the number obtained as a remainder, and now after forwarding, the matched alphabetic is obtained as the ciphertext form of that nibble.

(3) *Key Decryption.* For decryption, the same alphabet will be matched in the table to its corresponding number and backward move to the times the number obtained as remainder (this number will be shared between the sender and receiver especially for data sharing between the data owner and user) and now the matched number will be converted to binary (required nibble). Key encryption and decryption schemes can be seen in Figure 18.

FIGURE 19: The proposed system hierarchy.

TABLE 2: Inverse s-box.

	00	01	10	11
00	1010	1000	1110	1100
01	0010	0000	0110	0100
10	1011	1001	1111	1101
11	0011	0001	0111	0101

The proposed System Hierarchy is shown in Figure 19.

(4) *S-Box*. In a nonlinear way, s-box and inverse s-box are designed. Table 1 shows the s-box values.

(5) *Inverse S-Box*. Table 2 shows the inverse s-box values.

4.3.7. Significance of the Proposed Algorithm

- (i) Our proposed algorithm provides five keys (two extra keys than AES), but with no extra bit utilization (256 bits same as AES).
- (ii) It constitutes key encryption as well (to enhance security).
- (iii) It is a modular algorithm. Some modules are at the user side (autogenerated), though extender/contractor module data will be extended before uploading to any cloud and other modules for the intermediary cloud. Therefore, the overall benefit is that data will not be handed over in actual form to any third party.
- (iv) Every single key is treated in two different ways, i.e., half key for XORing and half key for XNORing (except key whitening).
- (v) Every single key transforms the data two times (except key whitening). Therefore (except key whitening), the key intermix 18 times with data instead of 9 times (for 9 rounds), as key addition and key subtraction are, in actual, the data and key subsuming. This is the major reason behind 64-bit data and using a large size key to subsume the single block of data (8x8 matrix) by a single key twice.

This technique is proposed against insider and outsider's attack due to multilevel security and multicloud utilization. Comparative analysis of the proposed scheme with commonly used algorithms also has been presented in Table 3.

4.3.8. Mathematical Description of Encryption and Decryption

D° = resulting actual data to be transmitted

D = data after extension

E = expansion bits

P_i = permutation

S_i = substitution

TABLE 3: Comparative analysis.

Algorithm	Alg. structure	Key size	Block size	Rounds	S-box	Round modules (functions)	Key transformation/scheduling	Differentiating factors	Identification	Classification
DES [6]	Feistel structure	56 bits	64	16	8	Permutation, expansion, substitution	Permutation based	No key encryption, a single type of key	No	No
3DES [7]	Feistel structure	112/168 bits	64	48	8	Permutation, expansion, substitution	Permutation based	No key encryption, a single type of key	No	No
AES [8]	Substitution-permutation	128, 192, 256 bits	128	10, 12, 14	1	Shift rows, mix column, add keys	S-box based	No key encryption, three types of keys	No	No
Blowfish [8]	Feistel structure	32-448 bits	64	16	4	Substitution, swapping	Permutation, substitution based	No key encryption	No	No
RC5 [9]	Feistel structure	64	128	20	8	XOR, left rotation, right rotation	"Magic constant" utilization	No key encryption	No	No
MES (proposed)	Substitution-permutation	128, 160, 192, 224, 256 bits	56 bits by extender/contractor module	9	1	Even and odd round permutation; substitution, shifting, key addition	Contraction, mathematical operations, encryption	Key encryption, five types of keys	Yes	Yes

K_0 = key for key whitening

K_{Li} = left key with a 64-bit chunk

K_{Ri} = right key with a 64-bit chunk

(1) *Transmitter*

Step 1: here, we get actual 56-bit data:

$$D^\circ = 56\text{-bit plaintext.} \quad (1)$$

Step 2: 56-bit data extension to 64 bits for equation (1):

$$D = \text{extension}(D^\circ). \quad (2)$$

Step 3: temporary data extension from 64 bits to 128 bits for key whitening for equation (2):

$$D + E \oplus K_0. \quad (3)$$

Step 4: data contraction from 128 bits to 64 bits for equation (3):

$$D + E - E \oplus K_0. \quad (4)$$

Step 5: permutation based on even or odd round for equation (4):

$$P_i(D \oplus K_0). \quad (5)$$

Step 6: substitution based on s-box (i.e., lookup table). Key addition and key subtraction for equation (5):

$$(S_i(P_i(D \oplus K_0)) \oplus K_{Li}) \oplus K_{Ri}. \quad (6)$$

(2) *Receiver*

Step 1: now at the receiver side, key subtraction will first cancel out the effect of key subtraction done at the sender side for equation (6). XOR effect of K_{Ri} on the receiver side will cancel out the XNOR effect of K_{Ri} on the sender side:

$$(S_i(P_i(D \oplus K_0)) \oplus K_{Li}) \oplus K_{Ri} \oplus K_{Ri}. \quad (7)$$

Step 2: XNOR effect of K_{Li} on the receiver side will cancel out the XOR effect of K_{Li} on the sender side for equation (7):

$$S_i(P_i(D \oplus K_0)) \oplus K_{Li} \oplus K_{Li}. \quad (8)$$

Step 3: inverse substitution on the receiver side will cancel out the effect of substitution on the sender side for equation (8):

$$S_i(P_i(D \oplus K_0)) \cdot S'_i. \quad (9)$$

Step 4: inverse permutation on the receiver side will cancel out the effect of permutation on the sender side for equation (9):

- (1) Key selection out of five keys based upon identification and classification
- (2) Entropy-based key generation through the autogenerated key module
- (3) Extension of 56-bit data chunks to 64-bit through the extender
- (4) Key transformation
- (5) Expansion of 64-bit to 128-bit data
- (6) XNOR key 0 with expanded 128-bit data
- (7) 128-bit data contraction to 64-bit
- (8) **if** the round is odd **then**
- (9) Odd round permutation
- (10) **else**
- Even round permutation
- (11) **end if**
- (12) Shifting of the 64-bit matrix in “x” like manner
- (13) Substitution of the 64-bit matrix through substitution-boxes
- (14) Key addition
- (15) Key subtraction
- (16) Repeat steps 8 to 15 till 9 rounds
- (17) Exit

ALGORITHM 1: Data encryption.

- (1) Key generation
- (2) The key division into chunks
- (3) From HCF, get a single chunk
- (4) Fix polynomial division to get standardized form
- (5) Four bytes' reduction to a single bit
- (6) Single-bit processing with the output of step 5
- (7) The output of step 7 will XNOR with key obtained from step 1
- (8) Resulting key
- (9) Exit

ALGORITHM 2: Key transformation.

$$P_i(D \oplus K_0) \cdot P'_i. \quad (10)$$

Step 5: this key addition will actually cancel out the effect of key whitening on the sender side for equation (10):

$$\left. \begin{array}{l} D \oplus K_0 \oplus K_0 \\ D(\text{data left}) \end{array} \right\}. \quad (11)$$

Step 6: contraction of 64-bit data to 56-bit data:

$$\left. \begin{array}{l} D^\circ = \text{contraction}(D) \\ D^\circ(\text{resulting plaintext}) \end{array} \right\}. \quad (12)$$

Equation (12) shows the resulting plaintext.

4.3.9. The Proposed System Hierarchy. Algorithm 1 shows data encryption scheme, Algorithm 2 shows key transformation, and Algorithm 3 shows the key encryption scheme.

- (1) From the 4 chunks of the key, we will have 4 bits
- (2) Convert them to decimal and get a remainder with 26
- (3) The remainder will be used for forward and backward intervals
- (4) Begin
- (5) Convert each nibble to decimal
- (6) Match that nibble in the alphabetic table
- (7) The matched number will be forwarded to the times the number was obtained as the remainder
- (8) Resulting ciphertext
- (9) Resulting key
- (10) Exit

ALGORITHM 3: Key encryption.

5. Performance Analysis

The proposed encryption protocol is implemented based upon the following hardware and software specifications. This section introduces the outcomes we obtained for the performance analysis of our proposed scheme. We performed MES modular CPU time consumption analysis as well as MES and AES encryption time-based comparative analysis. Experimental setup description is in Table 4.

5.1. CPU Consumption. Table 5 shows modular-based CPU consumption in seconds. The CPU utilization is the calculation of the time that a CPU consumed for a specific computing process. It mirrors the CPU load. The more the CPU is utilized in the encryption procedure, the greater the CPU load will be. The execution time of the major modules of the proposed scheme is provided here. We calculated the CPU cycle time for each round by taking different data sizes as input. Key transformation consumes relatively more CPU cycles than other modules. The obtained time in the execution of the proposed scheme shows its applicability.

Table 6 shows the simulation results of the performance analysis of MES on distinct processor types. This comparison is based on distinct data sizes. We performed this experiment on multiple Intel generations and calculated their CPU consumption time. The experiments are carried out to check the efficiency at distinct platforms for time calculation. Elapsed time calculation of MES and AES encryption is carried out. Figure 20 shows the performance analysis of AES with MES based on CPU cycle consumption.

Encryption time is the time taken by the encryption scheme for the conversion of plaintext to ciphertext. Encryption time for any scheme assists in calculating the throughput. It specifies the encryption speed. Less encryption time denotes more throughput; the higher the throughput, the lower will be the power consumption of the enciphering scheme. We took different input sizes and got the results below. The elapsed time for AES was relatively more than MES elapsed time. The following graph shows the superiority of MES over AES. The following graph Figure 20 shows the results based on seconds.

5.2. Memory Consumption. One of the most significant parameters for performance analysis is memory consumption.

TABLE 4: System specification.

	Hardware	Specification
1	Operating system platform	Windows 10
2	Processor	Intel (R) Core (TM) i7-7500U CPU @ 2.70 GHz 2.90 GHz
3	Development kit	Visual C++ (visual studio community 2017)
4	System type	64-bit operating system, X-64 based processor

Figure 21 shows the memory consumption graph of MES. The above experiment was done with the help of the visual studio analysis tab. The diagnostic session of MES was 10.804 seconds with memory consumption in kilobytes (kbs).

Figure 22 shows the memory consumption graph of AES. The diagnostic session of AES was 15.266 seconds, and the memory consumption was up to megabytes (Mbs).

5.3. Key Variances. Figure 23 shows the key variances, range of keys, or options of key utilization based on the user preferences towards attaining security level. MES provides five distinct types of keys against five different classifications of nuclear information. AES provides three types of keys, and DES and IDEA provide a single type of key. Therefore, Figure 23 shows the highest degree of key variances of MES.

5.4. Single-Round Key Subsuming. Every single key transforms the data two times (except key whitening). Figure 24 shows the comparative analysis of MES, DES, AES, and IDEA from a single-round key subsuming perspective, where MES transforms the data twice in a single round as compared to AES, DES, and IDEA.

6. Application of the Proposed Scheme

Few regions inside nuclear systems that could possibly be susceptible to cyberattacks:

- (i) Information among monitoring stations
- (ii) Information from monitoring centers to missiles and missile stations

TABLE 5: Modular CPU consumption.

Modules			Timing (sec)	
1	Key transformation	0.0620088	0.00341157	0.00275324
2	Key whitening	0.000007052	0.000019746	0.000001058
3	9 rounds	0.000371306	0.000035614	0.000025036
4	Key encryption	0.000394226	0.000004584	0.000003527

TABLE 6: CPU-based performance analysis.

Processor type	MES
Intel (R) Core (TM) M-5Y10c CPU @ 0.80 GHz 1.00 GHz	0.0542868 (sec)
Intel (R) Core (TM) i3-4030U CPU @ 1.90 GHz 1.90 GHz	0.515043 (sec)
Intel (R) Core (TM) i7-7500U CPU @ 2.70 GHz 2.90 GHz	0.330332 (sec)

FIGURE 20: CPU consumption-based performance analysis.

FIGURE 21: MES memory consumption.

- (iii) Telemetry information from the projectile to space- and ground-based monitoring resources
- (iv) Information from space-based frameworks including navigational, positional, and timing

- information for the systems that perform world-wide navigation
- (v) Climate information from ground-, air-, and space-based sensors

FIGURE 22: AES memory consumption.

FIGURE 23: Quantitative measure—key variances.

FIGURE 24: Quantitative measure—single-round key subsuming.

- (vi) Information about positioning to deploy platforms (for example, submarines)
- (vii) Ground station's information
- (viii) Information among joined controlling stations [20]

Such nuclear information is highly vulnerable to cyberattacks. Therefore, we proposed an enciphering scheme against nuclear information confidentiality.

7. Conclusion

The duty of guaranteeing the presence and efficient functioning of the nuclear security regime of a state relies on the state's government. Guaranteeing the security of confidential information is a fundamental constituent of the state's nuclear security regime that it ought to uphold. Cloud computing is prestigious for providing information technology administrations. These days, organizations and communities are keen on moving their enormous computations and data into the cloud. Since it requires to be protected all over its lifetime, data confidentiality in the cloud is a noteworthy issue to be tackled on, in light of the fact that the confidential nuclear information is in outsider's control. The threat against nuclear information confidentiality can be from insiders, i.e., CSP, or from outsiders, i.e., intruder. Applying security at different levels makes your system more secure than at a single level; with this intention, we proposed a protection-based modular encryption scheme (i.e., MES). Performance analysis of the proposed scheme shows its favorable results compared to the other commonly used schemes.

Data Availability

The proposed scheme data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] Y. Amano, *Security of Nuclear Information*, IAEA Nuclear Security Series No. 23-G, Vienna, Austria, 2015.
- [2] J. T. Harris, "Information security of nuclear systems," in *Advances in Systems Science. Advances in Intelligent Systems and Computing*, vol. 240, pp. 17–23, Springer International Publishing, Cham, Switzerland, 2014.
- [3] C. Gurkok, "Chapter 63—securing cloud computing systems," in *Computer and Information Security Handbook*, pp. 897–922, Terremark Worldwide, Inc., Miami, FL, USA, 2017.
- [4] M. A. Mizani, "Chapter 17—cloud-based computing," in *Key Advances in Clinical Informatics*, pp. 239–255, Middle East Technical University, Ankara, Turkey, 2017.
- [5] M. Rady, T. Abdelkader, and R. Ismail, "Integrity and confidentiality in cloud outsourced data," *Ain Shams Engineering Journal*, vol. 10, no. 2, pp. 275–285, 2019.
- [6] W. Kong, Y. Lei, and J. Ma, "Data security and privacy information challenges in cloud computing," in *Proceedings of the International Conference on Intelligent Networking and Collaborative Systems*, Ostrava, Czech Republic, September 2016.
- [7] D. Coppersmith, "The data encryption standard (DES) and its strength against attacks," *IBM Journal of Research and Development*, vol. 38, no. 3, pp. 243–250, 1994.
- [8] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice-Hall, New Jersey, NY, USA, 1999.
- [9] J. V. Shanta, "Evaluating the performance of symmetric key algorithms: AES (advanced encryption standard) and DES (data encryption standard)," *IJCEM International Journal of Computational Engineering & Management*, vol. 15, no. 4, pp. 43–49, 2012.
- [10] R. L. Rivest, "The RC5 encryption algorithm," in *Proceedings of the Second International Workshop on Fast Software Encryption (FSE)*, Leuven, Belgium, December 1994.
- [11] H. Kai and L. Deyi, *Trusted Cloud Computing with Secure Resources and Data Coloring*, IEEE, Piscataway, NJ, USA, 2010.
- [12] A. Bentajer, M. Hedabou, K. Abouelmehdi, and S. Elfezazi, "CS-IBE: a data confidentiality system in public cloud storage system," *Procedia Computer Science*, vol. 141, pp. 559–564, 2018.
- [13] S. D. Dorairaj and T. Kaliannan, "An adaptive multilevel security framework for the data stored in cloud environment," *The Scientific World Journal*, vol. 2015, Article ID 601017, 11 pages, 2015.
- [14] S. A. Hussain, M. Fatima, A. Saeed, I. Raza, and R. S. Khurram, "Multilevel classification of security concerns in cloud computing," *Applied Computing and Informatics*, vol. 13, no. 1, pp. 57–65, 2017.
- [15] U. M. Ismail, S. Islam, M. Ouedraogo, and E. Weippl, "A framework for security transparency in cloud computing," *Future Internet, MDPI*, vol. 8, no. 1, pp. 1–22, 2016.
- [16] R. R. Corpuz, B. D. Gerardo, and R. P. Medina, "Using a modified approach of blowfish algorithm for data security in cloud computing," in *Proceedings of the 6th International Conference on Information Technology: IoT and Smart City—ICIT 2018*, Hong Kong, December 2018.
- [17] I. Mohiuddin, A. Almogren, M. A. Qurishi, M. M. Hassan, I. A. Rassan, and G. Fortino, "Secure distributed adaptive bin packing algorithm for cloud storage," *Future Generation Computer Systems*, vol. 90, pp. 307–316, 2019.
- [18] Q. Zhang, X. Wang, J. Yuan et al., "A hierarchical group key agreement protocol using orientable attributes for cloud computing," *Information Sciences*, vol. 480, pp. 55–69, 2019.
- [19] J. Tang, S. Fu, and M. Xu, "An effective encrypted scheme over outsourcing data for query on cloud platform," *IEEE Access*, vol. 7, pp. 66242–66250, 2019.
- [20] B. Unal and P. Lewis, *Cybersecurity of Nuclear Weapons Systems Threats, Vulnerabilities and Consequences*, The Royal Institute of International Affairs, London, UK, 2018.
- [21] M. A. Elakrat and J. C. Jung, "Development of field programmable gate array-based encryption module to mitigate man-in-the-middle attack for nuclear power plant data communication network," *Nuclear Engineering and Technology*, vol. 50, no. 5, pp. 780–787, 2018.
- [22] S. A. El Rahman, "A comparative analysis of image steganography based on DCT algorithm and steganography tool to hide nuclear reactors confidential information," *Computers & Electrical Engineering*, vol. 70, pp. 380–399, 2018.