International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

An Efficient Reconfigurable Client-Side Prior Cryptography Algorithm for Data Security in Smart Cloud Computing System

Nithyashree B*[1], Dayananda P[2]

Submitted:14/03/2024 **Revised**: 29/04/2024 **Accepted**: 06/05/2024

Abstract: In recent years, a huge number of different formats of data have been communicated between source and destination. During this process, authentication plays a major role when different formats of data communication are used between the source and destination. It has been performed in association with cryptology and cryptanalysis, and a few more techniques have been performed during cryptography, such as blending words with images and microdots. In parallel, it operated on the masking and unmasking processes during storage and transit. Due to the smart device communication process, a huge amount of data has been stored in the cloud. During this process, more data modification has occurred during sign-up for desired services, which can cause unwanted data, and instructions have been executed on the sensitive data. To overcome the above-mentioned details, a proposed reconfigurable client-side prior cryptography algorithm (RCSPCA) has been implemented for data uploading and decrypting after downloading on the client-side using a key generated during encryption. The proposed algorithm shows better security on large files and varied format information with respect to the time consumption of the encryption algorithm of 0.56% as compared to the conventional methods by considering 4000 file sizes and with respect to the time consumption of the decryption algorithm of 0.26% as compared to the conventional methods by considering 4000 file sizes. The key value is also worked out by a different algorithm in this programme. As a result, our algorithm makes big files more secure and runs faster. So, we can add an extra layer of security that will stop unwanted strikes on private information and stop people from not following the same rules.

Keywords: *encryption, cryptography, cloud computing, standardization, cloud storage*

1. Introduction

People have a sizeable quantity of confidential information that has to be effectively managed. This information can be personal or corporate, which requires protection against malicious sites and ensures availability to only the relevant parties. This is achieved through cryptography. Throughout history, data has been protected and passed on only to relevant people through various practices. For example, the Greek aristocrats passed messages through tattoos on a slave's bald head, which are hidden by the hair that grows back. Earlier than the Greeks, the Egyptians in 1900 BC were the first to secure their important messages from outsiders through the use of cryptic messages, which can be read only by the relevant persons [1]. The Caesar cypher method [2] devised by the Roman military around 100 BC to send encrypted information to its army general is another such example. The first encryption method using a key came into existence in the mid-1400s. Various techniques resembling this original technique by Vigenère have been seen in the latter half of the 13th century, using multiple key characters leading to a modulo of 26 encrypted texts [3]. In spite of its cumbersome procedure, this method was widely popular for several centuries to protect data from irrelevant people. Though scientific methods of cryptography have achieved data security, such methods were developed only at the beginning of the 1800s. Further, with the advent of electricity, the utility of electrical and electromechanical devices for various purposes became the norm. At such a time, Hebern came up with a device known as the rotor machine, which inserted a secret key by rotating its disc. When this hidden key is operated on or activated, it generates encrypted text [4]. During World War II, cryptography advanced to a vast extent. After the war, encryption and decryption of data have been widely used by corporate businesses to protect their valuable information from competing businesses. The applications of cryptography have enormously expanded in the information age due to the increasing digital exchange of data and the necessity to protect digitally transmitted and received data. This age of emails and cloud storage of confidential data has seen more than its necessary share of data loss, piracy, and infringement. The various corporate and public sectors of the world are in need of a viable solution to effective data protection. Cryptography is the only viable technology that can effectively protect data on platforms that require digital transactions of data or on platforms that make use of shared data usage, such as in cloud computing. A popular example of cloud computing is Google Drive, which provides a platform to share and update common database data among

[1] *Department of Computer Science and Engineering, JSS Academy Of Technical Education, Bangalore -560060, Visvesvaraya Technological University, Belagavi, Karnataka – 590018, India*
ORCID ID: 0009-0008-4943-3137
[2] *Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, Karnataka 576104, India*
ORCID ID: 0000-0001-8445-3469
** Corresponding Author Email: nithi3231@gmail.com*

a number of users [5] through the use of the internet. Due to the innumerable advantages of such cloud computing services as Google Drive, they are widely used by developers, business entrepreneurs, academies, etc. Some of its many advantages include flexibility, data access irrespective of time and location, the availability of updated synchronised data among multiple users, etc. Cryptography involves encrypting this vast shared data before storage to ensure its protection [6].

One of the main services provided by cloud facilities such as Google Drive is the provision of shared memory, which not only provides flexibility and easy access to multiple users but also reduces the memory consumption of the users on their personal computer hard disk [7-10]. This third-party storage of data is always open to risk from various outside sources; hence, companies such as Google provide extensive security for user data through various layers of encryption in all their various products. The general security protocol followed by Google during data storage includes dividing the data into different parts, each of which is encrypted with a different encryption key, and each of these different parts carries its own ID necessary for identification. This aids in both privacy and the protection of data. Data is encrypted prior to transmission from the user; this uses the transport layer security encryption TLS. Similarly, data is encrypted after reception before storage through the advanced encryption standard AES, which uses a 256-bit encryption format [11-15]. Client-side cryptography is a process of encrypting data on the client side, typically within a web browser or a mobile app, before transmitting it to a server as shown in figure.1.

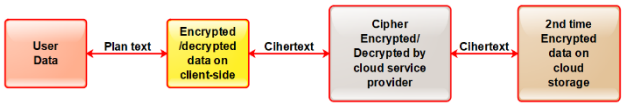

Fig.1 fundamental block diagram of client-side cryptography process

This approach enhances data security and privacy, as sensitive information is encrypted locally on the user's device before it leaves their control [16]. Here's a general overview of the client-side cryptography process:

 Key Generation:

The process begins with generating encryption keys. Typically, two types of keys are used: a public key and a private key for asymmetric encryption, or a shared secret key for symmetric encryption.

 Encryption:

Depending on the chosen encryption method (symmetric or asymmetric), the client-side code encrypts the data using the generated keys. Here are the two primary methods:

a. Symmetric Encryption:

In symmetric encryption, the same key is used for both encryption and decryption. The client generates a shared secret key, encrypts the data with it, and keeps the key secret.

b. Asymmetric Encryption:

In asymmetric encryption, there are two keys: a public key (used for encryption) and a private key (used for decryption). The client uses the recipient's public key to encrypt the data, ensuring only the recipient, who has the corresponding private key, can decrypt it.

 Data Transmission:

The encrypted data is then transmitted over the network to the server. Since the data is encrypted, even if intercepted during transit, it is virtually impossible to decipher without the corresponding decryption key.

 Data Storage:

On the server, the encrypted data is stored securely. The server does not have access to the plaintext data, as only the client holds the decryption key.

 Data Retrieval:

When the client needs to access the data, it requests the encrypted data from the server.

 Decryption:

The client-side code decrypts the data using the appropriate decryption key (private key for asymmetric encryption or shared secret key for symmetric encryption). Once decrypted, the data can be used by the client. It's important to note that client-side cryptography places a significant responsibility on the client-side application to securely manage keys and perform encryption and decryption operations. If the client's device is compromised, the security of the encrypted data can be at risk [17-20]. Popular libraries and tools for implementing client-side cryptography in web applications include WebCrypto API, OpenSSL, and various JavaScript libraries for encryption, such as CryptoJS and Stanford Javascript Crypto Library (SJCL). Additionally, mobile app development platforms provide APIs and libraries for implementing client-side cryptography in mobile applications. Client-side cryptography is essential for enhancing the security and privacy of user data, especially in scenarios where trust in the server or network cannot be assumed [21]. Similar to Google Drive, Amazon also provides cloud services via its Amazon Storage service, which provides seamless large-scale data sharing among multiple users. The user data is stored through Amazon S3 versioning to provide date protection, and the data is stacked and stored in the Amazon S3 bucket [22]. Similar to Google, Amazon also provides encryption at two levels: one is before transmission from the client side, called client-side encryption, and the other is to

ensure the security of data in storage, called server-side encryption [23]. Various intrusions, cyber-attacks, data losses, data leaks, etc. have been seen in cloud computing in recent times [24-28]. In order to prevent these attacks, a two-step encryption process is adopted, where the data is encrypted before transmission to the cloud services and after transmission during storage. The proposed method gives us a novel idea of encrypting and decrypting the data before transmission at the user end through the encryption key at the user end. This is done twice to ensure error-free and safe data passage from the user to the storage device or the cloud service

2. Related Works

Various methods to ensure data protection during transmission and reception have been devised, including cryptographic encryption methods. Symmetric key cryptography is a technique where both the sender and the receiver use the same key to encrypt and decrypt the data. This form of symmetry key cypher is a recent improvement or advancement on Alberti's poly-alphabetic cypher [29]. Due to improved internet connectivity and flexible, continuous, and easy access to the internet, cloud computing has emerged as a popular choice for various businesses and personal users. Balachandra et al. [30] provide a weighted insight into the different drawbacks and shortcomings of cloud storage and cloud computing. They also present the merits of service-level agreements (SLA), the challenges of SLA, the need for improved data protection, and the different policies for security used at its various levels. Dr. L. Arokhaim et al. [31] propose an algorithm to encrypt user data in the cloud, which provides improved data security and privacy. Niteen et al. [32] propose an algorithm that makes use of secure authentication to provide safe storage of user data in the cloud. This method is based on AES encryption. Srinivas et al. [33] present a survey on the various methods used for data security in cloud computing and shared data environments; they also provide insights on methods for improvement and possible techniques. BM Shereek [34] makes use of Fermat's little theorem method to design an open-key RSA through the use of prime numbers. Aldossary et al. [35] present in their paper a list of attacks and downfalls in cloud data storage, and they further present a set of methods to tackle each of the listed attacks and downfalls in cloud computing. Rachna et al. [36] present a cloud computation security measure using RSA and AES methods. The computation speed requirements for each of these methods are tested and compared. Tania et al. propose this method to improve client-side protection by combining the Diffie-Helmann algorithm and the AES algorithm. [37] They further test their method and show improved performance.

Nesrine et al. [38] propose a method to provide security for user data through a novel meta-data user authentication method, where the client data is encrypted using a separate key for each individual record. This improves client-side security through the repetition model. Mahdul et al. [39] make use of a one-dimensional matrix to integrate the hash algorithm with the AES algorithm. This method proposed by Mahdul provides security for data in storage and also during storage or upload. Prerna et al. [40] propose a unique localised user data encryption by the user prior to transmission, which provides an optional auxiliary data security mechanism to overcome data loss or theft during transmission.

Various cloud service providers provide security through various methods and techniques of data encryption and cryptography. Though the safety of the client data is ensured through these cloud services such as Amazon, Vivo, OwnCloud, Tresorit, etc., the privacy of the client data is a concern as these cloud services encrypt the client data using their own encryption technique, and the key for the encryption is also stored on their own servers. Choo et al. [41] throw light on the concern for the privacy of data stored using cloud services in their paper, where they discuss the need for improvement in the quality of data encryption provided by the cloud service providers. Further, Zargari et al. [42] present the challenges and concerns of cloud data storage. As the system data and the various user's data are stored together on a common storage platform, this is a concern. The system data is shared by all the different users, while the user data in the cloud is not shared. In the event that a malicious file finds its way into this common storage present in the cloud, it could corrupt all the user's data stored in it. Another concern presented in the paper is the fact that different cloud service providers follow their own encryption methods, which lack a generalised structure or procedure for encryption. For example, some service providers provide end-to-end encryption, while others do not. This huge storage space is a major concern in cloud computing.

Keiko et al. [43] present another security concern in cloud computing in their paper, where they discuss the lack of encryption of data when the data is transferred from one location to another within their own storage, causing data loss, leaks, and intrusions, which further increase the risk of IPR theft and loss of privacy. Dr. Subarna Shakya [44] proposes the use of SSL along with a transfer ticket to ensure safe data transfer within the cloud. The data transfer ticket in the SSL provides improved security. Pandian et al. [45] propose a memory-aware method that recognises the previous related data stored in the cloud and differentiates it from the outliers. The HHAR algorithm is extended to perform this memory-oriented data storage.

2.1. Conflict-Based Search (CBS) algorithm:

Conflict-Based Search (CBS) is a popular and effective algorithm used in the field of robotics and artificial

intelligence for solving multi-agent pathfinding (MAPF) problems. MAPF involves finding collision-free paths for multiple agents (robots, vehicles, or entities) in a shared environment while taking into account their individual goals and constraints. CBS is specifically designed to address such scenarios and is widely used in various applications, including robotics, video games, and traffic management [46].

CBS is a two-level search algorithm that decomposes the MAPF problem into two main phases: a high-level search and a low-level search. In the high-level search phase, CBS explores the space of possible paths for each agent independently and concurrently. Each agent's path is represented as a sequence of waypoints. The high-level search aims to find a set of non-conflicting paths for all agents while optimizing a cost function, such as the sum of travel times.

Once the high-level search identifies a set of individual agent paths, the low-level search comes into play. The low-level search focuses on resolving conflicts between agent paths. A conflict occurs when two or more agents attempt to occupy the same location at the same time. The low-level search aims to resolve these conflicts by adjusting the paths of the involved agents without violating their goals and constraints. CBS employs various conflict resolution strategies to handle conflicts. Some common conflict resolution techniques include prioritizing agents, delaying agents, swapping their positions, or dynamically adjusting their paths to avoid collisions. The choice of conflict resolution strategy depends on the specific application and requirements.

To improve the efficiency of CBS, heuristic functions are often used to guide the high-level search. These heuristics estimate the cost of reaching the goal from different states and help the algorithm make informed decisions about which agents to prioritize during conflict resolution. CBS is known for its completeness, meaning it can find a solution if one exists. However, it might not always guarantee optimality, as it might find a suboptimal solution in some cases due to the search space's complexity. The performance of CBS depends on various factors, including the number of agents, the size of the environment, the complexity of the goals and constraints, and the efficiency of the low-level search and conflict resolution strategies. CBS is a powerful algorithm for addressing multi-agent pathfinding problems and has been applied in real-world scenarios such as autonomous vehicle navigation, warehouse logistics, and multi-robot systems. Researchers continue to develop and refine CBS and its variants to tackle more complex and dynamic multi-agent scenarios efficiently.

2.2. User end Data Encryption for Efficient Cloud Service

Effective user data protection at the user end before transmission can be achieved through a symmetric key method. This kind of a user end encryption provides data safety from both external intrusions and from cloud service providers. The symmetric key algorithm is implemented at SaaS layer. This user-end data encryption also provides an authentication procedure which ensures data safety both during transmission and in storage. The user-end data encryption method stores the symmetry key used for user data encryption locally at the user side [47]. This local storage of encryption key keeps the data safe secure from the Cloud service provider. In order to decrypt these encrypted data stored in the cloud it has to be downloaded by the user and the downloaded data is further decrypted using the locally stored symmetric key. Due to this procedure which decrypts data only after download, the data is protected even during transfer within the cloud from one memory location to another. In addition to this the user-end data encryption method utilizes SSL secure sockets layer which forms encryption during each data transmission which provides additional security during cloud computing [48].

3. Functional work flow:

The proposed algorithm is designed to commensurate consistent service to the users along with easy anytime anywhere access and operation to them [49]. This proposed algorithm is implemented in an Amazon S3 bucket using eclipse IDE in a java Platform. The functional work flow of the proposed user-end data encryption algorithm is depicted in Figure 2.

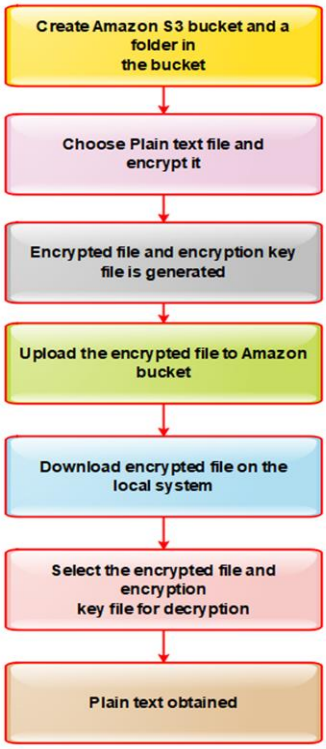

Fig 2 proposed functional work flowchart

3.1. Encryption algorithm:

Encryption algorithms are mathematical procedures or processes used to transform plaintext (unencrypted data) into ciphertext (encrypted data) to protect it from unauthorized access or interception [50]. These algorithms are a fundamental component of modern cryptography and are used in various applications, including data security, communication security, and digital privacy [51]. Here are some commonly used encryption algorithms:

3.1.1. Encryption algorithm:

 AES (Advanced Encryption Standard):

AES is one of the most widely used symmetric encryption algorithms. It supports key lengths of 128, 192, or 256 bits and is considered highly secure.

 DES (Data Encryption Standard):

DES was once a widely used encryption standard, but it is now considered obsolete due to its small key size. Triple DES (3DES) is a more secure variant that applies DES encryption three times with different keys.

 RC4:

RC4 is a stream cipher that was widely used in the past but has been deprecated due to vulnerabilities. It is no longer recommended for secure communications.

 Blowfish and Twofish:

These are symmetric block ciphers designed for fast encryption and are considered secure alternatives to DES.

3.1.2. Asymmetric Key Encryption Algorithms:

 RSA (Rivest-Shamir-Adleman):

RSA is a widely used asymmetric encryption algorithm for securing data transmission and digital signatures. It relies on the mathematical properties of large prime numbers and is widely used in SSL/TLS for securing web traffic.

 ECC (Elliptic Curve Cryptography):

ECC is another asymmetric encryption algorithm known for its strong security and efficiency. It is often used in resource-constrained environments like mobile devices and IoT.

 Diffie-Hellman Key Exchange:

While not an encryption algorithm itself, Diffie-Hellman is a key exchange protocol that allows two parties to securely agree on a shared secret key. It is often used in combination with symmetric encryption for secure communication.

 Hybrid Encryption:

Many secure communication systems combine both symmetric and asymmetric encryption for efficiency and security. In such systems, data is encrypted with a symmetric key, and the symmetric key itself is encrypted using asymmetric encryption. This combines the efficiency of symmetric encryption with the key distribution benefits of asymmetric encryption.

 Homomorphic Encryption:

This specialized encryption technique allows computations to be performed on encrypted data without the need to decrypt it first. It is used in privacy-preserving applications where sensitive data needs to be processed without exposing the plaintext.

 Post-Quantum Cryptography Algorithms:

As quantum computing advances, traditional encryption algorithms may become vulnerable. Post-quantum cryptography algorithms are being developed to resist attacks by quantum computers. Examples include lattice-based cryptography, code-based cryptography, and hash-based cryptography. The proposed algorithm has been shown in below:

Encryption algorithm_1:

Input: plain text

Output: cipher text, encryption key

Step_1: read ASCI value for single character

Step_2: convert ASCI to Binary

Step_3: if (value!=8bit)

Step_4: then add preceding 0’s

Step_5: else slice into 4bit

Step_6:swap positons

Step_7:slice 4bit into 2bit

Step_8:swap positons

Step_9: then reverse 8bit value

Step_10:convert Binary to ASCI value

Step_11: add keyValue

Step_12: return encryption key

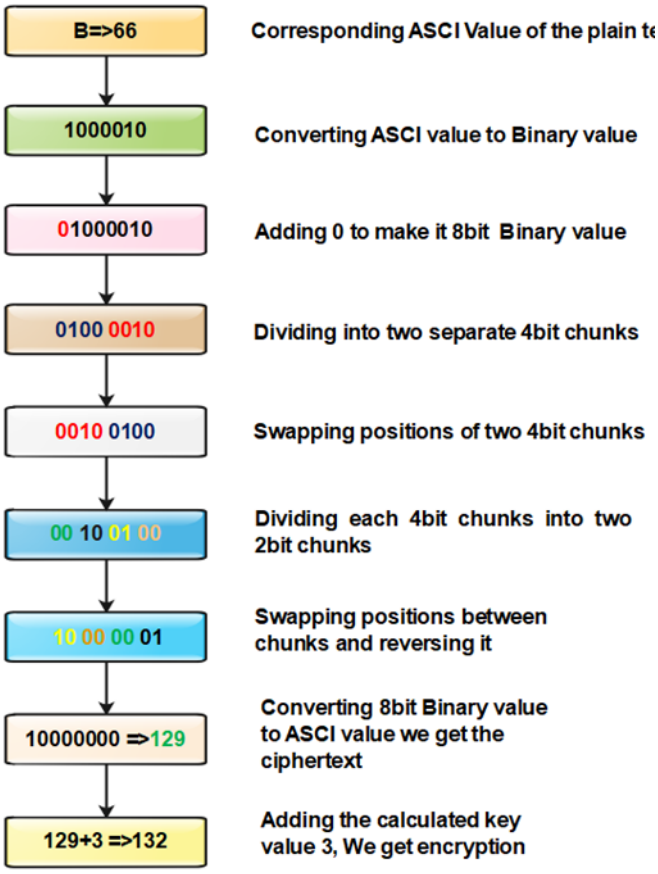

Fig 3 proposed encryption process algorithm

3.2. Key value calculation

Key value calculation can refer to several different contexts, depending on the specific domain or application. Here are a few common interpretations of key value calculation:

 Data Structures:

In the context of data structures like dictionaries, maps, or associative arrays, key-value calculation typically refers to retrieving or updating the value associated with a given key. You provide a key (e.g., a string or an identifier), and the data structure returns the corresponding value.

 Cryptography:

In encryption and decryption processes, key value calculation involves using cryptographic keys to transform plaintext into ciphertext or vice versa. The calculation could be as simple as XORing each byte of the plaintext with a corresponding byte of the key in a symmetric encryption scheme or using more complex mathematical operations in asymmetric encryption.

 Financial Calculations:

Key value calculation in finance often involves calculating the present or future value of an investment, loan, or financial instrument based on a set of parameters, including interest rates, time periods, and initial values. For example, you might calculate the future value of an investment using the formula for compound interest.

 Machine Learning and Statistics:

In machine learning and statistical modeling, key value calculation can refer to determining the importance or weight of a specific feature (key) within a dataset. This can be done through various methods such as feature selection or feature engineering.

 Database Queries:

In database systems, key-value calculation may involve querying a database using a key (e.g., a primary key or a unique identifier) to retrieve the corresponding values stored in the database tables.

 Hash Functions:

Key value calculation can also be related to hash functions, where a key is input into a hash function, and the function produces a hashed value (often a fixed-length string or number) that represents the original key. This is commonly used in data structures like hash tables for fast key-value lookups. Key value calculation algorithm_2 as shown in below

Algorithm_2

Input: Amazon S3 bucketName

Output: keyValue

Step_1: i=0, value=0;

Step_2: c= buscketName.length;

Step_3: while (i<=c)

Step_4: then value+=ASCI value[i]

Step_5: i++

Step_6: end while

Step_7: keyValue=value mod c

Step_8: return keyValue

3.3. Decryption algorithm

A decryption algorithm is a set of instructions or a mathematical process used to convert encrypted data back into its original, unencrypted form. The specific decryption algorithm used depends on the encryption method employed. Some common encryption algorithms and their corresponding decryption algorithms include:

3.3.1. Symmetric Key Encryption:

 Algorithm:

In symmetric key encryption, the same key is used for both encryption and decryption. Common symmetric encryption algorithms include DES (Data Encryption Standard), AES (Advanced Encryption Standard), and 3DES (Triple Data Encryption Standard).

 Decryption Process:

To decrypt data encrypted with a symmetric key, you apply the inverse of the encryption algorithm using the same key. This process typically involves reversing the transformations performed during encryption.

 Asymmetric Key Encryption:

Asymmetric key encryption uses a pair of keys: a public key for encryption and a private key for decryption. Common asymmetric encryption algorithms include RSA and ECC (Elliptic Curve Cryptography). To decrypt data encrypted with an asymmetric key, you use the private key corresponding to the public key used for encryption. The decryption process is different from encryption and involves mathematical operations based on the keys' mathematical properties.

 Hash Functions:

Hash functions are one-way functions that take input data and generate a fixed-size hash value. Hashing is not technically decryption, but it's worth mentioning because it's commonly used in cryptography. Hash functions are designed to be irreversible, so there is no direct decryption process. However, in some cases, attackers may attempt to find a collision (two different inputs producing the same hash) or use precomputed tables (rainbow tables) to reverse certain hash functions.

 Stream Ciphers and Block Ciphers:

Stream ciphers and block ciphers are used in various encryption methods. Stream ciphers encrypt data one bit or byte at a time, while block ciphers encrypt data in fixed-size blocks. Decryption for stream ciphers and block ciphers involves applying the inverse of the encryption algorithm using the appropriate key. The specific steps vary depending on the cipher used.

 Homomorphic Encryption:

Homomorphic encryption is a specialized form of encryption that allows computations to be performed on encrypted data without decryption. It's used in scenarios where data privacy is crucial. Homomorphic encryption doesn't require traditional decryption because computations can be performed directly on encrypted data. However, results can be decrypted if needed, using the private key. The choice of encryption and decryption algorithms depends on the specific security requirements and use cases. It's essential to use strong encryption methods and protect the decryption keys to ensure the security of sensitive data. The algorithm.3 shows the decryption algorithm,

Algorithm_3

Input: cipher text, encryption key

Output: plain text

Step_1: c=ASCI value of cipher text

Step_2: e= ASCI value of encryption text

Step_3: read c and e

Step_4: subtract keyValue from e

Step_5: if (value != c)

Step_6: then stop decryption

Step_7: else convert ASCI to Binary

Step_8: reverse binary value

Step_9: slice into two 4bit chunks

Step_10: again slice into two 2bit chunks

Step_11: swap positions between two 2bit chunks

Step_12: again swap positions between two 4bit chunks

Step_13: append chunks into binary value

Step_14: convert Binary into ASCI value

Step_15: return plain text

The work flow is initialized through the creation of a new folder within the Amazon S3 bucket and the Amazon S3 bucket. This folder acts as a storage to store the encrypted files. Once an input comprising of a text file is given, the input text file is chosen for encryption. The proposed Java algorithm converts this plain text into an encrypted file and an encryption key is generated along with it. The encryption key is stored in the local folder at the user end. After completion of encryption the cipher text is sent to the server through the Amazon bucket. Upon uploading the cipher text, it is encrypted once again at the server side before storage in the Amazon bucket folder. In order to retrieve this text file, the encrypted file is first downloaded, then this file is decrypted using the encryption key which is stored in local storage. After decryption the original plain text required, is received by the user. The plain text file is retrieved while ensuring data security during transmission and storage in the Amazon S3 bucket. The proposed Data Encryption method is also designed to ensure data safety against threats such as virus attacks etc. and prevents data corruption and data loss [52].

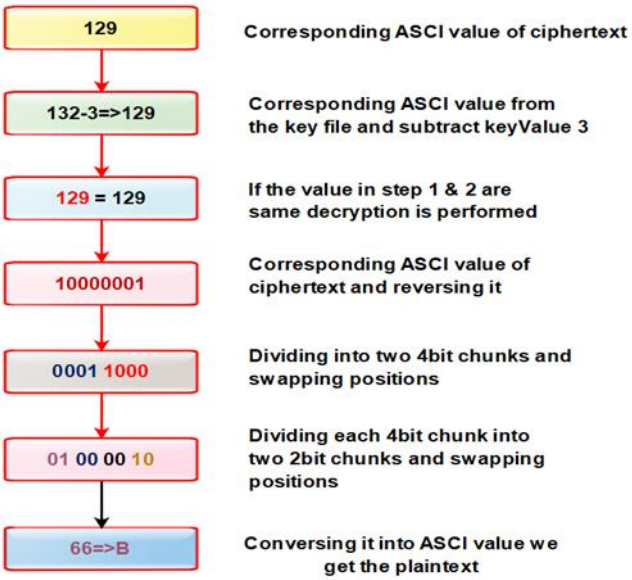

Fig.4 proposed decryption process algorithm

3.4. Amazon S3

The proposed encryption algorithm is implemented on the Amazon S3 bucket due to its user-friendly framework which provides a reliable and fast storage platform with reasonable price along with user friendly installations such as its ability to increment storage capacity. This storage provides seamless connection and data storage facilities to a number of Amazon services worldwide [53]. Amazon S3 also provides services to application peripheral interfaces (APIs) by providing access to create and manage their own buckets. Amazon S3 has provisions to create a maximum of 100 buckets in a single AWS account. The user can further increase the storage capacity through an agreement on service limit increase which further increases the capacity to 1000 buckets within a single AWS account. There is no limit to the number of objects that can be stored in each bucket. Each Amazon S3 bucket is created with a username and an AWS region through which the user wishes to operate. Flexible operation of various buckets with in various operating regions is an added advantage for the Amazon S3. The bucket is identified by a unique name [54-56] this name is also referred to as the key name of the bucket. The data is stored inside the bucket in the form of objects and can be accessed through the key name. The key name is the identifying factor which is necessary to access the Amazon storage services. The key name is also necessary to link it to the local storage in order to decrypt the cypher text stored in the Amazon S3 buckets.

4. Experimental Analysis

The cloud storage services provided through various commercial service providers is a very useful option for data storage to a number of global users. Security of data is the most important issue during communication and transmission of data. Apart from security the data encryption method used by the service provider also requires to be fast without latency to provide a seamless and flexible data usage to multiple users through the world wide web. The proposed to algorithm for data encryption is implemented on a Java Platform and its performance with respect to security and speed of encryption and data viability is tested and compared with existing methods. The user-end data encryption method utilizes a swapping technique which optimizes the algorithm and avoids looping every data file there by saving time for encryption. Proposed algorithm is tested and compared with the existing CBS algorithm [57-65] for various character sets with sizes of 1000, 2000 and 3000 characters each.

4.1. Encryption Algorithm Performance Comparison

Apart from the most crucial data security required, a commercial data storage service has to be user friendly and provide data transmission, reception and data storage services at a good data rate failing which the users will not opt for such a service. Faster data uploading and retrieval requires a faster rate of data encryption and decryption. The rate of data encryption and decryption for the user-end data encryption method is shown in table 1. The table elaborates the time taken by the user-end data encryption method to encrypt a text file with varying size of characters and generate an encryption key which is stored in the local storage, and upload the encrypted file to the cloud.

Table.1 parameter for proposed encryption algorithm

Number of characters	CBS Algorithm Execution time (Execution time)	Proposed algorithm (Execution time)
2000	2113ms	1875ms
3000	3218ms	2901ms
4000	4021ms	3750ms

The time taken to complete the encryption process using the proposed algorithm and the CBS algorithm is plotted in milliseconds Figure.5, this presents the comparison in performance using the user-end data encryption method and the existing CBS method. from the graph, the proposed user-end data encryption algorithm completes the encryption process much faster compared to the existing CBS algorithm. This shows the more efficient encryption process of the proposed method

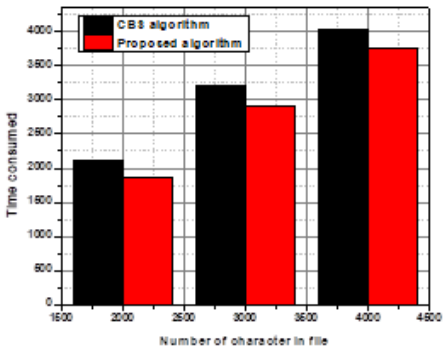

Fig 5 performance analysis of encryption algorithm between proposed and conventional methods

4.2. Decryption algorithm performance comparison

The proposed algorithm decrypts the cipher text downloaded by the user using the encryption key stored in local storage at the user end. Before decryption the proposed algorithm verifies whether the downloaded file is free from corruption by comparing it with the encryption key. Improved performance of the user-end data encryption algorithm is achieved through an optimum decryption.

The time taken by the user-end data encryption algorithm to decrypt the cipher text downloaded by the user into a plain text message is represented in milliseconds in Table 2. The time taken by the existing CBS algorithm to complete the decryption process is compared with the user-end data encryption algorithm.

Table.2 parameter for proposed decryption algorithm

Number of characters	CBS Algorithm Execution time (Execution time)	Proposed algorithm (Execution time)
2000	2356ms	2001ms
3000	3505ms	3085ms
4000	4125ms	4025ms

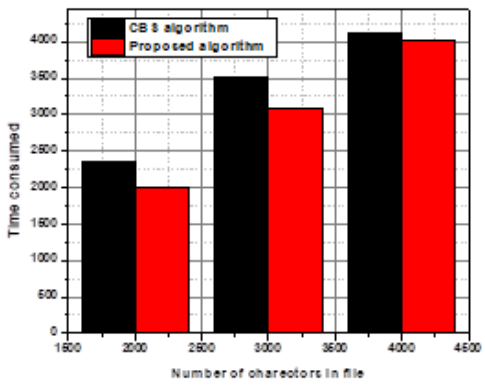

Fig 6 performance analysis of decryption algorithm between proposed and conventional methods

The time taken to complete the decryption process by the user-end data encryption algorithm and the existing CBS algorithms are plotted in Figure 6. This provides comparison of decryption performance of the two algorithms. From the graph it is noticed that the decryption performance of the user-end data encryption algorithm is much faster compared to the existing CBS algorithm. Along with reduced time of operation the proposed decryption algorithm ensures that the downloaded data is not corrupted and provides efficient data storage with security and data retrieval to the user.

5. Proposed Client-side Cryptography Based Security upload and download latency

Proposed Client-side cryptography is a security approach where encryption and decryption of data take place on the client-side (user's device) rather than on the server-side. This adds an extra layer of security to data in transit and at rest, as only the user with the decryption keys can access the data. However, client-side cryptography can introduce latency to the upload and download processes due to the computational overhead of encryption and decryption.

5.1. Encryption Overhead:

When a user uploads data to a server, the data must be encrypted on the client-side before transmission. Encryption can be computationally intensive, especially for large files, which can lead to increased upload times. Similarly, when downloading encrypted data from the server, the client-side must decrypt it, which adds processing time to the download process.

5.2. Key Management:

Managing encryption keys securely on the client-side is essential. Key generation, storage, and retrieval can introduce some latency. The client must also retrieve the decryption key before it can decrypt downloaded data. While not directly related to client-side cryptography, network latency (e.g., due to internet speed, server response times) can compound with encryption overhead, potentially leading to longer upload and download times. Some client-side cryptographic libraries and hardware acceleration can help mitigate latency by performing encryption and decryption tasks in parallel with other processes. This can help maintain a reasonable level of performance even with encryption overhead. Modern hardware, such as hardware security modules (HSMs) or specialized cryptographic co-processors, can significantly reduce the latency introduced by encryption and decryption processes. The choice of encryption algorithms can impact latency. Some algorithms are faster but may be less secure, while others are more secure but computationally intensive. The selection of algorithms should consider the trade-off between security and performance. Caching decrypted data on the client-side can help reduce latency for subsequent access to the same data, as there is no need to decrypt it again. Figure.7 shows

the performance analysis between proposed and conventional method with respect upload and download latency

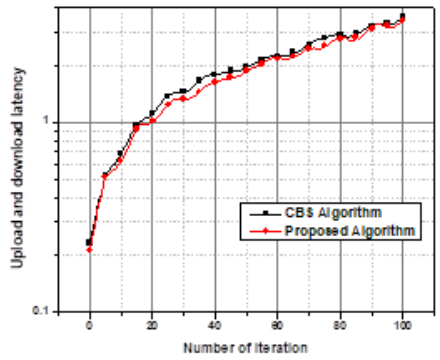

Fig 7 Performance analysis of upload and download latency between proposed and conventional methods

Proposed reconfigurable client-side prior cryptography algorithm for Data Security in smart Cloud computing system is a security approach that focuses on securing data on the client-side (user's device) rather than relying solely on server-side security measures. This approach involves encrypting and decrypting data on the client device before transmitting it to a server or after receiving it from a server.

The core of client-side cryptography is data encryption. When data is generated or processed on the client device, it is encrypted using strong cryptographic algorithms and keys. This ensures that even if the data is intercepted in transit or stored on a server, it remains confidential. Client-side cryptography often employs end-to-end encryption, where data is encrypted on the client device and only decrypted on the recipient's device. This means that even service providers or server operators cannot access the plaintext data, enhancing privacy. In client-side cryptography, users typically have control over the encryption keys. This ensures that the user, and not the service provider, holds the keys required to decrypt the data. As a result, the user has greater control over their data. Proper key management is crucial in client-side cryptography. Users need a secure way to generate, store, and back up encryption keys. Secure key management practices, including hardware-based solutions, are often employed to protect keys from unauthorized access. Strong user authentication is essential to ensure that only authorized users have access to the encrypted data. This can involve password-based authentication, multi-factor authentication (MFA), or other secure login mechanisms. Cryptography not only provides confidentiality but also ensures data integrity. Cryptographic hashes and signatures can be used to verify the integrity of data both in transit and at rest. Client-side cryptography allows users to access and work with their data even when they are offline. This is important for applications that need to function without a continuous internet connection.

Client-side cryptography is a critical component of secure communication and data protection in various applications, including web browsers, messaging apps, and cloud storage services. When communicating with a server, client-side cryptography can secure the transmission of data, making it resistant to eavesdropping and man-in-the-middle attacks. Implementations of client-side cryptography often need to work across various platforms and devices, such as web browsers, mobile apps, and desktop applications. While security is paramount, usability is also a critical factor in client-side cryptography. Developers must strike a balance between strong security measures and a user-friendly experience. In the event of key loss or device failure, client-side encryption systems may provide mechanisms for data recovery, often through secure key backup and retrieval processes. Client-side cryptography is commonly used in secure messaging apps, file storage services, and applications that handle sensitive user data proposed analysis between proposed and conventional methods as shown in figure.8. It provides users with a higher degree of control and privacy over their data, but it also requires careful implementation and ongoing management of encryption keys to ensure security

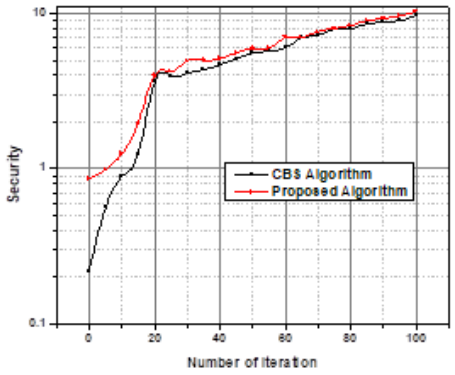

Fig 8 Performance analysis of security between proposed and conventional methods

6. Conclusion

In this paper, a proposed reconfigurable client-side prior cryptographic algorithm (RCSPCA) is effectively operated on different formats of data communication between source and destination with highly authenticated processes, and it is very difficult to identify the unauthorised person. Hence, the proposed cryptographic process operates on a 4000-file size. As per the process analysis, the proposed algorithm has reduced the data upload and download time by 0.65% as compared to the conventional methods. The proposed algorithm shows better security of 0.34% on larger files with varied format information as compared to the conventional methods. The proposed algorithm provides better security for large files and information in different formats. This is

shown by the fact that the encryption algorithm takes 0.56% less time than traditional methods when 4000 file sizes are taken into account, and the decryption algorithm takes 0.26% less time than traditional methods when 4000 file sizes are taken into account. In this programme, the key number is also determined by a different algorithm. Because of this, our algorithm makes big files safer and easier to run. So, we can add an extra layer of security that will stop unwanted attacks on private information and stop people from not following the same rules.

References

[1] N. Kabir and S. Kamal, "Secure Mobile Sensor Data Transfer using Asymmetric Cryptography Algorithms," 2020 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 2020, pp. 1-6, doi: 10.1109/ICCWS48432.2020.9292392.

[2] K. Gai, M. Qiu and H. Zhao, "Security-Aware Efficient Mass Distributed Storage Approach for Cloud Systems in Big Data," 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), New York, NY, USA, 2016, pp. 140-145, doi: 10.1109/BigDataSecurity-HPSC-IDS.2016.68.

[3] K. T. Priya and V. Karthick, "A Non Redundant Cost Effective Platform and Data Security in Cloud Computing using Improved Standalone Framework over Elliptic Curve Cryptography Algorithm," 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2022, pp. 1249-1253, doi: 10.1109/ICSCDS53736.2022.9761002.

[4] H. A. Farouk and M. Saeb, "An improved FPGA implementation of the modified hybrid hiding encryption algorithm (MHHEA) for data communication security," Design, Automation and Test in Europe, Munich, Germany, 2005, pp. 76-81 Vol. 3, doi: 10.1109/DATE.2005.58.

[5] A. Kumar, "Data Security and Privacy using DNA Cryptography and AES Method in Cloud Computing," 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2021, pp. 1529-1535, doi: 10.1109/I-SMAC52330.2021.9640708.

[6] "IEE Colloquium on 'Security and Cryptography Applications to Radio Systems' (Digest No.1994/141)," IEE Colloquium on Security and Cryptography Applications to Radio Systems, London, UK, 1994, pp. 0_1-.

[7] S. Al-Ghamdi and H. Al-Sharari, "Improve the security for voice cryptography in the RSA algorithm," 2022 International Conference on Business Analytics for Technology and Security (ICBATS), Dubai, United Arab Emirates, 2022, pp. 1-4, doi: 10.1109/ICBATS54253.2022.9759016.

[8] M. Shaar, M. Saeb, M. Elmessiery and U. Badawi, "A hybrid hiding encryption algorithm (HHEA) for data communication security," 2003 46th Midwest Symposium on Circuits and Systems, Cairo, Egypt, 2003, pp. 476-478 Vol. 1, doi: 10.1109/MWSCAS.2003.1562321.

[9] Y. Ma, Y. Zhao, Z. Zhang and J. Wang, "Distributed Data Multi-Level Storage Encryption Method Based on Full-Flow Big Data Analysis," 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2023, pp. 664-668, doi: 10.1109/EEBDA56825.2023.10090798.

[10] A. Anand, A. Raj, R. Kohli and V. Bibhu, "Proposed symmetric key cryptography algorithm for data security," 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Greater Noida, India, 2016, pp. 159-162, doi: 10.1109/ICICCS.2016.7542294.

[11] M. -Q. Hong, P. -Y. Wang and W. -B. Zhao, "Homomorphic Encryption Scheme Based on Elliptic Curve Cryptography for Privacy Protection of Cloud Computing," 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), New York, NY, USA, 2016, pp. 152-157, doi: 10.1109/BigDataSecurity-HPSC-IDS.2016.51.

[12] M. J. Dubai, T. R. Mahesh and P. A. Ghosh, "Design of new security algorithm: Using hybrid Cryptography architecture," 2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, India, 2011, pp. 99-101, doi: 10.1109/ICECTECH.2011.5941965.

[13] R. S. Shukla, "IoT Based Designing of Secure Data Storage System in Distributed Cloud System with Big Data using Cryptography Algorithm," 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 2022, pp. 264-270, doi: 10.1109/SMART55829.2022.10047177.

[14] W. Xu, H. Liang and Y. Ge, "Research on Data Security Protection System Based on SM Algorithm,"

2021 International Conference on Information Science, Parallel and Distributed Systems (ISPDS), Hangzhou, China, 2021, pp. 79-82, doi: 10.1109/ISPDS54097.2021.00022.

[15] Y. Zhang, "Research on the Application of Computer Big Data Technology in Information Security Management," 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2023, pp. 124-128, doi: 10.1109/EEBDA56825.2023.10090574.

[16] R. S. Kumar, E. Pradeep, K. Naveen and R. Gunasekaran, "Enhanced Cost Effective Symmetric Key Algorithm for Small Amount of Data," 2010 International Conference on Signal Acquisition and Processing, Bangalore, India, 2010, pp. 354-357, doi: 10.1109/ICSAP.2010.13.

[17] R. R. Prasad and A. Kumari, "Cloud Data Security using Balanced Genetic Algorithm," 2023 9th International Conference on Electrical Energy Systems (ICEES), Chennai, India, 2023, pp. 132-137, doi: 10.1109/ICEES57979.2023.10110211.

[18] Hashem Mohammed Alaidaros, M. F. A. Rasid, Mohamed Othman and Raja Syamsul Azmir Raja Abdullah, "Enhancing security performance with parallel crypto operations in SSL bulk data transfer phase," 2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, Penang, Malaysia, 2007, pp. 129-133, doi: 10.1109/ICTMICC.2007.4448620.

[19] A. Abduklimu, P. Yan, G. Wang, H. Liu and J. Xie, "A 5G-Based Big Data Security Access Processing Method and Device," 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2023, pp. 780-784, doi: 10.1109/EEBDA56825.2023.10090774.

[20] T. Ahmad, J. Hu and S. Han, "An Efficient Mobile Voting System Security Scheme Based on Elliptic Curve Cryptography," 2009 Third International Conference on Network and System Security, Gold Coast, QLD, Australia, 2009, pp. 474-479, doi: 10.1109/NSS.2009.57.

[21] G. N k and R. V, "Graph Theory Matrix Approach in Cryptography and Network Security," 2022 Algorithms, Computing and Mathematics Conference (ACM), Chennai, India, 2022, pp. 108-110, doi: 10.1109/ACM57404.2022.00025.

[22] Z. Zhou, Y. Tian, J. Xiong, J. Ma and C. Peng, "Blockchain-Enabled Secure and Trusted Federated Data Sharing in IIoT," in IEEE Transactions on Industrial Informatics, vol. 19, no. 5, pp. 6669-6681, May 2023, doi: 10.1109/TII.2022.3215192.

[23] M. Harini, K. P. Gowri, C. Pavithra and M. P. Selvarani, "A novel security mechanism using hybrid cryptography algorithms," 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE), Karur, India, 2017, pp. 1-4, doi: 10.1109/ICEICE.2017.8191910.

[24] E. M. R. Hamed, A. E. Taha and A. I. Hammoodi, "An Advanced Data Security Algorithm Using cryptography and DNA-Based steganography," 2014 24th International Conference on Computer Theory and Applications (ICCTA), Alexandria, Egypt, 2014, pp. 32-38, doi: 10.1109/ICCTA35431.2014.9521623.

[25] A. Bose, A. Kumar, M. K. Hota and S. Sherki, "Steganography Method Using Effective Combination of RSA Cryptography and Data Compression," 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichy, India, 2022, pp. 1-5, doi: 10.1109/ICEEICT53079.2022.9768402.

[26] H. Albataineh, M. Nijim and D. Bollampall, "The Design of a Novel Smart Home Control System using Smart Grid Based on Edge and Cloud Computing," 2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE), Oshawa, ON, Canada, 2020, pp. 88-91, doi: 10.1109/SEGE49949.2020.9181961.

[27] L. Zheng, Y. Hu and C. Yang, "Design and Research on Private Cloud Computing Architecture to Support Smart Grid," 2011 Third International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, 2011, pp. 159-161, doi: 10.1109/IHMSC.2011.109.

[28] A. Yusoff, I. S. Mustafa, S. Yussof and N. M. Din, "Green cloud platform for flood early detection warning system in smart city," 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW), Riyadh, Saudi Arabia, 2015, pp. 1-6, doi: 10.1109/NSITNSW.2015.7176406.

[29] Y. Lei and L. Zhang, "Construction of Smart City Information System Based on Cloud Computing and Internet of Things Technology," 2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2021, pp. 567-570, doi: 10.1109/ICISCAE52414.2021.9590807.

[30] C. Sivapragash, S. R. Thilaga and S. S. Kumar, "Advanced cloud computing in smart power grid," IET Chennai 3rd International on Sustainable Energy and Intelligent Systems (SEISCON 2012), Tiruchengode,

2012, pp. 1-6, doi: 10.1049/cp.2012.2238.

[31] Y. -D. Chen, M. Z. Azhari and J. -S. Leu, "Design and implementation of a power consumption management system for smart home over fog-cloud computing," 2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG), Yilan, Taiwan, 2018, pp. 1-5, doi: 10.1109/IGBSG.2018.8393553.

[32] L. Wieclaw, V. Pasichnyk, N. Kunanets, O. Duda, O. Matsiuk and P. Falat, "Cloud computing technologies in “smart city” projects," 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, Romania, 2017, pp. 339-342, doi: 10.1109/IDAACS.2017.8095101.

[33] S. Bertagna De Marchi, F. Ponci and A. Monti, "Design of a MAS as Cloud Computing Service to control Smart Micro Grid," IEEE PES ISGT Europe 2013, Lyngby, Denmark, 2013, pp. 1-5, doi: 10.1109/ISGTEurope.2013.6695381.

[34] B. Bitzer and E. S. Gebretsadik, "Cloud computing framework for smart grid applications," 2013 48th International Universities' Power Engineering Conference (UPEC), Dublin, Ireland, 2013, pp. 1-5, doi: 10.1109/UPEC.2013.6714855.

[35] P. Bellini, D. Cenni and P. Nesi, "A Knowledge Base Driven Solution for Smart Cloud Management," 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, USA, 2015, pp. 1069-1072, doi: 10.1109/CLOUD.2015.154.

[36] T. Rajeev and S. Ashok, "Operational Flexibility in Smart Grid through Cloud Computing," 2012 International Symposium on Cloud and Services Computing, Mangalore, India, 2012, pp. 21-24, doi: 10.1109/ISCOS.2012.23.

[37] M. Kumar, K. Dubey and R. Pandey, "Evolution of Emerging Computing paradigm Cloud to Fog: Applications, Limitations and Research Challenges," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021, pp. 257-261, doi: 10.1109/Confluence51648.2021.9377050.

[38] Yuanpeng Xie et al., "User privacy protection for cloud computing based smart grid," 2015 IEEE/CIC International Conference on Communications in China - Workshops (CIC/ICCC), Shenzhen, China, 2015, pp. 7-11, doi: 10.1109/ICCChinaW.2015.7961570.

[39] Y. Tyagi and A. Goyal, "Stacker: A Holistic Cloud Computing Based Framework for Smart Cities," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2020, pp. 827-830, doi: 10.1109/ICACCCN51052.2020.9362884.

[40] A. Mohanty, S. Samantaray, S. S. Patra, M. A. ahmad and R. K. Barik, "An Efficient Resource Management Scheme for Smart Grid Using GBO Algorithm," 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2021, pp. 593-598, doi: 10.1109/ESCI50559.2021.9396784.

[41] G. Shrimal and Sandeep, "Using Heterogeneous Cloud Computing to Manage Resources in Sustainable Cyber-Physical Systems," 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 2022, pp. 153-157, doi: 10.1109/SMART55829.2022.10047642.

[42] J. M. Navya, H. A. Sanjay and K. Deepika, "Securing smart grid data under key exposure and revocation in cloud computing," 2018 3rd International Conference on Circuits, Control, Communication and Computing (I4C), Bangalore, India, 2018, pp. 1-4, doi: 10.1109/CIMCA.2018.8739496.

[43] P. Kumar K, S. Itagi, N. C, A. S. S and S. N. N, "Intelligent Transport System using Cloud Computing & PSY Key Generation for V2V Communication," 2022 Fourth International Conference on Cognitive Computing and Information Processing (CCIP), Bengaluru, India, 2022, pp. 1-5, doi: 10.1109/CCIP57447.2022.10058634.

[44] B. Bitzer and T. Kleesuwan, "Cloud-based Smart Grid monitoring and controlling system," 2015 50th International Universities Power Engineering Conference (UPEC), Stoke on Trent, UK, 2015, pp. 1-5, doi: 10.1109/UPEC.2015.7339938.

[45] S. Kumar, N. M. G. Kumar, B. T. Geetha, M. Sangeetha, M. K. Chakravarthi and V. Tripathi, "Cluster, Cloud, Grid Computing via Network Communication Using Control Communication and Monitoring of Smart Grid," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 1220-1224, doi: 10.1109/ICACITE53722.2022.9823552.

[46] U. Sakthi and J. D. Rose, "Smart Agricultural Knowledge Discovery System using IoT Technology and Fog Computing," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2020, pp. 48-53, doi: 10.1109/ICSSIT48917.2020.9214102.

[47] X. Zheng, S. Xue, H. Cao, F. Wang and M. Zhang, "A Cost-efficient Smart IoT Device Controlling System Based on Bluetooth Mesh and Cloud Computing,"

2020 Chinese Automation Congress (CAC), Shanghai, China, 2020, pp. 3374-3379, doi: 10.1109/CAC51589.2020.9326634.

[48] P. Wang, L. T. Yang and J. Li, "An Edge Cloud-Assisted CPSS Framework for Smart City," in IEEE Cloud Computing, vol. 5, no. 5, pp. 37-46, Sep./Oct. 2018, doi: 10.1109/MCC.2018.053711665.

[49] Ling Zheng, Shuangbao Chen, Yanxiang Hu and Jianping He, "Applications of cloud computing in the smart grid," 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), Dengleng, 2011, pp. 203-206, doi: 10.1109/AIMSEC.2011.6010461.

[50] J. Tong, Z. Li and Z. Qiao, "Online Legal Cloud Computing Sharing Application for Smart Medical System Management," 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2022, pp. 1290-1293, doi: 10.1109/ICSCDS53736.2022.9760713.

[51] S. S. Vellela, B. Venkateswara Reddy, K. K. Chaitanya and M. V. Rao, "An Integrated Approach to Improve E-Healthcare System using Dynamic Cloud Computing Platform," 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2023, pp. 776-782, doi: 10.1109/ICSSIT55814.2023.10060945.

[52] K. Shahryari and A. Anvari-Moghaddam, "Demand Side Management Using the Internet of Energy Based on Fog and Cloud Computing," 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 2017, pp. 931-936, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.143.

[53] Yuanpeng Xie et al., "A hierarchical key management system applied in cloud-based smart grid," 2015 IEEE/CIC International Conference on Communications in China - Workshops (CIC/ICCC), Shenzhen, 2015, pp. 22-26, doi: 10.1109/ICCChinaW.2015.7961573.

[54] Q. Li et al., "A Risk Assessment Method of Smart Grid in Cloud Computing Environment Based on Game Theory," 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, China, 2020, pp. 67-72, doi: 10.1109/ICCCBDA49378.2020.9095625.

[55] F. -L. Huang and S. -Y. Tseng, "Predictable smart home system integrated with heterogeneous network and cloud computing," 2016 International Conference on Machine Learning and Cybernetics (ICMLC), Jeju, Korea (South), 2016, pp. 649-653, doi: 10.1109/ICMLC.2016.7872964.

[56] B. A. Ugale, P. Soni, T. Pema and A. Patil, "Role of cloud computing for smart grid of India and its cyber security," 2011 Nirma University International Conference on Engineering, Ahmedabad, India, 2011, pp. 1-5, doi: 10.1109/NUiConE.2011.6153298.

[57] N. Dezhabad, S. A. Motamedi and S. Sharifian, "A proposed architecture for soft computing in smart grid as a cloud-based service," 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran, 2015, pp. 386-391, doi: 10.1109/KBEI.2015.7436076.

[58] S. Sharma, A. Sharma, T. Goel, R. Deoli and S. Mohan, "Smart Home Gardening Management System: A Cloud-Based Internet-of-Things (IoT) Application in VANET," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-5, doi: 10.1109/ICCCNT49239.2020.9225573.

[59] J. Y. Zhang, P. Wu, J. Zhu, H. Hu and F. Bonomi, "Privacy-Preserved Mobile Sensing through Hybrid Cloud Trust Framework," 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, 2013, pp. 952-953, doi: 10.1109/CLOUD.2013.108.

[60] P. Maiti, H. K. Apat, A. Kumar, B. Sahoo and A. K. Turuk, "Deployment of Multi-tier Fog Computing System for IoT Services in Smart City," 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Goa, India, 2019, pp. 1-6, doi: 10.1109/ANTS47819.2019.9117921.

[61] V. Goswami, B. Sharma, S. S. Patra, S. Chowdhury, R. K. Barik and I. B. Dhaou, "IoT-Fog Computing Sustainable System for Smart Cities: A Queueing-based Approach," 2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC), Jeddah, Saudi Arabia, 2023, pp. 1-6, doi: 10.1109/ICAISC56366.2023.10085238.

[62] V. Mishra, S. S. Yau and C. Yenugunti, "Recovering Decentralized Critical Archival Data From Tampering in Smart City Environment Using Blockchain," 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/S

CI), Leicester, UK, 2019, pp. 1972-1977, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00344.

[63] B. Naets, W. Raes, R. Devillé, C. Middag, N. Stevens and B. Minnaert, "Artificial Intelligence for Smart Cities: Comparing Latency in Edge and Cloud Computing," 2022 IEEE European Technology and Engineering Management Summit (E-TEMS), Bilbao, Spain, 2022, pp. 55-59, doi: 10.1109/E-TEMS53558.2022.9944509.

[64] N. Mishra, V. Kumar and G. Bhardwaj, "Role of Cloud Computing in Smart Grid," 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, 2019, pp. 252-255, doi: 10.1109/ICACTM.2019.8776750.

[65] M. Vögler, J. M. Schleicher, C. Inzinger, S. Dustdar and R. Ranjan, "Migrating Smart City Applications to the Cloud," in IEEE Cloud Computing, vol. 3, no. 2, pp. 72-79, Mar.-Apr. 2016, doi: 10.1109/MCC.2016.44