Journal of Intelligent Information Systems (2024) 62:641–663
https://doi.org/10.1007/s10844-022-00715-7

Design of fully homomorphic multikey encryption scheme for secured cloud access and storage environment

Dilli Babu Salvakkam[1] · **Rajendra Pamula**[1]

Received: 25 January 2022 / Revised: 20 April 2022 / Accepted: 3 May 2022 /
Published online: 21 May 2022

Abstract
Cloud hosting is a kind of storage that enables users to access, save, and manage their data in a secure and private cloud environment. As a result of this choice, users are no longer need to maintain and build their storage infrastructure on their computers or servers. Many businesses are hesitant to embrace cloud storage because of the complexities of data privacy and security issues. An easy-to-use and secure method for cloud storage sharing and data access is proposed in this study, which may be implemented quickly and easily. This solution requires users to have a secure password and biometric data in order to function properly. Their capacity to deceive consumers into disclosing critical information to their service providers is the primary reason for this problem. Cloud storage systems must have a secure framework in place in order for users to connect to and interact with one another. Many benefits of cloud storage exist, including enabling users to store and manage their data in a safe environment. Users can regulate and manage their data security while using cloud storage services. While implementing a safe and authenticated data storage model, this article addresses the different elements that must be taken into consideration. Several procedures have been established to deal with this problem. Unfortunately, they are not sufficiently secure to prevent a wide variety of security intrusions from taking place on them. When encrypting stored cloud data, the Fully Homomorphic multikey Encryption (FHE) algorithm is utilized. They also have a vulnerability in their protocol that makes it susceptible to both user and serverside attacks. When it comes to remote access, cloud data and data sharing between geographically dispersed devices is a reliable protocol to use.

Keywords Cloud access · Storage · Data sharing · User authentication · Fully homomorphic multikey encryption

✉ Dilli Babu Salvakkam
dillibabusalvakkam@gmail.com

Rajendra Pamula
rajendra@iitism.ac.in

[1] Department of Computer Science and Engineering, Indian Institute of Technology (ISM), Dhanbad, India

1 Introduction

Cloud computing is a new paradigm that enables businesses to provide their customers with on-demand access to computing and file storage capabilities. The on-Demand Routing protocol easy to share routes for broadcast or non-broadcast media, and it enables users to use cloud computing services as needed without having contact between consumers and service providers (Karati et al., 2021). Its emergence has raised concerns about privacy and the integrity of their data. The use of server-side hardware is very cheap and easy to provide security to cloud data. However, it is not as secure as cloud computing due to its limited availability and resource constraints. Gaffled circuits are often used in cloud computing. In this paper, we introduce a method for safely reusing garbled circuits for different inputs. In terms of cloud privacy, there are various approaches like twin cloud and token-based cloud computing. Cloud tokenization exchange sensitive data for an irreversible, non-sensitive placeholder known as a token and securely preserving the original, sensitive data. However,these methods are very hard to parallelize. For researchers, one of the most promising approaches is FHE which is a type of Homomorphic multikey encryption (Ghaffar et al., 2020; Zhou et al., 2019). The rapid emergence and evolution of communication and information technology have greatly changed the computational model. The rise of cloud computing was directly caused by the changes in the computational model. It is mainly built on the principle of distributed computing, which is a type of multi-core computing. Due to the increasing popularity of data storage, the existing storage models are not able to handle the influx of data. Data storage management is a set of processes i.e., network virtualization, replication, mirroring, security, compression, deduplication, traffic analysis, process automation, storage provisioning and memory management to improve the performance of data storage resources. The cloud computing provides the data storage as a service and it deliver the on-demand access in order to eliminate buying and managing your own data storage infrastructure. This is where the need for storage solutions comes from. Cloud storage is a webbased data storage mechanism that allows users to store and retrieve their data from a variety of distant servers (Ghaffar et al., 2020). Cloud storage is becoming more popular. A third-party cloud storage provider is in charge of providing the storage service. Users may purchase or rent the storage space that they need in order to save their data. It is a collection of multiple storage devices and servers (Li et al., 2017) that is known as cloud storage. Cloud storage is much more than just a storage system, though. It is also considered to be a kind of service. Because cloud storage services are provided by other parties, customers are not required to comprehend the numerous components of storage devices, as well as the administration and maintenance of such devices. They may easily take advantage of the advantages of cloud storage without the need for any specialised knowledge or experience. Cloud storage, in addition to minimising the amount of storage space necessary, provides a great deal of convenience to its customers as well. When it comes to growth, cloud storage architecture makes things easier by enabling service providers to acquire more storage servers and quickly enhance the available capacity. The movement of the majority of the data from on-premises storage to cloud storage makes data management much easier to handle. Using cloud storage space to migrate big amounts of data to the cloud, businesses may save a significant amount of money by renting or purchasing storage space from cloud providers. File versioning, automatic synchronization, data backups, security, and scalability are some of the features of cloud storage space. Enterprises may get the best cloud storage solution for their unique requirements with the assistance of cloud storage service providers. It not only ensures that they get the highest

possible quality of service, but it also helps to reduce security threats. As a result of the benefits of cloud storage, more businesses are beginning to provide their services via the cloud storage infrastructure. Google Drive, Microsoft's Windows Azure, Sync, Amazon Drive S3, Apple iCloud, MediaFire, Microsoft OneDrive, and pCloud are just a few examples of cloud storage services. The results of a poll carried out by cloud storage business many survey revealed that just around 20% of consumers are prepared to keep confidential data in the cloud. Even in the face of this, the vast majority of users are pleased with the service's dependability and overall functionality. One of the most common reasons why consumers do not utilise cloud storage services is because of concerns about security. This is one of the primary reasons why many people are skeptical about cloud storage systems.

2 Fully homomorphic multikey encryption security in cloud

Since the cloud storage system has various features and security concerns, it is often necessary to develop and implement different solutions for different issues. This paper aims to analyze and discuss the various security issues that cloud storage can face (Yang et al., 2021). Due to the separation between the data management and the ownership of the stored data, it is important that the security measures are implemented to prevent the unauthorized access to the data. The use of encryption to preserve the privacy of stored data is generally considered to be a good practise. When Alice wishes to communicate data to Bob, she uses this encryption mechanism. The method may be used when Bob has to communicate information to Alice. Homomorphic multikey encryption is a key component of cloud storagebecause it allows anyone to execute certain algebraic operations on encrypted data, which is significant in the field of quantum computation. Unfortunately, it is not extensively utilised in cloud storage environments at the present time (Zhu et al., 2021). Maintaining the integrity of data saved on a cloud storage provider's server is also a significant concern. The Provable Data Possession (PDP) scheme is a cryptographic mechanism that allows users to verify the availability and integrity of outsourced data on untrusted cloud storage servers (CSS). The majority of PDP schemes are publicly verifiable, however in some applications, private verification is required to prevent the publication of any relevant information and it is described in (Liu et al., 2021; Kaleem et al., 2021) for specifications, and it lets a client to prove that the server did not tamper with or delete the data. Their attention was drawn away from the problem of data updating in real time. Essentially, the idea behind this strategy is to ensure that the data saved in the cloud is not tampered with. This can only be accomplished via the use of dynamic data updates. Data sharing has become more popular among cloud service providers as a result of the growing number of situations in which it is necessary. It is a secure way of information transmission cloud data sharing method and provides ubiquitous access i.e., may access the data anywhere using network devices. An international team of academics suggested a solution for securing sensitive data using an elliptic curve encryption system in 2010.

This article covers cloud storage data access and sharing technologies in depth. There are three key components involved: a server, a user, and a third-party that has been vetted by the organisation, in that order. Initially, the system creates the global parameter KG for the system. To utilise the cloud storage service (Rawal & Vivek, 2017), the user needs first create an account with the CS. Accessed data cannot be accessed by an attacker over the public channel, on the other hand. The purpose of this article is to present the security standards that must be met by cloud storage services.

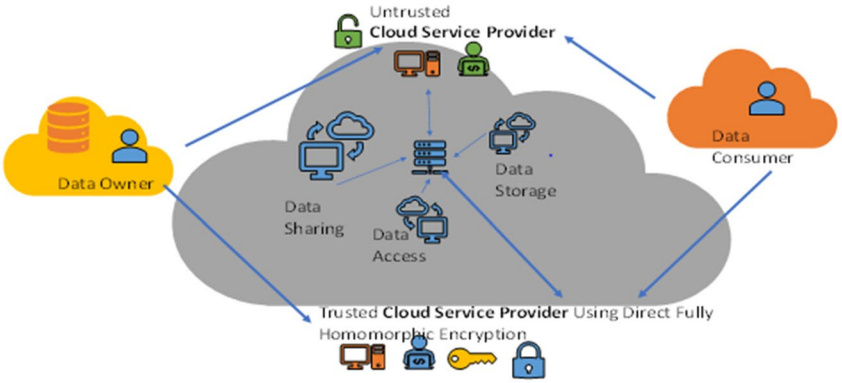

Fig. 1 Fully homomorphic multikey encryption security requirements

Table 1 List of notations

FHE	Fully Homomorphic Encryption, a trusted third party
$CUBIO_a$	Cloud User Bio-metric information of S_a
TCS	Server of the Trusted cloud storage provider
Gen	Λ generator of Z_q^*
$x_1(\cdot)x_2(\cdot)$	Function $\{0, 1\}^*$ to Z_q^* to calculate Hash
CU/D_i	identity of Cloud User CU $\_a$
$CUPW_a$	Password of CU_a
n	Λ large prime number
CS	Cloud Server
CU_a,CU_b	The a th and b th Cloud user

2.1 Homomorphic multikey security requirements in cloud storage and data access

In order for the user to have access to the TCS after being authorised, the user authentication scheme must be applied (Albrecht et al., 2019). Depending on his or her preferences, the user may personalise their passwords. In schemes saves the user's time and aids in the prevention of unauthorized access.

2.2 Mathematical background

Definition 1 The following properties are followed when ordering the elements in $\mathbb{G}.\{\mathbb{G}, \cdot\}$ (Fig. 1 and Table 1)

A group $\mathbb{G}$ has a set of elements that have a binary operation $\mathbb{G}.\{\mathbb{G}, \cdot\}$.

(a). Closure Property: The closure property of x,y provides with only one unique answer after adding or multiply in same $\mathbb{G}$.

(b). The Associative Property: The property's Associative Properties are defined as if x,y,z are all in $\mathbb{G}.x \cdot (y \cdot z) = (x \cdot y) \cdot z$, where $\forall x$,y, and z in $\mathbb{G}$

(c). An identity property is a unique element that can be used to identify a specific element. $x \cdot e = e \cdot x = x$,$y \cdot e = e \cdot y = y$,$z \cdot e = e \cdot z = z$.

(d). Inverse element: For any $x \in \mathbb{G}$, there is an element $x \cdot x' =$ and $x' \cdot x = e$.

2.3 Homomorphism in public key crypto system

Four parties are involved in this data exchange and access system, which is described in this section. In this scenario, there are four parties involved: the user; the system administrator; the data sharing scheme; and the fuzzy extraction algorithm. When given a specific input, fuzzy extractors are a biometric tool that enables for user authentication by employing a biometric template created from the user's biometric data as the key, with predictability indicating the likelihood of an attacker guessing the secret key and it is a process that can consistently extract uniform randomness from it. It is also error-tolerant in the event that the input is changed. Generation (Gen) is a technique that generates a biometric input string from which an extracted string may be generated. If the input string is not supplied, it will be outputted as an auxiliary string until otherwise specified. It is possible to retrieve V from the auxiliary string U and the vector CUBIO' prime that is near to U using this technique.

- RSA cryptosystem:

$$\begin{aligned} \mathcal{E}(x) &= x^e \text{modm} \\ \mathcal{E}(x_1) \cdot \mathcal{E}(x_2) &= x_1^e x_2^e modm = (x_1 x_2)^e modm = \mathcal{E}(x_1 \cdot x_2) \end{aligned}$$

- Paillier Cryptosystem

$$\begin{aligned} \text{Encrypt}\,(m;CUPK) &= g^{\text{Msg}} \cdot r^n \left(\text{mod} n^2\right) \\ c_1 \cdot c_2 &= g^{m_1} \cdot r_1^n \cdot g^{m_2} \cdot r_2^n = g^{m_1+m_2} \cdot \left(r_1 \cdot r_2\right)^n \text{mod} n^2 = c_3 \end{aligned}$$

- ElGamal Encryption

$$\begin{aligned} E &: G_q \to G_q \times G_q \\ E(m) &= (g^r, m * h^r) \\ E(m_1) * E(m_2) &= \left(g^{r_1}, m_1 * h^{r_1}\right)\left(g^{r_2}, m_2 * h^{r_2}\right) \\ &= \left(g^{r_1+r_2}, m_1 * m_2 * h^{r_1+r_2}\right) \\ &= E(m_1 * m_2) \end{aligned}$$

2.4 The proposed scheme FHE preliminaries

This algorithm generates a list of parameters that are used in HE algorithms. It takes the desired security level and outputs it as an input.

$$\text{ParamGen}\,(\lambda, PT, K, B) \to \text{Params}$$

This document only describes the underlying plaintext space of a parametrized format. It does not specify the type of approximate numbers that can be used in the space.

$$\left(V_1, \ldots, V_K\right) + \left(V_1', \ldots, V_K'\right) = \left(V_1 + V_1', \ldots, V_K + V_K'\right)$$

The encryption of a message is performed by parametrizing the digits with the plaintext space Zp. The message space is an integer that is equal to the range [0,1023).

$$\text{PubKeygen(Params)} \rightarrow \text{SK, CUPK, EK}$$

The extension rings and fields are parameterized by modulus p, and they are also specified by a polynomial $f(x)$, which is equal to the plaintext space $Z[x]$.

$$\text{SecKeygen(Params)} \rightarrow \text{SK, EK}$$

The dimension of the encrypted vectors is defined as the space where the messages are encrypted which is used to prevent a series of text that is identical to a prior sequence from creating the same exact ciphertext when encrypted by using a continuously changing integer in combination with a secret key. It is usually computed by defintion, which is the operation that is performed component-wise.

$$\text{PubEncrypt (CUPK, Msg)} \rightarrow \text{C}$$

As per the external sources, auxiliary parameter acts like supplementary which is used to encrypt the messages for the secure transmission. The auxiliary parameter B is used to specify the complexity of the programs and circuits that can be used to carry out encrypted messages. Generally, lower-complex programs and circuits are more efficient in their evaluation.

$$\begin{aligned} SecEncrypt(SK, M) &\rightarrow C \\ Decrypt(SK, C) &\rightarrow Msg \end{aligned}$$

A fuzzy extractor is a set of procedures that can reliably extract random bits from a given input. It is usually not error-tolerant if the input changes. Gen is a probabilistic generator procedure that outputs an extracted string from a biometric input which is get from the biometric characteristics that are acquired applying adequate sensors to extract biometric template in an enrolment process. It does so by extracting the specified string from the CUBIO distribution. $Rep\left(\text{FH}E', P\right) = Q$ if CUBIO' is reasonably close to CUBIO.

FHE distribution on M with min randomness m, the distribution's randomness is equal to the sum of the digits of the operation name. Gen is a cyclic generation procedure $Gen(CUBIO) = (R,P)$ that takes advantage of the input of Biometric input. It outputs an extracted string. Rep is a procedure that returns V from the string U and the vector CUBIO'. The $\left(CUBIO, CUBIO'\right) \in M$ does so by converting the data pair $\left(\text{CUBI}O'\right)$ to Q.KG chooses the system's global parameter q, and then generates public and private key pairs with a large prime number z_q.

$$\begin{aligned} &\left(\text{HS}K_a = k_a, CUPK_a = g^{k_a} \text{modn}\right), \\ &\left(\text{HS}K_b = k_b, CUPK_b = g^{k_b} \text{modn}\right) \text{and} \\ &\left(\text{HS}K_c = k_c, CUPK_c = g^{k_c} \text{modn}\right) \text{for} CU_a, CU_b \text{and TCS} \end{aligned}$$

2.5 Preliminaries

A fuzzy extractor is a procedure that can extract almost uniform randomness from a biometric input. It is error-tolerant if the input changes or the output is not sufficiently close to the original one.

1. Gen is a probabilistic generator that outputs an extractable string containing a biometric input as Gen(CUBIO) = (*Q*,*P*).

For any distribution of *m*, if the generating function $Gen(CUBIO) = (Q,P)$, then randomness collected the operator and the string is equal to the sum of the distributions CU_l.

2. The Repoperator $\left(FHE, CUBIO'\right) \in M$ is a predictable mechanism that allows for the recovery of information *V* concerning the *U* string $DIS\left(CUBIO, CUBIO'\right) \leq t$, if $Gen(CUBIO) = (Q,P)$ and the CUBIO'prime vector close to *V*.

2.6 System initialization phase

The first parameter KG chooses $x_1 : \{0,1\}^* \rightarrow \{0,1\}^n, x_2 : \{0,1\}^* \rightarrow Z_a^*$. At last, KG is the global parameter q, and the second one is the generator g. Then, it chooses hash functions $h1$: 1,1,1,1 *, and q public $\left(\mathrm{HS}K_a = k_a, CUPK_a = g^{k_a} \mathrm{modn}\right)$, $\left(\mathrm{HS}K_b = k_b, CUPK_b = g^{k_b} \mathrm{modn}\right)$ and $\left(\mathrm{HS}K_c = k_c, CUPK_c = g^{k_c} \mathrm{modn}\right)$ for CU_a,CU_b. CUPK distribute the public and private key pairs to the parties involved (Fig. 2).

3 Cloud server and user mutual authentication scheme for secured storage and access

There are some processes that must be completed in order to safeguard data access in diverse applications: user registration, authentication, and password changing. The steps of the user authentication procedure are shown in this session.

3.1 Cloud user registration phase

In order to make advantage of the cloud storage service offered by TCS, users must first create an Registration. This step is required to check that the user has been granted permission to access the service. When it comes to establishing trust, both sides must verify their communications with one another. In this case, the user CU_a sends a registration request to server TCS after forming a CS account and choosing an identity CIDi for themselves. It is regulated by TCS to save the information about the user Qi on a mobile device, then securely communicates the information to Ui. Following that, CU_a selects a password that is evenly spread across the system $\left(\mathrm{CUP}W_i\right)$. She also leaves a trace of her Bioi on the sensor, which may be detected. In the next phase, the variables *Y i*,*Mi*,*Ni*,*It*,*h*(),*Gen*(), and *Rep*() are inserted into a device with the values they represent.

Step 1: The Cloud Server CU_a selects and inputs the identity and password of the user. The biometric template created by the fuzzy extractor is subsequently imprinted on the Cloud Server. C transfers the variable A_i via a secure channel and saves the parameters $\left(g, h1, andb_a\right)$ in the Cloud Server's memory.

Step 2: CS sends $A_i = h_1 left, B_i = h_1$left, and submits the parameters through a secure channel.

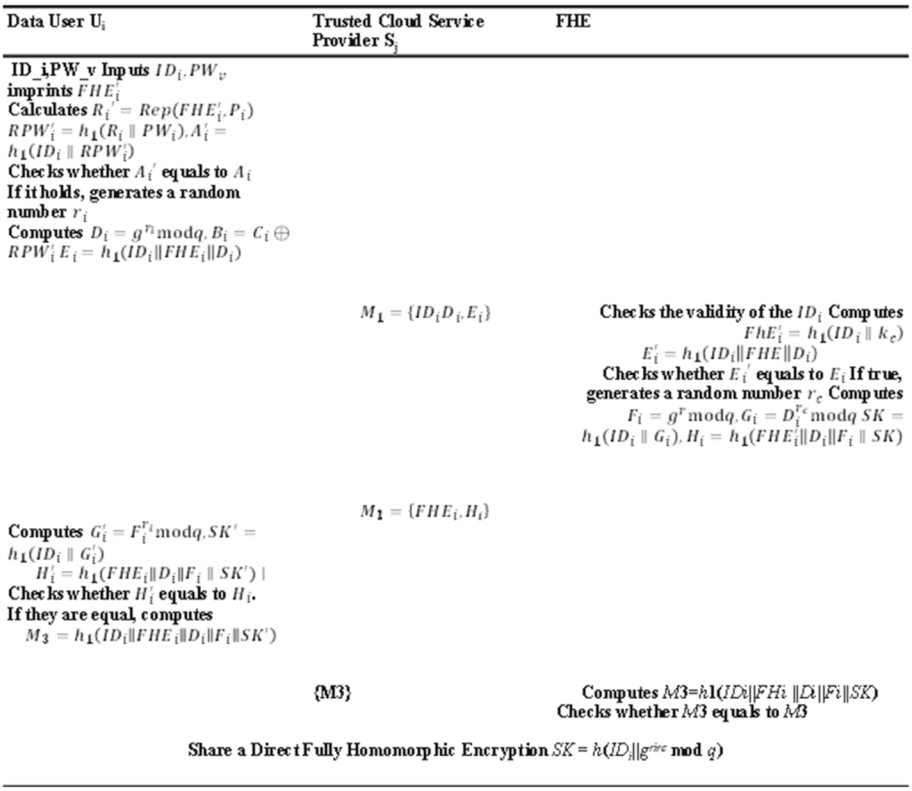

Fig. 2 Cloud user and Server transaction phases with FHE authentication mechanism for to provide safe data access

$$
\begin{aligned}
\mathrm{FHERPW}_a &= h\big(b \parallel CUPW_a\big) \\
Gen\big(\mathrm{Bio}_a\big) &= \big(\sigma_a, \tau_a\big) \\
N_a &= R_a \oplus h\big(\sigma_a \parallel \mathrm{FHERPW}_a \parallel b\big) \\
\mathrm{Msg}_a &= h\big(b \parallel ID_a \parallel \mathrm{FHERPW}_a \parallel \sigma_a \parallel R_a\big) \\
N_a &= b \oplus h\big(ID_a \parallel \mathrm{CUPW}_a \parallel \sigma_a\big)
\end{aligned}
$$

Cloud User Initial Login and **A**
ceess

3.2 Cloud user initial login and authentication protocol

In order to build confidence between the Data User and the Cloud Service Provider, these two parties must first authenticate one another. Babu et al. protocol is a cryptographic system using Blockchain based authentication that may be employed with or without a smart card, depending on the condition. Cryptography is an integral part of the inner-workings of blockchain technology and it provide the trait of immutability and improve the security, scalability, reliability. Here, a registered user CUi authenticates himself or herself by inputting the card reader and biometric sensor credentials into the appropriate fields. Ui is subjected to biometric scanning in order to determine the user's physical characteristics. Ui

then authenticates by entering the credentials IDi, PWi, and BIO_a into the authentication dialogue box. After that, the smart card creates a cancelable fingerprint C0T I, which is subsequently retrieved using error-correcting techniques. SCi determines if $h(r0i)$ equals h. (ri). If it fails to do so, the session is terminated without further delay.

Step 1: CU_a inputs CUIDi and $\mathrm{CUP}W_i$ in the login screen. The Cloud Server then calculates the number of sessions computed by the Cloud Server. If it is equal, the session is ended.

If the identity CUIDi is valid, TCS checks if the number B_{aprime} is equal to the one provided by the user. It is ignored if the two numbers are not equal in size.

Step 2: If the identity CUID i is valid, TCS checks if the request is equal. If it is $\mathrm{BIO}'_a = x_1\left(CSIDi \parallel k_c\right)$, it rejects the login request.

Step 3: V_a sends the TCS authentication message to the $U_p i$. If the session is ended, then the cloud user can verify the authentication of the TCS by $CU_p i$. CU_a computes $\mathrm{GR}P'_a = F_a^{r_a} modN, HSK' = x_1\left(CSIDi \parallel GRP'_a\right), X'_a = x_1\left(B_a \parallel D_a \parallel F_a \parallel HSK'\right)$.

$\mathrm{Ms}g_3 = x_1\left(\mathrm{CSIDi} \parallel B_a \parallel D_a \parallel F_a \parallel \mathrm{HS}K'\right)$ and submits $\left\{\mathrm{Ms}g_3\right\}$ to TCS.

Step 4: CS computes $\mathrm{Ms}g'_3 = h(CUIDi)\left(B'_a\right)(CUIDi)$.

$\mathrm{Ms}g'_3 = x_1\left(\mathrm{CSIDi} \parallel B'_a \parallel D_a \parallel F_a \parallel \mathrm{HSK}\right)$, and checks $\mathrm{Ms}g'_3 \stackrel{?}{=} \mathrm{Ms}g_3$.

CU_a and TCS For these secret communications, the two parties share a session key $HSK = x_1(CSIDi \parallel g^{r_a r_c} \mathrm{modN})$.

3.3 Password change phase

The CU_a user may make changes to his or her passwords without the assistance of the cloud storage service provider. In order to do this, the user must successfully enter the passwords and imprint the data on the screen. CUBIO'_a. The Cloud Server checks the computed $V'_a = h_1$ left $>$ to verify the user's identity. $V'_a = Rep\left(\mathrm{CUBIO}'_a, P_a\right)$, $\mathrm{FHERP}W'_a = x_1\left(R'_a \parallel CUPW_a\right), A'_a = x_1\left(CSIDi \parallel FHERPW'_a\right)$, and checks $A'_a \stackrel{?}{=} A_a$. If the passwords $\mathrm{CUP}W_a^{\mathrm{new}}$ are not equal, the Cloud Server will terminate the password change request. The user is asked to enter a new password. The Cloud Server will also replace the existing passwords with new ones. $\mathrm{FHERP}W_a^{\mathrm{new}} = x_1\left(R'_a \parallel CUPW_a^{\mathrm{new}}\right), A_a^{\mathrm{new}} = x_1\left(CSIDi \parallel FHERPW_a^{\mathrm{new}}\right)$, $C_a^{\mathrm{new}} = C_a \oplus FHERPW'_a \oplus FHERPW_a^{\mathrm{new}}$, and replaces A_a and C_a with A_a^{new} and C_a^{new}.

3.4 Storage and access in cloud

This section describes a secure data sharing scheme that enables users to store and share their data $m \in Z_q^*$ in cloud storage. For a user CU_a, sends data to the receiver the scheme requires that he or she register with the cloud storage provider.

Step 1: The user CU_a generates a random number that's $\alpha \in Z_a^*$, and stores it as the encrypted data of m on the device $m_e = m \cdot g^{\alpha}$. $\mathrm{CUP}K_a^{x_2(CSIDi|\alpha)} modq, m_V = x_1\left(g^{\mathrm{Msg}} \mathrm{modN}\right)$. In order to get the original data, user authenticates and obtains the original data stored on the server TCS.

Then, after storing the random number a *, the device stores the encrypted data of m * as well as the original data. When he wants to recover $m = m_e \cdot g^{-\alpha} \cdot \left(g^{x_2(CSIDi\|\alpha)}\right)^{-k_a} \mathrm{modN}$ the original data, he authenticates and retrieves the data whether m is valid by checking $x_1\left(g^{\mathrm{Msg}} \mathrm{modN}\right) \stackrel{?}{=} m_V$.

Step 2: The receiving user then submits CU_blogin to CS a data sharing request from CU_a.

Step 3: Login of U_a generates two random numbers $V_b = g^{r_b} \text{modN}. R_c = g^{r'_c} mod N, m_c^* = CUPK_c^{-r'_c} \cdot g^{-\alpha} mod N, m_b^* = CUPK_b^{-r_b} \text{CUP} K_a^{-x_2(CSIDi|\alpha)} \text{mod} N.$ and $r_b, r'_c \in Z_c^*$. Now Computes CU_a submits (CSIDj, R_c, m_c^*) to TCS, and submits $\left(R_b, m_b^*\right)$ to CU_b.

Step 4: After receiving the message $\left(R_c, m_c^*\right), CS$ computes $m_c = m_e. m_c^* \cdot R_c^{k_c} \text{modN}$, and submits $\left(m_c, m_V\right)$ to the user CU_b sends the message to the user.

Step 5: When receiving the message, enter username and the shared message m is obtained by calculating the sum of the two numbers. $\left(R_b, m_b\right)$ and $\left(m_c, m_V\right), CU_b$ obtains the shared data m by computing $m_b = m_c \cdot m_b^*. R_b^{k_b} \text{modN}$, and the validity of data $mx_1(g^{m_b} \text{modN}) \stackrel{?}{=} m_V$ validate that to receive the data. Store $\left(m_e, m_V \leftarrow 2.(CSIDi, CSIDj,\right.$ Request) (CSID j, Q_

4 Random oracle model (ROM) and BAN logic for formal security analysis and verification

In this study, we explain the notion of safe data storage and access using BAN logic, a formal technique. This technique accomplishes the aims of data protection for the user.

P believes that *X* is true.

(1) U believes M is true, i.e. $U| \equiv X : U$ believes M
(2) Someone sent a message which contains M to U, and U can read X. i.e. $U \lhd X : U$ sees M

This function returns the message containing M once sent.

(3) An earlier iteration of U delivered a message with M attached, i.e. $U \mid\sim X : U$ once said M
(4) M is subject to the jurisdiction of entity U, and U is trusted for M, i.e. $U \Rightarrow X$: U controls M

Since the present round of protocol, no entity has sent a message containing M.

(5) No entity sent a message containing M at any time before the current round of protocol, i.e. $\#(M) : M$ is fresh
(6) $U \stackrel{K}{\longleftrightarrow} V : U$ and V the two users and the server may interact with one another through the shared key K, where K is said to be more secure if no other entity can get it except for U,V and the entity trusted by U,V.

Rule 1 Message means that if U believes that he/she shares the key K with V, then U should believe that V once said M.

(7) $(M,N) : M$ and N are components of the message (M,N).
(8) $\{M,N\}_K : M$ and N are encrypted using the key K.
(9) $(M,N)_K$: Using the key K,M and N are hashed together.

Goal 1: $CU_a \mid\equiv \left(U_a \longleftrightarrow HSKCS\right)$.
Goal 2: $CU_a| \equiv CS| \equiv \left(U_a \longleftrightarrow HSKCS\right)$.
Goal 3: $CS \mid\equiv \left(U_a \longleftrightarrow HSKCS\right)$.
Goal 4: CS$\left|\equiv CU_a\right| \equiv \left(U_a \longleftrightarrow HSKCS\right)$.

Rule 1: Message meaning rule: Message means that if a person believes that he/she has a key K, then he/she should see the message X_K.

sees the message $\{X\}_K$,U believes that V once said M.

Rule 2: Nonce verification rule: If U believes that M is fresh and V once said M, then U believes that M.

$\frac{P|\equiv\#(X),P|=Q|\sim}{P|\equiv X}$, if U believes M is fresh and V once said X,U believes V believes M.

Rule 3: Jurisdiction rule: If V believes that it has jurisdiction over M, then it should believe that it has jurisdiction over M.

$P|\equiv X$, if U believes that V had jurisdiction right to M and believes V believes X,U believes M.

Rule 4: Freshness rule:

$\underset{P|=\#(M,N)}{P} |= \#(X)$, If message ($M$,$N$) contains message M, then message (M,N) must be fresh as well.

Rule 5: Belief rule: If U believes that the message is clear and unambiguous (M,N), then U believes that the message is clear and unambiguous (X).

$$P| = Q| \equiv (M, N)$$

Rule 6: Seeing rule: $\frac{P_a(M,N)}{\mathrm{PXX}}$, M is a part of the message (M,N), and if U sees (M,N),U also sees M.

Before the formal analysis, the two parties such as Data user and cloud service providers should first communicate the messages that they exchanged, first assume that the two parties are communicating through normal SMS messages.

The validity of A_1 and A_2 depends on the random numbers generated by r_a and r_c, which are both fresh random numbers.

A_1 and A_2 are valid since r_a and r_c random numbers produced by CUa and TCS to put it another way, because of the freshness in both ra and rc, $A3$ and $A4$ are reasonable choices. Using the device's information and the server's identification, the user may derive the secret key. A logical assumption is that the user's identity and Cloud Server information is known. User CU_a and server TCS can calculate $x_1\left(\mathrm{CSI}D_i \mid k_c\right)$ from the Cloud Server information and the secret key k_c'' and user's identification, and A_5 and A_6'' are also reasonable."

In this paper, we prove that a proposed protocol can meet the goals of its intended users. We provide a detailed description of the proposed protocol.

$S_1 : CU_a| \equiv CS| \sim \left(D_a, F_a, CU_a \longleftrightarrow HSKCS\right)$ We use the freshnessconjuncatenation rule when it comes to selecting fresh produce.

$S_2 : CU_a |\equiv \#\left(D_a, F_a, CU_a \longleftrightarrow HSKCS\right)$ based on the premise of S_3 and S_4, the nonce-verification rule is used in order to get the result.

$S_3 : CU_a| \equiv CS| \equiv \left(D_a, F_a, CU_a \longleftrightarrow HSKCS\right)$ based on the premise of S_5, we use the belief rule in order to attain our goals.

$S_4 : CU_a| \equiv CS| \equiv \left(U_a \longleftrightarrow HSKCS\right)$ (Goal 2). based on the premise of A_7 and S_6, we apply jurisdiction rule for the belief rule in order to attain our goals.

$S_5 : CU_a |\equiv \left(U_a \longleftrightarrow HSKCS\right)$ (Goal 1) based on the premise of message 3 , we can get

$S_6 : CS \lhd \left(CSIDi, CD_a, F_a, CU_a \longleftrightarrow HSKCS\right)_{x_1\left(CSIDi\|k_c\right)}$ based on the premise of S_8 and A_6, we employ the message meaning for belief rule in order to attain our goals

$S_7 : CS|\equiv CU_a| \sim \left(CSIDi, CD_a, F_a, CU_a \longleftrightarrow HSKCS\right)$. based on the premise of A_4, we apply freshness-conjuncatenation belief rule in order to attain our goals

$S_8 : CS |\equiv \#\left(CSIDi, CD_a, F_a, CU_a \longleftrightarrow HSKCS\right)$ based on the premise of S_9 and S_{10}, we apply nonce-verification for belief rule in order to attain our goals

$S_9 : CS|\equiv CU_a| \equiv \left(CSIDi, CD_a, F_a, CU_a \longleftrightarrow HSKCS\right)$ based on the premise of S_{11}, we apply belief rule to obtain

$S_{10} : CS|\equiv CU_a| \equiv \left(U_a \longleftrightarrow HSKCS\right)$ (Goal 4) based on the premise of A_8 and S_{12}, we apply jurisdiction to use the belief rule in order to attain our goals $S_{13} : CS \mid\equiv \left(U_a \longleftrightarrow HSKCS\right)$ (Goal 3) The many security measures of the proposal are discussed in detail in this section.

4.1 Random oracle model (ROM)

The Random Oracle Model was used for our formal security analysis. The random-oracle model (ROM) used for designing and analysing cryptographic protocols. It gives random functions that would undoubtedly create excellent cryptographic hash functions and security proofs for extremely practical constructions of crucial cryptographic building blocks like digital signatures, public-key encryption, and key exchange. It is commonly regarded as strong evidence that a protocol would withstand assaults in practise, despite its recognised inability to provide verifiable assurances when instantiated with a real-world hash function. This framework offers a simple and effective security paradigm for our proposed solution. We validate the ROM scheme's security and privacy and use the same security model.

Theorem 1 An adversary $\mathcal{U}_A$ can execute multiple oracle queries with execution time of less than 2 minutes. The adversary can break the security of U_rp by using the hash function $h(cdot)$. P denotes the protocol's correctness. D denotes the password dictionary. If the U_{rp} protocol is not followed, then the query will be executed by an adversary.

$$\text{Adv}t_{P_p,D}^{\text{AKE}}\left(\mathcal{U}_A\right) \leq M' \cdot q_s^{N'} + \epsilon(w)$$

where M' and N / are the security parameter and trivial function of Z_{ipf}.

Proof In every game, the Test query $_0$ to Game $_{1\ -\ 6}$ is used to guess the correct bit. The result S_a and $U_r\left[S_a\right]$ is presented as the probability of the chosen bit being correct. The game is offered as S_a and U_r left.

$$\text{Adv}t_{P_{\text{rp}},D}^{\text{AKE}}\left(\mathcal{U}_A\right) = P_r\left[S_0\right]$$

□

Game 1: This game shows how to establish a hash list $h(cdot)$ with a secure hash function.

$$\left|P_r\left[S_1\right] - P_r\left[S_0\right]\right| \leq \epsilon(w)$$

Game 2: Collisions have been ruled out in all possible sessions. The game will be terminated if there is a collision.

$$\left|P_r\left[S_2\right] - P_r\left[S_1\right]\right| \leq \epsilon(w)$$

Game 3: The game's simulation rules have been altered using the execute query. For example, the way private key sessions are calculated has been altered. If an attacker properly calculates XCS during the passive session $\mathcal{U}A$, may get the difference between Games 2 and 3 . To solve the task, we need to select some numbers randomly r_a1,r_cs1,

and $r_c s2 r_{a1}$,r_{cs1},r_{a2} and r_{cs2} and compute $T_{\mathrm{Sk}} = r_{cs2} X_a$ and $T_{\mathrm{Sk}} = r_{a2} X_{\mathrm{cs}} \cdot \mathcal{U}_A$ can make a query X_{cs},N_{cs},T_{cs} to hash oracle.

$$\left| P_r\left[S_3\right] - P_r\left[S_2\right] \right| \leq \epsilon(w)$$

Game 4: In this Game, we are going to use the query method used to active session $\mathcal{U}_A$ determines the authenticated X_{cs} to masquerade $\mathcal{AU}_a$.

This rule is assigned with the following responsibilities: To Calculate $N_{\mathrm{cs}} = x_1\left(CUIDa \parallel CUIDs \parallel X_a \parallel X'_a \parallel T_a \parallel T_{\mathrm{cs}} \parallel T_{\mathrm{sk}}\right)$ and determines $N'_{\mathrm{cs}} \stackrel{?}{=} N_{\mathrm{cs}}$

If this is correct, then CS predicts a list. $\left\{PCUIDa, X^*_a, N_a, T_a\right\}$ presented in L_{hs}.. This method is computed by calculating the valid $X_c$ss to disguise the query (Table 2).

$$\left| P_r\left[S_4\right] - P_r\left[S_3\right] \right| \leq \epsilon(w)$$

Game 5: The game's active session is used to query. This game will be aborted if the query succeeds and finds the record left$X_c s$.

$$\left| P_r\left[S_5\right] - P_r\left[S_4\right] \right| \leq \epsilon(w)$$

Game 6: The session key is chosen at random in this game. S_k of $\mathcal{AA}_a$ and CS. The advantage of $\mathcal{U}_a$ is negligible to guess the session key

$$\begin{aligned} \left| P_r\left[S_6\right] \right| &\leq M' \cdot q_s^{N'} \\ \mathrm{Adv}t^{\mathrm{AKE}}_{P_{\mathrm{rp}},D}\left(\mathcal{U}_A\right) &\leq M' \cdot q_s^{N'} + \epsilon(w) \end{aligned}$$

It can be performed within polynomial time. For the algorithm's implementation, we need to know the self-reducibility $\mathrm{Adv}_{CDH,P_{\mathrm{rp}}}(C)$ of the problem. Because of its difficulty, the CDH problem is viewed as infeasible in polynomial time and hence as infeasible. It has been shown that the theorem is accurate. $=\mid Pr\left[C\left(P, r_{a2}P, cupk_{\mathrm{cs}}, r_{a2}\mathrm{cup}k_{\mathrm{cs}}\right) = 1\right] - Pr\left[C\left(P, r_{\mathrm{cs}}, r_{cs2}P \quad sk_{\mathrm{cs}}\right) = 1\right]$, where sk_{cs} is a fixed value.

Now, $\mathrm{Adv}_{CDH,P_{\mathrm{rp}}}(C) \geq \mid Pr\left[TextAnon\left(\mathrm{cCUID}C^c, cCUIDc^j\right) = 1\right] - Pr[\quad \mathrm{TestAnon}\left(\mathrm{cid}^c_C, \mathrm{cid}^k_C\right) = 1\right] \mid$.

5 Security and performance evaluation

The different security characteristics of a suggested user authentication mechanism are presented in this section.

Table 2 Specifications for implementation

Items	Specifications
Windows	Windows 10
Processor	3.60GHz
Hardware	core i7
Online cloud server	PythonAnywhere
Language	python

5.1 User friendly

The user can freely choose the username and passwords for secure data access. To update the passwords CU_a inputs CSID_a and CUPW_a, the user needs to input the data required to create the new password, V'_a according to CUBIO'_a and U_a, and computes FHERPW_{a }^ {\prime } , A _ { a } ^ {\prime }\text { { and the Cloud Server will } } reproduce the data.

After verifying the validity of the user's identity $A'_a \stackrel{?}{=} A_a$, password and biometric, the Cloud Server sends a message $\_a^{\text{new}}$ to compute FHERP$W_a^{\text{new}}, A_a^{\text{new}}, C_a^{\text{new}}$, and replaces A_a and C_a with A_a^{new} and C_a^{new} to the mobile app to reset the passwords. Once this is accomplished, the user will be able to change his or her password without having to contact the cloud storage service provider.

5.2 Safeguard against a stolen verifier attack

Specifically, the suggested approach offers a way of mutual authentication between a cloud storage provider and its user. There is a reliance on the shared secret knowledge $x1$ (CSIDillkc) for the approach to work. Utilizing the information supplied by the Cloud Server, CUa can extract the value $x_1\left(CSIDi \parallel k_c\right)$, which can then be calculated by TCS by using CU_a 's identity CSID_i and the secret key kc provided by the Cloud Server. When the cloud storage provider accesses the Cloud Server, they will be able to extract the value of the secret information.

5.3 Efficient wrong password detection

The proposed method will allow the Cloud Server to quickly identify the unauthorized access by the user when they input a wrong password. This method will prevent the cloud storage provider from checking the credentials of the users.

In the login phase, Uz authenticates CU_a inputs CSIDi and imprints the biometric CUBIO'_a by using the fingerprint CUBIO_I prime on the Cloud Server. CU_a inputs a wrong password CUP$W_a^*\left(\neq CUPW_a\right)$ by mistake. Then, the Cloud Server computes $V'_a = Rep(\text{CUBIO}'_a, P_a), FHERPW'_a = x_1(R'_a \parallel CUPW^*_a)\left(\neq x_1(R_a \parallel CUPW_a) = FHERPW_a\right)$, and it is obvious that $A'_a = x_1\left(CSIDi \parallel FHERPW'_a\right) \neq x_1\left(CSIDi \parallel FHERPW'_a\right) = A_a$. Then, the device computes the incorrect V_i * left = operatornameCUBIOprime.

5.4 Resist replay attack without use of clock synchronization

The goal of this proposed scheme is to prevent replay attack by generating a random number and a timestamp for each session. This method will prevent the attackers from accessing the synchronized clocks of all the entities in the network.

5.5 Authentication process and session key agreement

In terms of security, both authentication process and session key agreement are regarded to be among the most important factors to consider. At first, the authentication process request

$\mathrm{Msg}_1 = \{CSIDi, D_a, E_a\}$ from CU_a,CS computes $B'_a = x_1(CSIDi \parallel k_c), E'_a = x_1(\mathrm{CSIDi} \parallel B'_a \parallel D_a)$, the user can verify the validity $E'_a = E_a$ of their request by checking the box labeled "E $_i$ prime?"

Next, the recipient receives $\mathrm{Msg}_2 = \{F_a, X_a\}$ from CS,CU_a computes $\mathrm{GR}P'_a = F_a^{r_a} modN, HSK' = x_1(CSIDi \parallel GRP'_a), X'_a = x_1(B_a \parallel D_a \parallel F_a \parallel HSK')$, and can authenticate TCS by checking $X'_a = X_a$ the response message $\mathrm{Msg}_2 = leftF_i$ right from TCS. When receiving a mutual authentication message, TCS will ignore the M3prime value and verify $\{\mathrm{Msg}_3\}$ from CU_a in step 4, it computes $\mathrm{Msg}'_3 = x_1(\mathrm{CSIDi} \parallel B'_a \parallel D_a \parallel F_a \parallel \mathrm{HSK})$, and can verify the validity of CU_a by checking $\mathrm{Msg}'_3 = \mathrm{Msg}_3$ the validity of the message. The proposed scheme enables the user to authenticate with the cloud storage provider through a mutual authentication. The session key $HSK = x_1(CSIDi \parallel g^{r_a r_c} \mathrm{modN})$ As an additional step to the mutual authentication, which incorporates both r_a and r_c from each of the two member organizations, the cloud storage provider computes the information that a user provides to them using random numbers obtained after the mutual authentication, which is carried out by the cloud storage provider after the mutual authentication, and this is carried out by the cloud storage provider afterwards

5.6 Violation of user anonymity

The identity $\mathrm{CUI}D_A$ of $\mathcal{AU}_{\dashv}$ is not sent in plain text CUIDa. However, UCUID $a = \mathrm{CUID}\ a \oplus X_a$. This is done via the use of a secret channel to deliver the results of the calculation to a private key termed CS, which is encrypted. The $\mathrm{CUI}D_a$, in addition, can only be created by the authorised CS that is used to authenticate users, and it is not made accessible to the public.

5.7 Resist impersonation attack

In order to be successful in impersonating the user CU_a, the attacker must first provide a legitimate username and email address into the system. The attacker must next generate a fake login request message using the user's email address as a starting point, and send it to the victim. $\mathrm{Msg}_1^* = \{CSIDi, D_a^*, E_a^*\}$ and a valid response Msg_3^*. User also know the Cloud Server's $B_a = h(CSIDi \parallel r_c)$. Since r_c is and the user's response $h(CSIDi \parallel r_c)$ An attacker needs to know the Cloud Server's details to carry out an attack. However, since the device's details are only known to the network's random number, the attacker would be unable to get them.

6 Cloud storage and data sharing security

This section is applicable to the proposed data sharing scheme.

6.1 Confidentiality

With our data sharing system, users can be certain that your information is completely safe and secure, U_i stores the encrypted data of m in its leftm $\_e$,m_V right, and it uses the random number alphain Z_q * to determine the original $m.m$,CU_a stores $\{m_e, m_V\}$ as the encrypted data of m on TCS, where $\alpha \in Z_q^*$ is a random number, $m_e = m \cdot g^{\alpha} \cdot CUPK_a^{x_2(CSIDi|\alpha)} \text{mod} N, m_V = x_1\left(g^{\text{Msg}} \text{modN}\right)$. The Trusted Cloud Storage (TCS) Provider allows holistic security and serves as a reference point for users which is used to identify cloud providers that are aligned with their security requirements. Because TCS cannot estimate the original m without knowing the random number α. The hardness of finding discrete logarithms depends on the groups i.e., if the polynomial-time based on few groups means then it is easily finding the solution O (n) but the large number group means the complexity is much harder O (log k (n)).

6.2 Correctness

Users' personal information may be retrieved by accessing the shared data m when the data

$$\begin{aligned}
\mathrm{m_b} &= \mathrm{m_c} \cdot \mathrm{m_b^*} \cdot \mathrm{R_b^{k_b}}\ \mathrm{modq} \\
&= \left(\mathrm{m_e} \cdot \mathrm{m_c^*} \cdot \mathrm{R_c^{k_c}}\right) \cdot \left(\mathrm{CUPK_b^{-r_b}} \cdot \mathrm{CUPK_a^{-x_2}(CSIDi \| \alpha)}\right) \cdot \mathrm{R_b^{k_b}}\ \mathrm{modq} \\
&= \left(\left(\mathrm{m} \cdot \mathrm{g}^{\alpha} \cdot \mathrm{CUPK_a^{x_2}(CSIDi \mid \alpha)}\right) \cdot \left(\mathrm{CUPK_c^{-r_c'}} \cdot \mathrm{g}^{-\alpha}\right) \cdot \mathrm{R_c^{k_c}}\right) \cdot \left(\mathrm{CUPK_b^{-r_b}} \cdot \mathrm{CUPK_a^{-x_a}(CSIDi \mid \alpha)}\right) \cdot \mathrm{R_b^{k_b}}\ \mathrm{modq} \\
&= \mathrm{m} \cdot \mathrm{g}^{\alpha} \cdot \mathrm{CUPK_a^{x_2}(CSIDi \| \alpha)} \cdot \mathrm{CUPK_c^{-r_c'}} \cdot \mathrm{g}^{-\alpha} \cdot \mathrm{R_c^{k_c}} \cdot \mathrm{CUPK_b^{-r_b}} \cdot \mathrm{CUPK_a^{-x_2}(CSIDi \| \alpha)} \cdot \mathrm{R_b^{k_b}}\ \mathrm{modq} \\
&= \mathrm{m} \cdot \mathrm{CUPK_c^{-r_c'}} \cdot \mathrm{R_c^{k_c}} \cdot \mathrm{CUPK_b^{-rb}} \cdot \mathrm{R_b^{k_b}}\ \mathrm{modq} \\
&= \mathrm{m} \cdot \mathrm{CUPK_c^{r_c'}} \mathrm{c}^{r_c c^{k_c}} \cdot \mathrm{CUPK_b^{-r_b}} \cdot \mathrm{g}^{\mathrm{rbk}} \mathrm{k}^{\mathrm{kb}}\ \mathrm{modq}
\end{aligned} \tag{1}$$

sharing scheme is configured properly and the scheme is implemented correctly.

6.3 Verifiable

When U_i Before obtaining the original data, one must first authenticate with the C before obtaining it. leftm $\_e$, which will allow him to recover m. TCS and gets $\{m_e, m_V\}$, The user may then devise a strategy for retrieving the m. $m_e \cdot g^{-\alpha} \cdot \left(g^{x_2(CSIDi\|\alpha)}\right)^{-k_a} \text{modN}$, additionally, the correctness of the data m may be verified by the use of a verification technique. $x_1\left(g^{\text{Msg}} \text{modN}\right) \stackrel{?}{=} m_V$. When the user obtains U_i 's shared data, he can verify its validity by checking the validity of the data by checking h_1 left (gleft). Similarity, when the user obtains a certain amount of data, he or she can check the validity of the data by checking the h_1.

6.4 Non-transferable

According to the recommended strategy, it is possible that the user CU_l will get permission to share data from the user CU_b. This is taken into consideration $\left(R_b, m_b^*\right)$ and $\left(m_c, m_V\right)$

from CU_a and TCS who gets the data sharing permission of U_l is the one who needs to know the secret key k_b of CU_b, where k_b of the U_j to recover the original data. This scheme prevents the unauthorized transfer of the data sharing permission of U_j. It is necessary to recover the original data (Table 3).

7 Proposed FHE scheme experimental analysis

Using Table-3 can calculate the time required to store the data during the data storage phase. When it comes to secure data access, time cost of hash operation and modular inversion refer to the parts of the process that require the usage of hash operations. When it comes to authentication and login phases, both the user and the storage provider needs $2Te + 6Th$ for every session.

7.1 Theoretical analysis

Furthermore, the performance of our plan has been evaluated in relation to a number of other comparable schemes using the AVISPA tool. Automated Validation of Internet Security Protocols and Applications (AVISPA) is a push-button tool that provides a modular and expressive formal language for describing protocols and their security features. The various benefits of this tool include the ability to integrate various back ends in order to execute a number of automatic analytic methodologies. It has a great level of scalability and robustness. The AVISPA tool is a commonly used security verification tool that may be used to test a broad range of Internet Protocols and Applications, including, but not limited to, the HTTP and HTTPS protocols. The HTTP protocol for transfer hypertext over the Internet, whereas HTTPS is an extension of HTTP (HTTP). HTTP has been the most extensively used protocol for data transfer over the Web due to its simplicity. It operates at the application layer, while HTTPS is used for secure communication, which is a communication protocol that uses Transport Layer Security to encrypt data. It is also utilized to verify the security of our scheme's security measures AVISPA has four rear ends, which are as follows: There are four types of model verifiers: 1) on-the-fly model verifier, 2) Constraint-Logic attack searcher, 3) SAT-based model verifier, and 4) Tree Automata. All of the players' responsibilities are depicted as fundamental roles in this diagram. The composition roles are also referred to as composition roles in this document. A threat model, developed by DolevYoo, is used to predict the behavior of the attacker. The HLPSL2IF security

Table 3 Time complexity

Function Name	Notation Used	Time (in ms)
Bilinear pairing	T_{bp}	0.0045
ECC point multiplication	T_{ecm}	0.0171
Fuzzy extractor function	T_{fe}	0.0171
Hash function	T_h	0.00032
Modular inverse	T_{inv}	0.00004275
Modular multiplication	T_m	0.00001425
Symmetric encryption/decryption	T_{sym}	0.0056

protocol combines an Intermediate Form and an output format that is used to construct a security protocol. If a protocol fails, the attack trail of the failed protocol is included in the OF if the protocol is unsafe. Also included in this program is a display of overall operation statistics (OI). Specifically, simulation results for the different rear ends of the proposed method were left out of the paper. In addition, the fundamental roles for the different users were established. It is possible to layer the AVISPA and HLPSL implementations on top of one another. Ensuring the HLPSL implementation is done correctly will help guarantee that the security protocol can achieve a particular state. For the execution test, the scheme is carried out in batches and consists of several model checking sessions that are carried out simultaneously. The suggested technique enables authorized agents to carry out a given procedure while also searching for and identifying a passive intruder. This scheme is calculated using the OFMC and CL-AtSe back ends, which are both open source. The scheme is found in around 0.35 seconds after being searched. The depth of the network is around seven plies, and the number of visited nodes is approximately 128.

7.2 Computational complexity

We have computed the computational complexity T_h of the various schemes and cryptographic operations that we have used T_{pm}. The time necessary to do It takes 0.000732 ms to do a hash operation, whereas it takes 0.002975ms to calculate the result of point multiplication. ms. For calculating the computational complexity of this scheme, we have considered the various hash T_h functions that are involved in its operation. The time required for calculating the operation's duration and the number of operations performed are computed in the following tables. Results shows (Figs. 3 and 4, Table 4)

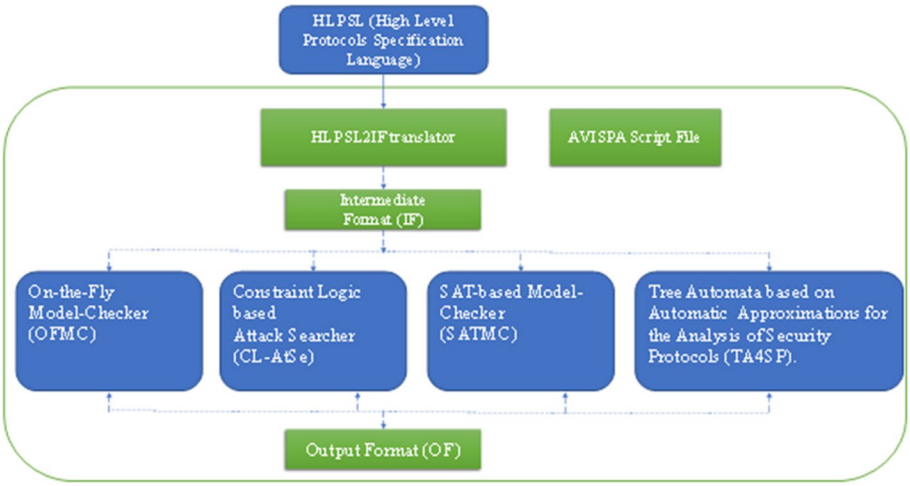

Fig. 3 AVISPA architecture

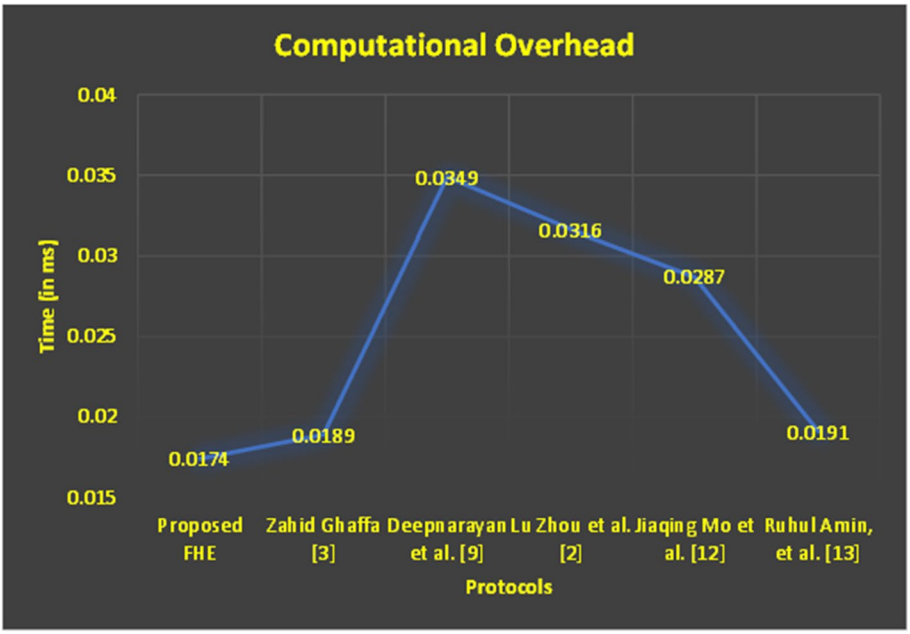

Fig. 4 After other factors, computational overhead

Table 4 Computational overhead

Protocols	Time (in ms)	Operations Performed
[10]	0.0349ms	$17T_h + 7T_{pm}$
[2]	0.0316ms	$38T_h$
[13]	0.0287ms	$13T_h + 6T_{pm}$
[14]	0.0191ms	$23T_h$
[3]	0.0189ms	$15T_h + 5T_{pm}$
Proposed FHE	0.0174ms	$13T_h + 5T_{pm}$

7.3 Storage overhead

The amount of space needed for keeping the different parameters of a specific scheme is referred to as the storage cost of the scheme. It is calculated by dividing the cost of storage by the number of bytes stored. In the table, it can be seen that our plan is around the same price as the other schemes. Our solution, on the other hand, is more cost-effective in terms of storage (Table 5 and Fig. 5).

7.4 Communication overhead

In computing the communication cost, the bit size of each entity's message is taken into account. It is expressed as a percentage of the total amount of bytes available. The establishment of a session between two parties results (Table 6 and Fig. 6).

Table 5 Analysis of Storage overhead

Protocols	Storage Cost(in bits)
Proposed FHE	728
[3]	858
[10]	928
[14]	1696
[2]	1088
[13]	1248

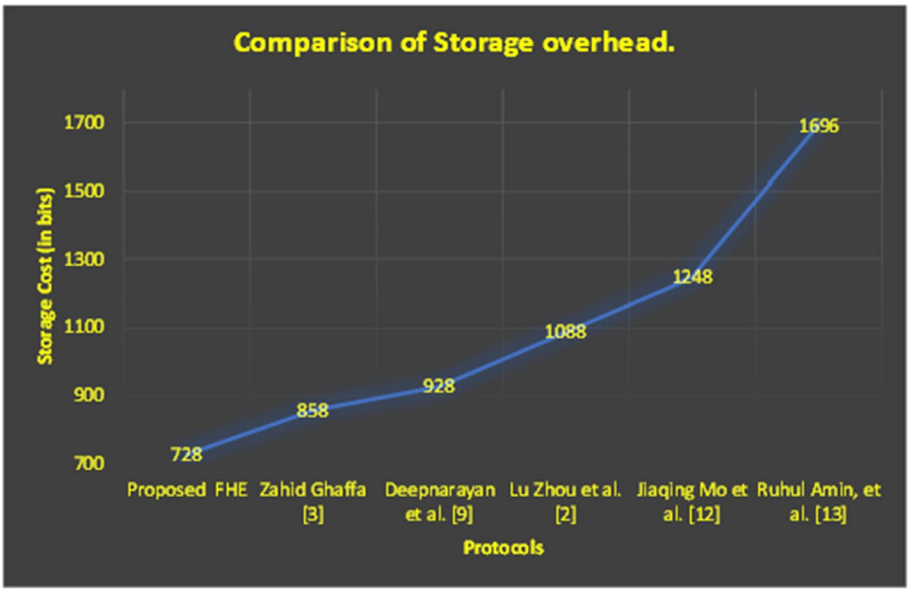

Fig. 5 Analysis of results in Storage overhead the computational cost of various proposed and related protocols. It is also shown in the Y-axis as the computation cost

Table 6 Analysis of Communication overhead in Various Protocols with proposed model

Protocols	Communication Cost(in bits)
Proposed FHE	2646
[3]	2750
[13]	2832
[10]	2896
[14]	6788
[2]	6548

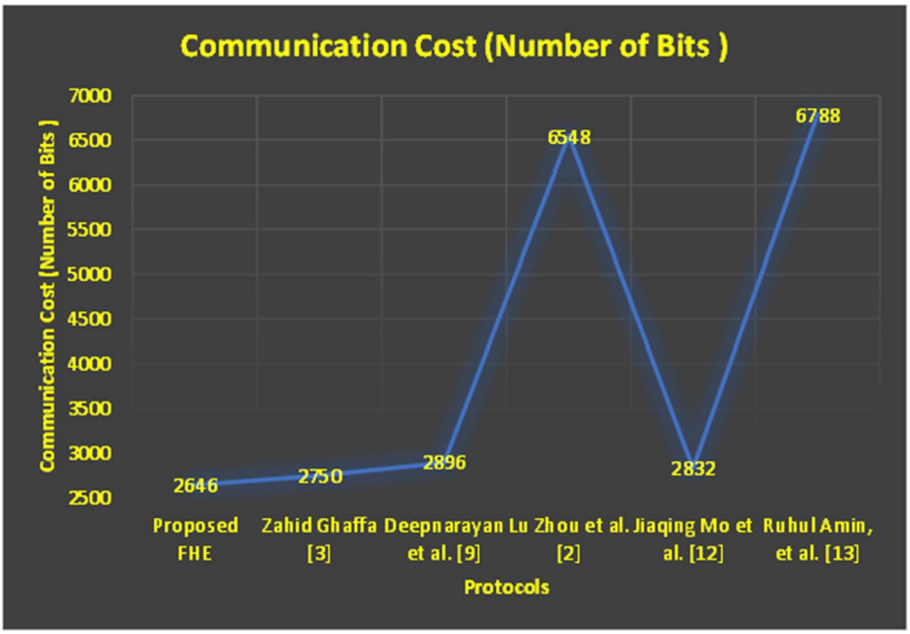

Fig. 6 Comparison of Communication Cost (Number of Bits) in establishing a mutually authenticated connection between the two parties. In data sharing services, the user authenticates by presenting a credential token to the service provider

7.5 Comparison of security features

Cloud storage $\mathcal{C}$ allows you to store and distribute encrypted data AS. The result shows the difference in the number of bits required for communication between the various protocols. Table-5 also shows the same efficiency comparison of the protocols with security features such as Data Confidentiality (F1), Flexible Data Access Control(F2), Man-in-Middle Attack (F3), Mutual Authentication(F4), Non-Repudiation (F5), Password Guessing Attack (F6), Password Stolen Attack(F7), Perfect Forward Secrecy (F8), Provide User Anonymity (F9), Server Impersonation (F10), Stolen Verifier and Privileged Insider Attack (F11) and User Impersonation Attack (F12). After analyzing the various aspects of our proposed protocol, we can conclude that it is more advantageous for our system's resource utilization. It also provides enhanced security features (Table 7).

8 Conclusion and future work

We presented a mechanism for user authentication that limits access to cloud storage to individuals who are not allowed to do so. Additionally, we proposed a safe data sharing system based on the difficult intractable discrete logarithm issue. Numerous elements must be taken into account when determining the security of data sharing for cloud

Table 7 Comparison of security features

Features	15	13	10	14	3	Proposed FHE
F1	X	N/A	X	X	N/A	✓
F2	N/A	N/A	X	X	N/A	✓
F3	X	✓	X	X	✓	✓
F4	✓	✓	✓	✓	✓	✓
F5	N/A	N/A	X	X	N/A	✓
F6	X	X	X	X	X	✓
F7	X	N/A	X	X	X	✓
F8	X	✓	X	X	X	✓
F9	X	✓	X	X	✓	
F10	X	✓	X	X	✓	✓
F11	X	✓	X	X	X	✓
F12	✓	✓	X	X	✓	✓

storage. Among them is ensuring that the data owner has access to the internet in order to spread the data. Cloud applications may benefit from this kind of secure data storage and access method. It incorporates the user attribute rules, biometrics, and Fully homomorphic double encryption necessary for storage provider access. If a user shares his data with another user, the system administrator must revoke the user's authorization to share private data. Data may be moved to another cloud storage provider only with the express permission of the user. Unfortunately, there are several drawbacks to relying on the internet to communicate data. For instance, if the data owner intends to share his or her information with a group, he or she should specify which group. This method has the ability to protect data from unauthorised access. We compared and contrasted the proposed technique to existing studies on cloud storage. Recent techniques have exposed users' privacy by making it simple for an attacker to identify a genuine user.

Data Availability Data analyzed during the research is available upon reasonable request.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., & et al. (2019). Homomorphic encryption standard. Cryptology ePrint Archive.

Ghaffar, Z., Ahmed, S., Mahmood, K., Islam, S.H., Hassan, M.M., & Fortino, G. (2020). An improved authentication scheme for remote data access and sharing over cloud storage in cyber-physical-social-systems. *IEEE Access*, *8*, 47144–47160.

Karati, A., Amin, R., Mohit, P., Sureshkumar, V., & Biswas, G.P. (2021). Design of a secure file storage and access protocol for cloud-enabled internet of things environment. *Computers and Electrical Engineering*, *94*, 107298.

Kaleem, M.A., Khan, P.M., & Khan, U.A. (2021). Strengthening of homomorphic encryption scheme for cloud environment using particle optimization algorithm. In *2021 Fourth international conference on computational intelligence and communication technologies (CCICT)* (pp. 397–405). IEEE.

Li, X., Kumari, S., Shen, J., Wu, F., Chen, C., & Islam, S.K. (2017). Secure data access and sharing scheme for cloud storage. *Wireless Personal Communications*, *96*(4), 5295–5314.

Liu, J., Wang, C., Tu, Z., Wang, X.A., Lin, C., & Li, Z. (2021). Secure KNN classification scheme based on homomorphic encryption for cyberspace. *Security and Communication Networks, 2021*.

Rawal, B.S., & Vivek, S.S. (2017). Secure cloud storage and file sharing. 2017 IEEE international conference on smart cloud (SmartCloud). https://doi.org/10.1109/smartcloud.2017.19.

Yang, X., Zheng, S., Zhou, T., Liu, Y., & Che, X. (2021). Optimized relinearization algorithm of the multi-key homomorphic encryption scheme. *Tsinghua Science and Technology*, *27*(3), 642–652.

Zhou, L., Li, X., Yeh, K.-H., Su, C., & Chiu, W. (2019). Lightweight iot-based authentication scheme in cloud computing circumstance. *Future Generation Computer Systems*, *91*, 244–251.

Zhu, H., Wang, C., & Wang, X. (2021). Quantum fully homomorphic encryption scheme for cloud privacy data based on quantum circuit. *International Journal of Theoretical Physics*, *60*(8), 2961–2975.