Received May 16, 2020, accepted June 13, 2020, date of publication June 23, 2020, date of current version July 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3004433

PSDS–Proficient Security Over Distributed Storage: A Method for Data Transmission in Cloud

FIZZA SHAHID[1], HUMAIRA ASHRAF[1], ANWAR GHANI[1], SHAHBAZ AHMED KHAN GHAYYUR[1], SHAHABODDIN SHAMSHIRBAND[2], AND ELY SALWANA[3]

[1]Department of Computer Science and Software Engineering, International Islamic University Islamabad, Islamabad 44000, Pakistan
[2]Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
[3]Institute of IR4.0, Universiti Kebangsaan Malaysia, Bangi 43600, Malaysia

Corresponding authors: Anwar Ghani (anwar.ghani@iiu.edu.pk) and Shahaboddin Shamshirband (shamshirbandshahaboddin@duytan.edu.vn)

ABSTRACT Cloud Computing facilitates business by storing an enormous amount of data in the cloud transmitted over the Internet with seamless access to the data and no hardware compatibility limitations. However, data during transmission is vulnerable to man in middle, known plain text, chosen cipher text, related key and pollution attack.Therefore,uploading data on a single cloud may increase the risk of damage to the confidential data. Existing literature study uncovered multiple cryptography techniques such as SA-EDS, Reliable Framework for Data Administration (RFDA), Encryption and Splitting Technique (EST) to secure data storage over multi-cloud.However, existing methods are vulnerable to numerous attacks. This article emphasis on data security issues over multi cloud and proposes a Proficient Security over Distributed Storage (PSDS) method.PSDS divides the data is into two categories; normal and sensitive, further more the sensitive data is further divided into two parts. Each part is encrypted and distributed over multi-cloud whereas the normal data is uploaded on a single cloud in encrypted form. At the decryption stage, sensitive data is merged from multi-cloud. The PSDS is tested against multiple attacks and it has been concluded that it is resistant to related key attack, pollution attack, chosen ciphertext attack, and known plain text attack. Furthermore, PSDS has less computational time as compared to the STTN and RFD encryption method.

INDEX TERMS Cloud computing, data transmission, security, cryptographic solution.

I. INTRODUCTION

Cloud computing environment offers enormous benefits over the local computing environment including financial cost, administrative and management overhead, adaptability, seamless office access, less memory utilization, and so on. Cloud computing provides a platform to users to utilize various assets provided on their request. Cloud computing provides adaptability by giving backup facility like Dropbox, Amazon, and Google Drive. Cloud computing also facilitates clients to cut down their expenses by providing environment for testing applications without building up a physical domain. It also provides administrative facilities, for example, Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [1], [2]. SaaS offers service to clients, for example, the virtual work area, webmail, and program interface. PaaS provides a stage for using programming language, administrations, libraries, and various apparatuses. IaaS offers online administrations to clients, for example, servers, load balancers, and other computing assets. Due to these reasons today businesses, organization and individual users are moving their data to the cloud.Various models of cloud such as private, public, hybrid and community cloud are defined [3] to provide various services such as infrastructure, software, and others.

The security of information with overwhelming size in the cloud is a noteworthy issue. Various strategies have been proposed not only to verify information but also to prevent unauthorized access. An over view of cloud computing and related applications is given in Figure 1. Data security is a

The associate editor coordinating the review of this manuscript and approving it for publication was Chenghong Gu.

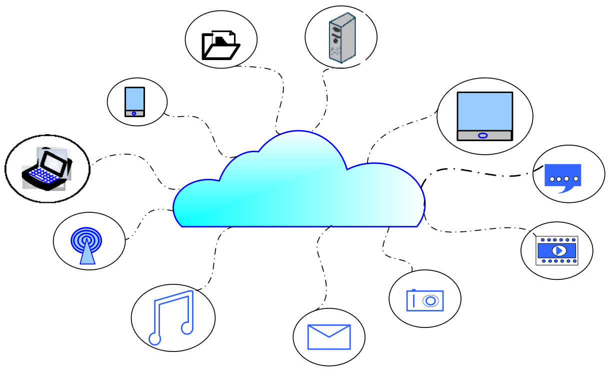


FIGURE 1. Cloud overview.

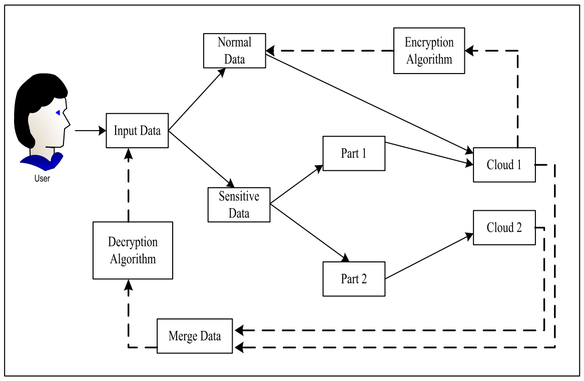


FIGURE 2. System architecture of the proposed PSDS.

crucial concern while transferring information over the system having various solution proposed in the literature. Be that as it may, cryptography is one of the principal methods used to encipher the data utilizing either symmetric key or asymmetric key. The asymmetric key is considered exceptionally secure as encryption and decryption use different keys. The key generation process of asymmetric key consumes a huge amount of energy and space [4].

Existing proposals have both positives and negatives, for example, Advanced Encryption Standard (AES) is a high-security method used for encryption [5]. Also, Shamir Secret Sharing Scheme is used for encryption [6]. In [7] sensitive data is encrypted by taking XOR with a random number, split and distributed over two clouds. In [8] the author uses a hybrid method such as Advanced Encryption Standard (AES), Rivest Shamir Adleman (RSA), Blowfish to secure data. Fully Homomorphic Encryption (FHE) is utilized to encrypt data and then encrypted data is distributed over multi-cloud [9]. In [5] to secure data, AES is used in combination with MD5, however, there is a possibility of a cache-based timing attack on AES [10]. There are chances of Biclique attacks on AES, as proved in [11]. Another approach to securely store/transmit data is to part information into equal parts and store it on multi-cloud. To access total information, split parts are blended. Parting information on multi-cloud improves security in such a way that even if an aggressor gains admittance to a part of the data may still be unable to access the full information [12].

This article proposes a symmetric key based cryptographic method named Proficient Security over Distributed Storage (PSDS) to secure client's information over the cloud as shown in Figure 2. The client chooses whether the information is private (sensitive) or typical (normal) information. The private information is split into two sections, part1 and part2 as appears in Figure 2. After splitting data, encryption steps of PSDS are applied to both parts. Both encrypted parts are uploaded on to two separate clouds, cloud 1 and cloud 2 in order to prevent loss or exposure of data. Normal data is encrypted with PSDS's encryption method and uploaded over a single cloud as shown in Figure 2. The solid line in Figure 2 shows encryption steps while the dotted line shows decryption steps. In the decryption phase, sensitive data from both clouds is downloaded and then merged both parts. After merging apply the decryption method of PSDS to convert ciphertext into plain text. In the decryption process of normal data, the data is downloaded from a single cloud and then decryption is applied to transform ciphertext to original text.

PSDS is an adaptable approach as it is designed to achieve high security by splitting sensitive data on a multi-cloud mechanism. The proposed approach resists against different attacks such as Chosen Ciphertext attack, related key Attack, and pollution attack. It also protects data against illegal access by the cloud service providers.

A. CONTRIBUTIONS

The general contributions of this article are:

- A proficient and secure data storage approach has been proposed that distributes sensitive users' data onto different cloud servers to avoid damage and vulnerability.
- A mathematical model has been developed and presented along with encryption and decryption algorithms to encrypt and decrypt sensitive as well as normal data.
- The proposed technique has been analyzed for security against various known attack to assess its security.
- The proposed technique has also been analyzed for computation and communication overhead in case of both sensitive as well as normal data to assess its complexity.
- A comparative analysis of the proposed technique has been presented with AES, STRRNS, RFDA, and SA-DES in terms of computation time both for sensitive data as well as the overall encryption/decryption time.

B. ARTICLE LAYOUT

The rest of the document is divided as follows: section II discusses the Literature Review, section III describes System Architecture of the proposed PSDS, section IV describes the proposed PSDS technique. In section V Test bed setup has been described. In section VI experimental results are discussed. In section VII security analysis is explained, and in section VIII the article has been concluded.

II. LITERATURE REVIEW

Cloud computing encourages individuals by giving a stage to store their information so as to spare their memory. Various strategies have been proposed to secure information over the cloud. such as AES, Homomorphic Encryption, Chaotic Map Method, BlowFish Algorithm is sub-order of the symmetric method.

Kumar et. al propose a Reliable Framework for Data Administration (RFDA) strategy. RFDA uses Advanced Encryption Standard (AES) for encryption. The author portrays the Data Splitting method [5]. AES uses larger space because it uses extra registers, comparators, multiplexers, and complex control as stated in [13]. Then it is seen that the key can also be attacked, by launching a related-key attack, as proved in [14]. Schindler et al propose Splitting through RRNS (STRRNS) for information security. In the proposed method information is part by Redundant-Residue-Number-System (RRNS) into pieces. Chunks are encoded using AES 256 bit [15].

There are chances of Biclique attacks on AES, as proved in [11]. Ximing *et al.* propose a Secure File Storage System among Distributed public Clouds (SFSS-DC) method to secure information over the cloud. Documents are isolated into pieces utilizing the Shamir Secret Sharing Scheme (SSSS) by the data owner. These pieces are encoded utilizing AES. Each encoded document is transferred to two clouds [16].

In [17] four types of attacks on SSSS are described. First is done by honest people and done accidentally and shared data synchronously, Second is done by honest people accidentally, data is shared asynchronously, Third is done by cheaters, the fourth type is done by cheaters and shares are disclosed asynchronously. Mudgal and Bhatia propose Encryption and Splitting Technique (EST). The record is encoded utilizing a mix of AES and SHA-1. The document is part of various segments to transfer on various clouds. Data on the transferred document is put away in the metadata the board server. Metadata is important to blend all these split segments [5], [18]. VR proposes Data Splitting with Dynamic Approach (DSDA). In DSDA the author utilizes a hybrid cloud to store information and metadata is put away on a private cloud. The record is encrypted utilizing AES 256 bit which is the symmetric block cipher. The encoded record is partitioned into parts and transferred on two distinct clouds [19].

Sant proposed Fully Homomorphic Encryption (FHE) to verify the information. Homomorphic encryption is such an encryption calculation that can play out a count on encoded data without decoding it. These encoded outcomes must be decoded by the customer who demands these calculations. The main information is encoded and afterward split over various clouds [9]. FHE is that it is vulnerable to malware attacks as stated in [20]. Tchernykh proposed WA-RRNS in order to secure data [21]. In WA-RRNS sift hold is utilized in mix with RRNS to partition information. The pieces are changed over into homomorphic ciphers.

In [21], it is proved that reliability is improved while compromising performance speed during encoding and decoding. For encoding WA-RRNS is three times slower and for decoding it is four-time slower as compared to the traditional system. In [22] article, with the built model called fuzzy-based semantic search for safe data discovery (FSS-SDD), the effective data discovery process is focussed and enhanced. A multi-valued logic called fuzzy logic is used, which has values of truth and variables ranging from 0 to 1. It is used where the meanings of truth vary from absolutely true to utterly false. The fuzzy semantic search based on logic increases the end user's search experience by locating and retrieving the same matching data provided by the user for the corresponding search data. In addition, the model uses semantine similarities to discover the closest related matches, in situations where the exact matches are not usable.

Multi-Cloud Approach for secure Data Storage (MCASD) is a methodology proposed by Karri and Mishra which uses the Chaotic Map for the security of information. MCASD split information on multi-cloud. In this model, the initial step is to partition, encoding and distribute information on multi-cloud. In this model for encryption, the Chaotic Map is utilized [23]. There can be two types of attacks on chaotic map A Chosen Ciphertext Attack and A known Plaintext Attack as stated in [24]. As proved in [25] there are four possible attacks on Chaotic Map Encryption Cipher text-only, Known-plaintext, Chosen plaintext, and Chosen ciphertext.

Dixit proposed a method named as Securing by Heterogeneous Cloud (SHC). In SHC the document is part into chunks utilizing SSSS. At that point, each document lump is encoded utilizing Blow Fish. It is proven that the reflection attack is possible on Blow Fish [26]. All weak keys of BlowFish can be recovered with 2^{48} chosen plain text against the number of rounds [27].

Sulochana proposed Data Confidentiality using Multi-Cloud Architecture (DCMCA) [28]. In DMCA, client's login into distributed storage utilizing a username and secret word. At the point when a document is transferred, the manager encrypts the record utilizing RSA. Administrator partitions scrambled document with division calculation and stores the pieces on various areas of another cloud [29]. Partial key exposure attacks can be possible on RSA [30]. RSA is vulnerable to Quantum polynomial-time fixed-point attack [31]. In [6] SSSS is used in combination with MD5 to secure data over the cloud. To keep the client's information private Pachipala utilizes Shamir's mystery sharing method (SSSS) in [6]. The information is transferred before transferring to the cloud and afterward transferred to multi-cloud. The document is encoded with the assistance of SSSS and secretes keys are created and these mystery keys are transferred to various clouds. In [32] the author describes 3 types of attack on SSSS. First is that the shareholders are honest and done errors accidentally. Second is that who is dishonest who modifies the data intentionally. They release their data synchronously. The third type is also dishonest who release data asynchronously. In [17] four types of attacks on SSSS are described. First is done by honest people and done accidentally, data is shared synchronously, second is done by honest people accidentally, data is shared asynchronously, Third is done by cheaters to fool the honest people and shares are disclosed synchronously, the fourth type is done by cheaters and shares are disclosed asynchronously. Qiu and Zaho proposed Security-Aware Efficient Distributed Storage (SA-EDS) algorithm in [7]. In the proposed calculation information is checked whether it is confidential information or not. On confidential information, encryption is connected and conveyed among various clouds. An arbitrary paired number is produced and subtracted from the content. The irregular key is produced and XORed with an arbitrary number. In [7] it is shown that SA-EDS have less encryption time as compared to AES on different data sets. It is proved in [33] that by brute force the keys can be extracted. In [34] it is proved that XOR is vulnerable to Pollution Attack.

The article [35] proposes an enhanced and safe authentication scheme free of issues of validity to promote a key agreement by trusted authority between user and cloud service. The proposed framework as an program often aims to achieve a central compromise with smart meters and cloud servers. Based on the Elliptic Curve Decisional Diffi-Hellman Problem (ECDDHP) principle of consistency, the standard Random oracle model is evidence of the reliability of the proposed scheme. The robustness of the scheme is further clarified by informal analysis. The framework suggested though offering all established security features has raised the cost of computing and connectivity marginally.

An improved symmetric key-based authentication protocol for IoT based WSN was introduced in [36] article. The protocol suggested will combat user traceability, compromised verifier, and DoS attacks. In addition, the proposed Protocol was tested and validated using the principles of Proverif and BAN. The suggested protocol has the same coordination cost as the baseline protocol, but it has an improvement of 52.63 per cent relative to the baseline protocol in computing costs.

The [37] suggest an improved SIP authentication protocol which covers the Dongqing *et al.* protocol limitations. Using BAN logic analysis proposed scheme is formally proved as stable. The performance analysis shows the contrast with the scheme introduced for similar schemes, and demonstrates the scheme's effectiveness and robustness over other schemes.

As proven in [38] timing attack is possible for DES, RSA, and Defi-Helman. As demonstrated in [39], DES is susceptible to known-plaintext attacks. In the [40] it was proven that the first known attack was used to break the DES 16 round AES uses more space because it uses extra registers, comparators, multiplexers, and complex controls as stated in [13]. The LIM based attack is vulnerable to AES [41]. There are chances of Biclique attacks on AES, as evidenced in [11]. Then it is seen that the key can also be attacked, by launching a related-key attack, as evidenced in [14]. There is the possibility of time-based cache attacks in AES [10].

The research [42] is restricted to supervised and unsupervised methods of machine learning (ML), considered the foundation of IoT smart data analyzes. This report contains analyses and discussions of significant problems relating to supervised and unsupervised machine learning methods, outlining the benefits and disadvantages of each algorithm, and addressing the scientific findings analysis.

Data Veracity for Cloud Storage through Dual Protection (DVCSDP) gives double security proposed by Kannan. In the initial step, the record is part into various lumps and dispersed over various servers [43]. Suwansrikham Proposed Asymmetric Secure Storage Scheme (ASSS) [44]. In ASSS information proprietor parts the document and approved the client by producing the token which contains the username, secret word, and area.

III. SYSTEM ARCHITECTURE

The system architecture of the proposed approach for file distribution over multi-cloud is shown in Figure 2. The system architecture consists of user and cloud storage.

A data owner is a person who owns data files and a data owner must Login to upload or download a file over the network. After a successful Login, a data owner may decide the file type. Cloud storage is a database that provides a storage place to the users to store their file(s). Different cloud providers have different policies. An application interface (API) is provided by cloud service providers enabling users to interact with its services.

The working of a the proposed system can be sum up in two phases; Data distribution over multi-cloud, and merging data from multi-cloud. In data distribution phase, confidential data is split into two parts to store each part on a different cloud ensuring security in case one cloud gets compromised. In merging phase, the chunks from multi-cloud are recollected and merged to obtain plain text.

However, uploading data on one cloud or multi-cloud does not guarantee that the data is secure. There is a need for an efficient encryption algorithm that provides high security with less computational time. Different encryption methods are proposed to secure data over the cloud as discussed in Section II, however, every proposal is not generic and every environment has its own variables.

IV. PROFICIENT SECURITY OVER DISTRIBUTED STORAGE (PSDS)

The proposed PSDS algorithm is based on symmetric key encryption to ensure the confidentiality of the data even if the data is uploaded in parts on different clouds. The proposed method uploads sensitive data on multi-cloud in order to protect it from unauthorized access as well as in case of any undesirable situation, all data must not be at a single location. The working of the proposed approach has been shown in Figure 3.

The working of the proposed PSDS approach may be divided into different parts including; Key Generation – to generate keys for symmetric encryption to ensure the confidentiality of the data to be stored on the cloud, a splitting algorithm to divide the sensitive data into parts, and

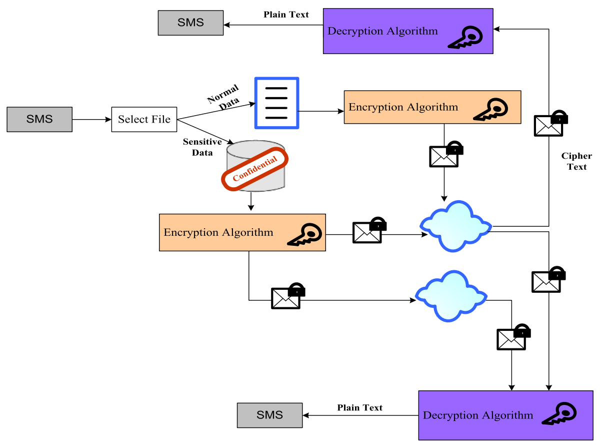


FIGURE 3. Overview of the proposed PSDS.

encryption and decryption algorithms to encrypt data while uploading and decrypt it back the owner wants to access the data. Different symbols used in the algorithms and flow diagrams of the proposed PSDS are presented in Table 1.

TABLE 1. Notation table.

Symbol	Description
ASCII	ASCII value
BIN	Binary value
RF	Value after applying Rail fence
$Keys_{S-Box}$	The key use for S-Box
T_{S-Box}	The computed value from S-Box
Key_{DNA}	Key used for DNA Table
T_{DNA}	The computed value from DNA tables
T_{shift}	Text after circular shift
$1 \leftarrow$	1 circular left shift
$1 \rightarrow$	1 circular right shift
B_{DNA}	Value from the DNA table
B_{S-Box}	Value from S-box Table
C_{ip}	Ciphertext
P	Plain text
H	Halve
Cl_1	Cloud 1
Cl_2	Cloud 2
Ch_1	The ciphertext of the first part
Ch_2	The ciphertext of the second part
E	Encryption Algorithm

A. KEY GENERATION

The key generation process of the proposed symmetric key-based PSDS approach is depicted in Figure 4.

The key generation process uses a random number generator to generate two random numbers and calculates its 1's complement. The $Keys_{s-box}$ takes the last two binary digits of the complemented number on the right and the first two binary digits of the complemented number on the left. This is the output of the $Keys_{s-box}$ which is fed into the $S-Box$. The second input is formed by combining the first two binary digits of the complemented number on the right and the last two digits of the complemented number on the left. Using these two inputs, the value from the $S-Box$ is calculated that is treated as key for the DNA algorithm.

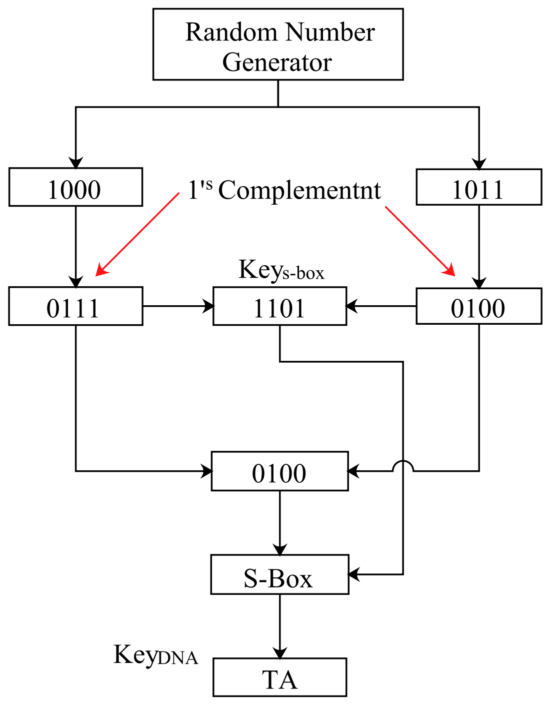


FIGURE 4. Key generation process of the proposed PSDS.

The description of a key generation process is given in Algorithm 1.

```
Algorithm 1 Key Generation
Input: binary number
Output: Keys-box, keyDNA
1: Generate two random binary number
2: Take 1's complement of both binary numbers.
3: Select last two digits of the first and the first two digits
   of the second number.
4: $Keys_{s-box}$ is generated.
5: Select first two digits of the first and last two digits of
   the second number.
6: Use this number and $Keys_{s-box}$ to calculate a new value
   using S-Box.
7: Use this value as a key for DNA algorithm ($Key_{DNA}$).
```

B. SPLITTING ALGORITHM

Before encrypting the data a client defines the category of data, whether his data is confidential and sensitive type or Normal data is encrypted using encryption algorithm 3. After encryption, the encrypted text is uploaded on a single cloud. While sensitive data is divided into two parts. Both parts are encrypted separately with the encryption algorithm. Both of these encrypted parts are uploaded on two different clouds. The Flow of splitting mechanism is shown in Figure 7.

The steps of the proposed splitting algorithm for PSDS are shown in in Algorithm 2.

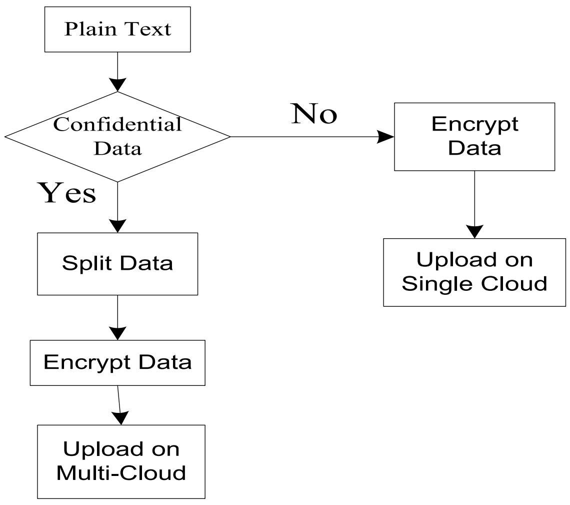


FIGURE 5. Flow and steps of the splitting method in the proposed PSDS.

```
Algorithm 2 Splitting Algorithm
Input: Plain text p
Output: Ch1, Ch2
    Step 1: Obtain plaintex P
    Step 2: Divide the plaintext into tow parts “$P_1$ and $P_2$”
    Step 3: Compute ciphertext $Ch_1 \leftarrow E(P_1)$
    Step 4: Compute ciphertext $Ch_2 \leftarrow E(P_2)$
    Step 5: Store one part on $Cl_1 \leftarrow Ch_1$
    Step 6: Store the other part on $Cl_2 \leftarrow Ch_2$
```

The description on the working of each of the steps is as follows:

1) STEP 1-4

Obtain plain text “P” and divide the length of the plain text with 2 which divides the plain text into two parts. After getting two parts of ciphertext encryption steps are applied to both parts to get ciphertext Ch_1 and Ch_2.

2) STEP 5-6

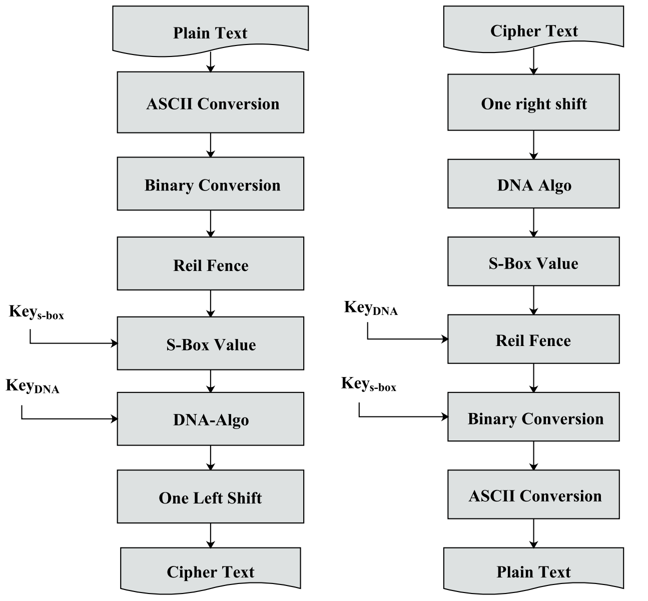
After getting ciphertext of both parts both parts are separately uploaded on two different clouds. upload Ch_1 to Cl_1 and Ch_2 to Cl_2.

C. ENCRYPTION AND DECRYPTION ALGORITHM

In the encryption phase of normal data, the data is converted into its ASCII values and then each ASCII value is converted into a binary number. Then the Rail Fence is applied to the binary text and then the values from S-Box have been calculated using Key_{s-box}. The S-Box has been represented in Table 2. After this step, the values using DNA algorithm have been computed using the the DNA Table 2. In the last step of the encryption process, a circular left shift of the text is performed to obtain the ciphertext.

TABLE 2. S-box and DNA tables.

S-Box					DNA Table				
	00	01	10	11		T	A	C	G
00	A	C	T	G	T	T	A	C	G
01	T	A	G	C	A	A	T	G	C
10	G	T	C	A	C	C	G	T	A
11	C	G	A	T	G	G	C	A	T



(a) Flow of encryption method (b) Flow of decryption method

FIGURE 6. Flow of encryption/decryption methods.

The Flow of encryption method is shown in Figure 6(a). In the Decryption Phase, the first step is to take One Circular Right Shift of Cipher Text. Then calculate the value using DNA algorithm. Then the value is calculated using S-Box. To calculate the value from S-box, Check the text value and then check the corresponding Key value. Then take Rail fence is applied on the text. Convert the text into their ASCII values. Then ASCII value is converted into plain text. The flow of the decryption method of PSDS is shown in Figure 6(b).

In Table 2 value of S-Box is given. The horizontal line shows Keys-box value and the vertical line shows the text value in S-Box. When Keys-box is 00 and text value is 11 then the new text will be C. When Key_{s-box} is 10 and text value is 01 then the new text will be G.

In Table 2 the value of DNA algorithm is given. The horizontal line shows Key_{DNA} value and the vertical line shows the text value in S-Box. When Key_{DNA} is C and text is A the text will be converted into G. When Key_{DNA} is T and text is C the text will be converted into C.

In algorithm 3 and algorithm 4 encryption and decryption algorithm of PSDS is given. The encryption and decryption algorithm is briefly described below.

1) ENCRYPTION METHOD

The encryption method of PSDS is shown in algorithm 3. The detail of the encryption method is described below.

Algorithm 3 Proposed Encryption Method

```
Input: Plain text P
Output: Cipher Text $C_{ip}$
 1: for all $k \in p$ do
 2:     Convert k into equivalent ASCII value
 3: end for
 4: for all $n \in ASCII$ do
 5:     Convert n into equivalent binary(bin) number
 6: end for
 7: for all $bin$ do
 8:     Apply reil fence algorithm
 9: end for
10: Calculate value using $keys_{S-Box}$ from S-Box.
11: Apply DNA algorithm and use $key_{DNA}$
12: for all $d$ do
13:     Apply once circular left shift
14: end for
```

Algorithm 4 Decryption Algorithm

```
Input: Cipher Text $C_{ip}$
Output: Plain text P
 1: for all $C$ do
 2:     Apply once circular right shift
 3: end for
 4: Apply DNA algorithm
 5: Using S-Box calculate value from S-Box.
 6: for all $s$ do
 7:     Apply reil fence algorithm
 8: end for
 9: Divide the data into 4 bits each.
10: for all $r$ do
11:     Convert bits into equivalent ASCII
12: end for
13: for all $ASCII$ do
14:     Convert ASCII into equivalent plain text
15: end for
```

In Step 1-4 ASCII conversion, binary conversion and rail fence is described. In step 5-8 the detail of S-box and DNA algorithm is described and last steps of conversion of plain text into ciphertext.

$$M = \sum (N^{1-n}, X^{1-n}, R^{1-n}, d^{1-n}) \tag{1}$$

Step 1–4: Obtain plain text which the user wants to upload over the cloud. Convert the plain text into their equivalent ASCII value. (E.g. for Z its ASCII value is 90). After getting ASCII value, now convert ASCII values into binary values. (E.g. for 90 its binary value is 01011010). Apply the rail fence algorithm on the binary value obtained by step 3. (E.g. for 01011010 rail fence value is 01110001).

$$N^{1-n} = ASCII(P_i) \tag{2}$$

In Eq. (2), P_i is the Plain text N is the ASCII converted value of P_i,

$$X^{1-n} = Bin(N_i) \tag{3}$$

In Eq. (3) X is stores value of each Ni binary equivalent

$$R^{1-n} = RF(X_i) \tag{4}$$

InEq. (4) R^{1-n} stores the plain text values after applying rail fence.

Step 5-8: After applying the rail fence, compute a value from S-Box using Keys-box. (E.g. the key bit is 00 and the text bit is 10 the new text will be G).The value obtained from Step 5 is used to calculate the value from DNA Table using KeyDNA. (E.g. the key bit is C and the text bit is G the new text will be A). The circular left shift is applied to the results of Step 6. (E.g. TGCAA becomes GCAAT). After once circular left shift ciphertext is obtained.

$$K^{1-n} = DNAsubstituteS - box(R^1 - n) \tag{5}$$

In Eq. (5) K stores value of each value of R after DNA substitution through S-BOX

$$d^{1-n} = CLS(K) \tag{6}$$

In equation 6 the d preserve the value K after circular shift

2) DECRYPTION METHOD

In the decryption, method ciphertext is converted into plain text. The conversion process of ciphertext into the plain text of PSDS is shown in algorithm 4. Step 1-4 describes a circular shift, DNA algorithm, and process of S-box is explained. In Step 5-8 rail fence, the binary conversion is described. At last plain text is obtained.

Step 1-4: Obtain the ciphertext downloaded from the cloud. Take the once circular right shift on the ciphertext. (E.g. GCAAT becomes TGCAA). Compute value from DNA table using Key_{DNA} on a result obtained from Step 2 (e.g. key bit is C and text bit is A the new text will be G) Use S-Box to calculate the value using Keys-box. (E.g. the key bit is 00 and the text bit is 10 the new text will be G).

Step 5-8: Apply the rail fence on the results of Step 4. (E.g. for 01110001 rail fence value is 01011010). Result of Step 5 is converted into their equivalent ASCII value (E.g. for 01011010 its ASCII value is 90). The results of Step 6 are converted into character values (E.g. 90 is converted into Z). Plain text is obtained.

V. TEST BED SETUP

The Proficient Security over Distributed Storage (PSDS) has one client and one server for normal data. In the case of sensitive data, PSDS has one client and two servers. Sensitive data is split into two halves and both halves are uploaded on two different servers. Normal data is encrypted and pass over a single cloud while sensitive data has two parts; both parts are encrypted individually and uploaded on two different servers. The test bud setup is shown in Figure 8.

TABLE 3. Computational time.

Parameter			Computational Time	Ciphertext Length (Bytes)
Message	Data Length (Bytes)	Key size (Bits)	Total Time (MS)	
Hello	5	4	0050	5
Cloud Computing	15	4	0089	15
Security Over Network	21	4	0120	21

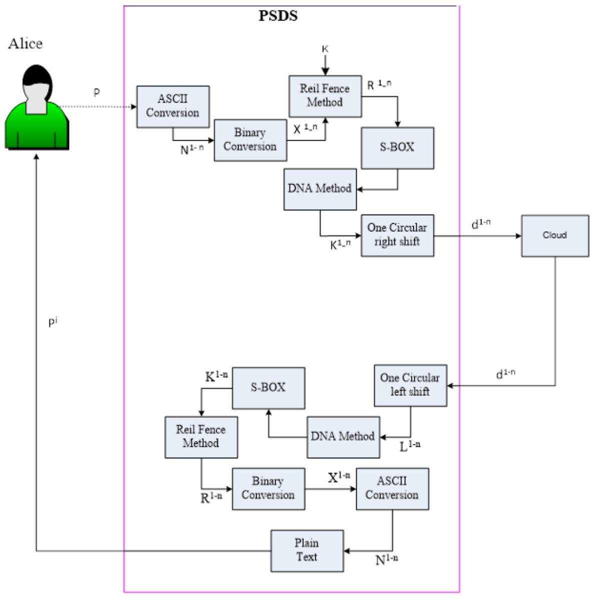


FIGURE 7. System model.

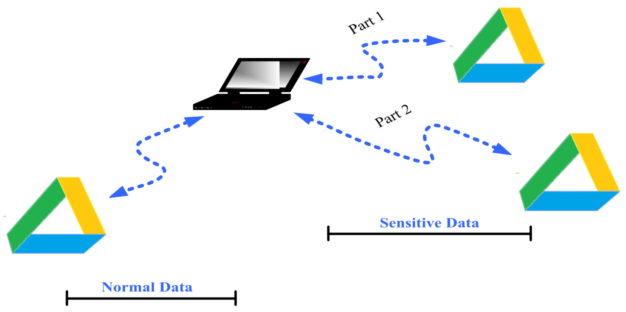


FIGURE 8. Test bed of the proposed PSDS.

VI. EXPERIMENTAL RESULTS

Experimental results are measured in terms of encryption and decryption time. Results are also analyzed from a security perspective.

A. COMPUTATIONAL TIME

The computational time is an important performance parameter for an encryption approach that shows how many times a certain operation is performed and what is the total number of operations performed during one transactions of a protocol. Computational time is the total amount of time taken by an algorithm to complete a specific amount of computation. Table 3 explains the time taken by key generation process, Encryption time and decryption time for different lengths of data. Computational time is shown in Table 3. The total computational time for 15 bytes is 0089 ms. It can be seen from the table that for 21 bytes, the computational time is 0120 ms whereas the total computational time for 5 bytes is 0050 ms.

B. ENCRYPTION TIME

Systems with complex and sophisticated security require more time to convert a plaintext into a ciphertext as the encryption process performs complex computation. However, the encryption time and complexity requirement is a trade-off and must be kept at a balance to provide better security in less time. The amount of time required to convert a plaintext into a ciphertext is known as encryption time. It can be used as a parameter to gauge computation efficiency of a particular encryption algorithm. Similarly, to check the performance of the proposed PSDS approach, its encryption time both in case of normal as well as sensitive data has been computed. Figure 9 shows encryption time in milliseconds (ms) of PSDS in case of normal data. The input data is considered byte by byte. It can be clearly observed from Figure 9, when size of the data is of 182 bytes, the encryption time is 1000 ms, the encryption time is increasing with the increase of data size due huge data has to be substituted through S-BOX and DNA table increases the time consumption after binary conversion.

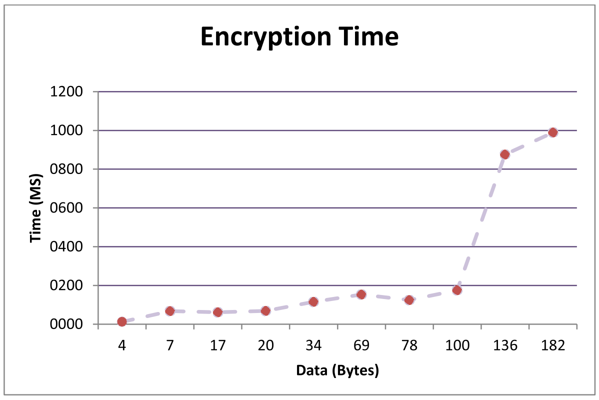


FIGURE 9. Encryption time of normal data.

Similarly, Figure 10 shows the encryption time in case of the sensitive data. In this case a total number of 78 bytes of data has been encrypted taking a time of 198 ms, due to the fact that sensitive data splited before encryption.This research is tested only for KB's because our machines were with low RAM which does not support the encryption of MB files.

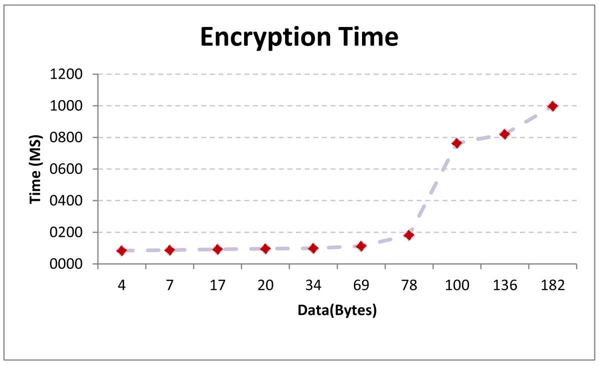


FIGURE 10. Encryption time of sensitive data.

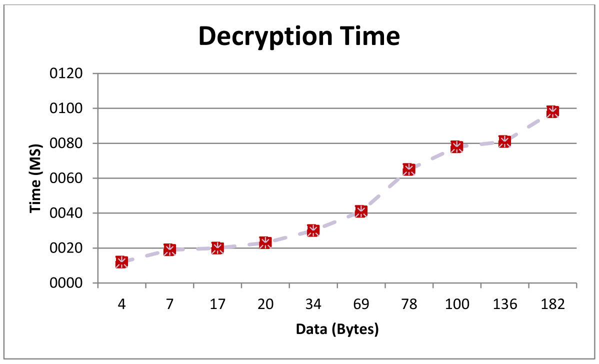


FIGURE 11. Decryption time for normal data.

C. DECRYPTION TIME

The total time used to transform cipher text into plain text is known as decryption time. In Figure 11 decryption time of PSDS is shown for normal data. Data is shown horizontally in bytes and time is shown vertically in ms. As shown in Figure 11, when the data is of 17 Bytes the decryption time is 0020 ms. In Figure 12 decryption time of sensitive data is shown. To decrypt 182 bytes PSDS takes 0100 ms.

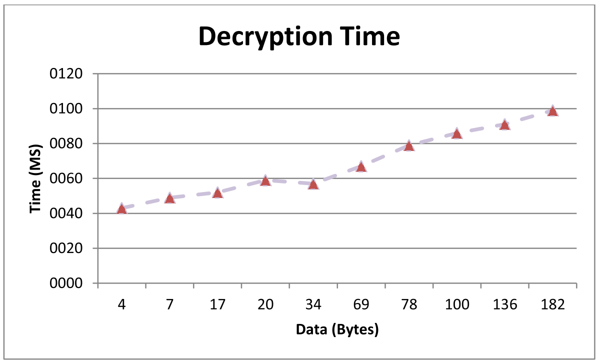


FIGURE 12. Decryption time for sensitive data.

D. COMPARATIVE ANALYSIS

In this portion comparison of PSDS is made with SA-EDS and AES [7] in respect of encryption time and decryption time of normal data and sensitive data. Figure 13 and figure 14 shows the encryption time comparison of PSDS and SA-EDS for normal data. Data is taken in KB and time is calculated in milliseconds. When data is 2 Kb, the encryption time of PSDS is 2334315 ms and in case of SA-EDS the encryption time for the same amount of data is 2009806 ms. When the data is 8 Kb, 11344318 ms is the encryption time for PSDS whereas the encryption time of SA-EDS for the same amount of data is 8765409 ms. As shown in figure 13 and figure 14, the encryption time of PSDS both in case of normal and sensitive data is less then AES, therefore, it can be concluded that the proposed PSDS approach is computationally efficient in comparison to AES [7].

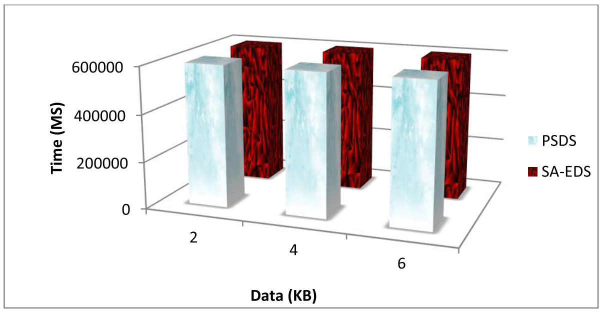


FIGURE 13. Decryption time for sensitive data.

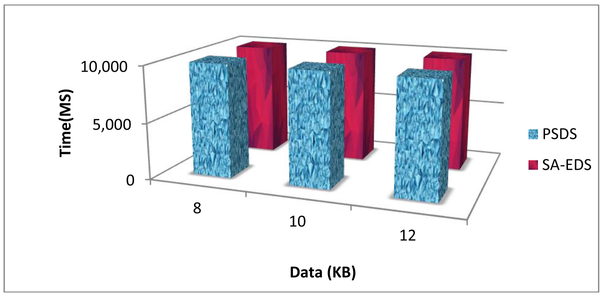


FIGURE 14. Encryption time for norma data PSDS VS. SA-EDS.

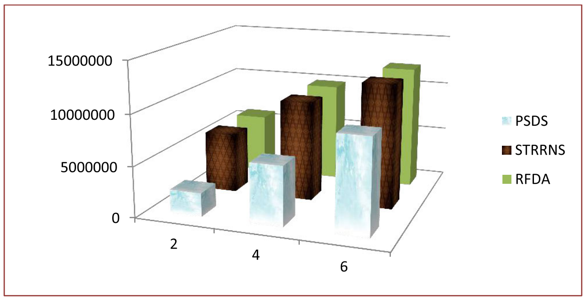


FIGURE 15. Comparison of encryption time for sensitive data of PSDS VS. SA-EDS.

Figure 15 and Figure 16 shows the encryption time comparison of PSDS and SA-EDS for sensitive data. Data is taken in Kb and time is calculated in milliseconds. When data is 3 Kb encryption time of PSDS is 4519652 ms and

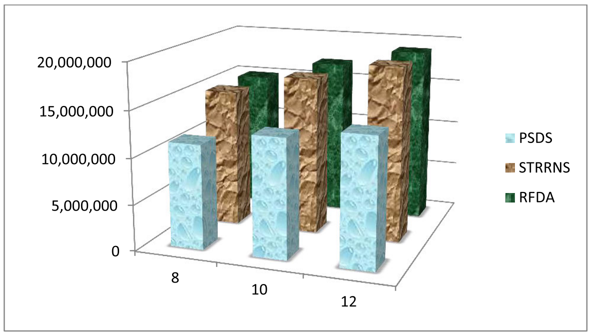


FIGURE 16. Comparison of encryption time for sensitive data of PSDS VS. SA-EDS.

for SA-EDS 3017218 ms. When data is 7 Kb, the encryption time is 9018432 ms for PSDS and for SA-EDS 6024241 ms. As shown in figure 15 and Figure 16 Encryption time of PSDS is greater than SA-EDS. As PSDS provides higher security as compared to SA-EDS.

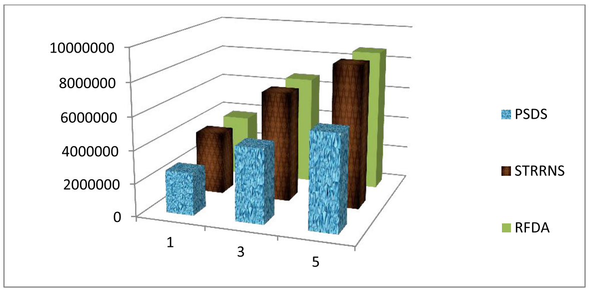


FIGURE 17. Comparison of encryption time of normal data.

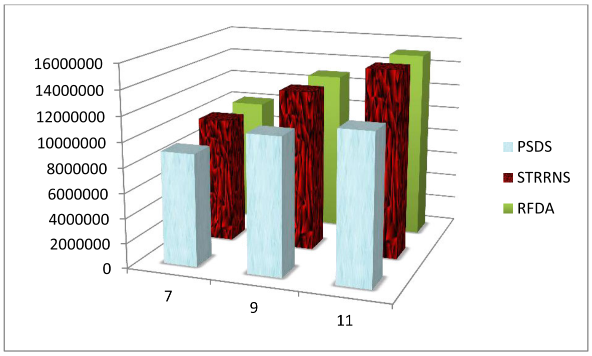


FIGURE 18. Comparison of encryption time of normal data.

Figure 17 and figure 18 shows the encryption time comparison of PSDS and STRRNS [11] and RFDA [6] for sensitive data. Data is taken in KB and time is calculated in milliseconds. When data is 3 kb encryption time of PSDS is 4519652 MS and for STRRNS [11] and RFDA [6] encryption time is 6710930 MS. When data is 7 kb encryption time is 9018432 MS for PSDS and for STRRNS [11] and RFDA [6] 10,089,233 MS. As shown is figure 22 and figure 23 encryption time of PSDS is less than STRRNS [11] and RFDA [6]. STRRNS [11] and RFDA [6] has greater encryption time.

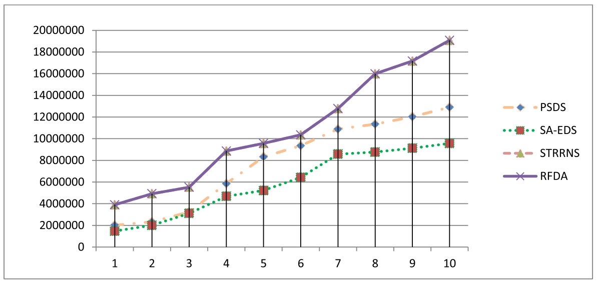


FIGURE 19. Comparison of encryption time of sensitive data.

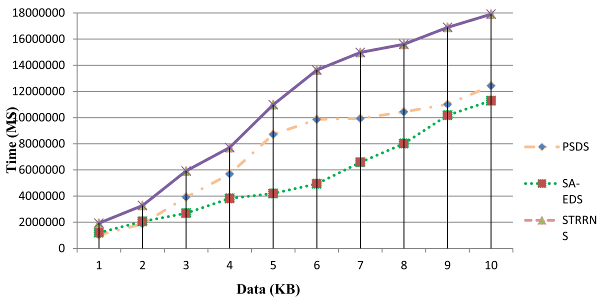


FIGURE 20. Comparison of encryption time of sensitive data.

Figure 19 and Figure 20 illustrates the comparison of PSDS, SA-EDS [8] and STRRNS [11] and RFDA [6]. The same input data size is used for PSDS, SA-EDS [8] and STRRNS [11] and RFDA [6]. In Figure 24 comparison of encryption, time is shown between PSDS, SA-EDS [8] and STRRNS [11] and RFDA [6] for normal data. When the encryption time of SA-EDS[8] for 2kb is 2009806 MS, the encryption time of PSDS is 2334315. For 6kb input size, PSDS requires 9,354,035 MS, while SA-EDS [8] requires 6428088 MS and STRRNS [11] and RFDA [6] needs 12250891 MS for encryption. For 10kb the encryption time of PSDS is 12,902,334 MS, the encryption time of SA-EDS[8] is 9,565,231 MS and STRRNS [11] and RFDA [6] takes 17078652 MS. The encryption time increases with an increase in data size. As shown in Figure 24 STRRNS [11] and RFDA [6] has the highest encryption time. SA-ESD [8] has the lowest encryption time due to which it is not suitable for providing data security. PSDS provides high security with less encryption time.

Figure 21 and Figure 22 illustrates the comparison of PSDS, SA-EDS [8] and STRRNS [11] and RFDA [6]. The same input data size is used for PSDS, SA-EDS [8] and STRRNS [11] and RFDA [6]. In Figure 26 comparison of encryption, time of sensitive data is shown between PSDS, SA-EDS [8] and STRRNS [11] and RFDA [6]. When the encryption time of SA-EDS[8] for 2kb is 2940321 MS, the encryption time of PSDS is 3460723. For 6kb input size, PSDS requires 7903112 MS, while SA-EDS [8] requires 5591313 MS and STRRNS [11] and RFDA [6] needs 9292486 MS for encryption. For 10kb the encryption time of PSDS is 11874645 MS, the encryption time of SA-EDS[8]

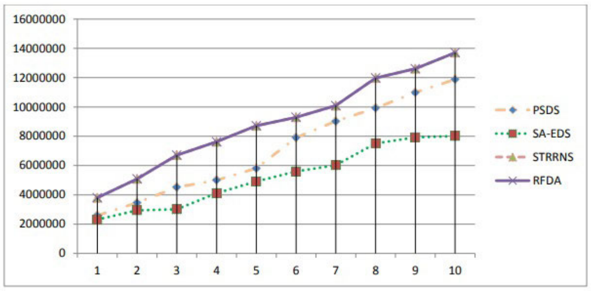


FIGURE 21. Comparison of decryption Time.

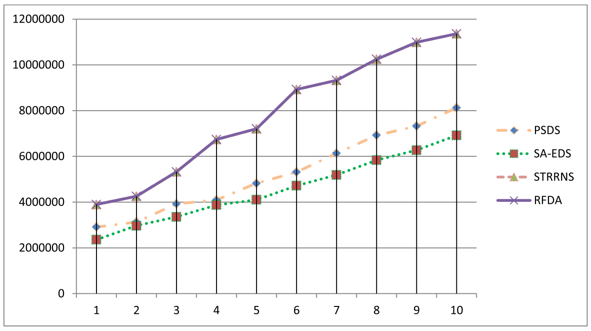


FIGURE 22. Comparison of decryption time.

is 8012314 MS and STRRNS [11] and RFDA [6] takes 13703242 MS. The encryption time increases with an increase in data size. As shown in Figure 25 STRRNS [11] and RFDA [6] has the highest encryption time. SA-ESD [8] has the lowest encryption time due to which it is not suitable for providing data security. PSDS provides high security with less encryption time.

In Figure 22 comparison of decryption, time for confidential data is shown between PSDA and SA-EDS [8] and STRRNS [11] and RFDA [6] When decryption time of SA-EDS[8] for 4kb is 3872404 MS, the decryption time of PSDS is 4093232 and for STRRNS [11] and RFDA [6] is 6739732 MS. For 8kb the decryption time of PSDS is 6924215 MS, the SA-EDS [8] takes 5836454 MS and STRRNS [11] and RFDA [6] requires 10246962 MS. Decryption time of STRRNS [11] and RFDA [6] is very high which make it unsuitable for data security.

Figure 23 and Figure 24 illustrates the comparison of PSDS, SA-EDS, STRRNS [11] and RFDA [6]. To test the behaviour of all the three techniques, the same input data size is used for PSDS, SA-EDS, STRRN and RFDA. In Figure23 comparison of encryption, time of sensitive data is shown between PSDS, SA-EDS and STTN. When the encryption time of SA-EDS for 2 KB is 2940321 ms, in the same case, the encryption time of PSDS is 3460723 ms. For 6 Kb input size, PSDS requires 7903112 ms, while SA-EDS requires 5591313 ms and STTN needs 9292486 ms for encryption. For 10 KB data size, the encryption time of PSDS is 11874645 ms, in the same case the encryption time of

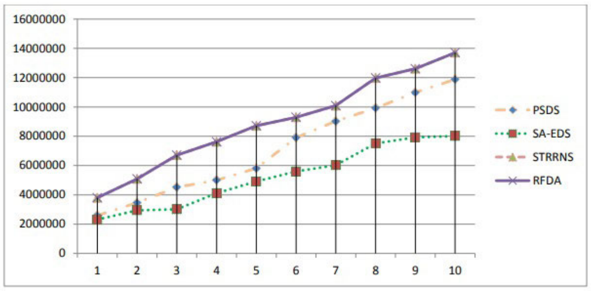


FIGURE 23. Encryption time comparison.

SA-EDS is 8012314 ms while that of STTN is 13703242 ms. The encryption time increases with an increase in data size. As shown in Figure 23 AES has the highest encryption time. Furthermore, SA-ESD is having less encryption time, however, it is proved to be highly vulnerable to pollution attack and is therefore, unsuitable for providing data security. It can be observed that PSDS encryption time is higher than SA-ESD however, it is more secure. Therefore, it can be concluded that PSDS provides high security with less encryption time.

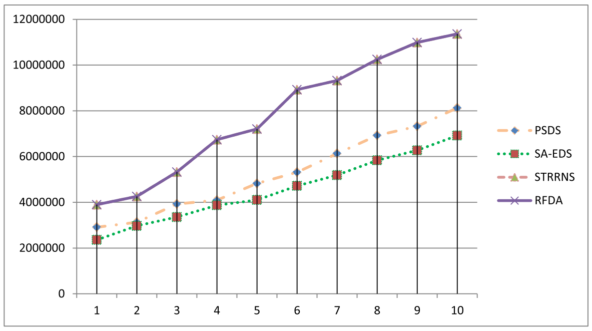


FIGURE 24. Decryption time comparison.

In Figure 24 comparison of decryption, time for confidential data is shown for PSDA, SA-EDS, RFD and STTN. It can be clearly seen from the figure that when decryption time of SA-EDS for a 4 Kb data is 3872404 ms, the decryption time of PSDS is 4093232 and for STTN it is 6739732 ms. For a data size of 8 Kb the decryption time of PSDS is 6924215 ms, the SA-EDS takes 5836454 ms while STTN requires 10246962 ms. It can observed from the figure that the decryption time of STTN and RFD grows very fast with the size of the data. Consequently, its decryption time is very high that makes it unsuitable for data security specially in case large data.

VII. SECURITY ANALYSIS

While uploading data over the cloud, there is the possibility of multiple attacks on our data. While providing a cryptographic algorithm, a measurement should be taken to avoid attacks over the network.

A. CHOSEN CIPHERTEXT ATTACK

In the chosen-ciphertext attack, the attacker has part of the ciphertext. The attacker uses this part of ciphertext to recover the plain text by decrypting this ciphertext. The attacker can also try to recover the secret key using the chosen ciphertext. The scenario of the Chosen Ciphertext attack is shown in Figure 25.

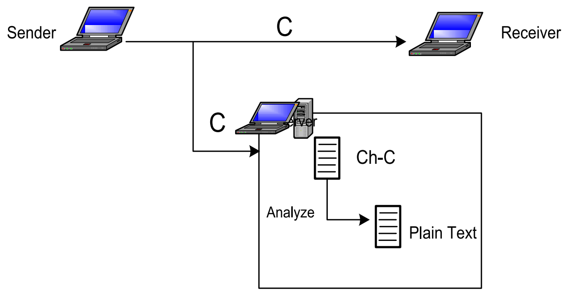


FIGURE 25. Chosen cipher-text attack scenario.

In the PSDS it is not possible to recover data from the chosen ciphertext. As the text value does not remain the same. As shown in Table 2 the value of C is not always converted in G. Their value varies with the text bit. So it is impossible to obtain the plain text from the chosen ciphertext.

B. KNOWN PLAIN TEXT ATTACK

Known Plain text attack is a very basic attack as an attacker knows both plain text and ciphertext. When data is sent for encryption, the attacker captures the part of plain text. The attacker has also knowledge of part of the ciphertext. The key is passed over a secure medium so the attacker is unaware of the key. The attacker uses a chunk of plain text and cipher text to recover plain text. The scenario of a known plain text attack is shown in Figure 26.

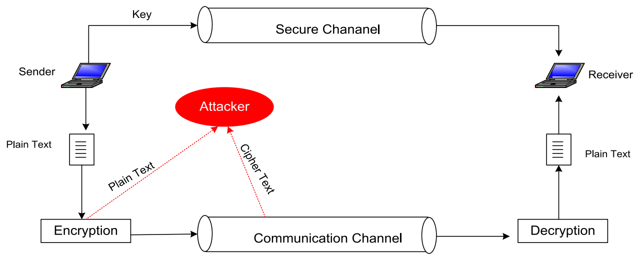


FIGURE 26. Known plain text attack scenario.

In PSDS plain text is not sent over the network. The only ciphertext is shared over the network. If an attacker successfully gets access to part of ciphertext, he'll not be able to obtain plain text as plain text is not shared over the network.

C. RELATED-KEY ATTACK

The Related-key Attack is such a type of attack in which the attacker knows some mathematical relationship of the key where initial values are unknown for the attacker and can observe the cipher under several keys. For example, the attacker knows that last bits are always the same but have not the knowledge of first bits. In Figure 27 Related-key attack scenario is shown.

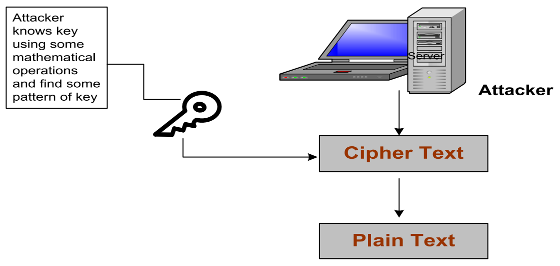

Attacker knows key using some mathematical operations and find some pattern of key
Server
Attacker
Cipher Text
Plain Text

FIGURE 27. Related-key attack scenario.

In PSDS, on every time, the new key is generated for encryption. In a case, the attacker finds a key, he is not able to decrypt all files, and he can only decrypt one encrypted file.

D. POLLUTION ATTACK

Pollution attack is such type of attack in which malicious node floods the infected packet in the network which prevents the receiver to decode the ciphertext. As shown in Table SA-EDS [7] is vulnerable to pollution attack as SA-EDS[8] uses XOR to encrypt plain text. The scenario of pollution Attack is shown in Figure 28.

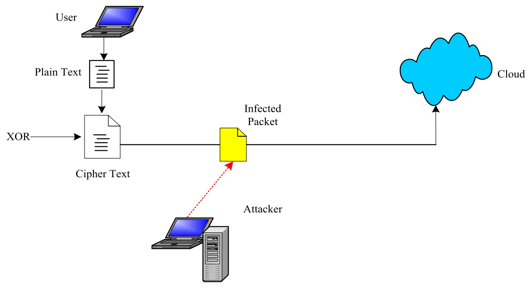


FIGURE 28. Scenario of pollution attack.

In PSDS, we use a Rail Fence, Circular shift, DNA algorithm, and S-box to encrypt plain text. XOR is vulnerable to pollution attacks. We do not use XOR in PSDS to avoid pollution attack.

E. MAN-IN-MIDDLE ATTACK

In cryptography, the man-in-the-middle is an attack where an unauthorized attacker positions himself between the two communicating parties and secretly intercepting their communication while the sender and receiver believe that they are communicating with each other. Man (attacker) in the middle is able to capture secrete conversation between two parties without their knowledge. The scenario of a man-in-middle attack is shown in Figure 29.

To test the resilience of the proposed PSDS approach against man-in-the-middle attack, the attack was launched

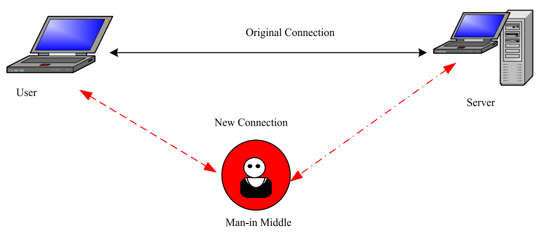


FIGURE 29. Man-in-middle attack.

using on the proposed approach using Wireshark. Due to space limitation, the screen shorts cannot be shown here. However, it can be clearly observed from the packet snipped through wireshark that the packets are moving in encrypted form and it is almost impossible for any attacker to recover the contents of the packet even if the attacker intercepts the traffic. Therefore, it can be concluded that the proposed PSDS prevents man-in-middle attack.

VIII. CONCLUSION

As technology is increasing very rapidly, the number of data raises terrifically. Cloud computing becomes common among people, as people can easily save their huge amount of data in order to save memory consumption. The major issue in storing data on clouds is data security. There is a need to transfer data into ciphertext. Multiple approaches are used to secure data over the cloud. Computational time is focused on data security approaches. A complex algorithm is not suitable for data security due to their increasing computational time. The less complex algorithm has security issues. In this paper, we propose PSDS in order to solve the issue. We divide the data in normal and sensitive part. Normal data is encrypted and uploaded over a single cloud while sensitive data is divided into two parts, then encryption steps are applied on these two halves and uploaded on separate clouds. At the time of downloading, these two separate halves are merged and the decryption algorithm is applied in order to obtain plain text. The proposed approach is secure against chosen ciphertext, known the plain text, related-key attack, pollution attack, and main-in middle attack.

REFERENCES

[1] P. P. Kumar, P. S. Kumar, and P. J. A. Alphonse, "Attribute based encryption in cloud computing: A survey, gap analysis, and future directions," *J. Netw. Comput. Appl.*, vol. 108, pp. 37–52, Apr. 2018, doi: 10.1016/j.jnca.2018.02.009.

[2] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Gener. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[3] S. Ramgovind, M. M. Eloff, and E. Smith, "The management of security in cloud computing," in *Proc. Inf. Secur. South Africa*, Aug. 2010, pp. 1–7.

[4] S. Chandra, B. Mandal, S. S. Alam, and S. Bhattacharyya, "Content based double encryption algorithm using symmetric key cryptography," *Procedia Comput. Sci.*, vol. 57, pp. 1228–1234, Jan. 2015.

[5] R. F. Olanrewaju, B. U. I. Khan, A. Baba, R. N. Mir, and S. A. Lone, "RFDA: Reliable framework for data administration based on split-merge policy," in *Proc. SAI Comput. Conf. (SAI)*, Jul. 2016, pp. 545–552.

[6] D. P. Yellamma, D. B. C. Narasimham, and M. T. Kumar, "Cloud computing security using secret sharing algorithm over singleto multi-clouds," *Latest Res. Eng. Manag.*, vol. 1, pp. 1–6, Apr. 2016.

[7] Y. Li, K. Gai, L. Qiu, M. Qiu, and H. Zhao, "Intelligent cryptography approach for secure distributed big data storage in cloud computing," *Inf. Sci.*, vol. 387, pp. 103–115, May 2017.

[8] B. S. Rawal, V. Vijayakumar, G. Manogaran, R. Varatharajan, and N. Chilamkurti, "Secure disintegration protocol for privacy preserving cloud storage," *Wireless Pers. Commun.*, vol. 103, no. 2, pp. 1161–1177, Nov. 2018.

[9] O. Zibouh, A. Dalli, and H. Drissi, "Cloud computing security through parallelizing fully homomorphic encryption applied to multi-cloud approach," *J. Theor. Appl. Inf. Technol.*, vol. 87, no. 2, pp. 300–307, 2016.

[10] K. Subramanian and F. L. John, "Secure and reliable unstructured data sharing in multi-cloud storage using the hybrid crypto system," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 6, pp. 196–206, 2017.

[11] S. Bogos, J. Gaspoz, and S. Vaudenay, "Cryptanalysis of a homomorphic encryption scheme," *Cryptogr. Commun.*, vol. 10, no. 1, pp. 27–39, Jan. 2018.

[12] B. Ul Islam Khan, A. M. Baba, R. F. Olanrewaju, S. A. Lone, and N. F. Zulkurnain, "SSM: Secure-split-merge data distribution in cloud infrastructure," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Aug. 2015, pp. 40–45.

[13] R. Mudgal and M. K. Bhatia, "International journal of engineering sciences & research technology enhancing data security using encryption and splitting technique over multi-cloud environment," *Eng. Sci. Res. Technol.*, vol. 7, no. 8, pp. 440–449, 2018.

[14] V. Balasaraswathi and S. Manikandan, "Enhanced security for multi-cloud storage using cryptographic data splitting with dynamic approach," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, May 2014, pp. 1190–1194.

[15] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection of fault-based side-channel cryptanalysis of 128-bit symmetric block ciphers," in *Proc. 38th Conf. Design Autom. DAC*, 2001, pp. 579–584.

[16] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2009, pp. 1–18.

[17] A. El-Yahyaoui and M. D. Elkettani, "Cryptanalysis of fully homomorphic encryption schemes," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 5, p. 677, 2016.

[18] A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems," *J. Netw. Comput. Appl.*, vol. 59, pp. 208–218, Jan. 2016.

[19] O. Acıiçmez, W. Schindler, and C. C. K. Koç, "Cache based remote timing attack on the AES," in *Proc. Cryptographers' Track RSA Conf.* Berlin, Germany: Springer, 2007, pp. 271–286.

[20] A. Tchernykh, M. Babenko, V. Miranda-Lopez, A. Y. Drozdov, and A. Avetisyan, "WA-RRNS: Reliable data storage system based on multi-cloud," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2018, pp. 666–673.

[21] G. Alvarez, F. Montoya, M. Romera, and G. Pastor, "Cryptanalysis of a chaotic encryption system," *Phys. Lett. A*, vol. 276, nos. 1–4, pp. 191–196, Oct. 2000.

[22] M. Ananthi, R. Sabitha, S. Karthik, and J. Shanthini, "FSS-SDD: Fuzzy-based semantic search for secure data discovery from outsourced cloud data," *Soft Comput.*, pp. 1–10, Jan. 2020, doi: 10.1007/s00500-020-04701-5.

[23] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full aes," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2011, pp. 344–371.

[24] R. Dixit, A. Gaikwad, O. Pal, M. Tawaskar, and A. Sankpal, "Heterogeneous-cloud for improving cloud data security," *Int. J.*, vol. 3, no. 6, pp. 13–19, 2018.

[25] O. Kara and C. Manap, "A new class of weak keys for blowfish," in *Proc. Int. Workshop Fast Softw. Encryption*. Berlin, Germany: Springer, 2007, pp. 167–180.

[26] S. Vaudenay, "On the weak keys of blowfish," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 1039, D. Gollmann, Ed. Berlin, Germany: Springer, 1996.

[27] A. Sakr, E. Yaacoub, H. Noura, M. Al-Husseini, K. Abualsaud, T. Khattab, and M. Guizani, "A secure client-side framework for protecting the privacy of health data stored on the cloud," in *Proc. IEEE Middle East North Afr. Commun. Conf. (MENACOMM)*, Apr. 2018, pp. 1–6.

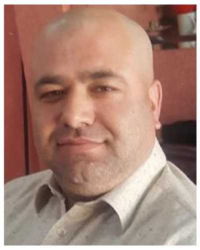
[28] M. Sulochana and O. Dubey, "Preserving data confidentiality using multi-cloud architecture," *Procedia Comput. Sci.*, vol. 50, pp. 357–362, Jan. 2015.

[29] Y. Yarom, D. Genkin, and N. Heninger, "CacheBleed: A timing attack on OpenSSL constant-time RSA," *J. Cryptograph. Eng.*, vol. 7, no. 2, pp. 99–112, Jun. 2017.

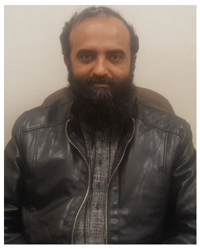
[30] A. Takayasu and N. Kunihiro, "Partial key exposure attacks on RSA: Achieving the boneh-durfee bound," in *Proc. Int. Conf. Sel. Areas Cryptogr.* Cham, Switzerland: Springer, 2014, pp. 345–362.
[31] Y. Wang, H. Zhang, and H. Wang, "Quantum polynomial-time fixed-point attack for RSA," *China Commun.*, vol. 15, no. 2, pp. 25–32, Feb. 2018.
[32] L. Harn and C. Lin, "Detection and identification of cheaters in (t, n) secret sharing scheme," *Des., Codes Cryptogr.*, vol. 52, no. 1, pp. 15–24, Jul. 2009.
[33] M. R. Reddy, R. Akilandeswari, S. Priyadarshini, B. Karthikeyan, and E. Ponmani, "A modified cryptographic approach for securing distributed data storage in cloud computing," in *Proc. Int. Conf. Netw. Adv. Comput. Technol. (NetACT)*, Jul. 2017, pp. 131–139.
[34] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing XOR network coding against pollution attacks," in *Proc. IEEE INFOCOM 28th Conf. Comput. Commun.*, Apr. 2009, pp. 406–414.
[35] S. A. Chaudhry, T. Shon, F. Al-Turjman, and M. H. Alsharif, "Correcting design flaws: An improved and cloud assisted key agreement scheme in cyber physical systems," *Comput. Commun.*, vol. 153, pp. 527–537, Mar. 2020.
[36] A. Ghani, K. Mansoor, S. Mehmood, S. A. Chaudhry, A. U. Rahman, and M. N. Saqib, "Security and key management in IoT-based wireless sensor networks: An authentication protocol using symmetric key," *Int. J. Commun. Syst.*, vol. 32, no. 16, 2019, Art. no. e4139.
[37] M. Ul Hassan, S. A. Chaudhry, and A. Irshad, "An improved SIP authenticated key agreement based on dongqing et al.," *Wireless Pers. Commun.*, vol. 110, no. 4, pp. 2087–2107, Feb. 2020.
[38] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1996, pp. 104–113.
[39] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1993, pp. 386–397.
[40] E. Biham and A. Shamir, "Differential cryptanalysis of the full 16-round DES," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1992, pp. 487–496.
[41] V. Karuvandan, S. Chellamuthu, and S. Periyasamy, "Cryptanalysis of AES-128 and [AES]-256 block ciphers using lorenz information measure," *Int. Arab J. Inf. Technol.*, vol. 13, no. 6B, pp. 1054–1060, 2016.
[42] M. H. Alsharif, A. H. Kelechi, K. Yahya, and S. A. Chaudhry, "Machine learning algorithms for smart data analysis in Internet of Things environment: Taxonomies and research trends," *Symmetry*, vol. 12, no. 1, p. 88, Jan. 2020.
[43] A. Kannan, "Proficient public substantiation of data veracity for cloud storage through dual protection," *Int. J. Sci. Res. Comput. Sci., Eng. Inf. Technol.*, vol. 3, no. 2, pp. 107–117, 2018.
[44] P. Suwansrikham and K. She, "Asymmetric secure storage scheme for big data on multiple cloud providers," in *Proc. IEEE IEEE 4th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2018, pp. 121–125.

FIZZA SHAHID received the B.S. degree in computer science from the University of Poonch Rawalakot, Azad Jammu and Kashmir, in 2016. She is currently pursuing the M.S. degree in computer science with the Department of Computer Science and Software Engineering, International Islamic University Islamabad, Pakistan. She is working towards her master's thesis. Her broad research interests include security of cloud technologies, information security, and cryptographic protocols.

HUMAIRA ASHRAF received the Ph.D. degree in computer science (with majors in cellular mobile networks and wireless networks). She is currently an Assistant Professor with the Department of Computer Science and Software Engineering, International Islamic University Islamabad, Pakistan. She has worked with Sardar Bahadur Khan Women University, Quetta, for ten years. She is an Experienced Researcher and affiliated with teaching profession for the past 14 years.



ANWAR GHANI received the B.S. degree in computer science from the University of Malakand, Pakistan, in 2007, and the M.S. and Ph.D. degrees in computer science from the Department of Computer Science and Software Engineering, International Islamic University Islamabad, in 2011 and 2016, respectively. He has worked as a Software Engineer with Bioman Technologies, from 2007 to 2011. He was selected as an Exchange Student under the EURECA Program, in 2009, for VU University, Amsterdam, The Netherlands, and EXPERT Program, in 2011, for Masaryk University, Czech Republic, funded EUROPEAN Commission. He is currently a Faculty Member with the Department of Computer Science and Software Engineering, International Islamic University Islamabad. His broad research interests include wireless sensor networks, next-generation networks, information security, and energy efficient collaborative communication.



SHAHBAZ AHMED KHAN GHAYYUR received the Master Computer Science Degree from International Islamic University Islamabad, in 2007, and the Ph.D. degree in computer sciences from Preston University Islamabad, Pakistan, in 2018. He is currently serving as an Assistant Professor with the Department of Computer Science and Software Engineering, International Islamic University Islamabad. He has authored various articles in International Journals and conferences across the globe. His research interests include software security, software requirements, software process engineering, software paradigms, could and Internet computing. He also served as an Editor and a Co-Editor for various international journals.

SHAHABODDIN SHAMSHIRBAND received the M.Sc. degree in artificial intelligence from Iran, and the Ph.D. degree in computer science from UM, Malaysia, in 2014.

He was an Adjunct Assistant Professor with the Department of Computer Science, Iran University Science and Technology. He also severed as a Senior Lecturer with the University of Malaya, Malaysia, and he assigned as a Senior Lecturer with Islamic Azad University, Iran. He participated in many research programs within the Center of Big Data Analysis at IUST and IAU. He was associated with Young researchers and elite club, from 2009 till now. He has supervised and co-supervised undergraduate and postgraduate students (Master and Ph.D.) by research and training. He is also the author and a coauthor of papers published in IF journals and attended to high-rank A and B conferences. He is a Professional Member of IEEE and ACM. He is an Associate Editor, a Guest Editor, and a Reviewer of high-quality journals and conferences.

ELY SALWANA received the M.A. degree in information technology (management) from the University of Technology Malaysia, in 2005, and the Ph.D. degree in computer science from the University of Malaya, in 2016. She is currently a Research Fellow with the Institute of IR4.0, Universiti Kebangsaan Malaysia (UKM), Bangi, Malaysia. Her primary research interests include information work, data analytics, knowledge organization and participatory information practices. The contexts of her research ranges from public organization, education, virtual worlds, and corporate. She has 11 years of working experience related to her field of study in university as a Lecturer, a Researcher, and a Supervisor for undergraduate and postgraduate student. She has involved in many research projects and grants that related to computer sciences especially in her area of interest as the Group Leader and a Researcher, and all the projects are successfully delivered on time and followed project schedule.