Chapter 3

SECURE AND PRIVACY ENHANCED AUTHENTICATION FRAMEWORK FOR CLOUD COMPUTING

3.1. Introduction

Cloud computing facilitates the industries with high performance, larger scale of use, low cost, multi-tenancy and many other benefits. Yet frequent adoption of cloud computing brings more concerns towards the safety and security of data and information. Several methods and a mechanism for providing detailed security in cloud computing have been proposed and implemented. In this chapter, we proposed another method of providing security using a combination of PGP and Kerberos applications in cloud computing. The proposed method provides authentication, confidentiality, integrity, confidentiality and non-inclusion functions for cloud service providers and cloud computing users.

We believe that the cloud privacy model should be customizable and user-centric. To protect sensitive data, the cloud client must be able to flexibly manage and manage various privacy mechanisms [99].

3.2. Cryptography

Cryptography is the discipline of using arithmetic to encode (encrypt) an algorithm for encrypting and decoding (decrypting) data using some inverse algorithm of the encryption algorithm (decryption algorithms). Cryptography allows us to transfer or store sensitive data or information through insecure networks (for example, the Internet) so that no one can read it other than the recipient.

Terminology of Cryptography

- **Plaintext** - the original intelligible message.
- **Ciphertext**- the transformed message.
- **Key** - some acute data used by algorithm, known only to the sender & receiver.
- **Encryption** –The way of hiding plaintext in a method to hide its facts.
- **Decryption**-The procedure of regressive cipher text to its unique plaintext is called decryption.
- **Cryptanalysis (code breaking)** - studying the principles and methods of transforming an incomprehensible message into an understandable message without knowing the key

3.3. Cryptographic Algorithms

This is a mathematical function used in the process of encryption and decryption. The cryptographic algorithm works in conjunction with the key. The same clean text can create a different encryption text with the corresponding different keys.

The security of the encryption text depends entirely on two factors: the strength of the cryptographic algorithm and the secret of the key [100]. In principle, there are two types of cryptographic algorithms:

1. Conventional or symmetric key cryptography
2. Public key cryptography or asymmetric key cryptography.

3.3.1. Conventional Cryptography

Conventional cryptography is also known as a shared secret key or symmetric key cryptography, as shown in figure 3.1, where a common key is used for encryption and decryption mechanisms. The standards for data encryption (DES) and Advanced Encryption Standard (AES) are examples.

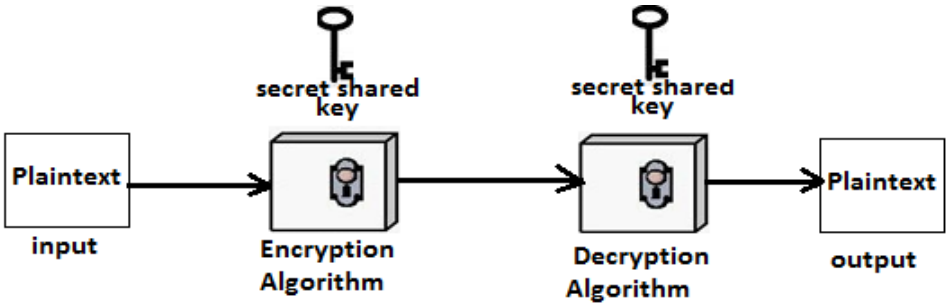

Figure 3.1: Working of Shared key Cryptography

3.3.2. Public Key Cryptography

This is an asymmetric scheme in which a key pair is used for encryption: the public key that encrypts the data, and the corresponding private or private key for decryption, as shown in figure 3.2. This algorithm must publicly publish the public key, keeping the private key. Anyone who has a public key can encrypt the message, and only the owner of the private key can read it. RSA, Diffie-Hellman, Elgamal, etc. are examples [101].

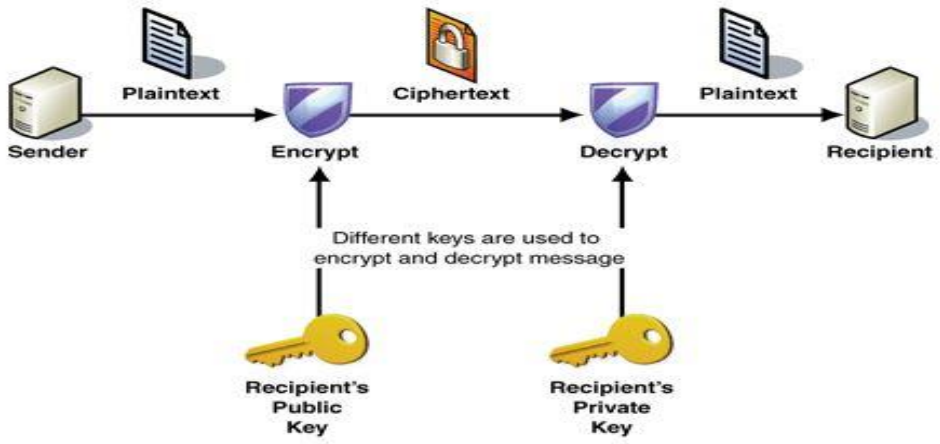

Figure 3.2: Overview of Public Key Cryptography

Working with a public-key encryption system has mainly three phases:

I. **Key Generation:** Anyone who wants to receive secret messages creates a public key (which is published) and a private key (kept secret). The keys are created in such a way that they hide their construction and make it difficult to find the private key, only knowing the public key [101].

II. **Encryption:** A secret message for any person can be encrypted with their public key (which can be officially displayed as phone numbers).

III. **Decryption:** Merely the individual you are dealing with can easily decrypt a secret message using a private key.

3.4. Problem Statement

Traditional authentication methods such as user id and password are not capable to addressing some new challenges in cloud computing. Kerberos has a weakness and limitation such as non-repudiation, which can be exploited by attacker to breach the Kerberos system. In

the cloud, resources are distributed among several tenants, which cause a high risk of attacks. As a rule, a loophole in authentication schemes causes a hacking threat. Surprisingly, attacks based on authentication, which include hacking, four out of five violations due to lack of proper authentication. Attackers reduce the credentials for reuse to gain unauthorized access through social engineering, and sometimes malicious programs capture them using keystrokes, browser caches, or system files. Therefore, an inadequate or unreliable authorization mechanism can significantly increase the risk of unauthorized use of cloud resources and services.

3.5. Existing Work

A lot of work has been done to analyse security and privacy in cloud computing. The requirements for confidentiality and security have been analysed for ubiquitous computing environments (PCEs), and they understand that access control schemes that retain confidentiality do not fully satisfy [102]. This scheme is approaching the control of access to privacy in the PCE to improve confidentiality by achieving traceability and unpretentiousness even against malicious people. Authentication protocol and authorization of anonymous communication in the cloud are described in [103], which presents a solution to existing standards, which facilitates their integration and compatibility with existing standards.

The authors extend the optimization before the federation of identifiers in the Marketplace. This optimization is achieved by entering security steps to establish trust between business applications and resource servers. This trust is associated with the authorization server and clients interested in accessing protected resources. In this architecture, trust is provided and synchronized as the previous step for authentication between all the objects that exchange data in the OAuth protocol. Bind points are used to create a trusted federation for Marketplace applications in all organizations. In their paper [104], the authors proposed a user authentication scheme for cloud computing that provides mutual authentication and negotiation of the session key in a cloud computing environment, but in the end they have to do a lot of mathematics calculations for verification. the cloud computing scenario like in cloud computing, we strive not to impose an additional burden on the user.

An innovative mechanism was proposed to provide flexible access control with a coordinated exchange of resources in the business environment [105]. The mechanism offers a model for users, resources and their relationships that define coordination and access control. The solution is aimed at meeting the requirements that are usually found in the business

environment with respect to controlled exchange of resources. A flexible scheme for biometric authentication of a remote user on paper is proposed [106]. This scheme is vulnerable and can be easily encrypted. They show that their scheme performs only one-way authentication (client authentication only) and that there is no mutual authentication between the user and the remote system, so their scheme is vulnerable to attacking the server spoofing. They also proposed a scheme to improve the security of flexible biometric authentication of a remote user. The author of the article [107] tried to combine a number of passwords and properties based on smart cards and proposed an authentication scheme for two smart cards and passwords, which is still vulnerable to many attacks.

In [107], authors described a method for implementing two-factor authentication using mobile phones. Document [108] proposed authentication based on sending a password of time to a registered mobile phone number. The SMS system does not guarantee delivery of the token in real time. Data can be intercepted by intruders [109].

Bottleneck/parameter: Memory, Bandwidth, Network traffic, CPU sharing management and Disk I/O management are major issues of bottleneck in cloud computing.

- **Memory:** The amount of memory is a key bottleneck for the density of virtual machines. Usually, memory is used, and then stores information to retrieve it later, so memory usage tends to be higher than the processor, and is usually the first bottleneck that virtual environments face.
- **Bandwidth:** For cloud computing, the bandwidth of the cloud provider and from it is a bottleneck. For some applications, the problem is full bandwidth. Some applications use or generate large amounts of data, so the application user may find that there is not enough bandwidth to send data, given the network bandwidth available to the respective operators.
- **Network Traffic:** it affects cloud computing Availability, applications and architectures for the conceivable future. As a cloud user, the network traffic is becoming a far larger part of application deployment will affect cloud computing services.
- **CPU Sharing Management:** Avoid performance bottlenecks: CPU sharing is often a bottleneck, which is due to problems with disks or I / O memory. It is necessary to control the infrastructure so that each element receives all allocated resources.

- **Disk I/O management:** Disk I / O operations are usually a bottleneck, followed by a memory or processor problem. A delay is a measure of the time it takes to return a disk command after it is entered and measured in milliseconds.

3.6. Existing Methods

We have used Kerberos and Pretty Good Privacy in our proposed method.

3.6.1. Kerberos

For network security, Kerberos is an authentication protocol that is used based on cryptography. It ensures the integrity of the message and the confidentiality of the data. It uses cryptography of a secret key to verify the authenticity of parties communicating over networks, while avoiding spyware or repeated attacks [110].

3.6.1.1. Components of the Kerberos (Servers)

There are three servers i.e. Authentication Server (AS), Ticket Granting Servers (TGS) and real server (CSP) that provides services to others.

a) Authentication Server (AS)

It is included in a separate KDC group recorded using Authentic Servers, and receives a user ID and PIN, and stores those identifications in its record of each user. Authentic Server confirms the customers and provides a session key that will be used among the customers and the TGS.

b) Ticket Granting Servers (TGS)

It facilitate the communication between user (A) and the real server (B) by issuing a session key (K_{AB}) for A and B.

c) Real Server

It is the actual server that provides services to the users.

3.6.2. Working of the Kerberos

3.6.2.1. Authentication

When B (user) wants to access a service from server (N), then validation process is as depicted in figure 3.3.

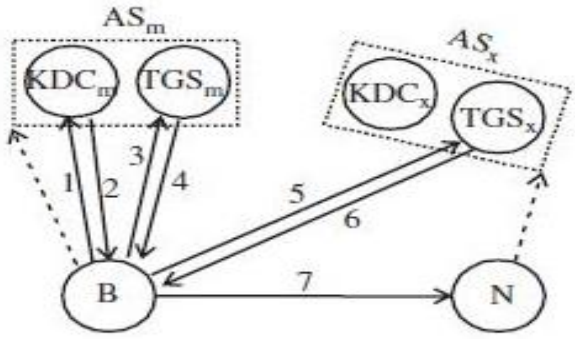

Figure 3.3: Authentication

We explain the steps of communication as follows.

Step1- The client logs on the workstation and sends the requests access a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with TGS ID, indicating a request to use the TGS service. (1) U $\oplus$AS: $ID_c ID_{tgs}$ TS_1

Step2- The AS transmits a note encoded by User's (A) key as

AS$\rightarrow$ U: $Ek_c[k_c, tgsID_{tgs}]$TS2 lifetime2 $Ticket_{tgs}$.

Step3- User (A) now sends three items to the TGS. The first is the ticket received from AS and the second is the name of the real server (B) (i.e. Cloud Service Provider), and the third is a timestamp that is encrypted by $K_{A\text{-}TGS}$. The timestamp prevents a replay by Eve.

Step4- Now, the TGS sends two tickets, each containing the session key between user(A) and real server(B). $K_{A\text{-}B}$, the ticket for user (A) is encrypted with $K_{A\text{-}TGS}$; the ticket for server (B) is encrypted with B's public key $K_{TGS\text{-}B}$. Note: Eve cannot extract $K_{A\text{-}B}$, because Eve does not know $K_{A\text{-}TGS}$ and $K_{TGS\text{-}B}$, even she cannot replay step-3 because Eve does not replace the timestamp with new one (she does not know $K_{A\text{-}TGS}$).

Step5- User (A) sends Server (B) ticket with the timestamp encrypted by $K_{A\text{-}B}$.

Step6- Real server B confirms the receipt by adding 1 to the timestamp. The message is encrypted with $K_{A\text{-}B}$ and send to user (A). Since PGP support digital signature and public key cryptography. After successful authentication by Kerberos the user (A) initiate the PGP for next authentication and data encryption process for confidentiality and data integrity. We know that the digital signature provides message authentication and integrity. So the sender (User) and the receiver (CSP) agree on the PGP.

3.6.3. Pretty Good Privacy (PGP)

PGP was invented by Philip R. Zimmermann in 1991 [111]. PGP provides cryptographic authentication and confidentiality for data transmission. It is used for e-mail applications and file storage to ensure confidentiality, integrity and authentication [112].

3.6.3.1. Working of PGP

a) Session Key Generation

First, PGP generates a random phrase (session key), which is a one-time secret key. This session key used to encrypt plain text.

b) Data Encryption

This encryption process can be understood in figure 3.4. Simple text or data is encrypted with a session key, and after the data is encrypted, this session key also ciphered with the public key of the recipient. This session key, encrypted with a public key, is sent to the recipient along with the encryption text [113].

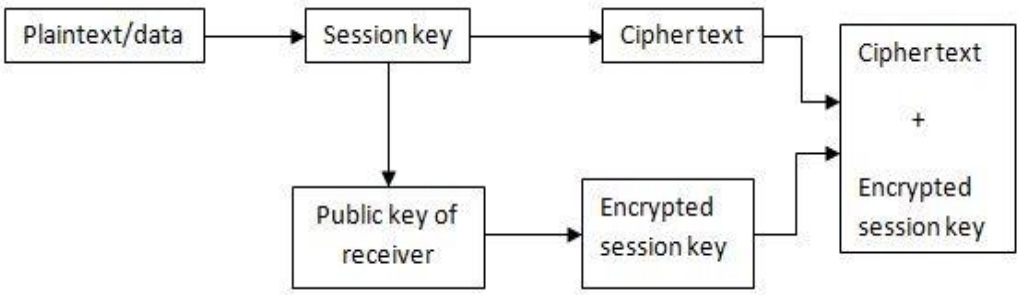

Figure 3.4: PGP Encryption

We need to share session key with authorized data consumers, so only they can decrypt the data.

c) Decryption

Decryption process shown in figure 3.5, it is reverse process of the encryption. The user uses his or her private key to recover the temporary session key, after getting session key user decrypts the data with this session key.

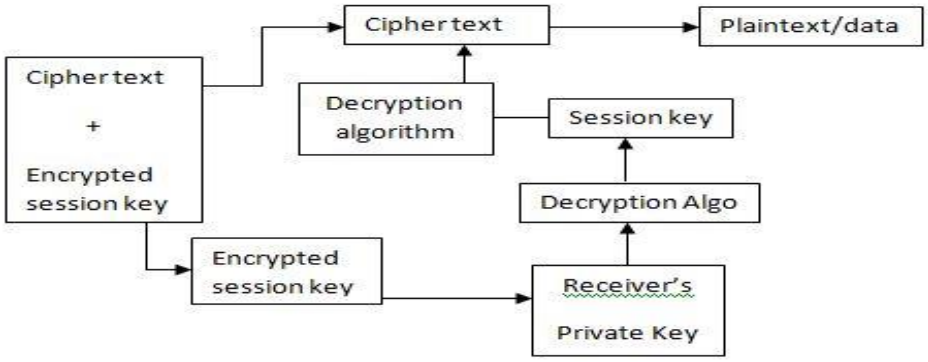

Figure 3.5: PGP Decryption

3.6.3.2. Keys and Key Rings

A key is a value to create a cipher-text using a cryptographic algorithm.

a) ***One-time session symmetric keys***

It is a random session key used for once.

b) Public keys or key ring

This makes it easier for the user to use multiple public key / private key pairs. As a result, there is no individual correspondence between users and their public keys [114]. Therefore, some means are needed to identify certain keys. The public keys of other users are stored in the user's open ring. This contains the following;

- Timestamp: The time the record was created.
- Key ID: 64 least significant digits of this entry.
- Public key: the public key for recording.
- User ID: the identifier of the owner of this key. Multiple identifiers can be associated with one public key.
- The public key can be indexed using a user identifier or a key identifier.

c) Private keys or key ring

Each PGP user must maintain a file with their own public / private key pairs, as well as a public key file of the corresponding correspondent. Each user supports two data structures with a ring: a ring with private keys for their public / private key pairs and an open key ring for public key correspondents. Private keychains:

- Timestamp: when a key pair was created.

- Key ID: 64 least significant public key bits.
- Public key: the open part of the key.
- Private key: private part, encrypted using a passphrase.
- User ID: usually the user's email address. It can be different for different key pairs.

d) Passphrase based symmetric keys

An exhibition to confirm the authenticity of the sender. A passphrase is a sequence of words or other text used to control access to a computer system, program, or data. The passphrase is similar to the password used, but overall it is more secure. Password phrases are often used to control the access and operation of cryptographic programs and systems.

3.6.4. Authentication by PGP

The authentication uses digital signature scheme with hashing in the following steps:

Step1-Sender A has (private/public) K_{pr} / K_{pb} key pair and she wants to send a digitally signed message *m* to receiver B.

Step2-A calculate digest of the message using SHA-1 to obtain *SHA(m).*

Step3-A calculate ciphertext c as $c=pk.encrypt K_{pr}(SHA(m))$

Step4-A sends pair *(m,c)to* receiver.

Step5-B receives *(m,c)* and calculate D(c) using sender's public key K_{pb} to obtain signature *s as* $s=pk.decrypt K_{pb}(c)$

Step6-B calculates the digest of *m* using SHA-1 and if this digest value is equal to *s* then the message is authenticated.

In this process B is sure that the message is correct and came from authentic A. Furthermore, A cannot later deny sending the message since only the sender has access to his/her private key K_{pr} which works in conjunction with the public key K_{pb}.

3.7. The Proposed Method

Since Kerberos is incompatible with non-repudiate, this Kerberos weakness can be reduced with public key cryptography and digital signature, so we implemented PGP with Kerberos, because PGP supports digital signatures of public key cryptography. Figure 3.6 illustrates the complete architecture of the proposed methods.

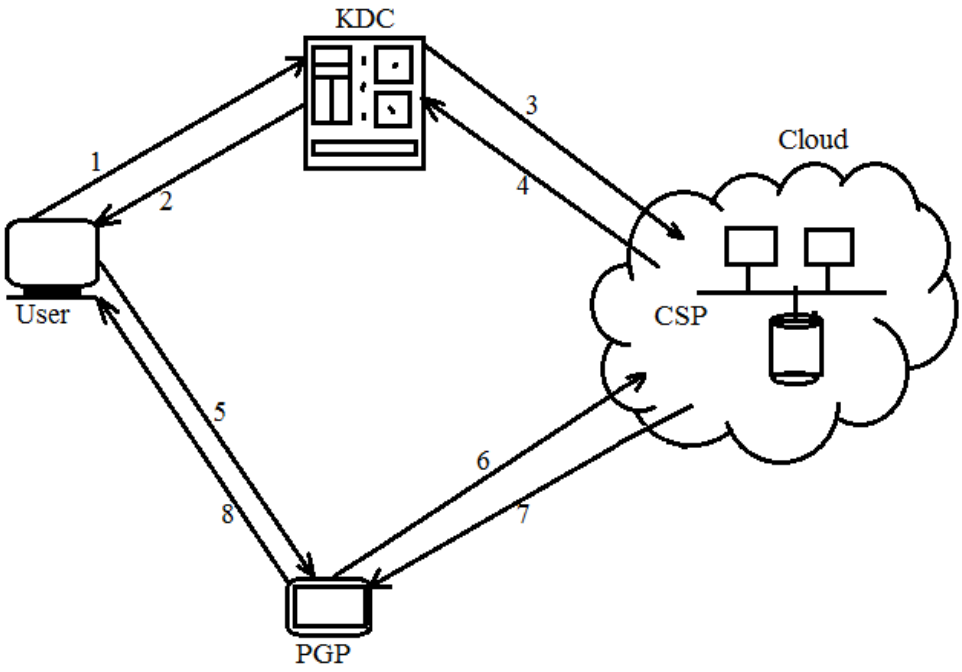

Figure 3.6: Proposed Method for Cloud Computing

Step1- User registers his identity to Kerberos (KDC).

Step2- KDC provides ticket to user to communicate with CSP.

Step3- KDC also sends a ticket and user identity to CSP, now CSP stores these credentials for future use.

Step4- CSP acknowledge to KDC about user's credentials storage.

Step5- User encrypts his data before sending to cloud.

Step6- PGP authenticate user and send information to CSP. Also PGP send user's encrypted data to cloud.

Step7- The CSP send the desired data to PGP requested by user.

Step8- The PGP decrypts the data and user authentication information. If user is authorized to access that data the PGP will send the decrypted data to user.

3.7.1. Working of Proposed Method

Figure 3.7 depict the Kerberos Authentication in Cloud Computing.

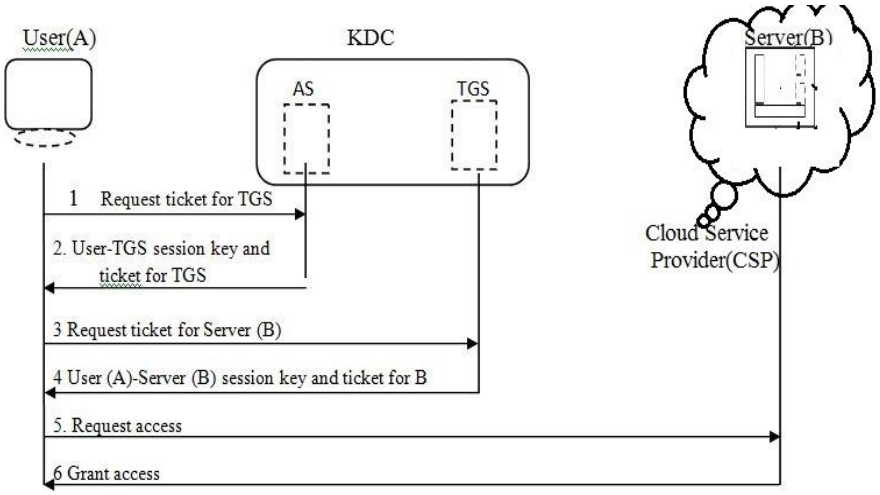

Figure 3.7: Kerberos Authentication in Cloud Computing

Step1- The client login the workstation and sends the requests access a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with TGS ID, indicating a request to use the TGS service. (1) U $\oplus$AS: $ID_c ID_{tgs}$ TS_1

Step2- Authentication Server transmits a note encoded with $K_{A\text{-}AS}$. As

AS$\rightarrow$ U: $Ek_c[k_c, tgsID_{tgs}$ TS2 lifetime2 $Ticket_{tgs}$.

Step3- User (A) now sends three items to the TGS. The first is the ticket received from AS and the second is the name of the real server (B) (i.e. Cloud Service Provider), and the third is a timestamp that is encrypted by $K_{A\text{-}TGS}$. The timestamp prevents a replay by Eve.

Step4- Now, the TGS sends two tickets, each containing the session key between user(A) and real server(B). $K_{A\text{-}B}$, the ticket for user (A) is encrypted with $K_{A\text{-}TGS}$; the ticket for server (B) is encrypted with B's public key $K_{TGS\text{-}B}$. Note: Eve cannot extract $K_{A\text{-}B}$, because Eve does not know $K_{A\text{-}TGS}$ and $K_{TGS\text{-}B}$, even she cannot replay step-3 because Eve does not replace the timestamp with new one (she does not know $K_{A\text{-}TGS}$).

Step5- User (A) sends Server (B) ticket with the timestamp encrypted by $K_{A\text{-}B}$.

Step6- Real server B confirms the receipt by adding 1 to the timestamp. The message is encrypted with $K_{A\text{-}B}$ and send to user (A). Since PGP support digital signature and public key cryptography. After successful authentication by Kerberos the user (A) initiate the PGP for next authentication and data encryption process for confidentiality and data

integrity. We know that the digital signature provides message authentication and integrity. So the sender (User) and the receiver (CSP) agree on the PGP.

3.7.1.1. Authentication and Integrity

During the validation steps, the sender first computes a summary of the data message whose picture illustrates the process of the digital signature service provided by the PGP, as shown in the figure 3.8.

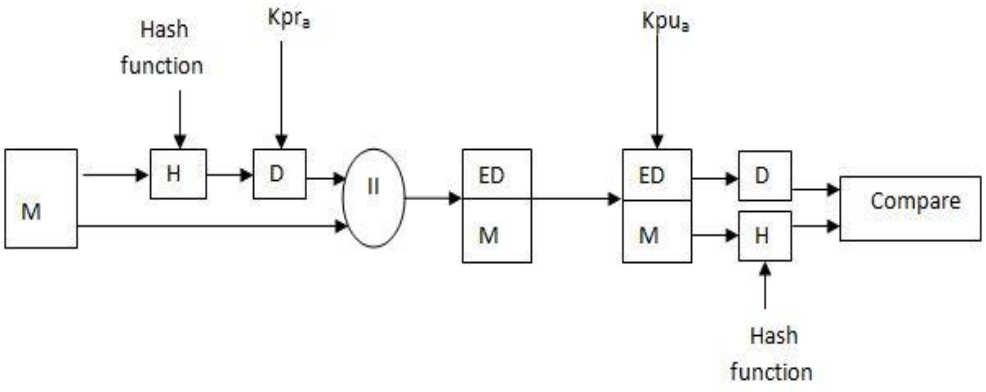

Figure 3.8: Authentication by Proposed Framework

Step1- User calculates the message digest of the message.

Step2- After calculating digest he encrypts this digest with his private key (put his digital signature).

Step3- He concatenates the original message with encrypted message digest and sends to the Cloud service provider.

Step4- the CSP decipher the digest by using user's public key and get the original digest of the message.

Step5- the CSP calculates the message digest of the message received using same hash function.

Step6- If both digest comparison calculates same; it shows that the sender is authentic user, whose public key is available to CSP repository. Also calculated digest show that the integrity of the message is uniform.

3.7.1.2. Confidentiality

The PGP provides confidentiality using several steps, which are shown as follows

Step1- User compresses his message using appropriate compression algorithm and then encrypts the compressed message with a session key.

Step2- After encrypting the compressed data the session key is also encrypted with public key of CSP.

Step3-The encrypted data and session key are concatenated and sent to the CSP.

Step4- At CSP end, after receiving data from user, the CSP decrypts the session key using his private key and finds session key.

Step5- After getting the session key, he decrypts the message using that session key and finds the compressed message.

Step6- After getting compressed data in step-5, the CSP uses appropriate decompression algorithm and finds the original message.

Figure 3.9 shows the overall processes of confidentiality as

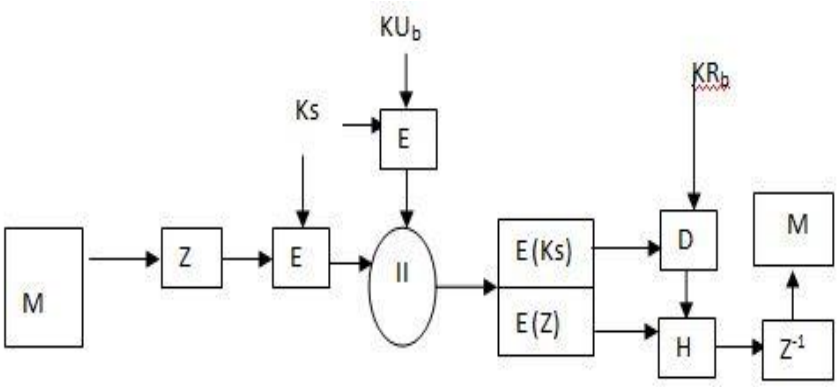

Figure 3.9: Confidentiality by Proposed Framework

3.7.1.3. Non-Repudiation

An Authentication, which can be considered genuine with a high degree of security. This property of the message, which provides proof of the integrity and origin of the data [113]. Non-negativity is a property that prevents a physical or legal person from refusing to perform a certain action related to data through cryptographic methods [114]. The stages are shown in the figure 3.10.

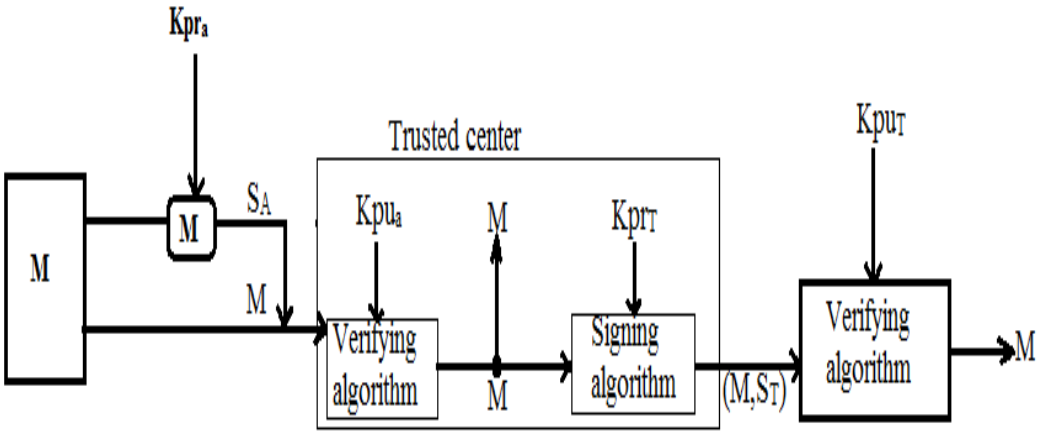

Figure 3.10: Non Repudiation Proposed Framework

Step1- User (A) generates a sign by his note (S_A) and transmits the message, his ID, CSP's id to the center.

Step2- The KDC, verify the user's public key (Kpu_a) is valid, to ensure that message is came from user A.

Step3- The KDC keeps the message sender id, recipient's id and a timestamp in its record.

Step4- The KDC also generates additional sign S_T by using its private key from the message the center sends the message, the new signature user A's identity and CSP's identity to CSP. CSP verifies the message using the public key of the trusted center.

Now, if in the future the user (A) denies that he sent the message, the center can now display a copy of the saved message. If the CSP message is a duplicate of the message stored in the center, the user (A) will lose the argument. To make everything confidential, you can add the encryption / decryption layer to the scheme.

3.8. Security Analysis

In this section, we analyze our proposed method using the SPAN tool (WASP) in various security configurations. SPAN is designed to help protocol developers write HLPSL specifications. Beginning with the HLPSL specification, SPAN helps you interactively create a sequence of graphical messages (MSCs) to execute the protocol. Because SPAN implements an active attacker, it can also be used to interactively search and create attacks on protocols.

3.8.1. Automated Validation of Internet Security Protocol and Application (AVISPA)Tool

The AVISPA provides a language called the High Level Protocol Specification Language (HLPSL), for describing security protocols and determining expected security properties, and a set of tools for their official confirmation [115]. Figure 3.11 shows the architecture of the AVISPA tool.

.

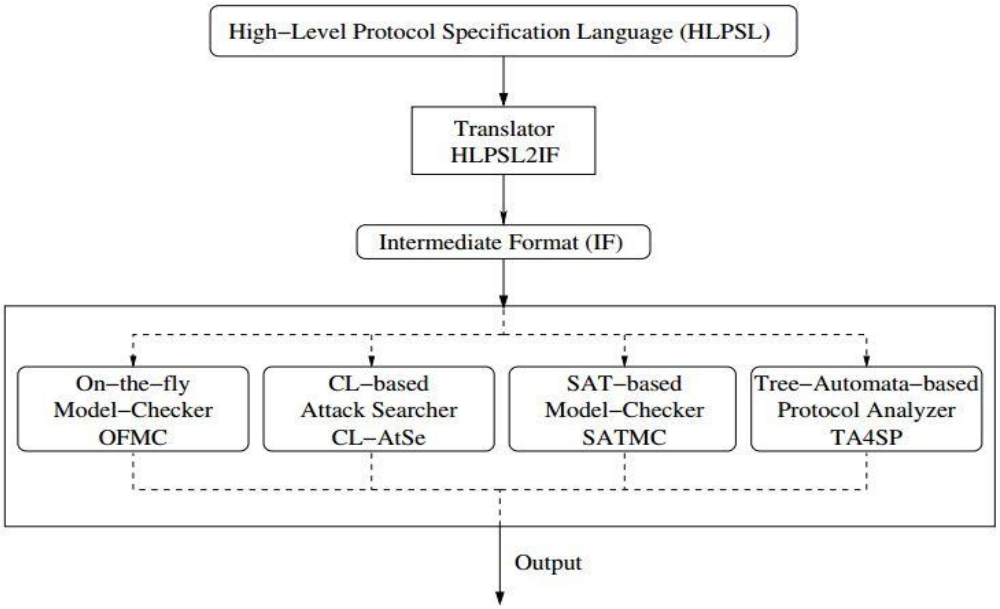

Figure 3.11: AVISPA Tool Architecture

3.8.1.1. The IF

The IF specification describes the protocol from the point of view of rewriting rules that describe an infinite state transition system with initial state, transition rules and state-based state, that is, an objective predicate (attack) that determines whether a particular state is an attack or not. (Tracking attacks is a route that leads from the initial state to the state of attack).

The AVISPA Tool integrates four back-ends:

- On-the-fly Model-Checker OFMC
- Constraint-Logic-based Attack Searcher CL-AtSe
- SAT-based Model-Checker SATMC, and
- TA4SP protocol analyse

3.8.2. Experimental Analysis

In this section, we analyzed the TLS protocol using SPAN, and based on this analysis, we tested our protocol and analyzed the results of our protocol, and we will work with some other protocol.

- **Visualization:** TLS as defined in AVISPA can be visualized using SPAN as shown in figure 3.12 and figure 3.13 depicts the intruder simulation of TLS protocol.

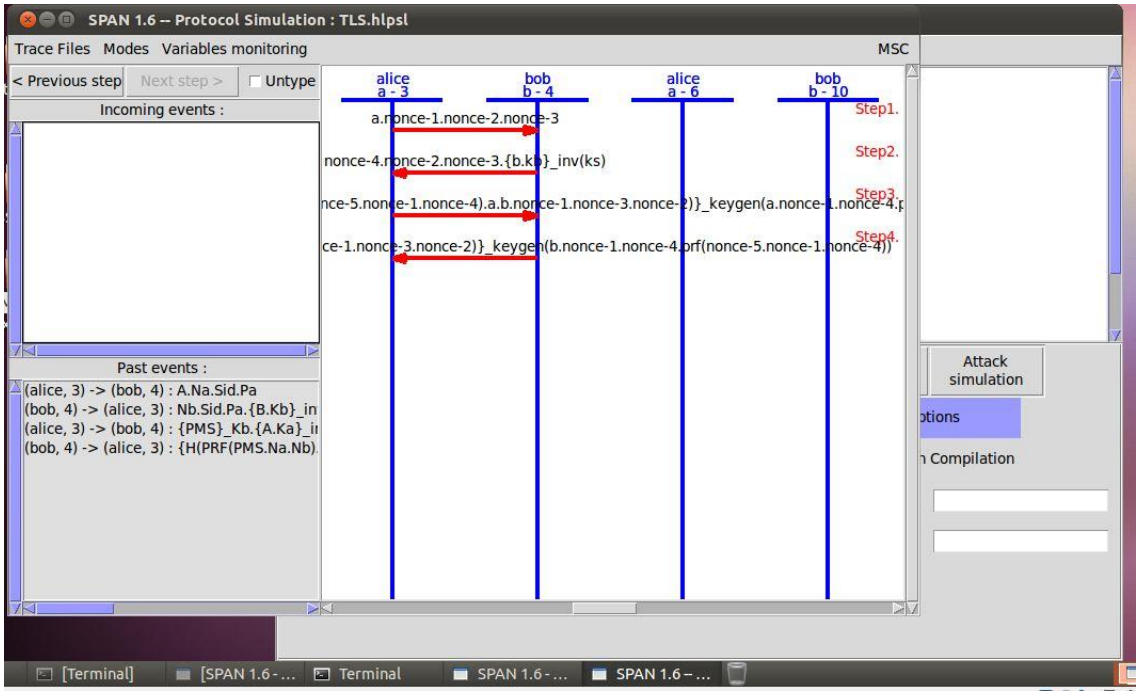

Figure 3.12: TLS visualized using SPAN

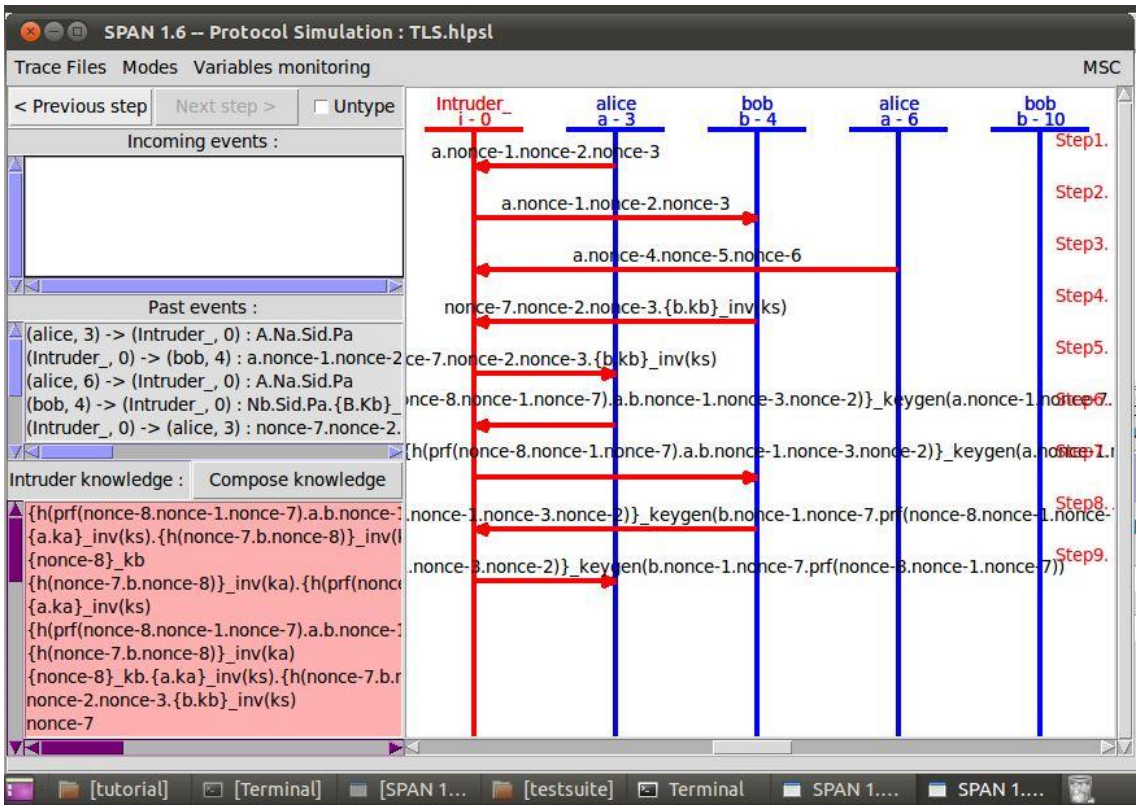

Figure 3.13: TLS Intruder Simulation

- **Result Summery:** The table 3.1 shows the result summery of the TLS protocol in various backend.

BACKEND	**OFMC**	**ATSE**	**SATC**	**TA4SP**
SUMMARY	SAFE	SAFE	SAFE	Inconclusive
GOAL	Authentication	Integrity	privacy	Secrecy
ATTACK TRACE	No attack found	no attacks	no attacks	Use a sub-approach to show a possible attack. An attacker can learn critical information

COMMENTS	parseTime: 0.00s searchTime: 0.01s visitedNodes: 4 nodes depth: 2 plies			TA4SP uses the abstractions '2AgentsOnly' Use approximate approximation to show potential attack An attacker may know some critical information

Table 3.1: TLS Protocol Analysis

- **The proposed Protocol**

Step1- The user request to AS for required service. The client login into the workstation and sends the requests access a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with TGS ID, indicating a request to use the TGS service. (1) U $\oplus$AS: $ID_c ID_{tgs}$ TS_1

Step2- The AS transmits a note encoded by User's (A) symmetric key, $K_{A\text{-}AS}$. $K_{A\text{-}TGS}$ and the ticket are extracted.AS$\rightarrow$ U: $Ek_c[k_c, tgsID_{tgs}]TS2$ lifetime2 $Ticket_{tgs}$.

Step3- User (A) now sends three items to the TGS. The first is the ticket received from AS and the second is the name of the real server (B) (i.e. Cloud Service Provider), and the third is a timestamp that is encrypted by $K_{A\text{-}TGS}$. The timestamp prevents a replay by Eve.

Step4- Now, the TGS sends two tickets, each containing the session key between user(A) and real server(B). $K_{A\text{-}B}$, the ticket for user (A) is encrypted with $K_{A\text{-}TGS}$; the ticket for server (B) is encrypted with B's public key $K_{TGS\text{-}B}$. Note: Eve cannot extract $K_{A\text{-}B}$, because Eve does not know $K_{A\text{-}TGS}$ and $K_{TGS\text{-}B}$, even she cannot replay step-3 because Eve does not replace the timestamp with new one (she does not know $K_{A\text{-}TGS}$).

Step5- User (A) sends Server (B) ticket with the timestamp encrypted by $K_{A\text{-}B}$.

Step6- Real server B confirms the receipt by adding 1 to the timestamp. The message is encrypted with $K_{A\text{-}B}$ and send to user (A). Since PGP support digital signature and public key cryptography. After successful authentication by Kerberos the user (A) initiate the PGP for next authentication and data encryption process for confidentiality and data integrity. We know that the digital signature provides message authentication and integrity. So the sender (User) and the receiver (CSP) agree on the PGP.

3.8.2.1. Protocol Simulation with AVISPA

We execute the HLSPL code in AVISPA the corresponding Message Sequence Chart is the outcome of the tool. Figure 3.14 shows part of the specification of the Kerb-PGP protocol and the message sequence chart of the attack trace that the AVISPA tool found when analysing the protocol in the bottom window.

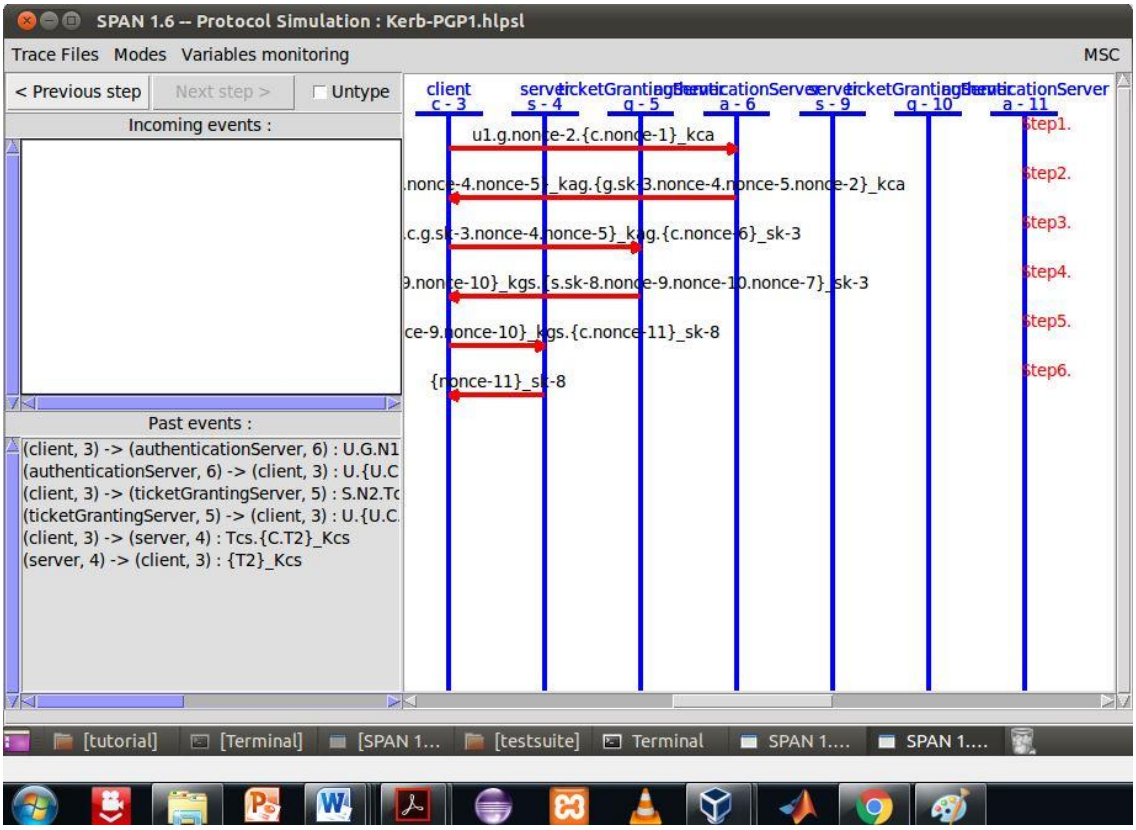

Figure 3.14: Protocol Simulation (Message Sequence Chart) Result

Summary

The validation results of Kerb-PGP protocol are shown in table 3.2 to perform strong authentication of both participants and to successfully negotiate keys.

BACKEND	OFMC	ATSE	SATC	TA4SP
SUMMARY	SAFE	SAFE	Inconclusive	Inconclusive
GOAL	Authentication	Integrity	Confidentiality	Non repudiation
ATTACK TRACE	No attack found	no attacks	no attacks	no attacks

COMMENTS	parseTime: 0.00s searchTime: 0.49s visitedNodes: 99 nodes depth: 10 plies	Analysed: 837 state Reachable: 483 States Translation: 0.04 seconds Computation: 0.57 seconds		TA4SP uses abstractions '2AgentsOnly' Use an under-approximation in order to show a potential attack The intruder might know some critical information

Table 3.2: Kerb-PGP Protocol Analysis

3.8.3. Analysis

We tested several existing network security protocols, such as TLS and many others, as well as our proposed protocol (for example, Kerb-PGP), and we compared the result with several parameters, such as execution time, vulnerability (number of threats) and satisfied properties for protocol, that is, authentication, confidentiality (confidentiality), integrity and non-repudiation. From the comparison table 3.3, we can say that our protocol is much safer than other protocols, as shown in the table. Our protocol satisfied the properties of security, that is, authentication, confidentiality (confidentiality), integrity and non-repudiation through PGP. But execution time is slightly longer than TLS and EKE2, as shown in table 3.3, whereas TLS and other protocols only satisfy three security properties.

Property: Includes Confidentiality, Authentication, Integrity and Non-repudiation.
Threats: Attacks including vulnerability (intruders knowledge about protocol secrete keys)
Time: execution of the protocol

Protocol	**Properties**				**Threats**	**Time**
	C	**I**	**A**	**NR**		
TLS	Y	Y	Y	N	0	0.01
EKE2	Y	Y	Y	N	0	0.16
SPEKE	Y	Y	Y	N	0	3.11
IKEv2-DSx	Y	Y	Y	N	0	42.56
Kerb-PGP	***Y***	***Y***	***Y***	***Y***	***0***	***0.28***
IKEv2-MACx	Y	Y	Y	N	0	40.54
h.530	Y	Y	Y	N	1	0.64

Table 3.3: Comparative Analysis of the Proposed Method with existing Protocol

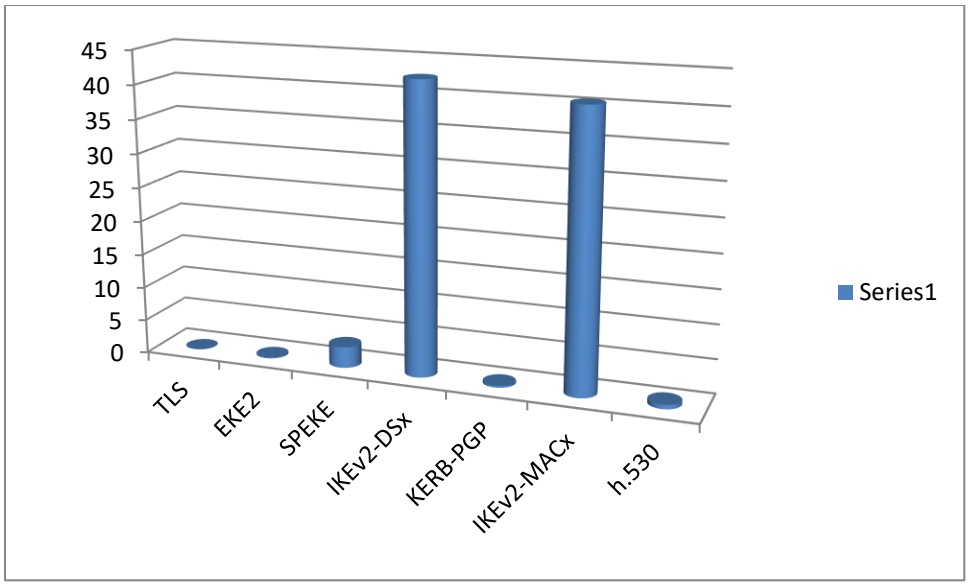

Figure 3.15: Existing Protocols Vs. Kerb-PGP (Execution Time)

3.8.3.1. Security against brute force Attack

The attacker tried to use all possible combinations to guess the private key during the attack of brute force. In the original RSA, the likelihood of rejecting this attack will be significantly reduced by selecting exponents greater than 2,048 bits.

3.8.3.2. Mathematical Attack

A mathematical attack will occur when determining p, q or pq, and it could be avoided using the 2048 bits in RSA. This could also be prevented by increasing the value of the digest *h*, the possibility of a successful mathematical attack will be greatly reduced.

3.8.3.3. User Privacy

The proposed scheme never transmits the user's personal data in text format. Messages are transmitted through a public channel. Obviously, these messages cannot be easily decoded to obtain ID, PW, etc. So the scheme provides privacy for the user.

3.8.3.4. Identity Management

KDC and CSP store all registered identifiers in the database and verify the presence of a unique identifier in each new record and provide a certificate for managing the user's identity.

3.8.3.5. Session Key Agreement

In the proposed method, the secret key (K) is shared by both the AS and the user. Using this key, they can communicate with each other for a specific session. Since this key is generated randomly, it cannot be easily broken.

We compared our method to the existing system [117] based on several parameters, such as 1. Security against a person in the middle attack, 2. User privacy, 3. User identification and 4. Non-Repudiation, which can be shown as figure 3.16.

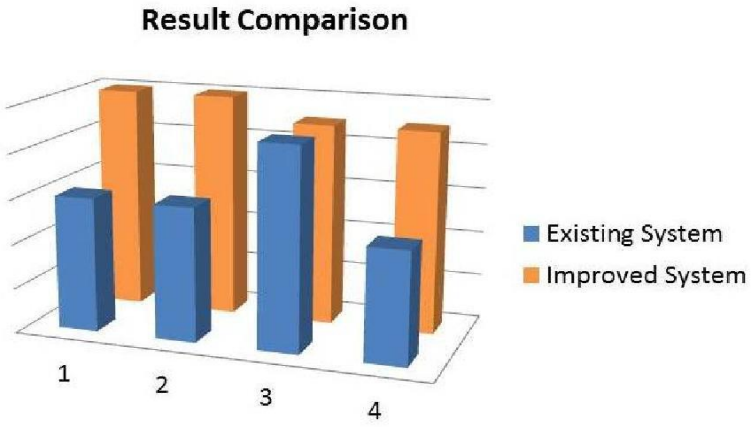

Figure 3.16: Result Comparison

3.9. Technical Comparison of Various Authentication Modes

The table 3.4 shows the technical comparison of various authentication models.

Modes	PPPoE	WEB Portal	802.1X	SSO	Kerberos	PGP
Standard level	RFC 2516	Private	IEEE Standard	No	RFC1510/4120	RFC 3156
Encapsulation Overhead	Relatively high	Low	No	Relatively high	Very High	high
Access Control	User	VLAN-MAC-IP	User	User/system control/ Agent	Centralized user / system default	user
IP Address Allocation	After authentication	Before authentication	After authentication	Not need to allocate	May not allocate	Not allocate
Multicast Support	Poor	Good	Good	Support	Support	

VLAN Demand	No	Much	No	No	No	no
Client Software	Need	Browser	Need	Need/Browser	OS build-in	Need
Device Support	Public protocol	Manufacturer private	Public protocol	Public and compatible	Public protocol	Public
User Connectivity	Good	Poor	Good	Good / With DB ticket user can shares resources	Relatively good	good
Device Performance	Relatively High (BAS)		Low	Relatively High (Multi-system	High	high

Table 3.4: Technical Comparison of the Various Authentication Modes

3.10. Implementation

We have implemented our protocol and run it on localhost. Following specifications are required for this purpose

- **Minimum System Requirements**
  - Machine : Pentium 4, 64 bit OS
  - Processor: 1.90 GHz
  - Ram:4 Gb
  - HDD: 100 GB
  - Operating System: Windows7 SP1
  - Front end: eclipse-jee-kepler-SR2 and jdk-8u66-windows
  - Back end: apache-tomcat-7.0.65
  - Server: XAMPP-win 32-1.8.1-VC9
- **Client Side**

In client side the user has provided registration page in which he enter his personal information like first name, last name, date of birth, email id and mobile number etc as shown in

figure 3.17. after entering these information's, the user gets login id and password. Now the user log in into the login page and after validation of his credentials, he/she can request the service from cloud service provider. For getting the services now user will be further authenticated by Kerberos and PGP. After successful verification user is able to access the services from cloud computing. The following screen shots depict the various steps to access the services from cloud computing.

1. **Client Registration**

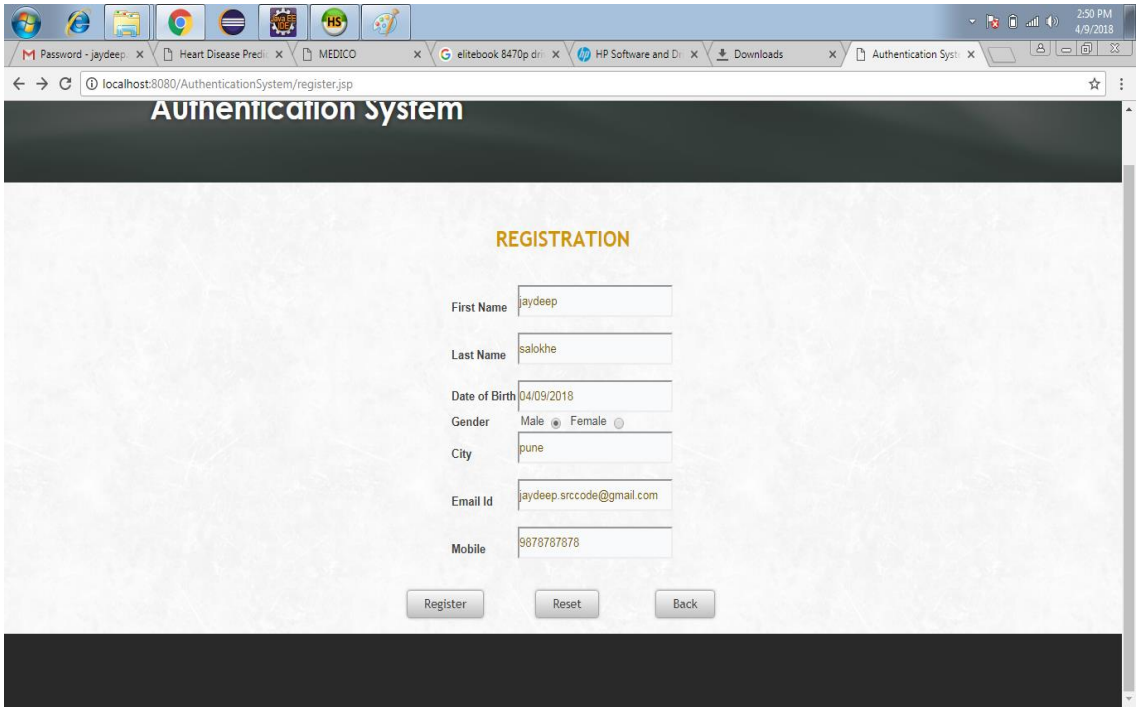

Figure 3.17: Client Registration

2. **Login –** During login process user enter his user id and server sent an otp as password for successful login as figure 3.18 depicts the login step.

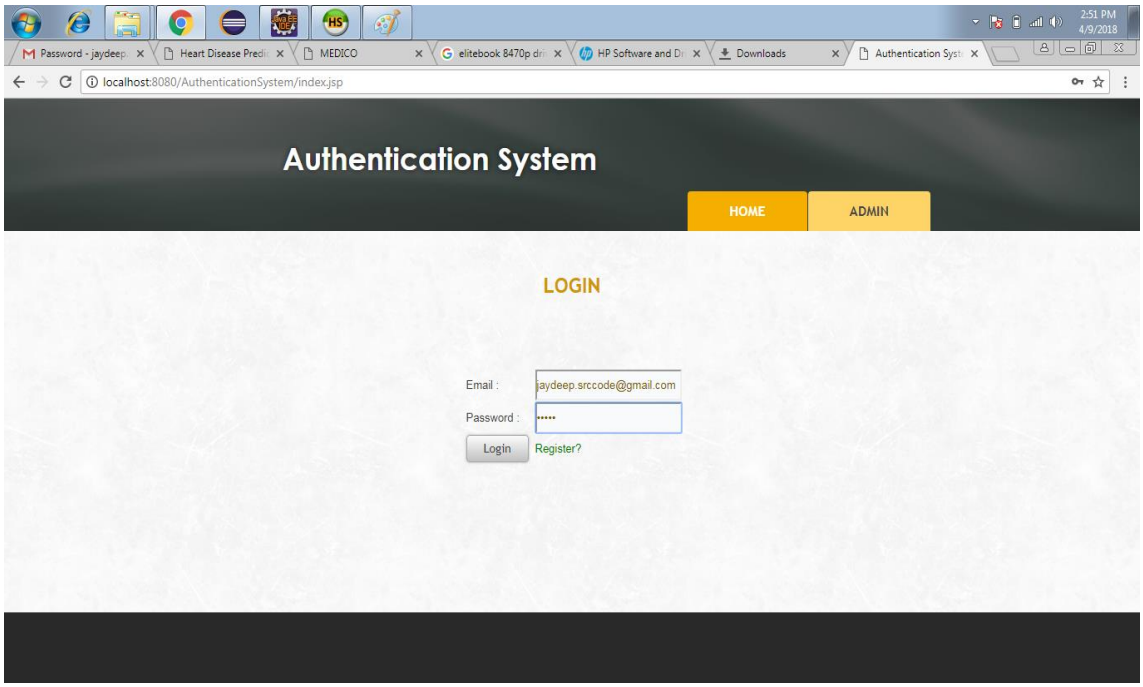

Figure 3.18: Login - by entering username and password

3. **Service Request**: The figure 3.19 depicts the sending request to authentication server for service.

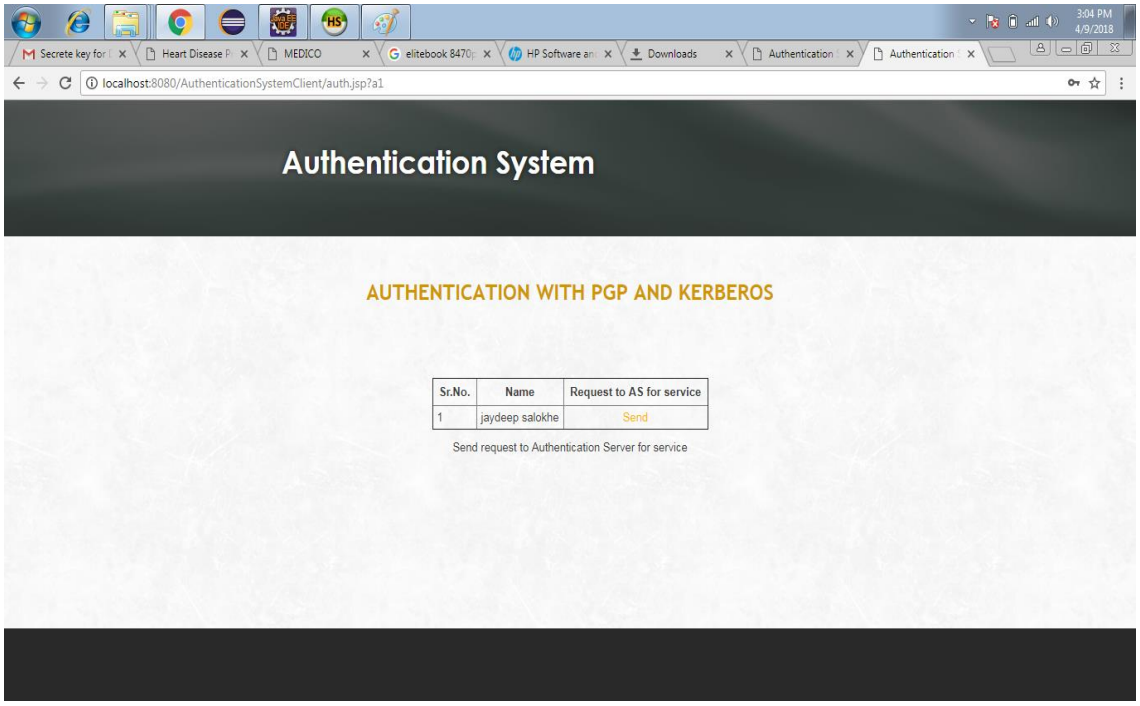

Figure 3.19: Send request to authentication server for service

4. **Public Key generation:** Authentication server sends you encrypted message to check authorization. If you can decrypt this message using your public key then you became authorized user and public key generated on your maid id as shown in figure 3.20.

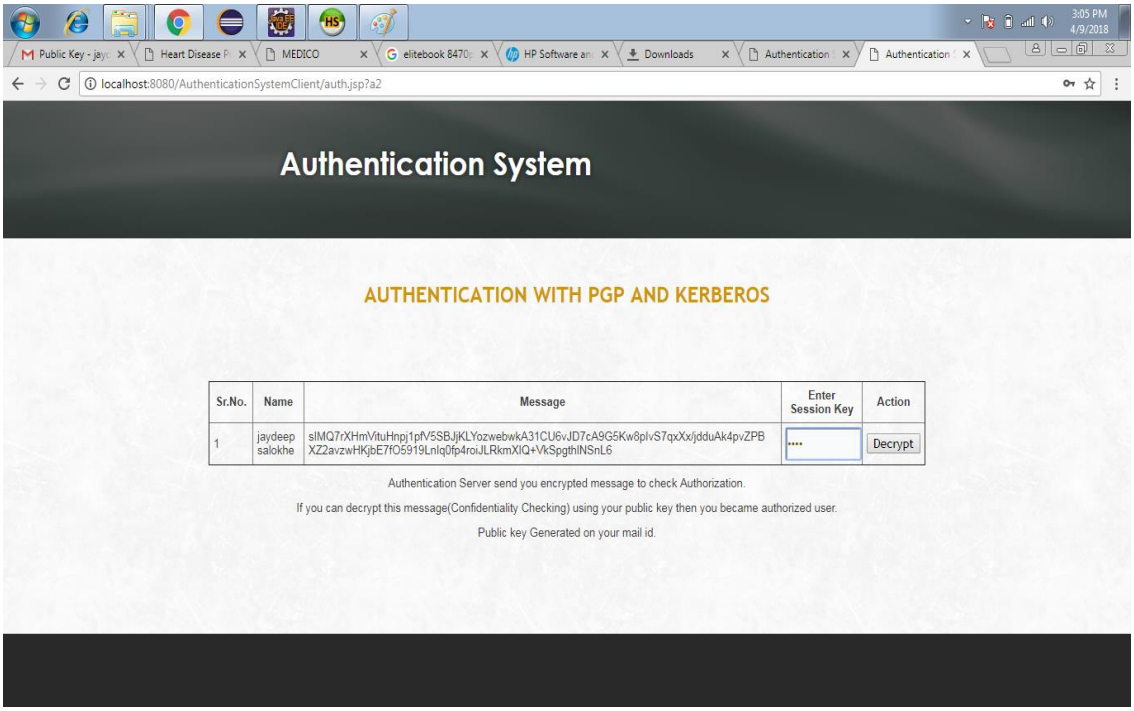

Figure 3.20: Public Key Generation

5. **Acknowledgement**: Now you are authorized. Send acknowledgement to authentication server. Then authentication server sends you ticket. The figure 3.21 dipicts the acknowledgement of receiving the public key from server and the server generate ticket to communicate with CSP

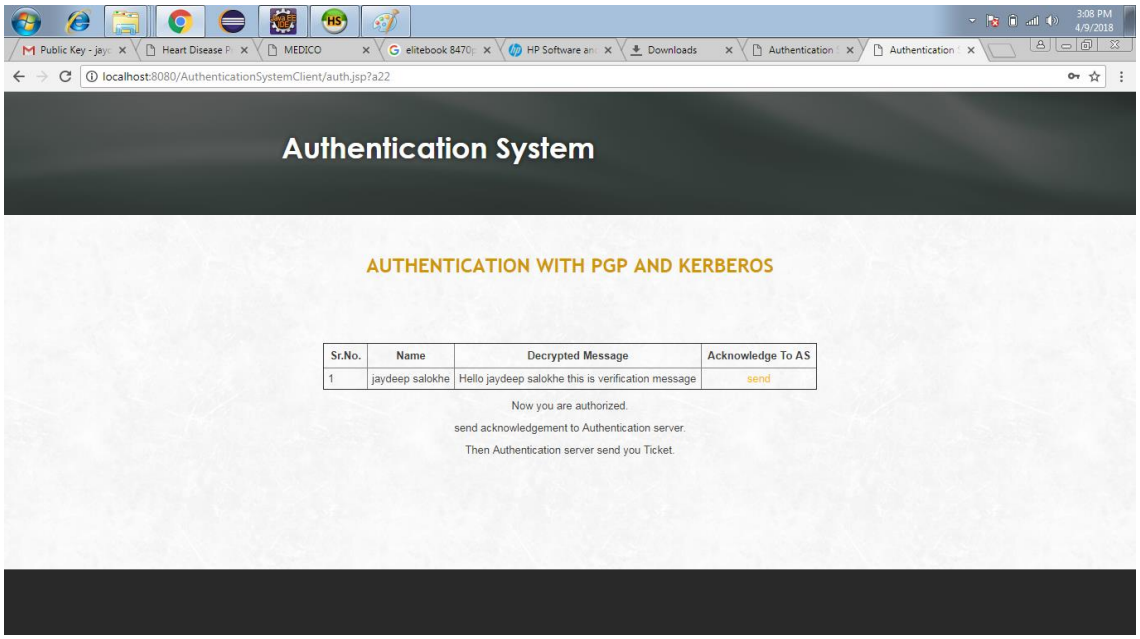

Figure 3.21: Ticket Generation

6. You received the ticket. Now can communicate with CSP using this ticket for service. Send ticket and your signature to CSP for verification. The figure 3.22 depicts the ticket generation for communication with user and CSP.

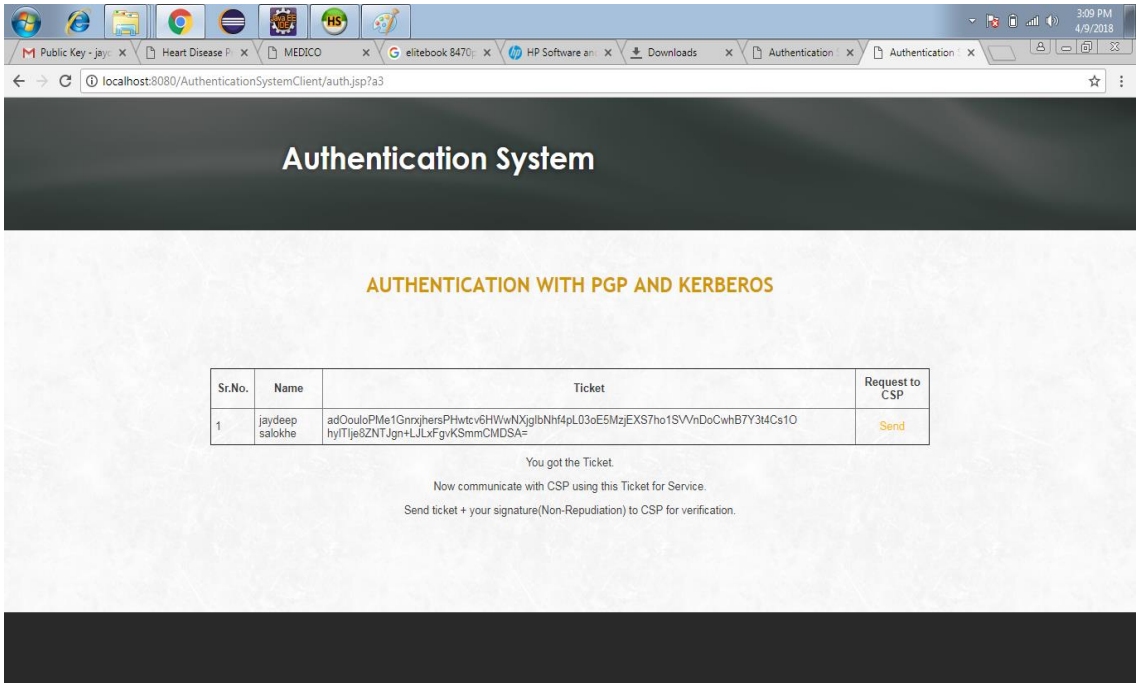

Figure 3.22: Service Request to CSP

7. **Non Repudiation:** In figure 3.23, the CSP verified your signed ticket and send back with sign, now in future you can't denies because you already verified. Now click on access to access center.

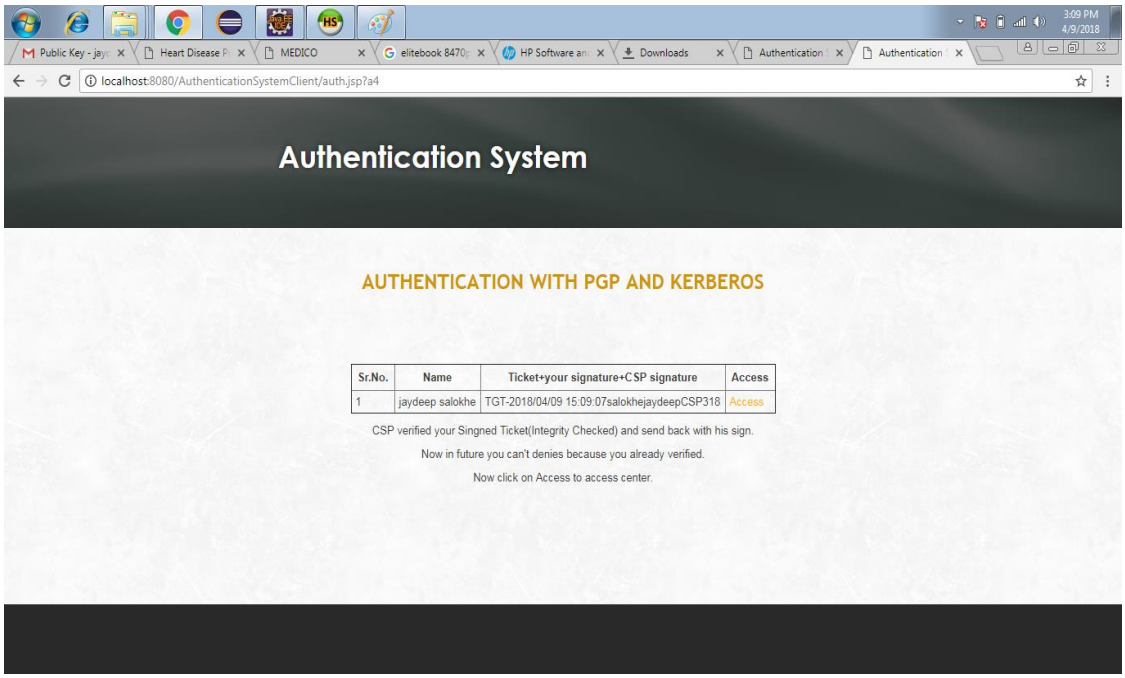

Figure 3.23: Non Repudiation

3.11. Conclusion

To achieve enhanced security in cloud we have proposed a framework which uses Pretty Good Privacy (PGP) and Kerberos based security features in cloud computing. Kerberos provides identity of users over networks and provides data integrity and authentication. Kerberos performs verification of users and services based on the concept of a trusted third party. But one of the weaknesses of Kerberos is that it cannot provide the non-repudiation features in communication. We have enhanced this feature of non-repudiation in our proposed work by using the Pretty Good Privacy. Since PGP uses the digital signature features in communication. After comparative analysis we observe that our proposed framework provides better features that many existing methods with respect to security parameters i.e. authentication, confidentiality, integrity, and non-repudiations in cloud environment.