cryptography

MDPI

Article

A New Hybrid Automated Security Framework to Cloud Storage System

Noha E. El-Attar [1,*], Doaa S. El-Morshedy [2] and Wael A. Awad [3]

[1] Faculty of Computers and Artificial Intelligence, Benha University, Benha 13518, Egypt
[2] Faculty of Science, Port Said University, Port Said 41523, Egypt; doaa_morshady@yahoo.com
[3] Faculty of Computers and Artificial Intelligence, Damietta University, Damietta 34711, Egypt; wael_abdelkader@du.edu.eg
* Correspondence: noha.ezzat@fci.bu.edu.eg

Abstract: The need for cloud storage grows day after day due to its reliable and scalable nature. The storage and maintenance of user data at a remote location are severe issues due to the difficulty of ensuring data privacy and confidentiality. Some security issues within current cloud systems are managed by a cloud third party (CTP), who may turn into an untrustworthy insider part. This paper presents an automated Encryption/Decryption System for Cloud Data Storage (AEDS) based on hybrid cryptography algorithms to improve data security and ensure confidentiality without interference from CTP. Three encryption approaches are implemented to achieve high performance and efficiency: Automated Sequential Cryptography (ASC), Automated Random Cryptography (ARC), and Improved Automated Random Cryptography (IARC) for data blocks. In the IARC approach, we have presented a novel encryption strategy by converting the static S-box in the AES algorithm to a dynamic S-box. Furthermore, the algorithms RSA and Twofish are used to encrypt the generated keys to enhance privacy issues. We have evaluated our approaches with other existing symmetrical key algorithms such as DES, 3DES, and RC2. Although the two proposed ARC and ASC approaches are more complicated, they take less time than DES, DES3, and RC2 in processing the data and obtaining better performance in data throughput and confidentiality. ARC outperformed all of the other algorithms in the comparison. The ARC's encrypting process has saved time compared with other algorithms, where its encryption time has been recorded as 22.58 s for a 500 MB file size, while the DES, 3DES, and RC2 have completed the encryption process in 44.43, 135.65, and 66.91 s, respectively, for the same file size. Nevertheless, when the file sizes increased to 2.2 GB, the ASC proved its efficiency in completing the encryption process in less time.

Keywords: advanced encryption standard; cloud computing; cryptography; data privacy; improved data encryption standard

Citation: El-Attar, N.E.; El-Morshedy, D.S.; Awad, W.A. A New Hybrid Automated Security Framework to Cloud Storage System. *Cryptography* **2021**, *5*, 37. https://doi.org/10.3390/cryptography5040037

Academic Editors: Jim Plusquellic and Josef Pieprzyk

Received: 25 November 2021
Accepted: 17 December 2021
Published: 20 December 2021

1. Introduction

The benefits of cloud computing technologies such as rapid scalability, cost reduction, and high reliability make it an attractive and widespread environment for various users [1]. Cloud computing is a flexible and elastic environment for various computing services such as servers, storage, networks, development platforms, and applications, which can be delivered on-demand with payment based on usage [2].

In general, cloud computing offers three main categories of service models; Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS provides the infrastructure as virtual computing machines (e.g., CPU, memory, storage, and network) accessible by cloud users [3]. PaaS is another kind of cloud computing service that offers a platform for users to create, run, and manage applications without going into detail about their infrastructure construction [4]. Finally, SaaS is one of the business services models cloud computing provides and allows users only to use the provided applications [5]. One of the significant shortcomings in a cloud environment is the lack

of complete control over data when saved in a cloud data center, which makes the data sources more vulnerable to different types of threats, to name a few, unauthorized access, unauthorized modification, and loss of data integration [4,6]. Therefore, storing data in the cloud environment raises different security issues, such as confidentiality, availability, and integrity [7]. Data availability is the process of making data accessible and usable on demand for authorized users [8]. Data integrity is the process of ensuring data reliability and accuracy, where the data should not be deleted or modified by unauthorized users [9]. Data confidentiality is another critical kind of challenge that may confront cloud environments, where the cloud service provider (CSP) must ensure authorized access to sensitive and private data stored in the cloud.

The obvious risk appeared when the CSP resorted to using a cloud third party (CTP). The CSP usually gives CTP several significant roles in dealing with data. To name a few, it is responsible for data analysis, data validation, data integrity, data auditing, managing and monitoring storage media, data outsourcing to third-parties storage, receiving responses from CSP, and sending reports to users [10–12].

Despite these delicate functions of CTP, it may morph into a vulnerability in the cloud environment. In more clarity, the third party is usually a human element, who could occasionally be a non-neutral or untrusted entity that poses a malicious insider threat to the cloud community [13]. As known, data confidentiality can be threatened in several ways, such as vulnerability and patch management, threat-aware identity and access management, and encryption and decryption methods [14]. Therefore, a large segment of cloud providers has tended to protect the user's data confidentiality away from the CTP by encrypting the data before hosting it on the cloud resources [6,15]. Thus, the main issues that will be handled in this paper can be summarized as follows:

- The problem of CTP as it may convert into a vulnerable point inside the cloud environment.
- The problem of simple keys in the encryption process, which can be readily broken.
- The problem of easier data breaching when using one encryption scheme.

This paper will handle the above data confidentiality issues in cloud computing systems by developing an automated data encryption system based on a novel modified AES algorithm to encrypt the user's data before exporting it to the cloud storage media. The main contributions of this proposed module are as follows:

- Curbing the CTP functions and enhancing data privacy.
- Dividing the original data into blocks with random sizes.
- Generating random encryption keys equal to the number of data blocks.
- Encrypting the random encryption keys by both RSA and Twofish algorithms.
- Saving the keys encrypted by the RSA algorithm in a private Mongo DB to be used in the decryption process.
- Using the keys encrypted by the Twofish algorithm as keys to encrypt the original data.
- Encrypting the blocks of data by two novel approaches using AES and DES.
- Improved AES by using a dynamic S-box for each block in the ARC approach.
- Comparing the proposed approaches with other state-of-the-art encryption algorithms, such as DES, 3DES, and RC2.

The rest of this paper is organized as follows; Section 2 discusses the related works that handle the security concerns of cloud computing. In Section 3, the preliminaries for the typical cryptographic algorithms used in our model are presented. The proposed automated encryption system for data storage and its architecture are discussed in Section 4. Section 5 presents the experiments and the comparison of statistical results. Finally, the paper ends with conclusions and the future work in Section 6.

2. Literature Reviews

The direction of eliminating the role of the CTP or reducing the potential risks of its existence has become the core of several recent studies that concern cloud security issues. In [16], S. Chakraborty et al., (2018) proposed an approach to ensure the integrity of

data stored on a remote cloud server with the assistance of a trusted third-party auditor (TTPA). TTPA has been proposed as an expert agent in the data auditing process. TTPA's role is to retrieve a file tag, check its signature, and quit if it fails. In the same context, G. F. Nadlamani and S. Shaikh (2016) [17] have presented a technique for encoding files using the (AES and Blowfish) algorithms, also based on the third-party auditor TPA. In this technique, the cloud provider (CP) stores a file in a database, and the user sends a report to the TPA to test the validity of a particular file. TPA forwards this demand to CP to produce a new hash for this specific file and transfer it to TPA. TPA compares each hash. If both are equivalent to the user integrity test result, TPA sends the crash report to CP. In [18], S. More and S. Chaudhari (2016) have suggested a new mechanism to protect data integrity and confidentiality. This approach also utilizes user-based data encryption and cloud third party (CTP) using a different technique. At first, the data owner must allow some transactions on his data, including splitting the file into blocks, encrypting it, creating a hash value for each block, concatenating it, producing a signature on it, and finally uploading it to the cloud. Posteriorly, the responsibility of CTP appears in public data auditing. The CTP tests the validity of the data by generating the hash value for the encrypted blocks obtained by the provider and for each signature concatenated and produced in the cloud environment. The third party eventually matches the two signatures to test whether or not the encrypted data has been changed. This strategy has struggled with some concerns about the CTP, and it also raises the consumer workload.

Another strategy for the stated security problems has been proposed by A. P. Singh and S. K. Pasupuleti (2016) [19]. They have developed their approach based on two sequential ways of data auditing. Firstly, during the dynamic system update process, the client demands the data blocks from the cloud storage server (CSS). The CSS checks the accuracy of the data for this block and whether the previous update was successful or not. After that, the third party starts the public auditing process, called the "Third Party Audit (TPA) process". The TPA process is based on checking the probabilistic proof of integrity provided by the CSS. The main security issue is identifying the TPA as an employee, which may become a malicious insider even if it is a weak possibility. F. O. Catak and A. F. Mustacoglu (2018) [20] have provided a private classification protocol using a privacy-preserving protocol method for the extreme learning machine algorithm. The suggested methods would use a distributed multi-party calculation (or cloud computing) technique to encrypt the hidden layer output matrix H. This study illustrates how to create a classification model with encrypted data that protects confidentiality. This strategy depends on the user for encrypting their data input, generating arbitrary and vertically divided data, and then uploading encrypted data to a cloud system. Another technique for the third-party auditing process has been presented by R. Saxena and S. Dey (2016) [21]. They have developed a strategy to check data validity and integrate auditing services to execute load balancing functions and implement batch auditing by several third-party auditors. Third-party auditing typically manages the original data during the process of public auditing. Furthermore, B. Adokshaja and S. Saritha (2017) [22] have proposed a system that is being developed to verify the accuracy of cloud data with the help of a third-party auditor (TPA). The system can audit the data on a periodic or on-demand basis without retrieving all the data or imposing additional online burdens on cloud users and servers. The proposed system ensures that no data content is leaked to TPA during the auditing process. The data blocks are encrypted by the data owner using the Advanced Encryption Standard (AES) algorithm. Another direction to limit the role of the third party was presented by K. M. Akhil et al., (2018) in [23]. They have developed a system that provides secure data without interference from a third party. Data are sent to the cloud server to be stored; it will be encrypted using the AES encryption technique. In this technique, the third-party auditor is not aware of encryption and decryption inside the system. The Advanced Encryption Standard (AES) is used for the encryption of data stored on servers. Furthermore, P. Sivakumar et al., (2019) [24] have applied the AES algorithm for data protection in the Heroku cloud. Releases from using AES require well-built security from

third parties where the key used in the encryption process is the same as the decryption process. This issue may make it easy to break into the original data and retrieve it. Another strategy to present a security system that does not depend on a third party is proposed by A. Orobosade et al., (2020) [25]. They have proposed a hybrid encryption method that includes symmetrical and asymmetrically designed cryptographic systems to preserve the privacy and safety of users in the cloud. They have proposed a privacy model based on Advanced Encryption Standard (AES) and Elliptic Curve Cryptography (ECC). In [26], M. Sohal and S. Sharma (2018) have presented a new symmetric-key multifold cryptography approach based on DNA cryptography, which leverages client-side data encryption to encrypt data before uploading it to the cloud. Another novel encryption algorithm for improving data security on the cloud has been presented by F. Thabit et al., (2021) in [27]. They have proposed a new Lightweight Cryptographic Algorithm based on 16 bytes of a block cipher and 16 bytes for the encryption key. This proposed algorithm is dependent on Feistel and substitution–permutation methods to increase the encryption complexity.

From the above, we can conclude the shortcomings in the recent approaches used to save data confidentiality in cloud environments as follows:

1- Using a single algorithm (non-hybrid methodologies) is insufficient to provide high levels of security because, in symmetric algorithms, only one encryption key is used to encrypt and decrypt data [28].
2- The non-automated systems that use cloud third parties must ensure that this CTP is trustworthy and not turned into a malicious insider. This issue is challenging to be guaranteed in actual practices.

The authors will address the data confidentiality issue from another perspective that differs from the above studies in this paper. The proposed automatic cryptographic system aims to curtail the role of the CTP by developing an Automated Encryption/Decryption System for Cloud Data Storage (AEDS), which adopts a fully automated strategy for data encryption and decryption processes. The AEDS is based on transferring all the security operations on the data to an autonomous system, beginning with uploading the data by the user on the cloud platform, passing through encrypting it to be ready for storage, and ending with decrypting it when its owner asks to retrieve it. AEDS is a hybrid cryptography framework that is implemented based on four encryption algorithms; Twofish [29], Advance Encryption Standard (AES) [30], and Data Encryption Standard (DES) [31] as symmetric encryption algorithms [26], and Rivest–Shamir–Adleman (RSA) as an asymmetric encryption algorithm [32].

Table 1 summarizes the characteristics of the works mentioned above and our proposed framework.

Table 1. Comparison between the related mentioned works.

Ref.	Algorithms Used	Hybrid Approach	Automated Approach	CTP	Calculate Processing Time	Lightweight	Split the Data		Calculate Throughput
							Equal Size	Random Size	
[16]	AES			√			√		
[17]	AES			√			√		
[18]	AES, RSA digital signature and SHA-2	√		√		√			
[19]	Chameleon Authentication Tree and Homomorphic Linear Authenticators	√		√			√		
[20]	Paillier Homomorphic Encryption			√			√		
[21]	Paillier Homomorphic Cryptography	√		√			√		
[22]	AES			√			√		
[23]	AES			√					
[24]	AES						√		
[25]	AES and ECC	√			√				
[26]	DNA		√		√	√			√
[27]	NLCA					√	√		
The current work	RSA, AES, DES, and Twofish	√	√		√	√		√	√

3. Materials and Methods

3.1. Twofish Algorithm

Twofish is a symmetric algorithm created by Bruce Schneier et al. [33]. It uses a 256-bit key length to encrypt a block size up to 128-bit [33]. The primary characteristic of the Twofish method is its complicated encryption scheme, which includes pre-calculated S-blocks that are also key-dependent. The n-bit keys are divided into two portions; half are used as encryption keys, while the other half is utilized to change the algorithm. The Twofish algorithm has a total of 16 rounds for any key size [34].

The design aspects of the Twofish algorithm are as follows, the input and output data are XORed using eight sub-keys K_0 to K_7. Input and output whitening are the terms for these XOR procedures. The F-function consists of five types of component operations: fixed left rotation by 8 bits, key-dependent S-boxes, Maximum Distance Separable (MDS) matrices, Pseudo-Hadamard Transform (PHT), and two sub-key additions modulo 2^{32} [35].

3.2. Rivest–Shamir–Adleman (RSA) Algorithm

The RSA encryption algorithm is an asymmetric cryptography technique developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977 [36]. As with any asymmetric technique, RSA uses two key pairs for encrypting and decrypting data. The data are encrypted with a public key shared with other users, while the private key is only used to decrypt the data and must be secret [37]. The encryption and decryption processes in the RSA algorithm could be calculated as follows:

1. Key generation: Select two separate prime numbers, N and M.
2. Calculate the value of L, which is the product of N and M.
3. Calculate the function of Euler's totient φ by $\varphi(L) = (N - 1) \times (M - 1)$
4. Find the value of *e*, under this condition: $1 < e < \varphi(L)$ and $(e,\varphi(L)) = 1$
5. Calculate p, where $p = e^{-1} \pmod{\varphi(L)}$
6. Use this equation for encryption: $C = K^e \pmod{L}$
7. Use this equation for decryption: $K = C^p \pmod{L}$

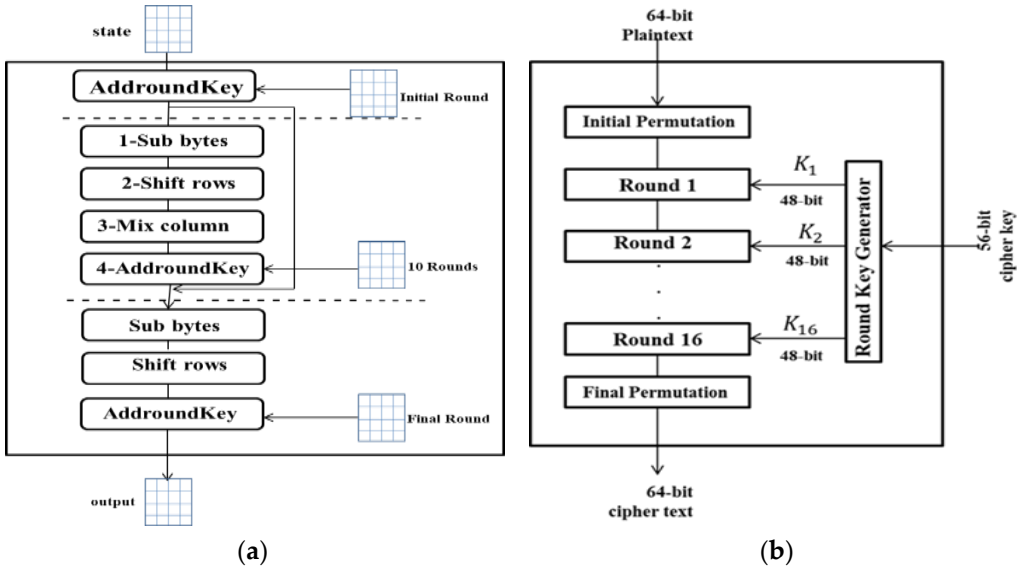
Where C is the cipher data, K is the original data, (e, L) is the public key, and (p, L) is the private key [38,39].

3.3. Advance Encryption Standard (AES) Algorithm

Joan Daemen and Vincent Rijmen created the AES algorithm as one of the symmetric encryption techniques [40]. AES is generally based on a substitution–permutation network. It consists of a sequence of connected processes, some of which require substituting specified outputs for inputs (substitutions), and others involve shifting bits around (permutations). In the AES algorithm, all calculations use bytes instead of bits. This means the 128 bits of plain text are treated as 16 bytes, organized into four rows and four columns to be processed as a matrix. In AES, the number of rounds is adjustable depending on the length of the key. It utilizes ten rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys [41]. The encryption process of AES contains the following operations:

1. Byte substitution: These operations are based on a substitution box (S-box) created, particularly for this purpose. This operation results in a four-by-four matrix for data.
2. Shift Rows: All of the four rows of the matrix are shifted to the left, and any items dropped are re-inserted to the right side of the row.
3. Mix Columns: A particular mathematical function is now used to alter each column of four bytes. This function takes four bytes from one column as input and returns four new bytes that replace the original column. As a result, a new matrix with 16 additional bytes is created. This stage is skipped in the final round.
4. Addround Key: The 16 bytes of the four-by-four matrix are now treated as 128 bits, then they are XORed with the round key's 128 bits. If this round is the last one, the output will be the ciphertext. Otherwise, the 128 bits are transformed into 16 bytes,

and another round will begin [26,42]. Figure 1a shows the encryption process of the AES algorithm [43].



(**a**) (**b**)

Figure 1. The encryption process of AES algorithm (**a**) and DES (**b**).

3.4. Data Encryption Standard (DES) Algorithm

The DES is a symmetric key block cipher technique created by IBM [39]. This technique employs a block size of 64 bits and a key size of 56 bits and uses 16 rounds of the Feistel structure, with a different key in every round [26]. The main phases of DES can be discussed as follows;

1. The 56-bit key is split into two parts, with 28 bits on the left and 28 on the right. These two parts are merged and compressed. Each half of the key is moved by one or two bits depending on the round. In this round, the plain text block is encrypted with a 48-bit compressed key.
2. The 64-bit data are split into two halves, each of which is 32 bits long. The size of the block is increased to 48 bits by expanding one-half of the block. Then the output is XOR'ed with the compression key of 48 bits, which is produced in step 1.
3. The output is passed to the S-box, which changes key bits and decreases the 48-bit block to 32-bit. The output of the S box is sent into the P-box, which permutes the bits. The two blocks are then switched and become the following round's input. The key halves that were shifted in step 1 are applied [26]. Figure 1b shows the encryption process in the DES algorithm [44].

This study has applied the AES, DES, RSA, and Twofish cryptographic algorithms due to their proven efficiency. For instance, AES is adopted as an efficient encryption algorithm for large volumes of data [23]. At the same time, the DES is known as a standard approach for protecting confidential data [45]. Moreover, both algorithms are characterized by their reliability, speed, efficiency, and flexibility in encrypting the data. On the other hand, although the RSA is slower than AES and DES algorithms in encrypting large amounts of data, it acts as a very effective complicated algorithm used to secure data [33]. In addition, the Twofish algorithm has high processing power and is considered one of the most highly secure techniques for data exchange [34]. Despite the benefits of the encryption algorithms mentioned above, utilizing a single encryption method to protect the confidentiality and privacy of data in public systems such as cloud environments makes it more vulnerable to hacking and privacy violations. As a result, we present novel approaches based on the hybridization of these conventional algorithms, as discussed in the following section.

4. The Proposed Automated Encryption/Decryption System for Cloud Data Storage (AEDS)

The proposed automated encryption and decryption system (AEDS) is a hybrid cryptography framework that integrates four encryption algorithms to enhance cryptographic power. It is based on two cornerstones: protecting user data and preserving encryption keys. Regarding user data, the AEDS combines a modified dynamic version of the AES algorithm (DAES) and DES algorithm in a novel manner to encrypt the original data. The DES is adapted in our proposed model because it is one of the standard techniques to protect confidential data. At the same time, AES is an effective methodology in protecting large volumes of data [23,45]. In our proposed model, we present a novel S-dynamic box based on a polynomial function instead of the standard static box used in the original AES to enhance the security level of AES.

On the other hand, both RSA and Twofish algorithms are combined as a protective approach for the generated encryption keys [33,34].

To enhance performance, accuracy, and security, the AEDS is implemented in three versions for encrypting blocks of data; the Automated Sequential Cryptography (ASC), the Automated Random Cryptography (ARC), and the Improved Automated Random Cryptography based on dynamic AES (IARC) versions.

As shown in Figure 2, the proposed AEDS begins with a Random Key Generator (RKG), responsible for creating a random key K_i for each block of data B_i, based on the RKG function shown in Figure 3. To increase the security of these keys, two separate encryption processes are performed on them as follows;

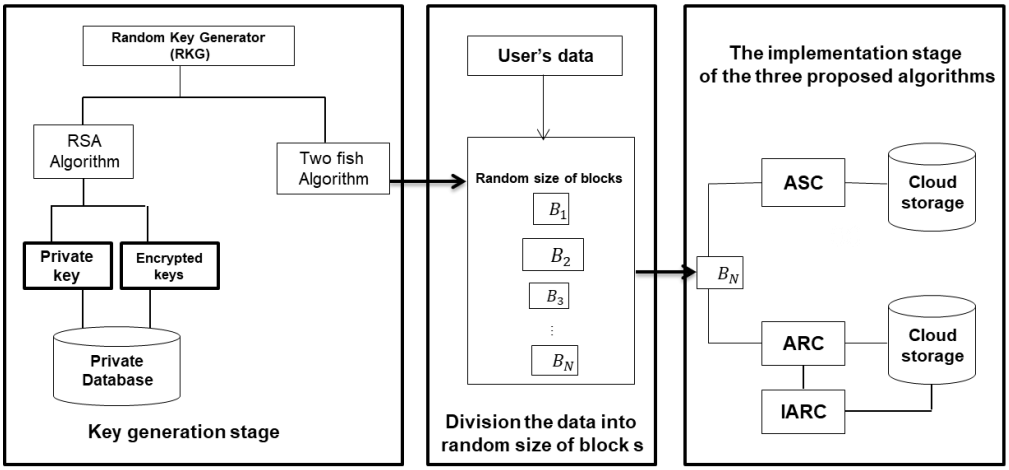


Figure 2. The proposed Automated Encryption/Decryption System for Cloud Data Storage (AEDS).

```
Where n=16
Def random _with_N_digits (self, n):
Range_start=10** (n-1)
Range_end = (10**n)-1
Random number = (range_start, range__end)
```

Figure 3. Random Key Generator (RKG) function.

(1) The RSA algorithm is used to encrypt these generated random keys K_i producing ciphered keys $K_{i,\,RSA}$, and private key (p, L), which will be used latterly to decrypt $K_{i,\,RSA}$. Both the private key and ciphered keys $K_{i,\,RSA}$ are stored in a private and secure database.

(2) The Twofish algorithm is also used in encrypting these generated random keys. Still, here, the produced ciphered keys $K_{i,\,Twofish}$ are used as encryption keys to encrypt the blocks of original data using the DES and AES algorithms consecutively using

the ASC version or randomly in ARC or DARC. The main idea is that the system automatically disposes of the public key used in encrypting the random keys to keep these keys safe and away from unauthorized access. Finally, all the encrypted blocks are sent to the cloud storage, and they are now ready to be stored. Figure 4 displays a brief scenario for the data storage process based on the proposed ADES module. It is worth mentioning that the original data are divided into the random size of blocks by using Equation (1).

$$\mathbf{B_{lenght} = 64\ MB \times S.} \quad (1)$$

where S is a random number [2:4], this interval is chosen for the following reasons:

- If S < 2, the stored data will be much smaller than the usual block size, making the performance suffer drastically.
- If S > 4, the size of data blocks will be substantially larger than the standard block size, and therefore, it will take a long time to upload and process data.

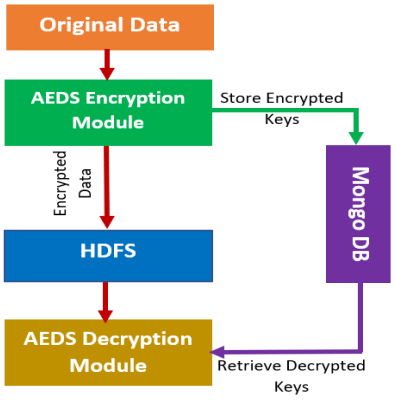


Figure 4. The prototype of the overall storing system in AEDS.

4.1. Automated Sequential Cryptography (ASC) for Data Blocks

ASC is the sequential version for ciphering the original data blocks in our proposed AEDS framework based on an automatic encryption operator (EO). Initially, the EO receives the encrypted random keys $K_{i,\ twofish}$ produced by the Twofish algorithm, and the blocks of data B_i, which are divided in the first stage, and begins the encryption process by switching between the two algorithms DES and AES sequentially (i.e., the first block B_1 is encrypted using the AES algorithm with an encrypted key $K_{1,\ twofish}$, the second block B_2 is encrypted by DES by $K_{2,\ twofish}$, etc., until the number of generated keys ends with the end of the number of data blocks), as shown in Figure 5a.

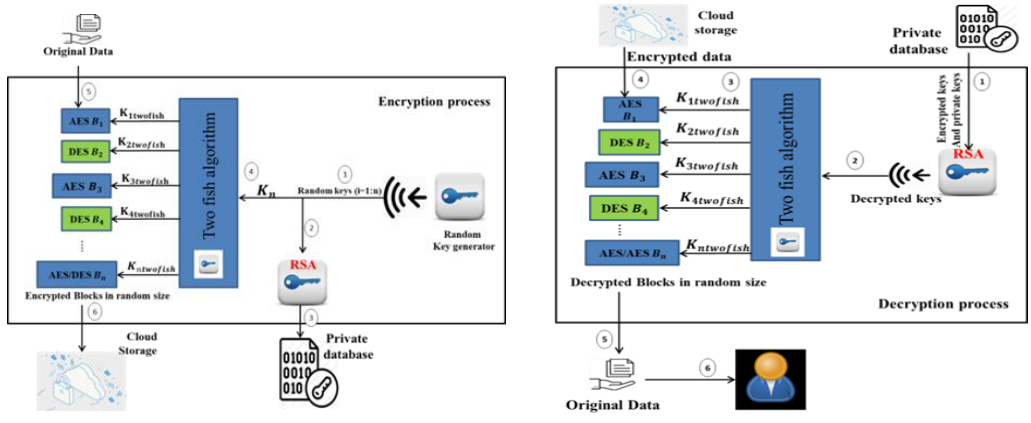


Figure 5. Methodology of the Automated Sequential Cryptography (ASC) for data blocks, encryption process (**a**) and decryption process (**b**).

The decryption process in the ASC is also performed automatically as the encryption process. The decryption operator (DO) is activated only when the authorized user requests to retrieve his original data. The DO procedure passes through two main reverse phases. The first phase begins by applying RSA as a decryption algorithm to decrypt all the encrypted keys stored in the private database using the private decryption key (p, L), which was previously stored in a private database. After obtaining the original keys, they will be encrypted by the Twofish algorithm to obtain the public keys of AES and DES and complete the decryption process, as shown in Figure 5b.

4.2. Automated Random Cryptography (ARC) for Data Blocks

The automated random cryptography technique is another novel strategy used for data encryption in our proposed framework to enhance data security and privacy. The novelty of this encryption strategy is to encrypt the data blocks. The ARC encrypts the data blocks by switching the original AES and DES randomly depending on a random integer number t, as shown in Figure 6. In more detail, for each block of data, a random number t is generated. If divided by 2, the data block will be encrypted by AES, otherwise, DES will encrypt the block.

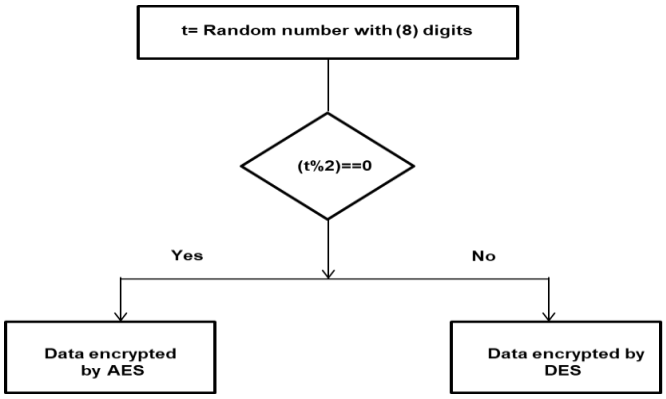


Figure 6. Flow chart of the randomize procedure in the ARC technique.

As shown in Figure 7, the encryption operator uses the $\mathbf{K_i}$, twofish produced by the Twofish algorithm as encryption keys to encrypt the blocks of data by AES or DES according to the value of t. Finally, these data blocks are collected as an encrypted file and sent to cloud storage.

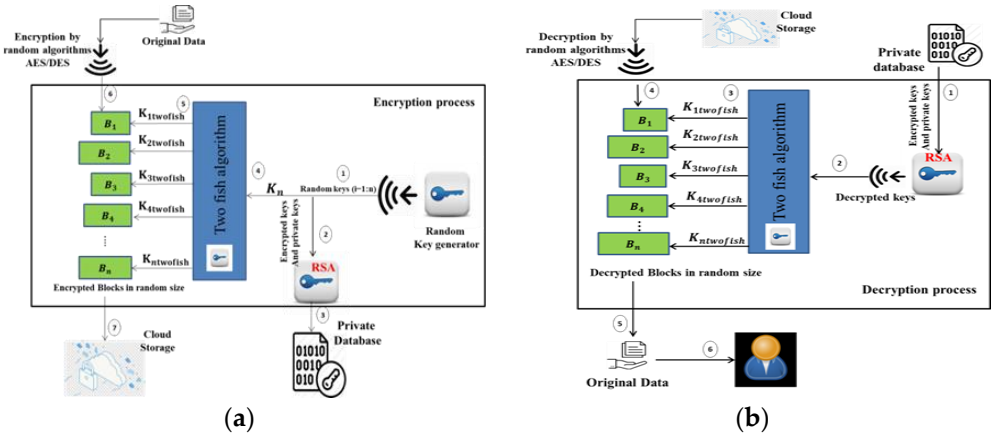


Figure 7. Methodology of the Automated Random Cryptography (ARC) for data blocks, encryption process (**a**) and decryption process (**b**).

Overall, the proposed Automated Encryption/Decryption System for Cloud Data Storage (AEDS), whether using sequential or random manners of encrypting blocks of data, is

primarily based on the automated execution of the encryption and decryption process without any interference from a third party and without any external pressure on the user's shoulders. The steps of the encryption and decryption processes are illustrated in Algorithms 1 and 2.

Algorithm 1: Encryption process.

Input: Plain text F, Random key K_i
Output: E_i *(encrypted text), private keys of RSA and* $K_{i,\ RSA}$ *(encrypted keys using RSA)*
Set $Y = F_{lenght}$
While (Y! = 0) do
Set F_{start} *= 0, i = 1 as number of current block,*
While (F_{start} *< Y) do*
Set t =0 ***/* in case of ASC encryption approach***
Generate t as a random number ***/* in case of ARC encryption approach***
Store t in the private database
Generate 16-byte random key according to RKG function
Use $K_{twofish}$ *to encrypt the 16-byte random key*
Encrypt the same 16-byte random key by K_{RSA}
Store K_{RSA} *and the private key in the private database*
Use the encrypted 16-byte random key to encrypt blocks of data
Determine random sizes of blocks n according to Equation (1)
For i = 1 to n
Calculate the $q_{end} = F_{start} + B_i$
While ($q_{end} > Y$*) do*
$q_{end} = Y$
End while
Read block data from file start at F_{start} *and end at* q_{end}
If (t mod2==0) then
Encrypt the block B_i *using AES algorithm by the* $K_{i,\ twofish}$*,*
Else
Encrypt the block B_i *using DES algorithm by the* $K_{i,twofish}$*,*
End if
t = t + 1 ***/* in case of ASC encryption approach***
Generate new t as a random number ***/* in case of ARC encryption approach***
Send the cipher block $\boldsymbol{E_i}$ *and send it to cloud storage*
End For
Set $F_{start} = q_{end}$
End while
End while
Store the encrypted blocks of data E_i *on the cloud storage, and store the private keys p and the encrypted keys by RSA* $K_{i,\ RSA}$ *on a private database.*

Algorithm 2: Decryption process.

Input: Encrypted blocks of data, $K_{i,RSA}$ *(encrypted keys) and the private keys by RSA*
Output: The original file
For i = 1 to n */* n is the number of ciphered blocks*
Select $K_{i,\ RSA}$ *and the private keys p from the private database*
Decrypt $K_{i,\ RSA}$ *using RSA decryption algorithm*
Read 16-byte of keys
Encrypt the 16-byte keys using the Twofish algorithm
Read the block E_i *from the cloud storage and read t from the private database*
If (t mod2==0) then
Decrypt the E_i *by the inverse of the AES algorithm by its* $K_{i,\ twofish}$
Else
Decrypt the ciphertext by the inverse DES algorithm by its $K_{i,\ twofish}$
Insert the decrypted block of data B_i *in a file F*
End if
End for

4.3. Improved Automated Random Cryptography (IARC) for Data Blocks

A novel modified version of AES based on a dynamic S-box generator is developed in the Improved Automated Random Cryptography technique. The original AES was based on using a static S-box, which is the component that is responsible for the substitution process in any symmetric algorithm [40]. The proposed novel Dynamic AES algorithm uses a dynamic substitution box (S-box) generated randomly for each data block without repetition to improve confidentiality and the security level.

The S-box in the proposed DAES algorithm is generated by using a polynomial function to compute each element in the new dynamic S-box for each data block, as shown in Equation (2):

$$S-box\ (t) = \left[\left(4t^4 + 3t^3 + 2t^2 + t + 1\right) \bmod N_r\ + (N_r \bmod 5)\right] \quad (2)$$

where *t* is an 8-digit random integer generated in the ARC technique's randomize procedure, as illustrated in Figure 7, and N_r is a counter that counts down until the dynamic S-box from [1:257] is completed. The changing value at each location in the dynamic S-box is determined by N_r. In many circumstances, the result of the phrase $\left[\left(4t^4 + 3t^3 + 2t^2 + t + 1\right) \bmod N_r\right.$ may be zero, thus the term $(N_r \bmod 5)$ is added to limit the result's number of zeros. To verify that each place in the dynamic S-box has a unique value and obtains a dynamic S-box with no repetition, we employed Algorithm 3.

Algorithm 3: S-box without repetition process.

Input: S-box (t) value
Output: S-box without repetition

```
Set i = 0
countval= S-box.count(S-box (t))
while(countval >= 1)do
i = i + 1
S-box (t) = S-box (t) + i
while (S-box (t) >= 256) do
S-box (t) = abs(S-box(t) − 256)
Endwhile
countval = S-box.count(S-box (t))
Endwhile
```

Finally, Table 2 displays the description of all the parameters utilized in the algorithms. All the symbols and abbreviations are mentioned in Appendix A.

Table 2. Description of all the parameters used in the following algorithms.

Parameter	Description
B_i	Blocks of data
K_i	Random key
F	Plain text file
F_{lenght}	The size of the file
F_{start}	the start of each block
q_{end}	the end of each block
$K_{i,\ twofish}$	Encrypted key using Twofish
$K_{i,\ RSA}$	Encrypted key using RSA
E_i	Encrypted block
i	Number of checks of repetition
Countval	Number of repetitions
S-box (t)	The value of each place

The complexity of the proposed algorithms has been calculated by Cyclomatic complexity, a standard software complexity metric for evaluating the complexity of a pro-

gram [46]. The parameters used in this software to calculate the algorithm complexity are the number of Line of Codes (LOC), Function Count (FC), count space token of functions (Token), and Cyclomatic Complexity Number (CCN). Table 3 displays the values of these parameters for the proposed cryptographic algorithms ASC, ARC, and IARC. The algorithm of the ARC technique used 33 functions in 826 lines of code, resulting in a complexity of 1.5. The ASC approach used 33 functions in 805 lines of code, resulting in a complexity of 1.4. Finally, the IARC utilized 33 functions with 822 lines, resulting in a complexity of 1.5.

Table 3. Comparison between the tested the complexity of the proposed approaches based on Cyclomatic complexity.

Algorithm	NLOC	Avg. NLOC	Avg. Token	Function Count	AvgCCN
ARC	826	22.9	185.2	33	1.5
ASC	805	22.4	180.9	33	1.4
IARC	822	22.8	183.1	33	1.5

5. Experiments and Results Analysis

The proposed encryption algorithms have been implemented and tested on a private cloud service provider using the open nebula as a cloud simulator, which provides an infrastructure for a cloud environment. The experiments have been performed using python as a programming language, running on VMware based on a virtual machine, Intel (R) Core i7 2.3 GHz CPU, 32 GB of memory, and Windows 10 operating system.

5.1. Configuration of Open Nebula Cloud Simulator

In the Open Nebula Hadoop simulator, the total configured capacity for our system is 80.47 GB, split into 57.86 GB for storage space and 22.6 GB for the operating system and other applications. We utilized 47.7 GB of the distributed file system (DFS) as the capacity used. Regarding the master server, it has a total configured capacity of 30.19 GB, of which 15.9 GB is utilized for DFS storage. Node 1 and node 2 have a total configured capacity of 25.19 and 25.09 GB, respectively. All the configurations are displayed in Table 4.

Table 4. Configuration of the Open Nebula.

Hostname	Master. Hadoop. Lan	Node1	Node2
Decommission Status	Normal	Normal	Normal
Configured capacity	30.19 GB	25.19 GB	25.09 GB
DFS Used	15.9 GB	15.9 GB	15.9 GB
Non DFS Used	8.48 GB	7.04 GB	7.08 GB
DFS Remaining	5.81 GB	2.25 GB	2.11 GB
DFS Remaining%	19.24%	8.93%	8.41%
DFS used%	52.67%	63.12%	63.37%
Cache used%	100%	100%	100%

5.2. Performance Evaluation Measures

The proposed encryption algorithms ASC, ARC, and IARC are compared to other popular encryption algorithms used in the cloud environment, including DES [44], 3DES [47], and RC2 [48]. The experiments were implemented and tested using different sizes of stored data; 500 MB, 730 MB, 1.2 GB, 1.7 GB, and 2.2 GB. Each file is divided into blocks of different sizes, ranging between 128, 192, and 265 MB. To evaluate the encryption algorithms, the statistical results for all the utilized algorithms used in the comparison are measured, including time of encryption process, time of decryption process, time of uploading and retrieving data, and throughput usage of both encryption and decryption files as follows:

- **Time consumption for encryption/decryption files (*T*):** is the total time the system takes to encrypt/decrypt all blocks of the selected file using the chosen algorithm from the start of the encryption/decryption process to the end [48,49]. It could be calculated by Equation (3).

$$T = \sum (End\_time - Start\_time)_{block} \quad (3)$$

- **Time consumption for upload/retrieving files:** is the time the system takes to send data from the user to the server and vice versa. Generally, the server's speed controls the upload/retrieve time and is calculated by Equation (4) [49].

$$Upload/Retrieve\ Time = Request\ time\ of\ the\ user + PT \quad (4)$$

- **Overall Processing time (*PT*):** is the system's total time for generating random keys and executing the encryption or decryption process. It could be calculated by Equation (5) [27].

$$PT = key\ generation\ time + T \quad (5)$$

- Throughput: is the amount of data that can be processed in a predefined time. It is calculated by Equation (6) [26,50]:

$$Throughput\ (MB/s) = \frac{\text{Size of the selected file (MB)}}{PT\ (\text{s})} \quad (6)$$

5.3. Discussion and Results Analysis

According to the comparison between the proposed cryptographic algorithms ASC and ARC and the other state-of-the-art cryptographic algorithms used in the evaluation, both ASC and ARC achieved better performance in encrypting and decrypting different files of various sizes. Furthermore, ASC and ARC removed the problem of the third party and achieved a high level of security; they performed encryption/decryption processes in better time than other algorithms. Tables 5 and 6 show the recorded time in encrypting and decrypting different data blocks sizes in several files of different types and sizes. For instance, as shown in Table 6, our proposed cryptographic algorithms ASC and ARC recorded the least time to encrypt and decrypt the various sizes of files. ARC was superior to the other algorithms used in the comparison. It consumed 22.90, 40.99, 83.79, and 100.24 s to complete the encrypting process for 500 MB, 730 MB, 1.2 GB, and 1.7 GB files, respectively.

In comparison, ASC recorded less time when the file sizes became 2.2 GB, consuming 127.37 s. Although ARC and ASC have complicated operations in the decryption process and have recorded more time than recorded in the encryption process, they outperformed the other algorithms in saving time. ARC recorded 101.27 and 158.45 s to decrypt the 500 and 730 MB files, respectively. Furthermore, ASC was better at decrypting the 1.2, 1.7, and 2.2 GB files.

Regarding time consumption in uploading and retrieving files displayed in Table 7, ARC and ASC recorded the best average time, except in a file size of 1.2 GB. DES recorded the least time for the file uploading process. While for the amount of data that can be processed in a second (i.e., the throughput), ARC and ASC were the best, followed by DES, as displayed in Table 8.

On the other hand, as IARC employed a dynamic S-box to enhance security, it definitely will enhance the security level of user data. However, as reported in the comparative tables, its continuous calculation process to generate the dynamic s-box value took longer to finish the encryption/decryption process.

Table 5. Comparison between the tested algorithms based on the overall processing time for encryption/decryption.

File Size	The Overall Time of Encryption Process/s						The Overall Time Decryption Process/s					
	DES	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**	**DES**	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**
500 MB	44.69	67.27	136.18	26.70	**22.90**	1092.41	115.22	139.43	243.04	103.26	**101.27**	1162.13
730 MB	63.24	102.81	182.84	46.72	**40.99**	3782.78	175.37	210.81	303.29	161.20	**158.45**	3972.41
1.2 GB	101.68	152.53	279.97	86.91	**83.79**	6157.41	285.36	325.95	469.01	**264.63**	267.36	6484.52
1.7 GB	152.50	236.39	406.76	104.05	**100.24**	1029.16	406.74	487.70	668.14	**379.95**	425.32	11,320.09
2.2 GB	194.00	302.07	509.57	**127.37**	162.76	12,702.62	521.86	620.32	831.47	**465.63**	486.22	14,195.37

Table 6. Comparison between the tested algorithms based on the consumed time for encrypting/decrypting a file in seconds.

File Size	Time for Encrypting a File/s						Time for Decryption a File/s					
	DES	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**	**DES**	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**
500 MB	44.43	66.91	135.65	26.41	**22.58**	1092.10	42.84	63.55	139.93	25.86	**21.27**	1082.47
730 MB	63.11	102.62	182.33	46.29	**40.47**	3783.11	61.26	98.39	178.10	45.13	**39.55**	3637.41
1.2 GB	101.30	151.84	279.41	86.48	**83.17**	6157.113	103.17	151.88	284.64	**74.83**	82.73	6057.11
1.7 GB	151.68	235.78	406.10	103.20	**99.70**	10,293.96	148.07	229.56	397.59	102.45	**94.40**	10,211.38
2.2 GB	193.30	301.02	508.65	**126.81**	161.43	12,702.17	185.38	293.27	504.15	**120.35**	156.60	11,801.31

Table 7. Comparison between the tested algorithms based on the consumed time for uploading/retrieving a file in seconds.

File Size	Time for Uploading a File/s						Time for Retrieving a File/s					
	DES	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**	**DES**	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**
500 MB	220.09	247.90	359.62	**137.14**	208.97	1285.33	115.56	139.79	243.78	103.65	**101.41**	1163.57
730 MB	324.31	364.12	463.90	**316.50**	326.18	4053.67	175.90	211.30	303.84	161.67	**159.07**	3297.32
1.2 GB	**528.33**	575.33	718.65	566.50	534.48	6601.83	286.38	327.29	469.96	**265.65**	266.33	5229.31
1.7 GB	768.27	850.46	1026.68	**745.61**	747.32	9318.29	408.43	489.39	669.88	**387.36**	427.82	8030.57
2.2 GB	977.90	1066.20	1291.87	**937.36**	966.01	13,195.55	524.35	622.51	831.47	**468.13**	488.45	11,059.28

Table 8. Comparison between the tested algorithms based on the throughput for encryption/decryption.

File Size	Throughput for the Encryption Process						Throughput for the Decryption Process					
	DES	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**	**DES**	**RC2**	**3DES**	**ASC**	**ARC**	**IARC**
500 MB	11.19	7.43	3.67	18.72	**21.83**	0.46	4.33	3.59	2.06	4.84	4.94	0.43
730 MB	11.54	7.10	3.99	15.62	**17.81**	0.19	4.16	3.46	2.40	4.52	4.61	0.18
1.2 GB	12.08	8.06	4.39	14.14	**14.67**	0.20	4.31	3.77	2.62	**4.64**	4.6	0.11
1.7 GB	11.42	7.36	4.28	16.73	**17.37**	0.17	4.28	3.57	2.61	**4.58**	4.09	0.15
2.2 GB	11.61	7.46	4.42	**17.69**	13.84	0.18	4.32	3.63	2.71	**4.84**	4.63	0.16

As shown in Figure 8, we can achieve fast big data processing, higher efficiency, and better performance by using the proposed algorithms ASC and ARC. However, these proposed algorithms are more complex but take less time than DES, DES3, and RC2 and increase security and improve cloud confidentiality.

Finally, a brief overview of ACS, RCS, and IARC approaches can be summarized in the following points to declare the novelties based on the data encryption features of cloud storage.

- Encryption of generated keys: Since the ACS, RCS, and IARC methods use two phases of encrypting the keys, the key generation procedure provides an efficient key generation process that helps resist brute-force assaults. One phase is used to encrypt the keys before encrypting the data, and another algorithm is used to safeguard the

keys to be securely stored. Since the key generation technique is utilized in ACS and RCS methods, the security level will be enhanced.

- Hybrid approaches: To improve security, we have combined four algorithms Twofish, RSA, AES, and DES, in different manners according to the adopted approach.
- Improved approach: To increase the security and achieve high confidentiality, we used a dynamic S-box without repeating the AES algorithm in the IARC approach.
- Automated approach: All encryption and storage operations are carried out without intervention from the cloud third party or the user.
- Time complexity: Although the operations are complicated in the proposed approaches, the processing time is still acceptable.
- Storage: The ACS, RCS, and IARC approaches proposed appropriate cloud storage data since they use concealed encryption to secure access against malicious parties.
- Security: ACS, RCS, and IARC approaches are secure algorithms because of the complicated operation and substitution–permutation and Feistel structure.

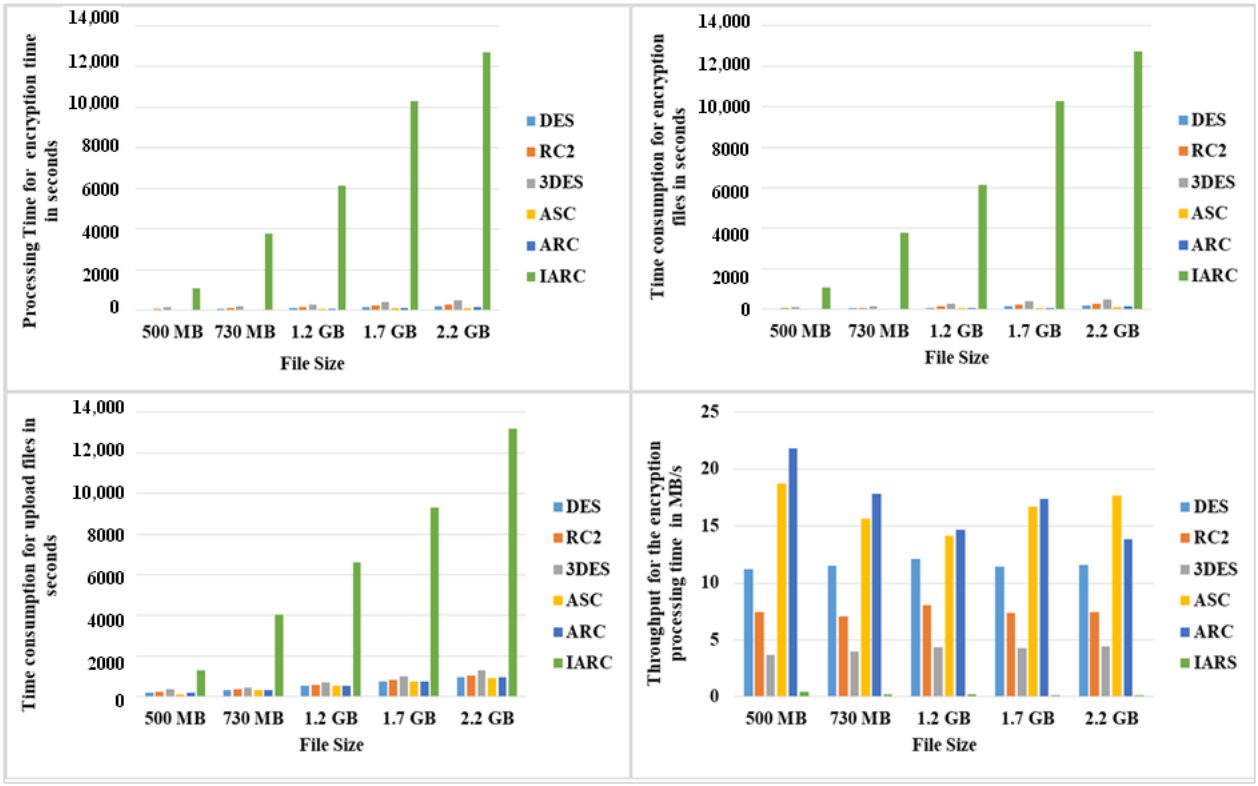


Figure 8. The comparison between the ACS, RCS, and IARC with other encryption algorithms.

6. Conclusions and Future Work

Increasing the volume of data has resulted in users saving data on remote access storage media, such as cloud computing storage infrastructures. Outsourcing the data makes it beyond the user's control and vulnerable to untrusted, anonymous operations. In this study, we have developed an automated hybrid cryptographic system that works without interference from cloud third parties to maintain data confidentiality and reduce the burden on users to secure their data by themselves. We have proposed three approaches; Automated Sequence Cryptography (ASC), Automated Random Cryptography (ARC), and improved automated random cryptography (IARC) for data blocks. Before using the keys to encrypt data, the keys are encrypted in two phases: initially, the keys are encrypted by the RSA algorithm to be stored in a private and safe database to be used later in decrypting the encrypted original data. The second phase to encrypt the keys is performed by the

Twofish algorithm, which is used to encrypt the keys utilized in encrypting the original data by AES and DES algorithms.

Furthermore, to increase security, data are divided into a random size of blocks and encoded by (DES, AES, and modified DAES) algorithms sequentially in the ASC approach and randomly in ARC and IARC approaches. To evaluate the performance of the proposed approaches, they have been compared with other state-of-the-art cryptographic algorithms, such as DES, 3DES, and RC2. This comparison has adopted various parameters, such as processing time, encryption/decryption time, upload/retrieve time, and encryption/decryption throughput. Experimental results indicate that, although some complex operations are carried out within the proposed ASC and ARC approaches, the results' analysis shows that our approaches have a high degree of security, more efficient data processing, and high throughput. Significantly, the ARC approach achieves a high level of security due to using the random manner in encrypting the blocks of data. On the other hand, the improved version "IARC" has been developed to increase and enhance the security level based on the dynamicity of the substitution process inside the modified version of the AES algorithm. Unfortunately, these high calculations, which are performed for every data block, lead to spend a lot of time the encryption and decryption processes.

Thus, in future work, we will need to employ the MapReduce programming model to apply concurrent processing strategy in running the proposed IARC model. In addition, an Auto Data Audits stage will be included in the proposed system to ensure the data saved is verified and integrated without interfering with the conventional cloud third party whenever the user wants to access it. This will ensure that the data are correctly integrated.

Author Contributions: All authors contributed equally to this paper, where D.S.E.-M. has designed and carried out the experiment, discussed the results, and written the paper. N.E.E.-A. designed the methodology, analyzed and discussed the results, and revised the paper. W.A.A. edited the manuscript, analyzed the results, and revised the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Academy of Scientific Research and Technology (ASRT), Egypt, Grant No. 6498.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in article.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

Appendix A

Table A1. A list of abbreviations.

Abbreviation	Meaning
CTP	Cloud third party
AEDS	Automated Encryption/Decryption System for Cloud Data Storage
ASC	Automated Sequential Cryptography
ARC	Automated Random Cryptography
AES	Advance Encryption Standard
DES	Data Encryption Standard
RSA	Rivest–Shamir–Adleman
3DES	Triple Data Encryption Standard
RC2	ARC2
IaaS	Infrastructure as a Service
PaaS	Platform as a Service

Table A1. *Cont.*

Abbreviation	Meaning
SaaS	Software as a Service
CSP	Cloud Service Provider
TTPA	Trusted Third-Party Auditor
CSS	Cloud Storage Server
TPA	Third-Party Auditor
TTP	Trusted Third Parties
MDS	Maximum Distance Separable
PHT	Pseudo-Hadamard Transform
RKG	Random Key Generator
EO	Encryption Operator
DO	Decryption Operator
DFS	Distributed File System

Table A2. A list of mathematical symbols.

Symbol	Meaning
N and M	prime numbers as random
φ	the function of Euler's totient
C	cipher data
K	original data
(e, L)	public key
(p, L)	the private key
K_i	Random key
$K_{i,\ RSA}$	Encrypted keys using RSA
$K_{i,\ Twofish}$	Encrypted key using Twofish
B_{lenght}	The size of the block
S	random number [2:4]
T	Time consumption for encryption/decryption files
PT	Overall Processing time

References

1. Mall, S.; Saroj, S.K. A new security framework for cloud data. In Proceedings of the International Conference on Advances in Computing and Communications, Kochi, India, 19–22 September 2018; Volume 143, pp. 765–775. [CrossRef]
2. Dhanapal, R.; Tharageswari, K.; Karthik, S. A decentralized accountability framework for enhancing secure data sharing through ICM in cloud. *Innov. Technol. Explor. Eng.* **2019**, *8*, 1505–1511, ISSN 2278-3075. [CrossRef]
3. Shahzadi, S.; Iqbal, M.; Qayyum, Z.U.; Dagiuklas, T. Infrastructure as a service (IaaS): A comparative performance analysis of open-source cloud platforms. In Proceedings of the International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, Lund, Sweden, 19–21 June 2017; pp. 1–6. [CrossRef]
4. Jathanna, R.; Jagli, D. Cloud Computing and security issues. *Int. J. Eng. Res. Appl.* **2017**, *7*, 31–38, ISSN 2248-9622. [CrossRef]
5. Kulkarni, G.; Waghmare, R.; Palwe, R.; Waykule, V.; Bankar, H. Cloud storage architecture. In Proceedings of the International Conference on Telecommunication Systems, Services, and Applications, Denpasar-Bali, Indonesia, 30–31 October 2012; pp. 76–81. [CrossRef]
6. Vurukonda, N.; Rao, B.T. A Study on data storage security issues in cloud computing. In Proceedings of the Inernational Conference on Intelligent Computing, Communications & Convergence, Odisha, India, 24–25 January 2016; Volume 92, pp. 128–135. [CrossRef]
7. Bentajer, A.; Hedabou, M.; Abouelmehdi, K.; Elfezazi, S. CS-IBE: A data confidentiality system in a public cloud storage system. *Procedia Comput. Sci.* **2018**, *141*, 559–564. [CrossRef]
8. Tawalbeh, L.A.; Saldamli, G. Reconsidering big data security and privacy in cloud and mobile cloud systems. *Comput. Inf. Sci.* **2019**, *33*, 810–819. [CrossRef]
9. Das, D. Secure cloud computing algorithm using homomorphic encryption and multi-party computation. In Proceedings of the International Conference on Information Networking, Chiang Mai, Thailand, 10–12 January 2018; Volume 1, pp. 391–396. [CrossRef]
10. Mohta, A.; Awasthi, L.K. Cloud data security while using third-party auditor. *Sci. Eng. Res.* **2012**, *3*, 1–9, ISSN 2229-5518.
11. Yusop, Z.M.; Abawajy, J.H. Analysis of insiders attack mitigation strategies. *Procedia-Soc. Behav. Sci.* **2014**, *129*, 611–618. [CrossRef]
12. Singh, S.; Thokchom, S. Public integrity auditing for shared dynamic cloud data. In Proceedings of the International Conference on Smart Computing and Communications, Kurukshetra, India, 7–8 December 2018; Volume 125, pp. 698–708. [CrossRef]

13. Potey, M.M.; Dhote, C.A.; Sharma, D.H. Homomorphic encryption for security of cloud data. In Proceedings of the International Conference on Communication, Computing and Virtualization, Mumbai, India, 26–27 February 2016; Volume 79, pp. 175–181. [CrossRef]
14. El-Attar, N.E.; Awad, W.A.; Omara, F.A. Empirical assessment for security risk and availability in public cloud frameworks. In Proceedings of the International Conference on Computer Engineering & Systems, Cairo, Egypt, 20–21 December 2016; pp. 17–25. [CrossRef]
15. El Makkaoui, K.; Ezzati, A.; Beni-Hssane, A.; Ouhmad, S. A swift Cloud-Paillier scheme to protect sensitive data confidentiality in cloud computing. In Proceedings of the International Conference on Mobile Systems and Pervasive Computing, Gran Canaria, Spain, 13–15 August 2018; Volume 134, pp. 83–90. [CrossRef]
16. Chakraborty, S.; Singh, S.; Thokchom, S. Integrity Checking using third party auditor in cloud storage. In Proceedings of the International Conference on Contemporary Computing, Noida, India, 2–4 August 2018; pp. 1–6. [CrossRef]
17. Nadlamani, G.F.; Shaikh, S. Preserving privacy using TPA for cloud storage based on regenerating code. In Proceedings of the International Conference on Recent Trends in Information Technology, Chennai, India, 8–9 April 2016; pp. 1–5. [CrossRef]
18. More, S.; Chaudhari, S. Third party public auditing scheme for cloud storage. In Proceedings of the International Conference on Communication, Computing and Virtualization, Mumbai, India, 26–27 February 2016; Volume 79, pp. 69–76. [CrossRef]
19. Singh, A.P.; Pasupuleti, S.K. Optimized Public auditing and data dynamics for data storage security in cloud computing. In Proceedings of the International Conference on Advances In Computing & Communications, Cochin, India, 6–8 September 2016; Volume 93, pp. 751–759. [CrossRef]
20. Ferhat, O.C.; Ahmet, F.M. CPP-ELM: Cryptographically Privacy-Preserving Extreme Learning Machine for Cloud Systems. *Int. J. Comput. Intell. Syst.* **2018**, *11*, 33–44. [CrossRef]
21. Saxena, R.; Dey, S. Cloud Audit: A Data Integrity verification approach for cloud computing. *Procedia Comput. Sci.* **2016**, *89*, 142–151. [CrossRef]
22. Adokshaja, B.L.; Saritha, S.J. Third-party public auditing on cloud storage using the cryptographic algorithm. In Proceedings of the International Conference on Energy, Communication, Data Analytics and Soft Computing, Chennai, India, 1–2 August 2017; pp. 3635–3638. [CrossRef]
23. Akhil, K.M.; Kumar, M.P.; Pushpa, B.R. Enhanced cloud data security using AES algorithm. In Proceedings of the International Conference on Intelligent Computing and Control, Coimbatore, India, 14–15 June 2018; pp. 1–5. [CrossRef]
24. Sivakumar, P.; NandhaKumar, M.; Jayaraj, R.; Kumaran, A.S. Securing Data and Reducing the Time Traffic Using AES Encryption with Dual Cloud. In Proceedings of the International Conference on System, Computation, Automation and Networking, Pondicherry, India, 29–30 March 2019; pp. 1–5. [CrossRef]
25. Orobosade, A.; Favour-Bethy, T.A.; Kayode, A.B.; Gabriel, A.J. Cloud Application Security using Hybrid Encryption. *Commun. Appl. Electron.* **2020**, *7*, 25–31, ISSN 2394-4714. [CrossRef]
26. Sohal, M.; Sharma, S. BDNA-A DNA inspired symmetric key cryptographic technique to secure cloud computing. *Comput. Inf. Sci.* **2018**, 1–8. [CrossRef]
27. Thabit, F.; Alhomdy, S.; Al-Ahdal, H.A.; Jagtap, S. A new lightweight cryptographic algorithm for enhancing data security in cloud computing. *Glob. Transit. Proc.* **2021**, *2*, 91–99. [CrossRef]
28. Maitri, P.V.; Verma, A. Secure file storage in cloud computing using hybrid cryptography algorithm. In Proceedings of the International Conference on Wireless Communications, Signal Processing and Networking, Chennai, India, 23–25 March 2016; pp. 1635–1638. [CrossRef]
29. Hoomod, H.K.; Hussein, A.M. New Modified Twofish for Data Protection Using Salsa20 and Lü system. In Proceedings of the International Conference on Intelligent Computing and Control Systems, Madurai, India, 15–17 May 2019; pp. 1189–1195. [CrossRef]
30. Zodpe, H.; Sapkal, A. An efficient AES implementation using FPGA with enhanced security features. *Eng. Sci.* **2020**, *32*, 115–122. [CrossRef]
31. Xu, X.; Tian, N. The search and improvement of DES algorithm for data transmission security in SCADA. In Proceedings of the International Conference on Intelligent Computing, Automation and Systems, Chongqing, China, 29–31 December 2019; pp. 275–279. [CrossRef]
32. Malgari, V.; Dugyala, R.; Kumar, A. A Novel Data Security Framework in Distributed Cloud Computing. In Proceedings of the International Conference on Image Information Processing, Shimla, India, 15–17 November 2019; pp. 373–378. [CrossRef]
33. Gupta, P.; Kumar, D.; Kumar Singh, A. Improving RSA Algorithm Using Multi-Threading Model for Outsourced Data Security in Cloud Storage. In Proceedings of the International Conference on Cloud Computing, Data Science & Engineering, Noida, India, 11–12 January 2018; pp. 14–15. [CrossRef]
34. Jintcharadze, E.; Iavich, M. Hybrid implementation of twofish, AES, ElGamal, and RSA cryptosystems. In Proceedings of the IEEE East-West Design & Test Symposium, Varna, Bulgaria, 4–7 September 2020; pp. 1–5. [CrossRef]
35. Ramtri, G.; Patel, C. Secure banking transactions using RSA and two fish algorithms. In Proceedings of the International Conference on Emerging Trends in Information Technology and Engineering, Vellore, India, 24–25 February 2020; pp. 1–5. [CrossRef]

36. Hemanth, P.N.; Abhinay Raj, N.; Yadav, N. Secure message transfer using RSA algorithm and improved Playfair cipher in cloud computing. In Proceedings of the International Conference for Convergence in Technology, Mumbai, India, 7–9 April 2017; pp. 931–936. [CrossRef]
37. Mehmood, M.S.; Shahid, M.R.; Jamil, A.; Ashraf, R.; Mahmood, T. A comprehensive literature review of data encryption techniques in cloud computing and IoT environment. In Proceedings of the International Conference on Information and Communication Technologies, Karachi, Pakistan, 16–17 November 2019; pp. 54–59. [CrossRef]
38. Mittal, S.; Arora, S.; Jain, R. PData security using RSA encryption combined with image steganography. In Proceedings of the International Conference on Information Processing, Delhi, India, 21–23 December 2017; pp. 1–5. [CrossRef]
39. Panda, M. Performance analysis of encryption algorithms for security. In Proceedings of the International Conference on Signal Processing, Communication, Power and Embedded System, Paralakhemundi, India, 3–5 October 2016; pp. 278–284. [CrossRef]
40. Rani, K.; Sagar, R.K. Enhanced data storage security in cloud environment using encryption, compression asplitting technique. In Proceedings of the International Conference on Telecommunication and Networks, Noida, India, 10–11 August 2017; pp. 1–5. [CrossRef]
41. Jayant, B.; Swapnaja, U.; Subhash, P.; Kailash, K.; Sulabha, A. Developing secure cloud storage system by applying AES and RSA cryptography algorithms with role-based access control model. *Comput. Appl.* **2015**, *118*, 46–52. [CrossRef]
42. Ametepe, A.F.; Ahouandjinou, S.A.; Ezin, E.C. Secure encryption by combining asymmetric and symmetric cryptographic method for data collection WSN in smart agriculture. In Proceedings of the International Smart Cities Conference, Casablanca, Morocco, 14–17 October 2019; pp. 93–99. [CrossRef]
43. Fauziah, N.A.; Rachmawanto, E.H.; Moses Setiadi, D.; Sari, C.A. Design and implementation of AES and SHA-256 cryptography for securing multimedia file over android chat application. In Proceedings of the International Seminar on Research of Information Technology and Intelligent Systems, Yogyakarta, Indonesia, 21–22 November 2018; pp. 146–151. [CrossRef]
44. Shivhare, R.; Shrivastava, R.; Gupta, C. An enhanced image encryption technique using DES algorithm with random image overlapping and random key generation. In Proceedings of the International Conference on Advanced Computation and Telecommunication, Bhopal, India, 28–29 December 2018; pp. 1–9. [CrossRef]
45. Yassein, M.B.; Aljawarneh, S.; Qawasmeh, E.; Mardini, W.; Khamayseh, Y. Comprehensive study of symmetric key and asymmetric key encryption algorithms. In Proceedings of the International Conference on Engineering and Technology, Antalya, Turkey, 21–23 August 2017; pp. 1–7. [CrossRef]
46. Subandri, M.A.; Cyclomatic, R.S. Complexity for Determining Product Complexity Level in COCOMO II. In Proceedings of the Information Systems International Conference, Denpasar-Bali, Indonesia, 6–8 November 2017; Volume 124, pp. 478–486. [CrossRef]
47. Reddy, I.R.; Murali, G. A novel triple DES to enhance e-governance security. In Proceedings of the International Conference on Energy, Communication, Data Analytics and Soft Computing, Chennai, India, 1–2 August 2017; pp. 2443–2446. [CrossRef]
48. Charbathia, S.; Sharma, S. A comparative study of rivest cipher algorithms. *Inf. Comput. Technol.* **2014**, *4*, 1831–1838, ISSN 0974-2239.
49. Malviya, S.; Dave, S. Secure data sharing scheme using cryptographic algorithm for cloud storage. *Appl. Eng. Res.* **2018**, *13*, 14799–14805, ISSN 0973-4562.
50. Srilaya, S.; Velampalli, S. Performance Evaluation for DES and AES Algorithms—An Comprehensive Overview. In Proceedings of the International Conference on Recent Trends in Electronics, Information & Communication Technology, Bangalore, India, 18–19 May 2018. [CrossRef]