SECURE AUTHENTICATION FRAMEWORK FOR CLOUD

A thesis submitted to the Christ University for the award of the degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

By

SUMITRA BINU

(Register Number: 1145007)

UNDER THE SUPERVISION OF

Dr. Pethuru Raj, Infrastructure Architect,

IBM Global Cloud Center of Excellence, Bengaluru

Dr. Mohammed Misbahuddin, Senior Technical Officer**,**

Center for Development of Advanced Computing, Bengaluru

CHRIST UNIVERSITY
BENGALURU, INDIA
Declared as Deemed to be University under Section 3 of UGC Act 1956

Centre for Research

Christ University, Bengaluru-560029

MAY 2016

DECLARATION

I, **Sumitra Binu**, hereby declare that the thesis titled "**Secure Authentication Framework for Cloud"** submitted to Christ University, Bengaluru in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy in **Computer Science** is a record of original and independent research work done by me under the supervision of

Dr. **Pethuru Raj,** Infrastructure Architect**, IBM Global Cloud Center of Excellence and Dr. Mohammed Misbahuddin,** Senior Technical Officer**, Centre for Development of Advanced Computing.** I also declare that this thesis or any part of it has not been submitted to any other University/Institute for the award of any degree.

Place: Bengaluru
Date**:** Sumitra Binu

CERTIFICATE

This is to certify that the thesis titled "**Secure Authentication Framework for Cloud**" submitted by **Sumitra Binu** to Christ University, Bengaluru in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy in **Computer Science** is a record of original research work carried out by her under our supervision. The content of this thesis, in full or in parts, has not been submitted by any other candidate to any other University for the award of any degree or diploma.

Place: Bengaluru
Date:

Dr. Pethuru Raj
Infrastructure Architect,
IBM Global Cloud Center of Excellence

Place: Bengaluru
Date:

Dr. Mohammed Misbahuddin
Senior Technical Officer,
Center for Development of Advanced Computing

Additional Director/Associate Director
Centre for Research, Christ University, Bengaluru-560029

ACKNOWLEDGEMENTS

First and foremost, I would like to bow my head before god almighty, without whose blessings nothing would have been possible. No words can articulate me deepest gratitude to you my Lord for the inspriration, strength and guidance I have received.

Interdependence is certainly more valuable than independence. This thesis is the result of five years of work whereby I have been accompanied and supported by many great minds and good hearted people. It is a pleasant aspect that I now have the opportunity to express my gratitude for all of them.

I am deeply indebted to my enthusiastic supervisor, Dr. Pethuru Raj Chelliah, Infrastructure Architect, IBM Global Cloud Center of Excellence, Bangalore, for his valuable guidance and motivation. I would like to express my sincere gratitude to Dr. Pethuru Raj for the continuous support of my research, for his patience and motivation.

I really deem it as a special privilege to convey my prodigious and everlasting appreciation and thanks to my co-supervisor, Dr. Mohammed Misbahuddin, Senior Technical Officer, Center for Development in Advanced Computing, Bangalore. His overly enthusiasm and integral view on the research and his mission for providing “only quality work”, has helped me immensely in coming out with this thesis. I owe him lots of gratitude for mentoring me, encouraging my research and for allowing me to grow as a researcher and for guiding me right from selection of topic to completion of this work.

I express my sincere thanks to Dr. Fr. Thomas C Mathew, Vice-Chancellor, Christ University and Dr. Fr. Abraham V M, Pro Vice-chancellor, Christ University for providing me an opportunity to undertake the research work and for their support and encouragement all through my research. I am grateful to Dr. Fr. Varghese, Finance Officer and Dr. Anil Pinto, Registrar, Christ University for their encouragement.

I would specially like to thank Dr. Shivappa Gudennavar, Associate Director, Center for Research, Christ University, Bangalore, for his valuable guidance, timely support and motivation for making this thesis a successful one. I am extremely appreciative to Dr. Gudennavar for his willingness to go through my work, and for being so dedicated to his role as the Associate Director for research. I am grateful to Dr. Tony Sam George, Associate Director, Center for Research, Christ University, Bangalore for his encouragement.

I express my profound gratitude to Prof. Joy Paulose, Head of the Computer Science department, Christ University, Bangalore, for his unbeilavable support, critical inputs and valuable suggestions. I would particularly like to thank Dr. Fr. Jossy P George, Director, Christ Institute of Management, Pune and Dr. Nachamai, Associate Professor, Christ University, Bangalore for providing me advises and inputs that helped me in staying on track. I take this opportunity to thank Dr. Srikantaswamy, Christ University, for his support during the initial period of my research work.

I profusely thank Dr. Bindiya Varghese, Head of the Computer Science Department, Rajagiri College of Social Sciences, Kochi, and Dr. P.X Joseph for rendering me invaluable support and thought provoking inputs during the course of this thesis work.

I specially thank my fellow-research scholar Mrs. Sreeja for her support, encouragement and constant willingness to help.

My heartfelt gratitude to my family. Words cannot express how grateful I am to my mother-in-law, father-in-law, my mother, father, brother, sister-in-law and my little niece, Ms. Ganga for all the support and prayers for me that has sustained me thus far. I would also like to thank all my friends especially Mrs. Deepti Thushar, and my ex-colleagues who supported me in my work and incented me to strive towards my goal.

My deepest gratitude is to my beloved husband Mr. Binu Rajendran, for his unwavering belief in me and who was always my support in the moments when there was no one to answer my queries. No words are enough to express my appreciation for him, without whose support I would not have achieved what I set out to do. My prodigious gratitude to my daughter Ms. Gayathri whose love, care and patience instilled in me all the energy that was needed to get me going ahead in full strength despite various pressures. My husband and daughter are the most important people in my world and I dedicate this thesis to them.

Abstract

The growing popularity of cloud based services is prompting organizations to consider shifting applications and data onto cloud. However, organizations dealing with highly sensitive information are apprehensive of moving its applications & data to public cloud owing to concern about security of its information. It is hence incumbent on service providers that only legitimate Users will access its services and resources in cloud.

Verifying authenticity of remote users is a necessary pre-requisite in a cloud environment before allowing access to secure resources/services/ applications. The simplest & most commonly used user authentication mechanism is password based authentication. However, Users tend to choose easy to remember password, and many a times use same password for multiple accounts, which makes it often the weakest link in security. Furthermore, service providers authenticating Users on the basis of password, stores password verification information in their databases and such authentication schemes with verification table are known to be vulnerable to various attacks.

From the perspective of authentication requirements, service providers in a cloud environment can be broadly categorized into two. Those service providers dealing with highly sensitive information and working in a regulated environment can be grouped into category one – as in those offering services for sectors like health care, finance. These providers require a strong and secure authentication mechanism to authenticate its users, without any additional functionality. Similarly, there is a second category of service providers dealing with secure information but operate in a

collaborative environment – as providers providing their applications bundled through a web portal. To provide the Users with a seamless authentication experience, while accessing multiple services during a session, the second category of service providers prefer to have Single Sign-on functionality.

Two-factor authentication technology overcomes the limitations of password authentication and decreases the probability that the claimant is presenting false evidence of its identity to verifier. If different service providers set up their own two-factor authentication services, Users have to do registration and login process repeatedly. Also, Users accessing multiple cloud services may be required to hold multiple authentication tokens associated with various service providers.

Authentication factors such as crypto-tokens and smart cards with cryptographic capabilities have been vastly used as a second authentication factor. However, Users are required to always carry these authentication tokens which make it cumbersome from practical usability perspective. Also its usage involves cost thus restricting its adoption to corporate environments. The authentication process can be made more user-convenient if the authentication factor chosen is such that it is commonly used by all types of Users. Leveraging the use of mobile phone as an authentication factor can help address issue of user convenience at no extra cost while improving the security of authentication schemes.

Though, there has been an increasing focus on strengthening the authentication methods of cloud service users, there is no significant work that discusses an authentication scheme that can be adopted by the two categories of cloud Service Providers.

Taking cognizance of aforesaid issues related to secured authentication in cloud environment, this research focused on designing secure Two-Factor authentication schemes that can be adopted by the two categories of service providers. This research carried out in different levels, proposes authentication architecture and protocols for the two categories of service providers.

At the first level, research proposes Direct Authentication architecture for cloud Service Providers who prefer to authenticate its users by using a strong authentication mechanism and does not require Single Sign-On (SSO) functionality. For those Providers who prefer to provide its user with a SSO functionality the research proposes Brokered Authentication architecture.

The next level of research focuses on proposing *User Authentication Protocols* for both Direct Authentication Service Providers (DASPs) and Brokered Authentication Service Providers (BASPs). The research proposes use of strong, Two-Factor Authentication Protocols without Verifier Table. The suggested protocols, provides Users with flexibility of using a Password and either a Crypto-token or a Mobile-token to authenticate with Service Providers. The proposed approach eliminates the requirement of the User to remember multiple identities to access multiple services and provides the benefit of a higher level of security on account of second authentication factor and non-maintenance of verifier table at server.

Access to different services offered by multiple service providers using a single authentication token requires interoperability between providers. Also, the Service Providers will have to address the task of issuing the second authentication factor to Users. As a result, the research intends to propose the utilization of proposed two-factor authentication scheme within a specific

environment which includes a trusted entity called an Identity Provider (IdP), with whom Users and Service Providers will be registered. The IdP is responsible for issuing and managing the second authentication factor.

In brokered authentication, the IdP playing the role of an authentication broker also provides Single Sign-on functionality. The Security Assertion Markup Language (SAML) is used by BASPs and the IdP to exchange authentication information about Users.

A major objective of this research is to propose an authentication model that can be adopted by both categories of service providers. Hence, this research proposes an authentication framework for cloud which supports an integrated authentication architecture that provides the service providers with the flexibility to choose between direct and brokered authentication. The integrated two-factor authentication protocol, which does not require the server to maintain a verifier table, supported by the frame work allows users to do a single registration and access services of both direct & brokered authentication service providers using the same crypto-token/mobile-token.

To verify claims about security strengths of the proposed authentication protocols, security analysis is done using theoretical intuition. The proposed protocols are found to offer desirable security features such as resistance to replay attack, stolen verifier attack, guessing attack, user impersonation attack etc. To verify the efficiency of the proposed protocols, the communication and computation costs are compared with similar schemes and it is seen that the costs are comparable. To validate the resistance of protocols to authentication attacks, they are analyzed using automated verification tool called ‘Scyther” and the protocol strength is verified by “*no attacks*” results.

TABLE OF CONTENTS

List of Tables

List of figures

CHAPTER 1
INTRODUCTION

Cloud computing, an emerging paradigm of Information Technology has off-late gained a lot of attention of both the industry and researchers. The ever increasing spread of resources on the Internet and the rapidly growing service providers have enabled cloud computing systems to grow as an "*anything-as-a-service (XAAS)*" model for distributed network environments. The much appreciable ability to abstract the intricacies of implementation and complexities of delivering services enables cloud computing technology to be applied in a completely different manner compared to traditional distributed systems. The use of virtualization technology to support resource pooling and sharing makes cloud resources highly scalable.

Organizations collaborating with other organizations to fulfill its business objectives, pharma companies that assimilate a lot of research data, government organizations offering e-governance services, financial institutions dealing with highly sensitive but voluminous data, etc. would like to shift their operations into cloud, so that they can benefit from ubiquitous access to resources, to scalability, to reduction in capital expenditure etc. However, the concerns related to security of information stored in the cloud are a major deterrent for many organizations from adopting cloud. According to a survey by International Data Corporation (IDC), 87.5% of the decision makers/influencers ranging from IT executives to CEOs cite "Security" as the top most challenge to be dealt with in every cloud service (Gens 2009).

The major reason for non-adoption of Public Cloud by organizations dealing with sensitive information is the lack of confidence in its information security. Many existing Public Clouds such as Amazon Web Services, Salesforce.com and Google App Engine have been victims of various attacks (Brian 2014, Entrust 2014, Darren 2014).

Authentication is one of the most important attribute for ensuring information security. An authentication mechanism which ensures that only valid users have access to data/information will help mitigate this concern to a great extent.

The simplest and the widely accepted mechanism for authenticating viz. Password based authentication, has many limitations that further corroborate these concerns. Today most of the cloud service providers protect the sensitive information residing with them with mere user name and password related to the Users account with provider, which renders the user authentication more critical in combating threats like unauthorized access and identity theft.

The ever increasing number of cloud based applications and services have raised another issue – about the number of accounts a User needs to maintain. Users of multiple applications have to manage many logins/passwords which is often difficult as the User may not be able to remember every identifier associated with each cloud service. Hence, Users tend to choose easy to remember passwords or use the same password for multiple accounts often leading to easy compromise of user accounts. Also a majority of Service Providers require the Users to store their account information in the Cloud and this information is accessible to the providers and their employees. This makes the authentication system susceptible to Insider attack and Stolen Verifier attack.

Cloud computing technology has created greater convenience by allowing sharing of resources, facilitating collaborative work environment, 24*7. However, the shared environment of Public Cloud and accessing of services over the Internet, raise questions about potential unauthorized access to cloud resources.

User authentication being the preliminary mechanism to ensure access control and the entry point for any network, including the Internet, there has been an increasing focus on deployment of strong authentication mechanisms to ensure utmost safety of user accounts maintained by cloud service providers. Strengthening the authentication includes ensuring security at client side, of data in transit and at server, besides offering strong two-factor authentication methods to verify the authenticity of Users accessing resources in the cloud.

1.1 MOTIVATION

Authenticating Users in a trustworthy and manageable manner is a vital and necessary requirement for organizations that adopt Cloud based services. Ease of implementation and cost effectiveness have prompted almost every new Cloud service to choose password based authentication methods for authenticating its end user. However, password authentication mechanisms could lead to major data breaches as had happened in the case of Utah Department of Technology Services (DTS) (Nicole 2012). On March 30 2012, hacker group from Eastern Europe succeeded in accessing the servers of DTS, compromising 181,604 Medicaid recipients and Social

Security Numbers of 25,096 individual clients. The hacker was able to retrieve the password of the system administrator, and gain access to the personal information of thousands of users. Though Utah DTS had proper access control mechanisms in place to secure sensitive data, a flaw in the authentication system rendered the system vulnerable to attack. In addition, the explosive growth of cloud services and web applications has made it near impractical for users to manage dozens of passwords for accessing different cloud services. Nevertheless, a majority of cloud service providers require the User to store the password at the service provider's end. Passwords of users are stored in a password verification table, making the system susceptible to security concerns like insider attack, stolen-verifier attack and denial-of-service attack. For example, on October 13th 2014, the Software-as-a-Service (SaaS) provider, Dropbox was victimized to a major security breach incident in which attackers managed to leak hundreds of passwords for various Dropbox accounts (Buchanan 2014). Dropbox claimed that hackers had stolen passwords from other sites and used them to login to Dropbox accounts as many customers use the same credentials for multiple services (Macmillan and Yadron 2014). Hence from a usability point of view, password authentication for cloud services faces a tremendous risk. These concerns clearly point out to the requirement of strengthening the authentication process and adopting a mechanism that eliminates the requirement of storing authentication credentials at the service provider's end.

Two-factor authentication technology which requires the User to provide more than one authentication information will drastically reduce the probability of the requestor presenting a fake identity. A two-factor authentication mechanism will make it difficult for the attackers to override the user authentication of cloud systems, since, in addition to guessing the user's password correctly, they also will have to acquire the second authentication factor.

The physical token, RSA SecurID (RSA Inc. 2015) and the software application, Google Authenticator (Google Inc. 2015) are among the popular two-factor solutions in cloud systems. Behind the scene, both the RSA physical token and the google Authentication software application shares secret seed with its corresponding authentication server. Coviello (2011) in his open letter to RSA customers says that a compromise of the servers of RSA SecurID, resulting in the exposure of secret seeds will enable the attacker to compute any pseudorandom authentication code, rendering RSA SecurID not fully secure to be utilized as a second authentication factor for cloud services. Also, the User will need to carry multiple devices to access services of different service providers since each service will be having a different secret seed (Zhu et al. 2014).

Authentication factors such as crypto-tokens and smart cards with cryptographic capabilities have been vastly used as a second authentication factor in many schemes for authenticating remote users. Ability to store authentication credentials, resistance to tampering of the stored contents and capability for computations make these tokens a preferred choice as a secure two-factor authentication mechanism. However, Users are required to always carry these authentication tokens which make it cumbersome from practical usability perspective. Also its usage involves cost thereby restricting adoption of these schemes to corporate environments. The authentication process can be made more user-convenient if the authentication factor chosen is such that it is widely used even by a layman user. This feature can be addressed by making use of a user owned device as an authentication factor. Last few years have seen the spread of mobile phone as a necessary personal gadget rather than an optional communication device. Leveraging the use of mobile phone as an authentication factor can help address issues of user convenience at no extra cost while improving the security of authentication schemes.

Service providers, offering their services from a cloud environment can be broadly categorized into two from the perspective of their authentication requirements. A set of service providers dealing with highly sensitive information and working in a controlled and regulated environment, such as those providing services for healthcare sector, can be grouped into one category. These service providers require a strong authentication mechanism to authenticate its Users. However, they do not require any additional functionality such as Single sign-on. Similarly, there are another category of service providers that deal with secure information but operate in a collaborative environment. Service providers whose applications are bundled through a web portal, SaaS services such as Ace project for project management and Assembla for code management which are simultaneously used by organizations for collaborative project management etc. can be grouped under the second category. If each of these providers has its own independent user management mechanism, then the Users will have to go through multiple registrations and multiple authentication processes. To provide the users with a seamless authentication experience, the second category of service providers prefer to have a Single Sign-on functionality by which the Users can authenticate to one of the service provider and can access multiple services without re-entering the credentials. However, if different service providers decide to offer its own two-factor authentication services, then the service providers will have the additional burden of issuing and managing the tokens. In addition, Users accessing multiple cloud services may be required to hold multiple authentication tokens associated with various service providers and may need to experience multiple registration and login process. If a single authentication token is to be used to access different cloud services, then it requires the involvement and interoperability between different service providers, owing to which the deployment of such strong authentication solutions is currently very limited (Stienne et al. 2013).

1.2 CURRENT ISSUES

In view of the aforesaid, the following salient shortcomings in the prevalent authentication schemes have been identified.

i. Password based authentication alone is not sufficient to ensure secure access to cloud resources/services/applications.

ii. Password based authentication requires the service providers to store the password information in a verification table, which leads to insider attack, stolen verifier attack etc.

iii. Prevalent two-factor authentication mechanisms used by service providers are not fully secure to be used as a second authentication factor.

iv. Lack of availability of an authentication mechanism, without verifier table, that uses a mobile token as the authentication factor.

v. Category 1 service providers preferring to adopt two-factor authentication, will have the added responsibility of registering Users and of issuing &

managing the tokens. Users will need to go through multiple registration processes and carry multiple authentication tokens to access multiple services.

vi. Category 2 service providers, preferring to adopt two-factor authentication, will have the added responsibility of registering users and issuing & managing the tokens. Users will need to go through multiple registration processes and carry multiple authentication tokens to access multiple services. Users will have to undergo multiple authentication processes to access different services simultaneously, in the same session.

vii. Lack of availability of an authentication model, without verifier table, that provides the service providers with the flexibility to directly authenticate its Users or delegate the authentication of its users to a third party, so as to achieve Single sign-on functionality.

1.3 REQUIREMENTS TO BE ADDRESSED

The above discussed issues related to authentication in a cloud environment motivated research to laydown the requirements to be addressed as follows: (a) an authentication architecture for direct authentication (b) an authentication architecture using third party Identity Provider (IdP) for brokered authentication (c) Two-Factor authentication protocols using Crypto-token and Mobile-token thus eliminating the need to maintain a verifier table at the server (d) a solution to the problem of service providers having to issue the tokens, and Users managing a multitude of tokens to access multiple services (e) a secure authentication framework for Direct and Brokered authentication.

1.4 RESEARCH OBJECTIVES

Primary goal of this research is to design, analyze and implement a secure authentication framework for cloud services, strong enough to overcome the limitations of currently prevalent mechanisms and be capable of providing a flexible authentication model to service providers. To achieve strong authentication, focus is to design two-factor authentication protocols without verifier tables. The target users of the proposed authentication model can be users accessing cloud services, and service providers facing an increasing need for cost effective solutions ensuring a more secure access to vast and sensitive data repository.

Considering the above required features for an efficient authentication framework, the following objectives are defined:

- To design the authentication architectures for direct authentication and brokered authentication.
- To design separate secure Two-factor authentication protocols without verifier table using Crypto-token/Mobile-Token for both direct authentication and brokered authentication environments.
- To analyze security of the proposed authentication protocols against common attacks on authentication schemes.
- To validate the proposed schemes through formal analysis methods.
- To propose an authentication framework for Cloud which supports an integrated authentication architecture that provides the service providers with the flexibility to choose between direct and brokered authentication.

1.5 ORGANIZATION OF THESIS

The Thesis is organized into six chapters. Chapter 1 provides an introduction to the topic of research along with an articulation of the motivation as well as research objectives.

Chapter 2 covers the literature survey which includes fundamentals of Cryptography, Cloud Computing and Authentication. It also includes the existing research in similar areas and its limitations, a description of common attacks on authentication protocols and a brief description of Scyther tool which is used for formal analysis of the protocols.

Chapter 3 elaborates on the authentication scheme for Direct Authentication. Chapter contents include the authentication architecture for direct authentication, Two-factor authentication protocol using crypto-token and mobile-token and the analysis of the proposed protocols.

Chapter 4 discusses the authentication scheme for Brokered Authentication. Chapter contents include the authentication architecture comprising of a centralized identity provider which provides a Single Sign-on functionality, Two-factor authentication protocol using crypto-token and mobile-token and analysis of the proposed protocols.

Chapter 5 presents the Authentication framework and its components. Integrated authentication architecture, integrated Two-Factor authentication protocol, and security, efficiency & formal analysis of the integrated protocol are included in the chapter.

Chapter 6 states a brief conclusion of the research work along with its limitations and possible future enhancements.

CHAPTER 2

LITERATURE SURVEY

2.1 BASICS OF CRYPTOGRAPHY

Prior to the introduction of data processing equipment's, security of valuable assets of an organization was ensured through physical and administrative procedures. Personnel screening procedures carried out by the human resources department, before hiring an employee can be cited as an example for the administrative procedure adopted to ensure security. Similarly, sensitive documents were protected by storing them in strong filing cabinets with combination locks.

The introduction of computers as data processing equipment's have contributed a lot towards efficient storage and processing of data. This automated processing of data also raised the need to have automated tools for protecting the sensitive information stored in the computers. Another major change that raised serious concerns about security was the introduction of communication networks and distributed systems which demands security of data in transit as well.

The rapid growth of Internet and related technologies such as Web 2.0, Cloud Computing etc. and the conveniences they offer have seen a wide spread adoption of these technologies by the corporate as well as by laymen. With due recognition to the fact that Internet has revolutionized communications, it should also be understood that the conveniences and uses offered by Internet and related technologies, come at the price of new perils. In the past few years, due to the rapid increase of information transmitted electronically, online fraud has been identified as a major source of revenue for criminals all over the globe. Hence detecting and preventing these nefarious activities have become a top priority for every major organization. It can be observed that almost everything that can be done offline has an online counterpart such as booking tickets online, paying bills online etc. All these applications have certain common requirements which includes protection from identity theft, data theft, assurance of privacy, confidentiality etc.

Information security plays a crucial role in safe guarding the resources and services that are available online. Confidentiality, Integrity and Availability are cited as the three security objectives for information systems, by the NIST (2004). These three concepts often referred to as the CIA triad are discussed in the following paragraphs.

Confidentiality: Maintaining confidentiality restricts unauthorized personnel from accessing sensitive data and ensures that authorized people are permitted access. A breach of confidentiality results in unauthorized disclosure of information.

Integrity: Maintaining integrity ensures that the data retains its consistency, trustworthiness and accuracy throughout its life cycle. A violation of integrity leads to unauthorized deletion or modification of information.

Availability: Availability property assures that authorized people are provided with a reliable and timely access to information. A breach of availability results in the disruption of access to a resource or the disruption of use of an information or a system providing information.

In addition to the three major security objectives discussed above, few more security concepts are considered relevant by some in the security field. These concepts include authenticity and accountability which can be explained as follows:

Authenticity: Authenticity property verifies that a person is who he claims to be and that an incoming message received by a system came from a valid and trustworthy source. This property enables the sender and the receiver to verify each other's identity and also the origin of the information.

Accountability: Enforcement of accountability property allows identifying the entity who caused an action. Accountability property, which supports non-repudiation, intrusion detection and prevention etc., allows us to trace a security breach to the corresponding entity responsible for the breach and execute a legal action (Stallings 2011).

Non-repudiation: The enforcement of non-repudiation property assures that a participant involved in a transaction cannot refute or deny a communication that they originated or received. This property enables a receiver to prove that a message received by him was sent by the alleged sender and vice-versa (Stallings 2011).

The need to achieve these security aspects arise when it is necessary to protect information flowing across a communication channel, being accessed and manipulated by an adversary thereby affecting its confidentiality, authenticity, integrity etc. To achieve the CIA triad and the other security objectives, cryptographic mechanisms are employed.

Cryptography is the science of using complex mathematics and logical principals for securing data by transforming into a form that is not legible to unauthorized users. Application of Cryptographic mechanisms such as encryption and hashing enable us to store sensitive information or transmit it across insecure networks without worrying about breach of confidentiality, authenticity, integrity and so on. The origin of the Cryptographic science can be traced back to Babylonian era during 3000 B.C (Thawte 2013). However, the usage was confined mostly to sending messages related to military operations during war times. History of cryptography speaks about the Roman emperor, Julies Caesar who used to send encrypted messages to his generals. While sending messages he didn't trust his messengers and hence he used to transform the original message by substituting each letter of the alphabet with the letter appearing after three places. For instance, the message "hello, how are you" will be encrypted or transformed into "KHLLR KRZ DUH BRX".

Until recently, cryptography was considered only as encryption and decryption. Encryption is the process of performing various substitutions and transformations on the plain text to generate the corresponding cipher text or encrypted text and the reverse process is termed decryption. The introduction of computers has extended the use of cryptography into the digital world to protect sensitive information by ensuring their confidentiality and integrity. With the advancement in time and the widespread use of Internet for online transactions, various innovative techniques such as microdots, merging words with images etc. have been proposed for securing or hiding information in storage or transit. Thus cryptography can be viewed as all the techniques adopted to protect the integrity or secrecy of digital information by converting them into a form that can be understood only by authorized users. There are three classes of approved cryptographic algorithms based on the number of keys that are used in conjunction with the algorithm. This includes symmetric key algorithms, public key algorithms and hash functions (Barker et al. 2012). The following sub section discusses the techniques used by cryptosystems to achieve the security objectives introduced in the section 2.1.

2.1.1 Hash Functions

A hash function takes as input, a block of data of varying length and generates an output of fixed length (Stallings 2011). The input is referred to as the message and the output is called the hash or digest of the message which is a fingerprint of the message. For eg. Given a message ‘M’ and the hash function ‘H’, the hash value ‘h’ of the message is calculated as h = H(M). Hash functions are one way functions, which makes it practically infeasible to retrieve the original message given its message digest. An efficient hash function should meet the criteria that the application of the hash operation will produce digests that are random in nature and are evenly distributed. Thus a change to a single bit or bits in the original message ‘M’ will produce a drastic change in the original message digest.

The type of hash function used for security applications is known as cryptographic hash function and this is an algorithm that satisfies the following properties *(*Stallings 2011).

Consistency: The consistency property of hash functions assures that the same input message to a hash function always produces the same hash result as the output. Thus irrespective of the number of attempts you make and regardless of the time, a particular input will always be mapped to the same hashed result, provided you are using the same hash function and case sensitiveness is considered.

One-way Property: This property assures that hash functions are irreversible. Therefore, given a hash value ‘h’ of a message ‘M’, it is computationally infeasible to derive the original plain text message ‘M’ from ‘h’. In other words, it is not possible to find a data object or a message that map to pre-specified hash result.

Collision-free property: The uniqueness or collision-free property of hash functions assures that it is computationally infeasible to find two different data objects or messages that map to the same hash value. This means, it is computationally not possible to find two different messages ‘M1’ and ‘M2’ such that H(M1) = H(M2) = h, where ‘H’ is the hash function and ‘h’ is the message digest. This property is required to overcome the problem of cryptographic hash collisions.

Pre-image resistance: This property assures that given a hash value ‘h’, it is infeasible to find a message ‘M’ such that h = H(M). The pre-image resistance property is related to the one-way property.

Second pre-image resistance: This property of hash functions assures that given an input message ‘M1’, it is computationally infeasible to find another message ‘M2’ such that H(M1) = H(M2).

These properties of hash functions are indicative of the fact that an adversary cannot modify or replace an input message without changing its corresponding message digest. This makes data integrity as the principal objective of hash functions. Cryptographic hash function does not require keys and virtually all cryptographic hash functions takes a variable length message as input to which compression functions are applied in an iterative manner to produce a fixed length output.

Cryptographic hash functions are used by cryptosystems for checking data integrity, authenticity, non-repudiation and for password verification by storing hashed passwords in the verification table at the server. A vast majority of authentication systems stores hashed passwords or password digests in the verification table maintained by the server and during login operation, the digest of the password submitted by the user is verified against the stored password digest.

There are many hash functions that have been proposed by researchers, though a few among them such as MD4, MD5, HAVAL-128, RIPEMD etc. were proved to be susceptible to collision attack (Xiaoyun et al. 2004). Until recently MD5 and SHA-1 were the most commonly used cryptographic hash functions (Cryptographic hash function, Wikipedia) and reports on attacks on MD5 (Schneier 2004) (Alexander et al. 2008) and weaknesses of SHA-1 (Xiaoyun et al. 2005) have prompted organizations to upgrade their security applications to use the next versions of SHA viz. SHA-2 and SHA-3 standard (NIST 2013). The outputs generated by SHA-256 hashing algorithm for various input messages are depicted in figure 2.1.

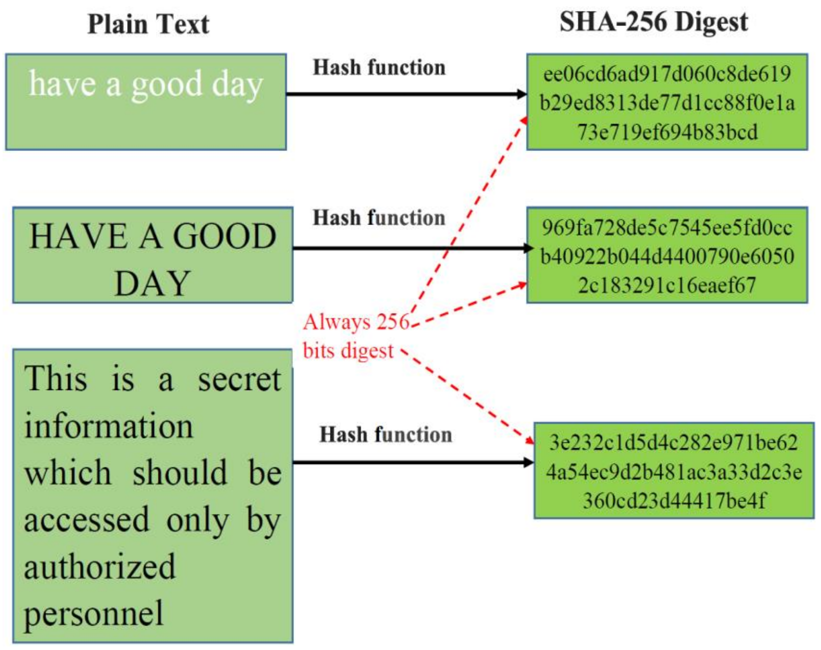


Figure 2. 1 SHA-256 Mapping of Input Messages to Corresponding Message Digests

2.1.2 Symmetric Cryptography

Symmetric key or shared key or secret key (Stallings 2011) cryptography is a form of cryptosystem, which uses a single key for both the encryption and the decryption process. This encryption mechanism requires both the sender and the receiver to share the same secret key value. In symmetric encryption, the sender transforms plain text into the corresponding cipher text by using a symmetric encryption algorithm such as AES. The plain text to be encrypted and a secret key shared with the receiver is given as inputs to the algorithm which produces the corresponding cipher text as the output. The cipher text is transmitted to the receiver across a communication channel. At the receiver, the same algorithm is used for decryption by giving the received cipher text and the shared secret key as inputs to obtain the corresponding plain text. Symmetric key encryption is much faster compared to asymmetric key encryption which uses two different key values. A simplified model of symmetric encryption to achieve confidentiality and authenticity is depicted in Figure 2.2.

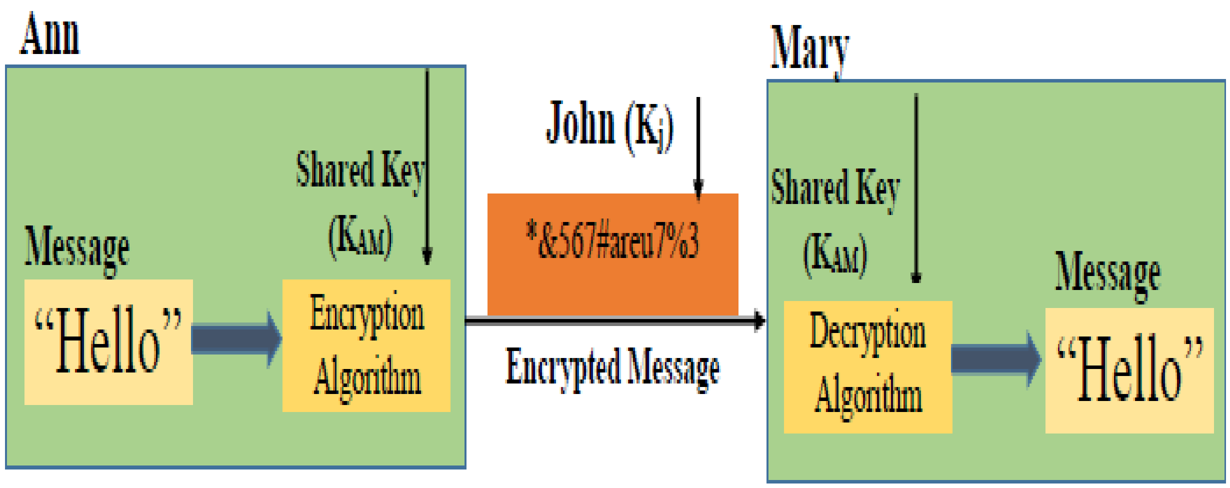


Figure 2. 2 Simplified Model of Symmetric Encryption

Here confidentiality property is achieved since the unauthorized person John, who does not possess the key 'K_{AM}' is unable to read the intercepted message. A few examples of popular symmetric algorithms include AES otherwise known as Rijndael (Joan and Vincent 2003), RC4 (Rick wash), 3DES (Hamdan 2010). Though symmetric key encryption is simple and faster in terms of computation, key distribution is a matter of concern (Blumenthal, 2007) since the communicating parties must use the same key for secure communication. Another limitation is the fact that a compromise of the shared key will result in the disclosure of all messages encrypted using this shared key. Also this encryption mechanism has scalability issues since 'k' secret keys are required to communicate with 'k' different people.

Though symmetric encryption can be efficiently used to achieve confidentiality, authenticity and integrity, it cannot be used to provide non-repudiation. For eg. If Ann sends a message to Mary encrypted using the shared key ‘K_{AM}’ and later denies that she did not send the message, then there is no way in which Mary can prove that the encrypted message was created by Ann. This is because Mary also possesses the same shared key ‘K_{AM}’ and Ann can very well argue that Mary herself might have created the message. The absence of a third party witness who could establish that Ann and Mary shared a secret key, makes it difficult to prove that Ann is the originator of the message. This limitation of symmetric key cryptography is addressed by asymmetric or public key cryptography.

2.1.3 Public Key Cryptography

Asymmetric key Cryptography otherwise known as Public key cryptography is a form of cryptosystem in which two different, related key values are used to perform the encryption and decryption process. Asymmetric encryption which was proposed by the researchers Diffie and Hellman (1976), is based on complex mathematical techniques such as finding the factors of the product of two large prime numbers as opposed to substitutions and permutations, used by symmetric encryption algorithms. In asymmetric encryption, the sender transforms the data to be protected (plain text) into the corresponding cipher text by using an asymmetric encryption algorithm such as RSA. The plain text to be encrypted and one of the keys in the key pair is given as inputs to the algorithm which produces the corresponding cipher text as the output. The cipher text is transmitted to the receiver across a communication channel.

At the receiver, the same algorithm is used for decryption by giving the received cipher text and the other paired key as inputs to obtain the corresponding plain text. One key in the key pair which is publicly available either via a public directory or via public key certificates is known as the Public key and the other key, which is a secret is known, as the private key.

The two related keys of asymmetric key cryptography which aids in performing encryption and decryption, signature generation and verification satisfies the property that with the knowledge of the public key, it is mathematically infeasible for an adversary to determine the corresponding private key.

Popular examples of public key algorithms include Diffie-Hellman key exchange algorithm which provides key distribution and secrecy, Digital Signature algorithm (DSA) (NIST 2013) which provides digital signatures and RSA (Rivest et al. 1978) which provides key distribution, secrecy and signatures. Though public key cryptography (PKC) addresses the weaknesses such as key distribution, scalability and non-repudiation, associated with symmetric key cryptography, it has got its own limitations. Major weakness of PKC is that the large key size and high computational complexity of asymmetric algorithms (Blumenthal 2007) makes it extremely slow compared to symmetric key algorithms. Also a public key infrastructure is required to implement public key algorithms. Hence to achieve the best of both symmetric and asymmetric cryptosystems, symmetric encryption is applied to encrypt large volumes of data and symmetric encryption keys are encrypted using asymmetric ciphers.

Asymmetric key or public key cryptography aids in achieving Confidentiality, Authenticity, Integrity and non-repudiation.

2.1.4 Digital Certificates

A digital certificate is a major Public Key Infrastructure (PKI) component that provides a mechanism for exchanging public keys between the participating entities without contacting a public key authority (Stallings 2011) (PKI, Wikipedia). A public key digital certificate issued to an entity contains information that binds a public key and the identity information of the owner of the public key. The certificates issued by a trusted third party commonly known as a certificate authority also includes information such as certificate serial number, validity period of the certificate, issuer name etc. along with a digital signature of the certifying authority (Stallings 2011). The digital signature is created by generating a digest of the contents of the certificate and then signing the hash using the private key of the certifying authority. The digest and the public key of the certifying authority, enables the clients to validate the certificate. Generally, X.509 formats are followed for the creation of digital certificates (Housley et al. 1999).

2.1.5 Message Authentication Code

Message authentication also called as data-origin authentication is a procedure adopted by cryptographic systems to validate the origin of a message and protect the integrity of a message during transit. Message authentication enables a message to be conveyed to the receiver by the sender with the assurance that the original message cannot be altered without the change being detected by the receiver.

The functions that are used to generate an authenticator, a value used to authenticate a message, can be grouped into following three classes:

Hash Function: A function that takes a variable length message as input and based on the hashing algorithm used, produces a message digest having fixed length as output. The generated hash value is used as the authenticator.

Message Encryption: The message is encrypted using a symmetric key or one of the keys of an asymmetric key pair. The resulting cipher text serves as an authenticator, since the message can be decrypted only by the shared key or the other key in the key pair, which ascertains the identity of the origin.

Message Authentication Code (MAC): A function that takes a variable length message and secret key as input and produces a fixed-length MAC value which serves as the authenticator. MAC values are used more commonly in scenarios where only authentication is needed or when we want authentication to persist longer than encryption. The MAC function requires two inputs viz. a secret key shared only between the sender and recipient of the message and a plain text message whose MAC value is to be calculated. A secure MAC function should satisfy the following requirements (Stallings 2011):

- If the adversary observes the authenticator MAC (K, M) and the original message M, then it should be computationally infeasible for the adversary to create another message M' such that

MAC (K, M') = MAC(K, M)

- The probability that two randomly chosen messages M and M' have the same authenticator, ie. MAC (K, M) = MAC(K, M') is 2^{-n} where n is the number of bits in the MAC value . This condition is satisfied if the authenticator MAC (K, M) is distributed uniformly.
- The authentication algorithm used to calculate the authenticator, should not be weaker with respect to specific bits or parts of the message than others. If this requirement is not satisfied then an opponent who knows M and MAC (K, M) can make variations on the known "weak spots" and arrive at a new message whose tag matches with the known tag. Thus if M' is a message obtained by applying some transformations on M such as inverting bits, ie. M'=f(M), then Pr[MAC(K,M)=MAC(K,M')] = 2^{-n}.

Several proposals have been put forward to incorporate a secret key into a proven hash algorithm to generate a MAC value based upon hash. Among the approaches that were proposed as part of keyed hashing for message authentication, the approach that received wide acceptance was HMAC, proposed by Bellare et al., (Bellare et al. 1996).

2.1.6 Hash Based Message Authentication Code(HMAC)

Hash based Message Authentication Code (HMAC): HMAC is a keyed hash message authentication code and is an authentication technique that generates a MAC value using a cryptographic hash function that takes a secret shared symmetric key and the message as inputs (FIPS 2002). HMAC is used to verify both the integrity and authenticity of a message. A major design objective of HMAC implementation is to allow for easy replicability of an existing hash function module with a new one, in case a faster or more secure hash function is required or is designed.

HMAC Algorithm: Figure 2.3 illustrates the working process of HMAC (Stallings 2011). Following are the variables used to generate a MAC value by HMAC.

MD = Hash function or Message digest used (MD5, SHA-1 etc.)

M = message whose MAC hash value is to be computed

L= number of blocks in message M

n = length of the hash code produced by the hash function

b = number of bits in each block

IV = initial value input to hash function

K = Secret key or the shared symmetric key in HMAC.

ipad (inner padding) = A string 00110110 (36 in hexadecimal) repeated b/8 times where b is the number of bits in each block

opad (outer padding) = A string 01011100 (5C in hexadecimal) repeated b/8 times where b is the number of bits in each block

Given these terms, HMAC value can be expressed as

$\text{HMAC}(K, M) = \text{MD}[(K^{+} \oplus \text{opad}) \,||\, \text{MD}[(K^{+} \oplus \text{ipad}) \,||\, M]$

The algorithm illustrated in figure 2.3 can be explained as follows:

Step1: Key length and b should be equal. Therefore, the algorithm includes three different scenarios, based on the length of the key K

Scenario 1: length of K is less than length of b. Here K has to be expanded by adding zero bits to its left until the length of k becomes equal to b. Thus if K = 150 bits and b = 512, then 362 bits of zero value will be appended to K. The new value is called modified K represented as K^+.

Scenario 2: If k and b are of same length then step 2 will be executed

Scenario 3: If key length is greater than the length of b then, K needs to be trimmed by passing it through a hash function chosen for the HMAC calculation. The hash function produces a key K (digest) containing n bits. The key K is then padded with (b-n) bits to make its length equal to b.

Step 2: The transformed key K^+ and ipad are XOR-ed to produce the b-bit block S1.

Step 3: The original message M is appended to the S1, the output of step 2. In other words, S1||M is computed.

Step 4: The hash function (MD5, SHA-1 etc.) is used to calculate the hash of the output of step 3. ie. MD (S1||M) is calculated. The result may be called as H.

Step 5: K^+ and opad are XOR-ed (exclusive-OR) to generate the b-bit block S2.

Step 6: The message digest calculated in step 4 ie. H, is appended to S2. In other words, S2||H is calculated.

Step 7: The hash function used by HMAC is used to calculate the digest of the value generated in step 6. ie. MD(S2||H) is calculated. The result is the final MAC value of the message M.

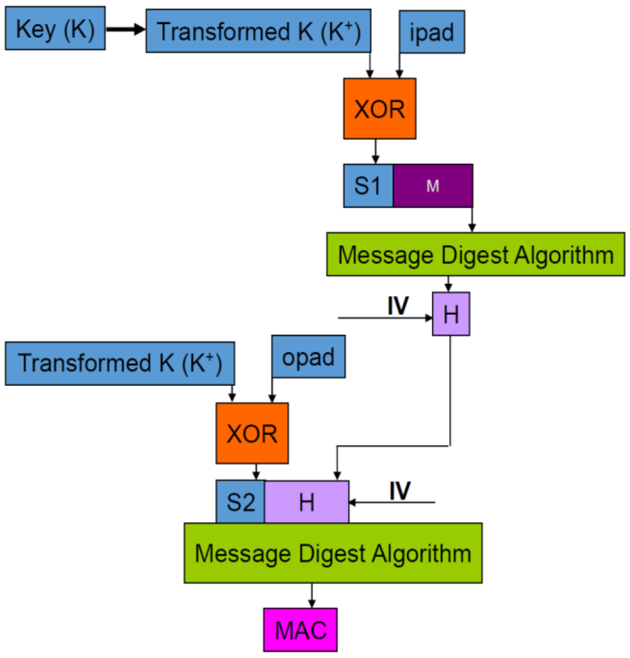


Figure 2.3 HMAC Structure

2.1.7 Password Based Encryption(PBE)

Password based encryption (PBE) enables secure encryption of files using key values generated from the user's password. There are situations in which a user would prefer to encrypt her file using simple and easy to remember password which serves as the encryption key. Simultaneously she needs the confidence that the file is secure from unauthorized access. One possible way to achieve this is by performing the encryption using the public key of the user and decrypting using the corresponding the private key. However, public key cryptography requires the secure storage of private key. A compromise of the private key can result in a breach of confidentiality of user data.

The above discussed requirement and problems can be addressed by Password Based Encryption (PBE). A PBE algorithm generates a secret symmetric key based on the password provided by the user and a randomly generated salt value (Atreya 2000). A salt value generated by a pseudo random number generator is used to strengthen the PBE algorithm by addressing the issue of dictionary attacks commonly applicable to password tables. The concatenation of the salt value with the password prevents dictionary attacks or pre-computation attacks. A PBE algorithm generates a digest of the password and the salt which can then be used as a cryptographic key for the subsequent encryption process.

2.2 CLOUD COMPUTING FUNDAMENTALS

Cloud computing is “gracefully losing control while maintaining accountability even if the operational responsibility falls upon one or more third parties” (CSA 2009). Cloud computing is an evolving computing model that concentrates on delivering computing resources over the Internet, in a shared manner that allows on-demand scalability, self-service and typically a pay-for-usage pricing model. This emerging computing model has evolved from the recent advances in existing technologies such as distributed systems, hardware virtualization, Web 2.0, service-oriented computing, utility computing and system automation.

Among the plethora of definitions attempting to address the cloud concept, from various perspectives, the widely accepted definition is the one published and standardized by National Institute of Standards and Technology (NIST). As per the NIST’s working definition published in January 2011, “Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell and Grance 2011). Cloud computing can also be perceived as a cost-effective model that promotes collaboration, enhances agility, availability and scalability simultaneously providing a different way to acquire and manage IT resources, contributing to cost reduction through optimized and efficient computing. This distributed computing model has five essential characteristics, three service delivery models and four deployment models (Mell and Grance 2011) which will be discussed in later sections. To better understand the scope of cloud computing and related concepts, the characteristics, participants, service delivery and deployment models, benefits and limitations, and security issues in cloud, are discussed in the following sections.

2.2.1 Cloud Computing Characteristics

Cloud computing has five indispensable characteristics (Mell and Grance 2011), the definition and scope of which are discussed in this section.

- On-Demand Self-Service: A consumer of cloud services can obtain services, on a need basis, at the Infrastructure, Platform and Application level through the self-managed interfaces without interacting with the service provider.

- Broad-Network Access: Cloud services are available over the Internet and accessible using standard mechanisms such as HTTP and SOAP that promote use by thin or thick client platforms such as mobile phones, PDAs, laptops etc.

- Resource Pooling: The computing resources such as storage, processor cycles, memory, network equipment's, network bandwidth, virtual machine instances etc. ,owned by the provider are pooled and shared to cater to the requirements of multiple customers using a multi-tenant model. Divergent physical and virtual resources are dynamically assigned and re-assigned to the consumer, on a need basis, in a time-sharing manner.

- Rapid Elasticity: As demand increases, computing resources can be rapidly provisioned to quickly scale out and as demand decreases, scaling in can be achieved by releasing the provisioned resources. Resource pooling helps to attain elasticity and three major features of elasticity include linear scaling, on-demand utilization and pay-as-you-go.

- Measured Service: Cloud systems control and optimize the usage of resources by making use of a metering capability at some level of abstraction appropriate to the type of service. For example, in IaaS, charges are often calculated based on the number of CPU cycles (processor by the hour), storage occupied (storage by the day), IP address allocated, number of virtual servers, network data transfers etc.

2.2.2 Cloud Participants

A cloud model has four main participants (Zarandioon 2012):

- Cloud Provider: A cloud provider is the organization that offers the cloud computing system. Cloud provider entity holds the responsibility of managing everything required to make the cloud service available.

• Cloud Consumer: A cloud consumer can either own a cloud service or consume a service available in the cloud. An application or individual who accesses and uses a cloud service is referred to as cloud consumer.

• Cloud Broker: A cloud broker acts as a mediator between service provider and consumer and provides the cloud consumer with services that caters to his requirements in the most appropriate manner.

• Cloud Auditor: A cloud auditor is a third party acting independently to provide an appraisal of security, availability and privacy level of a certain cloud service. This is done by examining the service stack of the cloud service and ensuring that the security clauses mentioned in the corresponding service level agreements (SLA) are satisfied. SLA's include the details and scope of auditing process.

2.2.3 Service Delivery Models

The definition of cloud computing provided by the U.S. National Institute of Standards and Technology (NIST) (Mell and Grance 2011) segregates cloud computing into three different models based on services provided. These models viz. Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS) and a few service providers offering these services are discussed in this section.

The three different services can be viewed as the multiple layers of services and each layer characterizes an entity that provides services to the layer above it and achieves some function by consuming services offered by the layer below it.

Software-as-a-Service: The SaaS layer provides the customer, with the capability to use an application owned by a provider and deployed on cloud infrastructure. The provider, who deploys the application in the cloud, controls the operating system, network and storage. The intricacies involved the management of physical settings are transparent to the SaaS consumers. The proper streamlining of applications and maintenance provided by SaaS model contributes to increased efficiency of operations and reduction in cost to be paid by customers. Some popular SaaS providers include (Lewis 2010):

- Google Apps: The most popular application of Google Apps is the web-based e-mail client viz. Gmail which does not require the customer to install or configure their e-mail client. Google docs for document management, Google Calendar etc. are among the other SaaS services provided by Google Apps.

- Zoho.Com: This SaaS provider offers a variety of products in the SaaS space which includes Zoho Writer, Zoho Sheet, Zoho Show, Zoho CRM etc. These web based applications are meant to cater to the requirements of small businesses.

Platform-as-a-Service: The PaaS layer appears below SaaS layer in the separation stack and here the customer represents a software developer. This model, allows consumers to develop their customized applications using the platforms, frameworks and program development tools provided as a service by the PaaS provider. Thus, using the resources of PaaS providers, the customers are able to create and host large scale applications than they would be able to handle as an individual. In the case of PaaS, the customers have full control over the applications created and deployed by themselves, but not over the underlying cloud infrastructure. Some PaaS examples include (Lewis 2010):

- Force.Com: Force.com offered by Salesforce.com, provides users with a computing platform which can be used by developers to build applications using AppExchange components or they can develop their own customized applications.

- Google App Engine: This service offered by Google, provides users with a complete stack of development tools for developing their custom application which can be executed on the underlying infrastructure provided by Google. This service provides Software Development Kits (SDKs) for programming in Python and Java.

Infrastructure-as-a-Service: This layer which is the last layer of the service stack, provides the customer with the physical infrastructure needed to offer their customized services. The ownership of the physical resources such as networking equipment, connectivity and physical hardware resides with the service provider while the customer would have control over anything including operating system, above those resources. The consumer will be typically allowed to create virtual machines on the physical resources and run his own operating systems and applications. Thus the IaaS consumer does not manage or control underlying cloud infrastructure but has control over operating systems, storage and deployed applications. A few popular services in this category are (Lewis 2010):

- Amazon Elastic Compute Cloud (EC2): EC2 provides users with an exclusive virtual machine known as Amazon Machine Image (AMI) that can be installed and executed on the infrastructure supported by EC2.

- Amazon Simple Storage Service (S3): This service provides users with data storage infrastructure that is highly-scalable, durable and secure.

2.2.4 Deployment Models

The cloud deployment models are categorized into four, on the basis of the masses who can access these resources. The four deployments models can be explained as follows (Mell and Grance 2011):

Public Cloud: Public cloud infrastructure and services are those that are accessible to the general public via an internet connection on a pay-for-service basis. Users need not invest on hardware to use the service and can scale their use on demand. The responsibility of setting up and managing the infrastructure and pooling in the resources as required by the user lies with the organization selling the cloud service. Examples for public clouds are:

- Amazon web services, Google.

Private Cloud: Private cloud infrastructure and services are dedicated for the operations of an organization. The resources are deployed inside a firewall and the user organization controls access to its resources. The software and hardware infrastructure required to set up and maintain the private cloud is owned by the user organization.

- Compute clouds owned by universities for the execution of scientific experiments with the intention of keeping potentially sensitive data within the university can be cited as an example for a private cloud.

Community Cloud: Community cloud infrastructure and services are shared by several individuals or organizations having specific needs or shared concerns such as mission, interest, security requirements, policy requirements etc. Community cloud has its place in between private and public cloud. As in the case of private clouds, it can avoid network bandwidth, security exposures and the usage can be restricted. As with Public cloud, setting up is made simple for individuals and it efficiently uses shared resources pooled in by different participating organizations.

- Example: To store and manage information related to residents of Karanataka, government organizations of Karnataka can share computing resources and computing infrastructure offered by cloud.

- Example: A health care cloud which offers HIPAA compliant services to the institutions under the umbrella of health care.

Hybrid Cloud: Hybrid clouds are a combination of two or more clouds. Standard and proprietary technologies are used to connect the individual clouds in a hybrid cloud to facilitate portability of application and data.

In certain cases, an enterprise will have a private cloud but might also use a public cloud, for meeting certain security requirements or as a backup or to handle peaks of load or as a test bed. A commonly observed usage scenario of hybrid cloud by organizations is, the hosting of non-critical information in the public cloud and hosting of sensitive, business-critical information in their private cloud which is completely under their control.

2.2.5 Cloud Computing Benefits and Limitations

The following features of Cloud computing can be viewed as the key attractions for the adoption of this emerging computing model (Lewis 2010).

Elasticity: The resources can be dynamically acquired and released by the users and the cloud service provider transparently manages the utilization of resources as per changing requirement. Physical resources are allocated to the consumers on a need basis, and hence the cloud services can scale on demand.

Scalability: Cloud service consumers have access to a resource pool that scale based on their demand.

Availability: Services offered by cloud are accessible to users 24*7 through different devices, via an Internet connection.

Accessibility and Usability: Cloud consumers have access to a virtually unlimited pool of shared resources. This enables them to develop large scale applications requiring huge amount of resources which may not be available within a regular organizational environment.

Mobility: The cloud services and resources can be accessed from anywhere around the globe, via an Internet connection, without regard to where the services are hosted.

Collaboration: Shared environment provided by the cloud enables users to work simultaneously on common data and information. In a cloud environment there are potentially no format incompatibilities when everyone is sharing applications and documents. Therefore, the collaborating users do not have to worry about the incompatibility of the documents created by independent users.

Cost-effectiveness: Purchasing of expensive hardware and software can be avoided by availing these as a service from a cloud service provider. Also, sharing of physical resources enhances its utilization and contributes to cost reduction.

Lower Infrastructure Costs: The users of cloud services need not invest on buying and maintaining infrastructure as they are owned and operated by the service provider. This cloud feature of “no upfront investment”, contributes to reduced CAPEX and OPEX.

Pay-as-you-go Pricing Model: Billing model based on consumption of services, makes cloud services affordable for start up firms and small & medium enterprises (SMEs). The consumers will have to pay only for the unit of resources they have consumed as opposed to paying for the entire resource as applicable in a traditional environment.

Easy Maintenance: Cloud providers undertake the responsibility of non-functional requirements such as maintenance of hardware and software, thereby enabling the cloud service consumers to focus on their core business requirements.

Virtualization: Hardware virtualization technology is used by IaaS providers for resource pooling. Irrespective of how the available resources are maintained in physical devices, each user has an illusion that he is accessing the resources from a single location. Users are made to feel that they have access to an infinite resource pool and are transparent to the underlying intricacies of sharing resources among the different users. Thus, virtualization of shared resources enables a provider to serve a greater number of users by using a limited number of physical resources.

Risk Reduction: Services offered by cloud can be used by organizations to do a test run of their ideas and concepts before making huge investments on new technology and infrastructure.

Despite its many attractions there are certain key concerns that can prevent the whole-hearted adoption of cloud computing. The limitations of cloud and how they act as barriers (Lewis 2010) are discussed in the subsequent paragraphs.

Security: Once the data is shifted from the boundaries of the organization to the premises of the service provider, the owners lack control over the data and they are worried about unauthorized access to their data.

Interoperability: Cloud service providers use proprietary standards and Application Programming Interfaces (APIs) for rendering their services, which makes it difficult to move hosted services from one vendor to another, resulting in vendor lock-in. There is a need to define common standards, to facilitate interoperability between service providers.

Latency: The cloud services being accessed via the Internet, the bandwidth constraints of Internet will introduce latency into communications between customer and provider.

Platform or Language Constraints: Some cloud providers support certain specific platforms and languages, which may not meet the requirements of the user. Some PaaS offerings may have their particular programming language to be used by developers or their specific API.

Regulations: There are concerns regarding the storage location of data by the cloud service providers, data protection, privacy of stored information etc. For e.g., as per U.S. PATRIOT ACT, the U.S government can access any data stored in the country, without requiring the consent of the data owner (Whittaker 2011). Some cloud service providers having their data centers in U.S and organizations considering a shift of their sensitive data to cloud should consider these regulations.

Reliability: To tackle emergency situations and disasters, cloud service providers should have back up and disaster recovery plans to bring the system up and back into operations with minimum down time. Lack of such planning and the use of commodity hardware many a times leads to failure of the cloud infrastructure.

Resource Control: Depending on the provider, the amount of control that a user can have over service provider and its resources will vary.

2.2.6 Cloud Security

In the recent past, the imploring features of cloud computing has motivated the amalgamation of cloud environment in the industry. As a consequence, there has been a lot of research on cloud and related technologies by both the industry and academia. The pay-as-you-go pricing model of cloud combined with on-demand delivery of scalable services has paved the way for a new enterprise computing model, wherein the on-premises infrastructure are shifted to off-premise data centers, accessed over the Internet and managed by cloud hosting companies. However, amount of security offered by a new computing technology, plays a major role in deciding its level of acceptance and despite its advantages, the shift to this computing paradigm raises several security concerns. This makes "cloud security" the primary topic of research in the realm of cloud. In addition to the security concerns contributed by web technologies and the Internet, architectural features of cloud poses security risks, which should be addressed immediately to promote cloud adoption.

As per a report by Gartner (Gartner 2012) (Vmware 2012), the cloud system infrastructure services market is heading for a strong growth and the world-wide growth is estimated to vary from $4.3 billion in 2011 to $24.4 billion in 2016. As per the forecast of Cisco (2015), by 2019, 56 percent of the cloud workloads will be in data centers in public cloud, which is 30 percent increase from 2014. These forecasts pinpoints where businesses and Information Technology industry is heading: towards a future dominated by cloud computing.

Despite the fact that the characteristics and attractions of cloud are well understood from a business perspective, the security state of cloud computing still lacks clarity. Although, there is a growth in cloud computing which implies that many businesses have adopted this computing model, there are several security issues which are preventing organizations from migrating their sensitive data onto cloud. Ambrust et al. (2009) in their report have mentioned hearing multiple times "My sensitive corporate data will never be in the cloud". Security, Interoperability and Portability have been identified by NIST, as the major barriers for a whole hearted adoption of cloud (NIST 2012). In addition, a survey was conducted in 2009 (Gens 2009), by International Data Corporation (IDC), a market research and analysis firm, to identify the most disturbing cloud issues. The results of the survey which was attended by masses belonging to varied levels from IT executives to top CEOs, highlight that security is the major concern as it ranked first with 87.5% of the votes, 12.9 % more than the study of the previous year (Gens 2008).

Security being identified as the major barrier for a broader adoption of cloud, many researches both from academia and industry have done several surveys in this area of knowledge and published many commendable works which will be discussed in this section.

Jensen et al. (2009) discusses the technical security issues arising from using cloud services and the intrinsic technologies used to frame these cross-domain, inter-connected collaborations. The work discusses various technologies such as WS-Security and TLS that are used and combined to secure cloud computing systems. Authors explore the issues with the application of XML Signature and Web Services Security Framework, relevance of browser security in cloud applications and analyze the impact of flooding attacks on cloud systems.

Takabi et al. (2010) have recognized cloud computing as an unrestrainable force because of its potential benefits. The authors thrust upon the need to have relevant mechanisms to handle the security and privacy concerns in cloud environment. Security challenges with respect to identity management, access control, policy integration, trust and service management are explored in the work, leading the authors to propose an exhaustive security framework for cloud computing.

Zhou et al. (2010) produced a detailed study on the privacy and security concerns experienced by many cloud service providers. The article discusses the five goals availability, confidentiality, integrity, control and auditing to ensure security of cloud computing systems and provides few strategies to achieve these goals.

IaaS security issues were explored in a very elaborate manner by Vaquero et al. (2011). The study covers in detail the security concerns derived from the multitenant environment induced by the virtualization technology. The work presents the most widely accepted cloud threats and relates these with three different domains of the IaaS model viz. the machine virtualization domain, the network virtualization domain and the physical domain. As a conclusion of their analysis, authors have identified that, most reported systems as part of securing the components present in a data-center shared by multiple-tenants, employ encryption mechanisms and access control techniques.

Subashini and Kavitha (2011) specifically studied security issues pertaining to various service delivery models in cloud. The security issues in the scope of the three models are analyzed individually, with a greater number of issues relevant to the SaaS model. Authors conclude their work by including a few security solutions proposed by researchers in this area and by proposing a framework that provides “security as a service” to the applications.

Hsin-Yi Tsai et al. (2011) in their work explores the security issues in different service delivery models from the perspective of Virtualization. The issues were discussed based on security bench marks of confidentiality, integrity and availability.

Bhadauria et al. (2011) explored the security threats applicable to cloud computing with the focus of the work on various network and application level threats such as cross-site scripting, SQL Injection attack, etc.

Ahuja and Komathukattil (2012) presented a survey on some common threats such as data confidentiality, insecure interfaces, malicious insiders, shared technology issues and the associated risks to clouds. The study revealed that there is a lack of well -defined procedure for extending the current access control mechanisms used by organizations up into the cloud.

A survey on the security state in PaaS environment was provided by Rodero-Meriono et al. (2012). Authors are focusing on issues arising to due to sharing of computing platforms. The work concentrates on Java and .NET platforms and their properties such as resource accounting, resource isolation, and safe thread termination (Fernandes et al. 2013).

A structured review of security issues in cloud was provided by Xiao and Xiao (2013) on the basis of five most representative security and privacy attributes. The attributes considered for categorizing security issues include accountability, availability, confidentiality, integrity and privacy-preservability. The study discusses the relationship among the attributes and vulnerabilities that may be exploited by the attackers.

In their book chapter, on issues and developments in cloud computing and storage security, Aguiar et al. (2013) discusses the recently discovered attacks on cloud providers and the corresponding counter measures. Protection mechanisms that focus on enhancing the integrity as well as privacy of client's data and computations are included in the study. The study overviewed several issues related to client authentication and authorization, hardware virtualization, cloud availability, cloud accountability and remote storage protection and suggests solution to address these issues. The discussion concludes by putting thrust on the need to have mechanisms for attaining storage privacy, accountability, and fool-proof verifiability on client's data and computations.

A comprehensive book chapter was published by Pearson (2013) relating the trust, security and privacy properties of cloud computing. Authors in their work, opine that only those cloud services dealing with personal information needs to take privacy into account. This chapter introduces basic concepts, and discusses the Trust, Security and Privacy issues in cloud computing, along with the approaches to address the issues.

Pearce et al. (2013) in their work discusses the basic concepts of virtualization and then slowly proceeds to system virtualization, which is used for isolation of guest operating systems, consolidation of physical servers etc. The work provides a comprehensive coverage of the threats affecting agents such as VMM, VMs, OSs in VMs, software installed on those OSs, and the environment in which the agents operate such as a network. The work concludes with recommendations for implementation and verification of secure virtual platforms.

A categorization of vulnerabilities on hypervisors is provided by Perez-Botero et al. (2013) with focus on Xen and Kernel based virtual machine (KVM). The study includes the classification of hypervisor vulnerabilities into three categories based on the hypervisor functionality, the trigger source and the target affected by the security breach. The authors have considered 11 functionalities that a hypervisor provides and have mapped the vulnerabilities to them.

➢ Security Issues Specific to Cloud Infrastructure

There are few threats that originated due to the inherent architecture of cloud and these threats require special mention. These threats create a greater impact in the cloud environment due to the architectural and operational features of cloud. The security issues specific to cloud infrastructure are explained in this section.

Side-Channeling: Virtualization technology facilitates sharing of resources, which in turn contributes to cost sharing and offering of services at low prices (Halpert 2011). To launch side-channel attack, the attacker first creates virtual machines on the physical machine which hosts victims VM. The attacker then exploits a shared CPU (Zhang et al. 2011) or observe patterns in traffic (Carlson 2012) to elicit sensitive information such as, password or secret key of the victim. For e.g., by using a virtual machine-based root kit, an attacker can create a “rogue hypervisor”, which can be placed below the original one. With the aid of this hypervisor, unauthorized code can be installed into the system by malicious users (Krutz and Vine 2010).

Shared Ecosystem and Fate Sharing: An organization deploying its application in the cloud will be provided with security guidelines by the CSP, based on best practices pertaining to a particular type of application and from his own knowledge base (Chen et al. 2010). However, a third party’s intervention could result in interruption of services to users of a shared cloud eco system. For example, in April 2009 many customers went out of business and several data center owners lost millions of dollars as an after effect of the FBI raid of data centers in Texas (Zetter 2009).

Vendor Lock-In: The usage of proprietary APIs, by SaaS providers, makes migrating from one cloud provider to another a difficult task. Vendor lock-in poses a serious threat when the services are terminated by the service provider. To cite an example, more or less, 45% of the customer data was lost and closely 20,000 consumers went out of business, when the cloud service provider "Link Up" providing online storage service, discontinued their services. (Ambrust et al., 2009).

Insecure interfaces and APIs: Cloud API's function as an interface that link applications, services and infrastructure. A set of API's manage and interact with cloud services for provisioning and monitoring of services. These API's are typically exposed and their security affects the security of cloud services. Furthermore, modifications in the cloud API's can contribute to malfunctioning applications, lost connectivity, and new vulnerabilities exposed due to bugs introduced in the new APIs (Carlin and Curran 2011).

Abuse and Nefarious Use: To gain unauthorized control over computing, network and storage resources, a malicious user launches this attack by taking advantage of the loopholes in the registration process for accessing services and by exploiting the anonymity in the usage models of resources.

➢ Authentication Challenges in Cloud

Wide array of cloud services and ever growing number of cloud service providers are beneficial to users from the perspective of scalability, ease of maintenance, elasticity etc. Permission to access the secure resources hosted by clould service providers is granted to the user only after successful user authentication which is the process of verifying the identity claimed by an individual (Meyer 2007). However, users accessing cloud services/ resources from multiple cloud service providers will have to address many authentication challenges (Liang 2011, Granneman 2012)

- .Customers are requested by the cloud service providers to store their account related information in the cloud. This information can be accessed by the service providers and customers are worried about unauthorized access to their stored information and service providers misusing the information.
- Majority of the cloud service providers use weak authentication mechanisms to authenticate users. Password based authentication is the most commonly used mechanism as it is simple, cheap and easy to deploy. However, human beings have a tendency to choose simple and easy to remember passwords often leading to data breaches.
- Password based authentication requires the cloud service providers to store the password information of the user. Owing to security reasons, this is stored either in the hashed form or salted hashed form in the server's database. However, if an attacker manages to gain access to the server's database, then he can retrieve this stored information and launch an offline dictionary attack.
- A user who uses different cloud services, will need to store his/her password information or authentication credential with every service provider. Many a times, a user uses the same password for different services and if a hacker manages to get hold of the password of a particular account of a legitimate user, then he can use the same password

to login into another account of the victim. This redundancy of information is a concern to both the customers and the service providers.

- When a user maintains different accounts to access different cloud services, then he will have to undergo multiple authentication processes. While authenticating to each service provider, he needs to exchange his authentication credentials. This redundant exchange of information can be exploited by an attacker to create a security loop hole in the system.
- The SLA’s of cloud service providers contains information pertaining to the mechanisms followed by the service providers to ensure the security of the information stored in the cloud. However, from a user’s perspective, verifying whether the rules are being enforced properly or not, is a very laborious process. This makes it difficult for the user to monitor the security of stored information.

➢ Authentication Attacks in Cloud

This section discusses various attacks that are launched by exploiting the loopholes in the authentication process.

Eavesdropping: This attack is carried out by an attacker who listens to the communication channel in between two legitimate users. In a cloud environment, a bit of code is loaded on a cloud server (Hardesty 2012) by a traffic eavesdropper to passively intercept the data transferred within a cloud or he passively listens to messages exchanged between provider and consumer to make an unauthorized replica. The illegally obtained information can be used by the attacker to retrieve credentials of a valid user which in turn can be used to impersonate the user. In a cloud environment, eavesdropping attack contributing to disclosure of information, can be controlled by adopting fool proof authorization procedures and by securing the communication channel using HTTPS.

Man-in-the-Middle Attack (MITM): MITM is a prevalent attack in SaaS environment, where in the attacker intercepts the messages exchanged between legitimate users and modifies the same without their knowledge (Misbahuddin 2010). Various types of MITM attacks are discussed in the following paragraphs.

Wrapping Attack: Meena and Chella (2012), in their work mentions that this attack is launched by modifying the Simple Object Access Protocol (SOAP) messages, exchanged between the browser and the server while establishing communication, so as to duplicate credentials for logging in. The request signed by a valid client is modified by the attacker by shifting the body of the original message body to a different wrapping element inside the SOAP header. The original message is replaced by a new body which includes the unauthorized operation the attacker wants to execute with the authorization of the original client. The attacker thus successfully gains access to the cloud and runs a code malicious in nature which interrupts the regular operations of cloud servers. The possible countermeasure would be using a combination of WS-Security with XML Signature to sign particular element and using digital certificates such as X.509 issued by trusted certificate authority (CA's).

Flooding Attack: In a cloud environment, the computation servers communicate among themselves and work in a service specific manner. An adversary carries out a flooding attack by sending bogus service requests to the server (Zunnurhain and Vrbsky 2010). The cloud server authenticates the request before providing service and the process requires the utilization of CPU cycles, memory etc. of the server. When the bogus service requests, exceeds the capacity of the server, request for service from legitimate users will starve and the server will offload its processing jobs to another server providing the same service, which will also eventually arrive at the same situation. The adversary thus succeeds in attacking one server and spreading the attack by flooding the entire cloud system resulting in engaging the resources of the whole system.

Impersonating Attack: The attacker pretends to be an authorized entity or a valid server and lures a valid user to share his/her credentials which in turn is used to impersonate the user. A verifier Impersonation attack involves an adversary who assumes to be a valid verifier and lures the client to reveal the authentication keys or information (NIST 2006), which can be used by the adversary to falsely authenticate to the verifier. A phishing attack, which also comes under this category, is launched by making the users to believe that a valid server is communicating with them, by displaying a web page that resembles a valid server page (Raza et al. 2012)

Session Hijacking: Once a user is authenticated to access a service, a session will be created for the user by the server and a session ID will be assigned for the session. Session ID's of authenticated users can be stolen by an adversary, if they are not protected properly, and the hijacked session ID can be used for identity spoofing (Gowrie 2014). Session hijacking can be addressed by encrypting the communication channel (Meier etal. 2006).

Cookie Poisoning: Cookies contain identity related credentials of an authenticated user. To launch this attack, adversary, modifies the contents of the cookies to gain unauthorized access to a resource (Bhadauria 2012). To a certain extent, this attack can be handled by regularly cleaning up cookies and by the use of products for Intrusion prevention which examines each HTTP request sent to a web server for modification of cookie information (Imperva, 2013).

Replay Attack: A replay attack involves sniffing of an authentication message exchanged between two honest communication partners and resending the same after some point in time (Misbahuddin 2010) (Stallings 2011). This replayed message contains an authentication token that was previously exchanged and hence the solution to handle replay attack is to ensure that there is some content that change every time, the message is transmitted.

Shoulder Surfing Attack: This attack which results in information disclosure is launched by monitoring the victim over his shoulders, without his knowledge, while he tries to login by entering his credentials via the keyboard (Raza et al. 2012). This attacked which is mostly launched while the victim is in a public place can also involve the use of spy cameras and the objective is to obtain the password information of a valid user.

Cloud Malware Injection Attack: The attack focuses on injecting a malicious instance of a virtual machine or a malicious service implementation, which appears to the cloud system as one among the valid service instances (Zunnurhain and Vrbsky 2010). This attack is launched by an adversary who creates its own IaaS or SaaS service implementation module, containing malicious code or a malicious VM instance (IaaS) and placing into the cloud. The attacker makes the cloud system to believe that the injected instance is a valid instance of a particular service the adversary has attacked. Thereafter, the cloud server redirects the service requests of valid users to the injected instance and the malicious code of the adversary is executed. Adversary’s code is capable of various activities ranging from subtle data modifications to full functionality changes.

Password Discovery Attacks: Multifarious approaches are adopted by attackers to obtain passwords stored by a system or are transmitted across the network. A few methods used to retrieve password based on the information available, are as follows:

Guessing Attack: Easy to remember passwords chosen by users, make them susceptible to guessing attack (Misbahuddin 2010). Based on some password related information obtained by the adversary, he guesses a password and tries to verify the correctness by logging in multiple times until he succeeds. Probability of guessing correctly is high in an offline scenario, as there is no restriction on the number of login attempts. However, the system places a restriction on the number of attempts in an online scenario, which makes guessing difficult.

Brute Force Attack: Attacker attempts to guess the correct password by trying out all possible combinations of numbers, alphanumeric characters and letters, until he gets the right password (Raza et al. 2010). Automated methods are used to launch a Brute force attack which requires a lot of computing time and computing power to be successful.

Customer Fraud Attack: In this attack, a valid user purposely compromises its authentication token, either for personal benefit or to bring bad reputation to the organization. This attack can be prevented by a verifier who can prove that the failure in authentication process is due to a wrong action by the victim (Ashraf, 2013).

2.3 AUTHENTICATION

Security of any network depends on the attainment of two simple objectives (i) Ensuring that unauthorized persons are denied access to resources and (ii) ascertaining that authorized users are allowed to access the resources they need. These objectives can be attained in many ways and one among them is to assign access permissions to resources which specify the category of users who can or cannot access a particular resource. Nevertheless, permission to access a resource can be granted only after verifying the claimed identity of the individual attempting to access a resource, and that's where authentication has a role to play. Authentication is the act or process of verifying the identity (Meyer 2007), claimed by an individual or an object prior to disclosing any sensitive information. Authentication process in turn allows authorized users and services to access sensitive resources in a secure manner, while denying access to unauthorized users, thereby supporting confidentiality and access control. Consequently, in most applications where security has top priority, it is necessary to attain authenticity which is an indispensable element of a typical security model.

There is marked difference between authentication and authorization. An authentication system ensures that a person is the one who he claims to be where as an authorization system verifies that you have the rights to use the resources in the manner you want to use. Authentication precedes authorization.

It is all the more important that remote users be authenticated properly, as they are more susceptible to security risks when compared to onsite users. Single Sign-on (SSO) authentication feature allows users to use a single credential (password, smart card, biometrics etc.) and prove their authenticity to multiple servers in a network without repeatedly-submitting the credentials. This relieves the user of the pain of remembering multiple passwords as well as the going through the authentication process multiple times to access multiple resources.

Authentication Types: There are different means for providing the authentication credentials to the verification system. The simplest and the commonly used remote user authentication mechanism is Password authentication, though it is not the most secure. In general, authentication can be classified into Single-factor, Two-Factor and Multi-Factor based on the type and number of factors used. An authentication factor is an independent category of credential that uniquely identifies an entity and it is a secret that is known to, possessed by or inherent to the owner. For instance, password is a secret known only to the owner and hence can be considered as an authentication factor whereas User-ID is public information and hence fails to qualify as an authentication factor. When network resources include highly sensitive data, authentication mechanisms that offer more protection such as Two-Factor and Multi-Factor authentication mechanisms based on Smart Cards, Crypto-tokens, Bio-metric authentication etc. are preferred.

Most widely used authentication factor include, Knowledge factor or Something the user knows such as Password and PIN, Possession factor or Something the user has (USB Token, Smart Card, Crypto-token, Mobile - Token), Inherence factor or Something the user is (Physiological or behavioral biometrics) and Location factor or Somewhere the user is (Geographic location at the time of login). Combination of any two of these factors offer higher level of security strength than single factor authentication and is referred to as Two-Factor authentication. Various single-factor and two-factor based authentication methods currently used for remote user authentication are discussed in the subsequent sections.

2.3.1 Single Factor Authentication

Single factor authentication is a process which requires only one category of credentials to identify a user requesting access to a secure resource. Single factor authentication methods include password based authentication (Cristofaro et al. 2014), challenge-response methods and bio-metric authentication. These methods are discussed in the subsequent sections.

Password Based Authentication (PBA): Cost-effectiveness, ease of implementation and simplicity makes PBA the most preferred authentication mechanism. For password based authentication, when attempting to logon to access a resource, the user is required to submit the user name /user ID and password corresponding to a particular account as the authentication factor. The authentication server maintains a data base of user accounts holding the credentials of authorized users and the submitted password is verified against the entries in the database.

In the case of password based authentication systems the security of the entire system depends entirely on a secret password. However, human beings have a tendency to choose simple and easy to remember and hence easy to guess passwords making them vulnerable to several attacks. To resist passwords from being guessed, users are recommended to secure their accounts with high entropy passwords (Cristofaro et al. 2014). Entropy, which is a measurement of unpredictability of the password, is calculated based on password length and combinations of characters it can hold. It is also recommended that passwords should not be personal details such as mother's name, favorite hobby, neither should they be words from dictionary nor from your mother-tongues.

It is typically difficult for a human being to guess a password without having some information about the owner of the password and if the value of the password is something representative of the user. However, there are software programs called "password crackers" which can be used by human beings to launch a "Brute force" attack on password systems (Password Cracking, Wikipedia). This means that an application program tries out each word in a pre-computed dictionary of terms until the correct combination of characters breaks the password (Dictionary Attack, Wikipedia). To prevent such attacks, it is advisable to choose strong passwords having alphabets, numbers and symbols and passwords should have high entropy with a minimum length of 8 characters.

Again PBA, requires the authentication system to store passwords and user name in a data base against which the password information submitted by the user can be verified. If the server is not provided with strong security, the stored passwords can be retrieved/modified by an adversary (Lee 2011) who manages to gain unauthorized access to the database. To address this issue, many authentication systems rather than storing plain text password, stores the password in its hashed form (Misbahuddin et al. 2006). This involves using a hash function, which takes the plain text password as input and produces a unique non-reversible digest as output (Stallings 2011). If the data base is breached the attacker will be able to read only the hash of the passwords and not the original password. However, storing password hashes is not an ultimate solution to ensure the security of stored passwords, since the attackers can use a rainbow table which is a pre-computed table for cracking password hashes (Rainbow Table, Wikipedia).

Passwords are also prone to shoulder surfing attack (Raza et al. 2012) and sniffing attack (Kulshrestha and Dubey 2014), which mostly happens when you attempt to log into various web sites using passwords while in a public place such as Internet cafes, CCD, libraries, Air terminals etc. Over the years, many enhanced authentication schemes have been proposed to overcome the limitations of password based schemes.

Challenge-Response Based Schemes: The Challenge-Response Based scheme requires the user to respond to a challenge received from the service providing server and based on the user response, the server will decide whether to grant permission to access the resource or not. Challenging the users with multiple questions and verifying the responses known only to the user, provides additional level of security and falls under the category of single factor authentication (Rouse 2015).

Biometric Authentication Schemes: Biometric authentication involves the automated mechanism of measuring a physical characteristic such as finger print or behavioral characteristic such as key strokes of an individual, and comparing the measured value with a previously stored value. The objective of the comparison process is to determine whether the similarity measure is satisfactory enough to confirm the identity (Allison 2000, Woodward 2000). As per biological statistics, the probability of two individuals having the same biological characteristics such as retina and iris pattern, finger print, handwriting etc. are negligibly small. This uniqueness of biometrics has paved the way for uniquely identifying and authenticating users based on their biometric traits.

Biometrics is difficult to forge since it is unique to the person and it is non-transferable. However, it is possible to replicate biometric data as it is converted into digital form before being passed onto the authentication system, and any information in digital form, can be easily replicated. Again biometric indicators are not only unique, they are unary as well, which means that they cannot be replaced at any cost. "A biometric is a unary identity: All of us have only one left thumb print" (Moskowitz 1999). The unary feature of biometrics raises two different problems. First, is the risk of biometrics getting disclosed to unauthorized individuals and second is the threat of losing the biometric indicator. Though this is very much similar to losing and disclosure of password, the unary nature of biometrics makes it infeasible to change a lost or disclosed biometrics of an individual.

However, the limitations of single factor authentication (SFA) such as vulnerability to guessing, phishing, social engineering attacks are making people to think about shifting from conventional SFA to adopt stronger authentication mechanisms (Misbahuddin 2010). This has urged many service providers offering high risk services and storing sensitive data to use an additional factor to authenticate the customers of their services. Considering this requirement to provide strong user authentication, several two factor authentication schemes were proposed by researchers.

2.3.2 Two-Factor Authentication

Over the years, many enhanced authentication schemes have been proposed to overcome the limitations of password based schemes. Two-Factor authentication is a process in which the authentication system requires two categories of credentials/factors (Misbahuddin et al. 2009) to identify a user requesting access to a secure resource. The factors considered for authentication include something you know? something you have? and something you are? (Abraham 2009)(Cristofaro 2014) (Allison 2000). Two factor authentication systems use a combination of any of these two factors for authenticating the users. Majority of two factor authentication schemes uses password as the first factor and what you have/what you are as the second factor. Probability of both the authentication factors getting compromised simultaneously is very less which decreases the chances of an unauthorized person circumventing the security system. This section discusses Two-Factor authentication and the factors used for authentication in detail.

Something you know (Knowledge Factor): A user is authenticated based on his knowledge of a secret value (Meyer 2007). The secret can be Personal Identification Number (PIN), Password, Answer to Secret Question etc (Misbahuddin 2010). which is expected to be memorized by the user and should be known only to him. Majority of the remote user authentication systems requiring the user to communicate over the Internet and Intranet based authentication systems use password based authentication to identify authorized users.

Something you have (Possession factor): In certain scenarios, a user is authenticated based on the possession of a factor (Misbahuddin 2010). For instance, to withdraw money from a bank's ATM, we need to possess the ATM card. Similarly, there are other devices such Crypto-tokens, Smart cards, RSA SecurID token etc. that serve as authentication factor. Generally, these devices are used in combination with a knowledge factor such as a PIN or password. Various devices and tokens that fall into this category are given below.

Crypto-token: A crypto-token otherwise known as a USB token, security token, authentication token, cryptoken etc. is a hardware device that provides secure storage of digital identities. The size of the token is typically small and crypto-tokens such as CryptoMate64 (CryptoMate 2016) weigh only around 6g so that it can be carried along with ease. A few of these tamper resistant tokens are designed to store cryptographic keys including Digital Signature, biometric data such as fingerprint minutiae (Security Token, Wikipedia) etc. Crypt-tokens have built-in smart card chips where all cryptographic operations such as SHA-1, SHA-256, AES-128/192/256, and RSA are performed (CryptoMate 2016) as opposed to performing in the PC or terminal. The tokens are primarily loaded with either of the two operating systems viz. JCOP and MULTOS which provide the ability to load customer specific applets and satisfies the common criteria for facilitating maximum security (Cryptoken 2016). To transfer a generated key value to a client system, crypto tokens are designed with additional features such as USB connector, RFID functions and Bluetooth wireless interface. Figure 2.4 shows a crypto-token (Cryptoken 2016)

Figure 2. 4 Crypt-Token

Smart Card: A Smart card resembles credit-card in size and has built-in integrated circuit that provides the capability to store and process data (CardLogix 2009). Based on the capability to process data and on the memory types, we have two categories of smart cards viz. memory cards and microprocessor cards. The smart cards are primarily loaded with any of the two types of smart card operating systems viz., a fixed file structure or a dynamic application system. Fixed file structure card OS makes the card usable as a secure computing and storage device. The dynamic application card operating system which includes MULTOS and JavaCard® (Multos and Javacard) varieties provides the developers with a platform for building, testing and deploying different on-card applications securely. Figure 2.5 shows a smart card. (Walter 2011)

Figure 2. 5 Smart Card

The computational capability, tamper-resistance property, and convenience in managing authentication parameters have made smart cards to be chosen as a second authentication factor for many remote user authentication schemes (Chien et al. 2002) (Hsiang and Shi 2009). Smart cards use cryptography based techniques to authenticate the user and offers stronger security than password authentication because in order to authenticate successfully to a system or a network, the user must be in possession of the card and he should know the Pin/Password. However, carrying around the cards and the reader remains a burden to users and hence these schemes are mostly constrained to corporate environments.

Time Synchronized Tokens: The time-synchronized one-time passwords generated by physical hardware tokens (Meyer 2007) are commonly used for remote user authentication. The token contains a built in accurate clock that has been synchronized with the clock on the authentication server. RSA SecurID is a commonly used time-synchronized token for performing two-factor authentication (RSA Inc., 2015) (RSA SecurID

2016). Every 60 seconds, an authentication code is generated by these token (Zhu et al. 2014) using a clock value and a random value called “seed”. The “seed” which is unique for each token is stored in the respective RSA SecurID authentication server. One commonly observed problem with time-synchronized tokens is that over a period of time they fail to be synchronous with the server. The cost of the hardware and the need to carry around the token are some of the concerns related to using this approach. Figure 2.6 shows the image of a RSA SecurID token (Ocrho 2008).

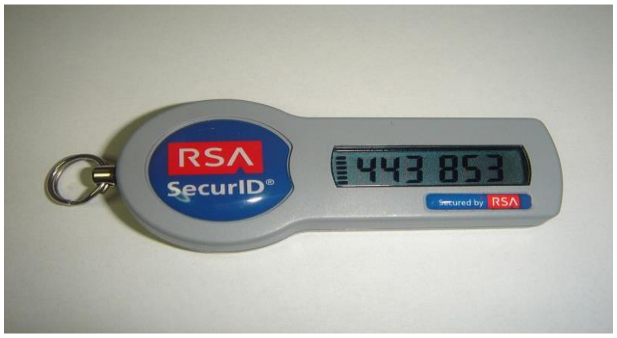
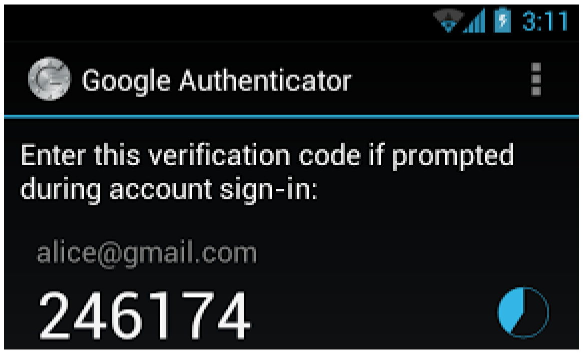


Figure 2.6 RSA SecurID

Google Authenticator: Google authenticator (Zhu et al. 2014) is an application that generates time based one- time passwords in user’s smart phone. Typically, to use this authentication factor, users will have to install the authenticator app on their smart phone. To use the authenticator app (Google Authenticator, 2016) a set up operation should be performed by the user. This involves storing of an 80-bit secret key in the authenticator app and the key is generated by the service provider uniquely for each user and is communicated over a secure channel. To log into a web site that requires two-factor authentication, user will have to first provide his user name and password. Then the user runs the authenticator app which generates a six to eight-digit time based one-time password (OTP) which is provided by the user as the second authentication factor. OTP is verified by the service provider prior to providing access to its resource. Figure 2.7 shows the image of Google Authenticator (Cristofaro et al. 2014).


Google Authenticator

Enter this verification code if prompted during account sign-in:

alice@gmail.com

246174

Figure 2. 7 Google Authenticator

Something you are (Inherence factor): Sometimes the authentication of a person can be based on “something that person is” or in other words his authenticity is ascertained by verifying his personal characteristics. Authentication based on personal characteristics will employ physiological biometrics or behavioral biometrics (Misbahuddin 2010). The Physical biometrics are related to the physical traits of an individual such as facial features, features of the eye, hand geometry and the associated authentication methods include facial recognition, iris & retina scanning, palm recognition, finger print recognition etc. The behavioral biometrics pertains to the behavior of a person and the authentication systems based on behavioral biometrics use verification methods such as handwritten signatures, keystroke dynamics, typing pattern, voice verification etc. (Allison 2000).

Commonly used biometric authentication mechanism are those categorized as physiological biometrics since the values of the corresponding authentication biometrics such as finger print minutiae are more consistent. In the case of behavioral biometrics, the values of the biometrics are also affected by a person’s mood variations, health conditions, posture etc. To cite an example, one profound problem of authentication systems based on keystroke dynamics is that, a person’s key strokes may vary substantially between different days and different times of the day. For eg. There may be variations in the typing done when the person is talking over the phone as compared to the typing done without any distraction. Similarly, if a person is to be authenticated by a voice recognition system with which his voice is already registered, and the person is suffering from sore throat, then the system may not recognize him due to changed voice. These variations may cause the authentication system to make false-positive and false-negative errors.

All the physiological biometrics such as finger print recognition, iris recognition etc. require additional hardware such as finger print reader, iris scanner etc. which is expensive to implement whereas majority of the behavior biometrics such as key stroke dynamics, voice recognition, signature dynamics etc. require software tools to verify the captured biometrics (Misbahuddin 2010). Every technology has pros and cons and so has biometrics. Biometrics have the advantage that it is unique and non-forgeable. However, the unary nature of biometrics has a disadvantage that lost or disclosed biometrics of a person cannot be changed whereas it is possible to change lost password (Allison 2000). Again, since the biometric information is converted into digital form and passed on to the authentication system, it can easily be replicated just like any other digital data (Allison 2000).

2.3.3 Single Sign-on Using SAML Standard

The research work discusses in this thesis uses Security Assertion MarkUP Language and hence this section discusses Single Sign-on functionality using SAML.

Single Sign-on: Collaborating organizations would like to provide their users with a seamless login experience while accessing different services. In a Single sign-on platform, if users are authenticated at one service, they do not have to re-enter their credentials and repeat the authentication process to log on to access another service (Hillenbrand et al. 2005). Most of the existing Single sign-on (SSO) solutions typically rely on browser cookies for maintaining state and exchanging identity information. Cookie poisoning is an authentication attack, which involves the modification of cookies of an authorized user to gain unauthorized access to resources. Hence cookies are not a reliable mechanism for sending authentication information. Browser cookies are not transferrable across DNS domains and hence the browser cookies, created from one security domain, for security reasons (same origin policy) can't be read from another one (Trosch 2008). Therefore, to solve cross domain SSO, proprietary mechanisms to pass the authentication data between security domains have been used. This solution which works fine for a single enterprise, becomes impractical when different organizations using different mechanisms collaborate. SAML provides a standard protocol and message format to exchange this security information.

Security Assertion Markup Language (SAML) is an open standard based on Extensible MarkUp Language (XML), developed by OASIS consortium (Organization for the Advancement of Structured Information Standards), an organization focusing on the inventing and adopting open standards for information technology. SAML is used (OASIS 2005) for exchanging security information between hosted SAML enabled applications and enables a user who has established and verified his identity in one domain to access services available in another domain.

Basic SAML Concepts: SAML consists of building block components whose functionalities are collaborated to support a number of SAML use cases. These components basically facilitate the exchange of identity, authentication, and attribute and authorization information between security domains.

Compared to other security systems SAML follows a different approach in providing security assertions about a principal that can be trusted by other applications within a network. To understand how this works, there is a need to introduce the two major actors in a SAML environment Viz., the Identity Provider (IdP) and the Service Provider.

The Identity Provider or asserting party is the system or administrative domain that makes assertions about a subject (OASIS 2005). Identity provider asserts that a particular user has been authenticated using a certain authentication mechanism and has been given the associated attributes. For example, the Identity Provider after validating the subject or user “Ann” can generate an assertion, this user is Ann, having an email address annmary@gmail.com and she has been authenticated to IdP’s system using an authentication mechanism based on password. The service provider or relying party is the system or domain that relies on information received from the asserting party. The service provider can use various mechanisms to verify the assertions supplied to it by the Identity Provider (OASIS 2005).

A SAML specification defines assertions, protocols, bindings and profiles. A SAML Assertion defined by an XML schema is a set of claims made by an asserting party about a subject. A SAML assertion is mostly received from the Identity provider in response to a request from the relying party. SAML has three kinds of security assertions which include Authentication statement, Attribute statement and Authorization decision statement. An authentication statement is issued by the asserting party after successfully authenticating a user. The statement includes authentication related information such as version of SAML used, issuer of the assertion, authenticated subject, validity period of the assertion, authentication mechanism used by the verifier etc.

- For example, the SAML version used is SAML 2.0, the assertion is issued at 2004-12-05T09:22:05 and the assertion is issued for verification by the relying party domain https://sp.example.com. Protected password mechanism was used to authenticate the subject at “2004-12-05T09:22:00”.

- A subject about whom SP and IdP communicate should be identified through a NAME-Identifier whose definition follows a format as defined by SAML. The format includes unspecified, persistent, transient, X.509SubjectName etc. A persistent identifier is stored in the data base entry for a particular user as the value of two attribute pairs. A transient identifier is temporary and no data will be written to the user's persistent data store.

•	There are certain restrictions under which the assertion is to be used. NotBefore restriction specifies the earliest time at which the assertion is valid and NotOnOrAfter specifies the latest time at which the assertion can be used. AudienceRestrictionCondition specifies that the assertion is addressed to a particular audience.

An attribute statement contains information related to the attribute value associated with the subject. For example, “Ann” in “Christuniversity.edu” is associated with the attribute “department” with the value “computer Science”. An authorization decision statement specifies what a user is permitted to do. For eg. The issuing authority of an authorization assertion decides whether to grant the request by the subject “Ann” for access of type “update” to the resource “Internal assessment marks of MCA course” given that she is a professor handling MCA course related topics.

SAML protocols encompass a number of request/response protocols to exchange messages between the asserting party and the relying party. Assertions to be requested and how to place the requested is defined by the SAML protocols. The structure and the content of the protocol messages specified as a set of requests and responses are defined using XML scheme. For example, the Authentication Request Protocol, typically used to support the web browser SSO profile defines a <AuthnRequest> message issued by the service provider to the Identity provider who is the asserting party. This protocol message causes a <Response> containing one or more assertions pertaining to the principal to be returned by the Identity Provider.

A SAML binding defines how SAML protocol messages can be transported by embedding them in communication protocols such as HTTP and SOAP. The bindings (OASIS 2005) defined are (1) SAML SOAP binding (2) Reverse SOAP (PAOS) binding (3) HTTP redirect binding (4) HTTP POST (5) HTTP Artifact binding.

A SAML profile typically defines which protocols and bindings can be combined and which data and assertion must be included. A number of use cases and profiles are supported by SAML among which the most important are (1) Web Browser SSO profile (2) Enhanced client and proxy profile (ECP) and (3) Federation (OASIS 2005).

The following Figure 2.8 illustrates the relationship between the basic SAML concepts.

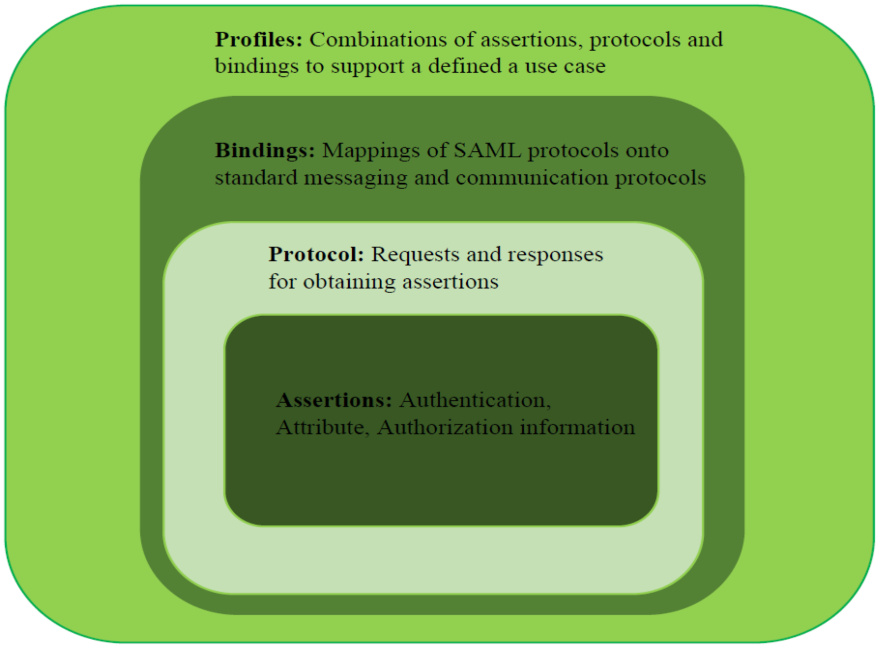

Profiles: Combinations of assertions, protocols and bindings to support a defined a use case
Bindings: Mappings of SAML protocols onto standard messaging and communication protocols
Protocol: Requests and responses for obtaining assertions
Assertions: Authentication, Attribute, Authorization information

Figure 2.8 Basic SAML Concepts

The Web Browser SSO profile with Service Provider initiated: Redirect POST→binding is used in this thesis for brokered authentication with SSO functionality. This is discussed in the following paragraph.

Web Browser SSO Profile: Two separate use cases are supported by web browser SSO profile, for users who are directly accessing the Identity Provider or are re-directed to the Identity provider by the Service Provider (OASIS 2005). Identity Provider (IdP) initiated and Service Provider (SP) initiated use cases are the two supported use cases. This section discusses the Service Provider (SP) initiated SSO profile with Redirect POST binding.

The process starts with the user requesting a resource hosted by the service provider, for example, safecloudsp1.com. As the user is currently not having any log on session or in other words a security context at this domain safecloudsp1.com, an authentication request is triggered to the IdP. The redirect message of HTTP is used to send this request to the browser (OASIS 2005). The HTTP header has a destination field which includes the location of the Single sign-on service of the IdP, for example, safecloudidp.com. The <AuthnRequest> generated by safecloudsp1.com is also included in the HTTP header in the form of a query variable. The browser processes the redirect message and sends a GET message to the URL corresponding to the Single sign-on service, safecloudidp.com along with the SAML authentication request. If the user is not currently having a session with IdP's domain, then IdP will initiate the authentication process and challenge the user to submit the valid credentials. On the submission of valid credentials by the user, authentication process will be executed and a security context is created in IdP's domain, for the user. A HTML form containing the SAML assertion is send back to the browser by the SSO service and this response must be signed by the IdP. The browser via an "auto submit" will issue a HTTP POST containing the SAML Response to the assertion consumer service (ACS) URL of the service provider. The digital signature of the IdP contained in the SAML assertion is validated by the service provider, safecloudsp1.com, and decides to grant or deny access to the resource.

2.3.4 Authentication Models for Service Providers

The following sub section discusses different authentication models adopted by service providers and where they can be used. A comparion of the two authentication models is shown in table 2.1.

i) Direct Authentication

Direct authentication is used by servers to authenticate remote users when the server maintains a database of user's record (Microsoft, Direct Authentication 2005). This authentication mechanism requires the user and the service to establish credentials (Microsoft, Web Services Security 2005), prior to the user accessing the services. For example, prior to accessing a SaaS application such as quickbook.com for calculating tax, user should first establish an ID and password with the provider by registering for the service, before calling the service.

Direct authentication can be used if any of the following conditions are satisfied (Microsoft, Web Services Security 2005):

1) The authentication credentials presented to the service by the client during the authentication process are based on shared secrets such as Password.

2) The server maintains an identity store such as a database of user credentials which helps the server to validate the submitted credential by comparing against the stored values.

3) The service is quite simple in nature and Single sign-on capability is not needed.

4) Client and service trust each other to exchange credentials securely.

Direct Authentication Process: As illustrated in Figure 2.9, a request containing the credentials of the client is send to the service by the client. The submitted credentials are verified against a database and after validation a response is send by the service to the client, after the validation is done.

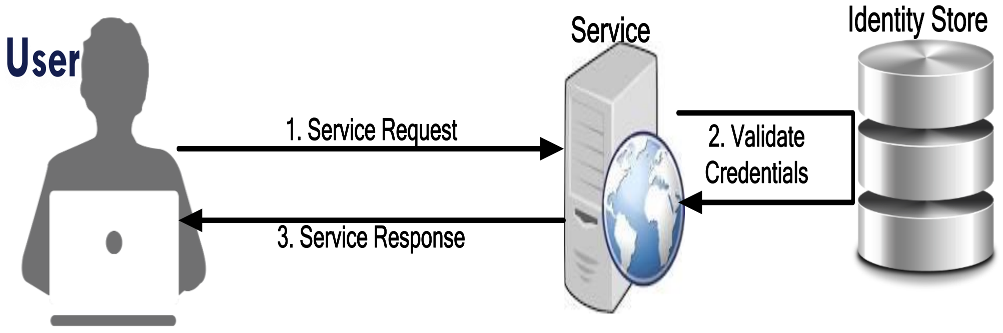


Figure 2.9 Direct Authentication Process

ii) Brokered Authentication

An authentication mechanism adopted by servers to authenticate remote users by directing them to a trusted third party. The credentials submitted by the users are verified by the third party who is otherwise known as the authentication broker (Microsoft, Brokered Authentication 2005). On successful authentication, a security token is issued by the broker, which is used by the client to authenticate to a service (Microsoft, Web Services Security 2005). Thus the authentication broker acts a broker for authenticating the client on behalf of the service and the service validates the credentials without having a direct trust relationship with the client. In this case, the identity store or databases of user credentials are maintained by the authentication broker. Brokered authentication is performed when there is no direct trust relationship between the client and the service and the service does not have direct access to the identity store.

Brokered authentication is used if any of the following conditions are satisfied (Microsoft, Web Services Security 2005):

1) Multiple services are accessed by the client resulting in the requirement of a Single sign-on (SSO) solution.
2) Direct trust relationship does not exist between the client and the service
3) Service do not have direct access to the identity store
4) A standard access control infrastructure is shared by the client and the service.

Brokered Authentication Process: As illustrated in Figure 2.10, before accessing a service the client sends an authentication request to the authentication broker.

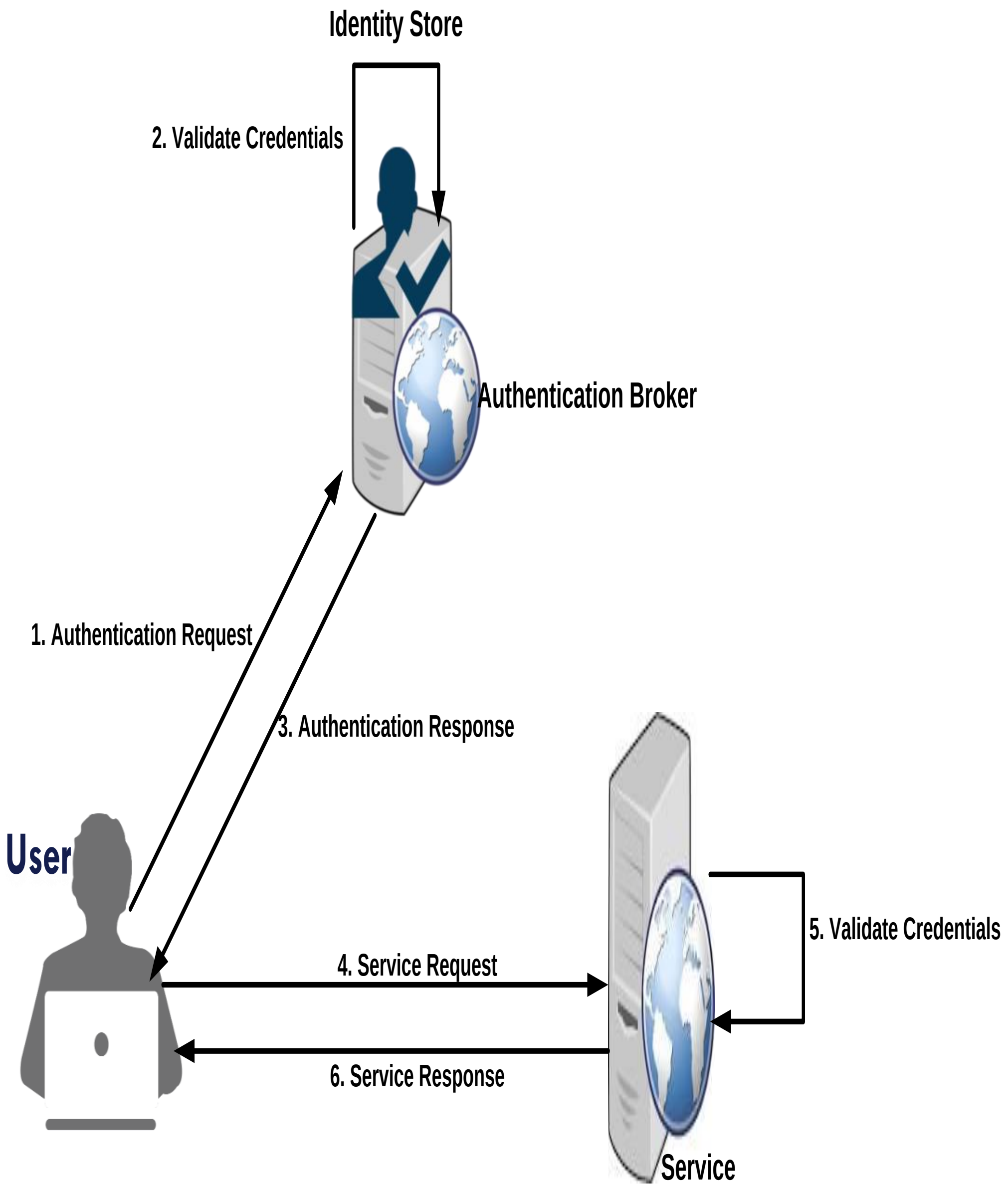


Figure 2.10 Brokered Authentication Process

The credentials of the client are included in the request and the broker verifies the submitted credentials against an identity store such as a database. The authentication broker, who vouches for the client, responds to client's authentication request and client is issued with a security token on successful authentication.

A request for service is send by the client to the service along with the token, which is used by the service to authenticate the client before providing the service. The service provides a response to the client after validating the token. The issued token is valid for a time period specified by the broker and the same token can be used by the client to authenticate requests to the service until the token expires.

The following Table 2.1 gives a comparison if direct and brokered authentication models.

Table 2.1 Comparison of Direct and Brokered Authentication

Direct Authentication	**Brokered Authentication**
Requires the presentation of the user credentials to the service, which is used by the service to authenticate the request	Requires the presentation of the user credentials to the authentication broker, which is used by the broker to authenticate the user. On successful authentication, the broker provides a token which validates the client and this token is used to authenticate with services.
Any infrastructure that supports credential management is adequate.	An infrastructure that supports different types of security tokens such as X.509 PKI, Kerberos, STS SAML Token etc. is required.
Provides no support for single Sign-on (SSO) functionality. The client needs to authenticate individually for every service which can have a negative effect on performance.	Security token is used to provide SSO functionality. Authenticating to a service is done using the security token and the same token can be used to access different services during the same session.

Types of Authentication Brokers: Based on the mechanism used to mediate the authentication process between the client and the service, authentication brokers are classified into different types. X.509 PKI, Kerberos and Security Token Service (STS) are the most prevalent examples for authentication brokers (Oracle). Public Key Infrastructure support is required for X.509 implementation and this follows an elaborate procedure for implementation and maintenance. Kerberos requires an Identity Provider supporting Kerberos protocol such as the Active Directory (AD). Security Token Service (STS) requires an STS implementation that issues and manages security tokens.

Brokered Authentication Using Security Token Service: The Protocols discussed in chapter 4 uses brokered authentication using STS and hence this section explains brokered authentication using security token service. The usage of Brokered Authentication using STS is justified under any of the following conditions:

1) The situation demands security tokens that are extensible and can contain claims that can address other security functions such as authorization, custom authentication, auditing etc.
2) Multiple services as accessed by the user and the user need not re-enter the credentials for each login process. This results in a requirement for Single sign-on (SSO) functionality.
3) The client needs to access services from multiple security domains and the it must possible for the client to use the same authentication token to access services in different security domains.

Brokered Authentication Using STS- Authentication Process: An authentication request along with credentials, is send by the client to a security Token Service (STS), trusted by client and service provider. On successful authentication, STS issues a token which proves the authenticityof the client. The tokens are interoperable which means that a standard protocol is used for generating the tokens which are acceptable to all service providers. The client then sends a service request along with the token, which is verified by the service to ascertain that the token is issued by a relaible STS and that the client has succesfully authenticated to STS. The client is then allowed to login to access the service. To allow for interoperable tokens, a protocol based on WS-Trust (Web Services-Trust) is used for issuing security tokens. The authentication request message sent by client to STS, for issuing a token is known as Request Security Token (RST) message. RST message includes the credentials such as the User ID and Password required for authenticating the client. Request Security Token Response (RSTR) message is the message send by STS to the client. An XML based Security Assertion Markup Language (SAML) Token will be included in RSTR, and the client uses the token to authenticate to a service.

2.4 REMOTE USER AUTHENTICATION

Remote user authentication process verifies the legitimacy of a remote login user (Lin et al. 2003). With the rapid increase in the demand for convenient and on-demand access to resources/services, more and more remote servers provide resources which can be accessed via communication networks. However, most of these resources which are hosted by remote servers are very sensitive and requires secure access. Hence a user who wants to access the resources in the remote servers must first authenticate to the server by undergoing a user authentication process. This section discusses the most relevant research publications related to remote user authentication, authentication schemes without verifier tables, authentication schemes for cloud and authentication schemes using mobile phones as had been analyzed/published across forums/journals and its interpretation as part of this Research. Security analysis of a few schemes, security attacks on authentication protocols and a brief description of the Scyther tool is also included in the section.

2.4.1 Remote User Authentication Schemes

Password authentication is the best-known and most commonly used mechanism for authenticating remote users (Chung and Wu 1991, Xiong et al. 2012). The advent of authentication using password, required the server to maintain the password of all the registered users in a password table in the clear text form (Li et al. 2001). Unfortunately, this paved the way for stealing the stored passwords of registered users and impersonating them by an adversary who has access to the server. Storing the password information at the server can also make the system prone to insider attack, since an administrator or employee who can access the verification information of a registered user can use it to gain access to other accounts of the user, in a scenario where the same password is used for multiple accounts. To address these issues of storing password information, authenticating server was required to store the verifier of password ie. password in hashed form (Li et al. 2001, Misbahuddin et al. 2006) and during the login time, the password submitted by the user is converted to its hashed form and compared against the stored passwords. There is a unique verifier corresponding to each user and these verifiers are stored in a verifier table in memory (Chien and Jan 2003). Hence an attacker who gains access to the server can steal the password verifier from the server and can use the verifier to launch an impersonation attack or a denial-of-service attack. This is referred to as stolen verifier problem (Bellowin and Merritt 1993, Chien and Jan 2003). Again, storing the verification information at the server makes the scheme susceptible to offline guessing attack, wherein an attacker can copy the verification information (eg. Hash of password) and guess the password while remaining offline (Gong et al. 1993). Guessing is carried out by comparing each entry in the verifier table with the hash of words in a dictionary (Gong et al. 1993) and the probability of successful guessing is high, since human beings have a tendency to choose dictionary words as their passwords (Asoke and Manish 2009). Also, an adversary who can access the verifier table stored at the server can launch a dictionary attack (Schenier 1996) to crack hashed passwords by comparing hashed passwords with values stored in a pre-computed table (Jin and Sunghwan 2013). This table contains the hash of most common passwords and is known as rainbow table (Rainbow table, Wikipedia). The limitation of hashed passwords was addressed by concatenating a random data referred to as salt with the password and the hash of the result is stored in the database, which increases the difficulty in guessing the right password. Though salt protects against general dictionary attack on a password file, it does not prevent attack on poorly chosen passwords (Schneier 1996).

These limitations of password based authentication schemes were addressed by academicians and researchers by proposing authentication protocols that provides strong authentication and resists stolen verifier problem.

Gong et al. (1993) in their paper mentions that weak passwords can be protected against strong attacks by using public-key encryption. Authors proposed a public-key based approach, wherein the authentication server is provided with a pair of private/public keys which serves to protect password against offline password guessing attack.

Bellowin and Merritt (1993), in their work discusses that though the original encrypted key exchange protocol (EKE) protocol protected passwords that are sent across the network, they required a trusted key distribution center. Authors proposed an extended version of the EKE protocol which protects the remote user's password. The proposed scheme uses RSA (Rivest et al. 1978) for generating digital signatures.

The above discussed schemes based on public-key crypto systems are known as weak-password authentication schemes and has the advantage that the remote server need not maintain a verifier table to validate the authenticity of the user (Das et al, 2004). However, authentication schemes using public-key cryptographic techniques cause heavy computational load on the system when compared to strong-password authentication schemes which are lighter because of using hash functions and xor operations (Das et al. 2004). One-way hash function is computationally infeasible to inverse (Schneier 1996). Considering this aspect of hash functions, many smart card based authentication schemes using hash functions were proposed to address the issues such as guessing attack and stolen verifier problem related to password based authentication.

The first remote user authentication scheme with smart cards (SC) was suggested by Leslie Lamport (1981) and the scheme used irreversible hash functions to create a chain of passwords. However, high computational overhead and the need for resetting password, affected the practical usability of the system. Added to that, in Lamport’s scheme the server maintains verification table leading to additional cost in maintaining the table and susceptibility to stolen verifier attack.

Sun (2000) improved the smart cased based authentication scheme proposed by Hwang and Li (2000) the security of which was based on difficulty in solving discrete logarithm problem. The security of Sun’s (2000) improved scheme was based on the irreversibility property of one-way hash functions and the scheme significantly reduces the communication and computation cost.

Chien and Jan (2003) in their paper recognizes that strong passwords with high entropy generated by computers are difficult for human beings to memorize. Hence, trusted devices such as smart cards are required to store strong passwords. Authors proposed a robust and simple protocol (ROSI) based on smart cards, using hash operations for providing security and the protocol addresses the stolen-verifier problem.

Juang (2004) in his work recognizes that smart card based remote user authentication is a very practical solution to create a distributed computing environment. Author discusses a remote user authentication scheme that uses symmetric encryption and hash functions. The proposed scheme uses nonce (number used once) values to resist replay attacks which are launched by an attacker who pretends to be a legitimate user and attempts to login to a server by transmitting messages send earlier by a valid user (Chen and Yeh 2005). In his paper author points out that the three criterions viz. (i) Session key security (ii) Forward secrecy and (iii) known key security are important for session key agreement.

Hsu (2004) in his paper has demonstrated how a parallel session attack will work on a smart card based authentication scheme and further he has pointed out that the attack is workable due to the symmetric structure of messages exchanged between user and server.

Das et al. (2004) in their work has observed that authentication schemes can be categorized into two types viz. schemes based on public key cryptographic techniques and other based on one-way hash operations. Authors have pointed out that the computational load of authentication schemes using one-way hash functions and xor operations will be less compared to schemes based on public key crypto systems. Considering this fact, Das et al. (2004) proposed a smart card based authentication scheme providing security using one-way hash functions and the scheme preserves user anonymity by making use of dynamic ID for each login which avoids the risk of ID-theft.

Das et al.'s scheme was proved to be a failure in providing user anonymity and an enhancement of that was suggested by Chien and Chen (2005). In the improved smart card based scheme using hash functions, Diffie-Hellman protocol is used by Chien and Chen (2005) to calculate the session key, which adds on to the computational complexity of the scheme.

Liao et al. (2005) in their work proves that Das et al.'s scheme cannot resist guessing attack and does not achieve mutual authentication since during the authentication phase, server can authenticate the user but user cannot authenticate the server. Hence an attacker can masquerade a server to get information from user (Liao et al. 2005). Authors proposed an improved scheme using one-way hash functions that addresses the limitations of Das et al.'s (2004) scheme. The proposed scheme uses time stamps to resist replay attacks which requires synchronization of clocks at the client and server. Also the scheme was proved by Yoon and Yoo (2006) to be prone to reflection attack.

From the work of Chen and Yeh (2005) it can be understood that Smart card based authentication schemes should with stand replay attacks and this can be achieved using either a time stamp based approach or a nonce-based approach. However, the time-stamp based approach faces some draw backs such as variation in time zone, the delivery latency etc. (Chang et al. 2006) and clocks can become unsynchronized due to faults in the synchronization mechanism (Gong 1992). Chen and Yeh (2005) proposed a smart card based authentication scheme using nonce values to resist replay attacks. The scheme provides security using one-way hash functions.

Even though the authentication schemes discussed in the previous paragraph were successful in addressing the stolen verifier problem, these authentication schemes for single server architecture/environment have got a major limitation. A remote user who needs to access multiple network services must register their identity and password individually at each server and must remember various identities and passwords which is very painful for the user (Li et al. 2001). From the work of Liao and Wang (2009) it can be understood that, an efficient and secure remote user authentication scheme for multi-server architecture allows the user to do a one-time registration at the registration center after which he/she can access the services of all registered service providing servers. The limitation of multiple-registration and maintaining multiple identities by users in a single server environment has been addressed by numerous works related to authentication schemes for multi-server environments. Most relevant user authentication schemes for multi-server architecture without verifier tables are discussed in the following paragraphs.

Li et al. (2001) proposed a user authentication scheme for multi-server architecture that allows users to choose password freely and does not require the system to maintain a verifier table. The scheme is based on artificial neural networks in which password authentication system is a pattern classification system. In this system, each user must hold a large amount of memory to store the public parameters required for authentication process which makes the communication cost of the system to be extremely high.

Lin et al. (2003) in their work observes authentication schemes based on neural networks requires a lot of time to train neural network. Authors proposed an authentication scheme which does not require the system to maintain a verification table and the security of the scheme is based on difficulty in solving discrete logarithm problem in a finite field. In this scheme every user must have a large number of memory to store public parameters.

Tsaur et al. (2004) in their work recognizes that constructing and maintaining neural networks in a neural network based authentication system will add extra manpower and cost. Authors proposed an authentication scheme for multi-server architecture based on RSA Cryptosystem and Lagrange Interpolating polynomial.

Juang (2004) has proposed a smart card based password authenticated key agreement scheme using hash functions and symmetric key crypto-system. From the work of Juang (2004) it can be understood that checking the freshness of nonce values is mandatory to resist replay attack. The scheme requires the service provider to maintain an encrypted key table burdening the provider’s memory.

Chang and Lee (2004) in their paper has pointed out that mutual authentication between remote server and user is a necessary requirement in the case of remote password authentication protocols. Authors also identifies that when the servers share a unique secret information with each user and the secret information for all the users is stored in the server’s database, then it will burden each server with need for memory, when the number of user’s is large. Chang and Lee (2004) proposed a smart card based mutual authentication scheme using hash functions and symmetric key crypto system, that does not require the server to maintain a key table.

Tsai et al. (2008) in their work has mentioned that using one-way hash functions in user authentication schemes reduces the communication and computation cost. They proposed a smart card based authentication scheme using hash functions which does not require the server to maintain a verifier table. However, the scheme requires the intervention of registration center to change user’s password. The proposed scheme permits a remote user to access services from multiple servers without individually registering at each server.

Since the current research, focuses on Two-factor authentication schemes using hash functions, that permits single registration to access multiple services and does not require the server to maintain verification table, the following section elaborates some recent and relevant two factor authentication schemes using hash functions. Discussion of the works involves description of various phases of each scheme which includes Registration, Verification and Password Change phase. Our observations are also included at the end of description of each scheme. Security analysis of a few schemes are also done to identify vulnerability of the scheme to various attacks on authentication protocols.

2.4.2 Authentication Schemes without Verifier Tables

In 2009 Liao and Wang (2009) proposed an authentication scheme using simple hash functions for multi-server environment. They focused on achieving mutual authentication and key agreement and address the time synchronization problem in a distributed environment using nonce values. It is assumed that the environment has three participants' viz. the user U_i, the service provider S_j and the registration center (RC).

Registration Phase: U_i sends his identity ID_i and password, PW_i to RC who calculates $T_i = h(ID_i \| x)$, $V_i = T_i \oplus h(ID_i \| PW_i)$, $B_i = h(PW_i) \oplus h(x)$ and $H_i = h(T_i)$ using the values send by U_i and his master secret key 'x'. RC chooses a secret number 'y' and stores the secret parameters (V_i, B_i, H_i, h(.), y), in the SC which is issued to U_i.

Login Phase: U_i keys in $ID_i{}^*$, PW_i^* and server identity SID_j. SC performs the following steps:

Step 1: Compute $T_i' = V_i \oplus h(ID_i' \| PW_i')$ and checks whether $H_i' = h(T_i') = H_i$. If equal user proceeds to next step. Otherwise the process is terminated.

Step 2: Calculates $CID_i = h(PW_i) \oplus h(T_i \| y \| N_i)$, where N_i is a nonce. $P_{ij} = T_i \oplus h(y \| N_i \| SID_j)$ and $Q_i = h(B_i \| y \| N_i)$ and sends login request $\{CID_i , P_{ij}, Q_i, N_i \}$ to S_j.

Mutual Verification and Session Key Agreement Phase:

Step 1: S_j on receiving login request computes $T_i = P_{ij} \oplus h(y \| N_i \| SID_j)$, $h(PW_i) = CID_i \oplus h(T \| y \| N_i)$, $B_i = h(PW_i) \oplus h(x)$ and compares $h(B_i \| N_i \| y)$ = received Q_i. If equal S_j authenticates U_i and computes $M_{ij1} = h(B_i \| N_i \| y \| SID_j)$ and sends (M_{ij1} , N_j) to U_i , where N_j s a nonce generated by S_j. If there is a mismatch, then request of U_i is rejected.

Step 2: U_i on receiving (M_{ij1}, N_j) calculates $h(B_i \| N_i \| y \| SID_j)$ and checks for equality with M_{ij1} in the acknowledgement from S_j. If so, U_i authenticates S_j, and sends $M_{ij2} = h(B_i \| N_j \| y \| SID_j)$ to S_j. S_j calculates $h(B_i \| N_j \| y \| SID_j)$ and checks with M_{ij2} to verify the identity of U_i.

Step 3: On successful mutual authentication a shared session key SK is computed by U_i and S_j where $SK = h(B_i \| N_i \| N_j \| y \| SID_j)$,

Password Change Phase: U_i can invoke this phase to change password at his end. U_i keys in ID_i^*, PW_i^* after inserting SC into the system upon which SC calculates $T_i^* = V_i \oplus h(ID_i^* \| PW_i^*)$ and checks whether $H_i^* = h(T_i^*)$ = stored H_i. If equal, U_i is allowed to update password. U_i submits PW_{inew} and SC computes $V_i^{new} = T_i \oplus h(ID_i^* \| PW_i^{new})$, $B_i^{new} = B_i \oplus h(PW_i) \oplus h(PW_i^{new})$.

V_i^{new} and B_i^{new} replaces V_i and B_i respectively stored in the SC.

- In this scheme password of user is transmitted directly to RC during registration and a privileged insider such as an administrator who learns the password of RC can launch an insider attack (Hsiang and Shih 2009). If the user is using the same password to access another server and the server is adopting a normal password authentication scheme, then the insider can use this password to impersonate the user to access the other server.
- A legitimate user who obtains values ‘y’ and ‘B_i’ from his own SC can launch a masquerade attack by generating a forged login message.
- The secret number ‘y’ is common for all the users, which permits a dishonest user to produce a login request that will be considered by the server as valid. Changing ‘y’ value for each U_i is inefficient thus contributing to poor reparability of Liao-Wang’s scheme (Hsiang and Shi 2009).
- The server does not check the freshness of nonce N_i, making the scheme susceptible to replay attack (Chen and Yeh 2005).
- Unsuccessful mutual authentication owing to wrong computation at server during mutual verification.
- The service providing server requires only ‘y’ and h(x) to verify the authenticity of a user and this information is accessible to all registered users. Hence a legitimate user with malicious intentions can launch a server spoofing attack by impersonating a valid service providing server and get the confidential information of other users (Chen and Yeh 2005).

Hsiang and Shih (2009) in their work produced a cost-effective, secure and efficient remote user authentication scheme suitable for practical application environment. The scheme identifies that the concept of sharing the master secret of the server and the secret number of registration center leads to server and registration center spoofing attack respectively. Hsiang and shih attempts to resolve the issues by proposing a scheme in which the master secret ‘x’ and two secret numbers ‘r’ and ‘y’are known only to RC and a secret key is shared by RC with each service provider S_j.

Registration Phase: Step 1: U_i selects a random number b and sends ID_i, $h(b \oplus PW_i)$ to RC.

Step 2: RC computes $V_i = T_i \oplus h(ID_i \| h(b \oplus PW_i))$, $A_i = h(h(b \oplus PW_i) \| r) \oplus h(x \oplus r)$, $B_i = A_i \oplus h(b \oplus PW_i)$, $T_i = h(ID_i \| x)$, $R_i = h(h(b \oplus PW_i) \| r)$ and $H_i = h(T_i)$.

Step 3: Server stores (V_i, B_i, H_i, R_i, h(.)) into SC and sends to U_i who inserts b into the received SC.

Login Phase: Step 1: U_i types in his ID_i and PW_i after inserting the card into the system. $T_i = V_i \oplus h(ID_i \| h(b \oplus PW_i))$ is calculated by smart card and checks whether $h(T_i)$ = stored H_i .If there is a match U_i is considered as a legitimate user and the login request is generated as in step 2. Otherwise the request is rejected.

Step 2: SC generates a nonce N_i and calculates $A_i = B_i \oplus h(b \oplus PW_i)$, $CID_i = h(b \oplus PW_i) \oplus h(T_i \| A_i \| N_i)$, $P_{ij} = T_i \oplus h(A_i \| N_i \| SID_j)$, $Q_i = h(B_i \| A_i \| N_i)$, $D_i = R_i \oplus SID_j \oplus N_i$, $C_0 = h(A_i \| N_{i+1} \| SID_j)$.

Step 3: SC sends (CID_i, P_{ij}, Q_i, D_i, C_0, N_i) to S_j

Mutual Verification and Session Key Agreement Phase: For authenticating U_i, S_j generates a nonce N_{jr} and executes the following steps.

Step 1: S_j calculates $M_{jr} = h(SID_j \| y) \oplus N_{jr}$, and sends (M_{jr} , SID_j , D_i , C_0 , N_i) to registration center.

Step 2: RC calculates $N_{jr}' = M_{jr} \oplus h(SID_j \| y)$, $R_i' = D_i \oplus SID_j \oplus N_i$, $A_i' = R_i' \oplus h(x \oplus r)$. Calculates $C_0' = h(A_i' \| N_{i+1} \| SID_j)$ and checks for equality with the received C_0. If there is no match, U_i is denied access. If equal, RC calculates $C_1 = h(N_{jr}' \| h(SID_j \| y) \| N_{rj})$, $C_2 = A_i \oplus h(h(SID_j \| y) \oplus N_{jr}')$ and sends (C_1, C_2, N_{rj}) to S_j, where N_{rj} is a nonce value of RC.

Step 3: S_j calculates $C_1' = h(N_{jr}' \| h(SID_j \| y) \| N_{rj})$ and checks for a match with C_1 from RC. If equal, RC is authenticated by S_j who calculates $A_i = C_2 \oplus h(h(SID_j \| y) \oplus N_{rj})$, $T_i = P_{ij} \oplus h(A_i \| N_i \| SID_j)$, $h(b \oplus PW_i) = CID_i \oplus h(T_i \| A_i \| N_i)$, $B_i = A_i \oplus h(b \oplus PW_i)$.

Step 4: S_j calculates $h(B_i \| A_i \| N_i)$ and checks for equality with the Q_i received in the request for login from U_i. If equal, U_i is successfully authenticated by S_j and proceeds to step 5. Otherwise request is rejected.

Step 5: S_j calculates $M_{ij}' = h(B_i \| N_i \| A_i \| SID_j)$ which is sent to U_i along with N_j, where N_j is a nonce of S_j.

Step 6: U_i on getting (M_{ij}', N_j) calculates $h(B_i \| N_i \| A_i \| SID_j)$ and checks for equality with the received M_{ij}'. On equality, U_i authenticates S_j and sends $M_{ij}'' = h(B_i \| N_j \| A_i \| SID_j)$ to S_j. Otherwise the session is terminated.

Step 7: S_j on receiving M_{ij}'' calculates $h(B_i \| N_j \| A_i \|SID_j)$ and checks for equality with received M_{ij}''. If there is a match U_i is authenticated.

On successful authentication of each other, both U_i and S_j calculates a shared key as $h(B_i \| A_i \| N_i \| N_j \| SID_j)$ which is used to secure future sessions.

Password Change Phase: This phase does not involve RC or S_j and proceeds as follows.

Step 1: U_i enters ID_i, PW_i after inserting SC into the system and submits a request for password change.

Step 2: SC calculates $T_i = V_i \oplus h(ID_i \| h(b \oplus PW_i))$ and checks whether $h(T_i)$ = stored H_i .On equality, U_i is permitted to enter new password PW_{inew} . Step 3: SC computes $V_{inew} = T_i \oplus h(ID_i \| h(b \oplus PW_{inew}))$, $B_{inew} = B_i \oplus h(b \oplus PW_i) \oplus h(b \oplus PW_{inew})$. The values V_{inew} and B_{inew} replace the values V_i and B_i in the SC.

Following are salient observations on the scheme:

- Authentication Phase involves both RC and Service Provider contributing to increased communication cost.
- Failure in the mutual authentication of RC and Service Provider owing to wrong computation by service provider during mutual verification and session key agreement phase (step 3).
- Susceptible to Masquerade attack and server spoofing attack by a legitimate user.

Lee et al. (2011) proposed a scheme to address the weaknesses in Hsiang and Shih scheme (2009). The scheme includes a onetime registration phase, login phase, verification phase and password change phase which permits the user to change password. The secrets ‘x’ and ‘y’ used to verify the user is known only to RC. The values h(x||y) and h(y) are calculated and shared by RC with each server S_j. Unlike Hsiang et al. scheme, this scheme does not require the involvement of RC during verification phase.

Registration Phase: Step 1: U_i submits ID_i, $h(b \oplus PW_i)$, to RC where b is a random number.

Step 2: RC calculates $B_i = h(h(b \oplus PW_i) \| h(x\|y))$, $T_i = h(ID_i \|x)$, $V_i = T_i \oplus h(ID_i \| h(b \oplus PW_i))$, and $H_i = h(T_i)$.

Step 3: SC containing (V_i, B_i, H_i, h(.), h(y)) is sent to U_i by RC. U_i stores b into his SC which contains (V_i, B_i, H_i, h(.), h(y), b).

Login Phase: Step 1: After inserting SC into the system, U_i keys in his ID_i and PW_i. SC calculates $T_i = V_i \oplus h(ID_i \| h(b \oplus PW_i))$ and checks whether $H_i^* = h(T_i)$ is equal to the H_i value in SC. If they are not equal, the login request is rejected. Upon equality U_i is considered a legitimate user and the request for login is generated as in step 2.

Step 2: SC calculates $A_i = h(T_i \| h(y) \| N_i)$, $CID_i = h(b \oplus PW_i) \oplus h(T_i \| A_i \| N_i)$, $P_{ij} = T_i \oplus h(h(y) \| N_i \| SID_j)$, $Q_i = h(B_i \| A_i \| N_i)$, where N_i is a nonce generated by SC . SC sends (CID_i, P_{ij}, Q_i, N_i) to S_j.

Verification Phase: On receiving S_j authenticates U_i as follows.

Step 1: S_j computes $T_i = P_{ij} \oplus h(h(y) \| N_i \| SID_j)$, $A_i = h(T_i \| h(y) \| N_i)$,

$h(b \oplus PW_i) = CID_i \oplus h(T_i \| A_i \| N_i)$, $B_i = h(h(b \oplus PW_i) \| h(x\|y))$.

Step 2: S_j calculates $h(B_i \| A_i \| N_i)$ and checks for equality with the received Q_i. Request for login is rejected if there is a mismatch. Otherwise S_j calculates $M_{ij}' = h(B_i \| N_i \| A_i \| SID_j)$ and sends M_{ij}' along with a nonce N_j generated by S_j to U_i.

Step 3: On receiving (M_{ij}', N_j), U_i computes $h(B_i \| N_i \| A_i \| SID_j)$ and checks for equality with the received value M_{ij}'. U_i fails to authenticate S_j if there is a mismatch. Otherwise U_i sends $M_{ij}'' = h(B_i \| N_j \| A_i \| SID_j$ to S_j.

Step 4: S_j on receiving the message M_{ij}'' calculates $h(B_i \| N_i \| A_i \| SID_j)$ and ckecks for equality with the M_{ij}''. On equality, U_i is successfully authenticated by S_j and the session key $SK = h(B_i \| N_i \| N_j \| A_i \| SID_j)$ is simultaneously generated by both U_i and S_j.

Password Change Phase: This phase requires involvement of RC and proceeds as explained:

Step 1: U_i keys in his ID_i, PW_i after inserting SC into the system and submits a request for changing password.

Step 2: SC calculates $T_i = V_i \oplus h(ID_i \| h(b \oplus PW_i))$ and checks whether $H_i^* = h(T_i)$ is equal to the H_i value in SC. If equal U_i chooses a PW_{new} and a new random number b_{new}. U_i sends ID_i and $h(b_{new} \oplus PW_{new})$ to registration center in a secure manner.

Step 3: RC calculates $B_{new} = h(h(b_{new} \oplus PW_{new}) \| h(x\|y))$ .The values V_i and B_i in the SC are replaced with V_{inew} and B_{inew}.

Relevant observations on the scheme are as follows:

- Failure in mutual authentication owing to wrong calculation by the service provider.
- The password change phase requires the involvement of RC.
- Improper authentication due to acceptance of fabricated authentication messages.

Security Analysis of Lee et al. Scheme (2011)*:* This section analyzes the resistance of Lee at al.'s scheme (2011) to various attacks and highlights the associated weaknesses. In this protocol, the messages in the login phase as well as verification phase are transmitted via an insecure channel making the scheme susceptible to various attacks.

Failure in Mutual authentication: During mutual authentication, in step 3, Ui calculates M_{ij}'' = h(B_i || N_j || A_i || SID_j) which is sent to S_j. S_j on receiving message M_{ij}'' calculates h(B_i || N_i || A_i || SID_j) and compares with the M_{ij}''. U_i is using the nonce N_j while computing M_{ij}'' whereas S_j is using the nonce N_i to computing M_{ij}''. Obviously the values will not match and the authentication will fail.

Denial-of-Service Attack: This scheme is susceptible to DoS attack owing to partial modification of authentication message by the attacker. In the step 2 of verification phase, S_j sends ($M_{ij,}$ N_j) to U_i. However, the nonce N_j is not used in the message M_{ij} and hence even if its value is modified, U_i will not reject the message.

To illustrate the scenario, assume that an adversary M intercepts the message ($M_{ij,}$ N_j) send by S_j to U_i. Now the adversary sends ($M_{ij,}$ N_m) to U_i where N_m is adversary's nonce. U_i calculates h(B_i || N_i || A_i || SID_j) and checks with the received value of M_{ij}. Since the change of nonce value from N_j to N_m does not modify the message, the computed value of M_{ij} will be correct and U_i authenticates S_j even though the message (M_{ij}, N_m) is a fabricated message. To achieve mutual authentication U_i computes $M_{ij}'' = h(B_i \| N_j \| A_i \| SID_j)$ and sends to S_j. In the correct scenario S_j should compute $M_{ij}'' = h(B_i \| N_j \| A_i \| SID_j)$ whereas S_j calculates $M_{ij}'' = h(B_i \| N_i \| A_i \| SID_j)$, which will result in an authentication failure even in the case of a honest user since $N_i \neq N_j$.

The fact that the protocol does not check the freshness of nonce N_j, permits an attacker to change the message partially. In this scheme a valid user U_i is not able to differentiate between forged and valid message which ultimately results in a denial-of-service to a legitimate user.

Forgery Attack: Lee et al. (2011) claims that a valid login message cannot be created by a malicious person without knowing A_i, B_i and PW_i. Also a legitimate user who is malicious in nature cannot impersonate another user since he cannot get B_i from the SC and intercepted login message (CID_i, P_{ij}, Q_i, N_i) without knowledge of $h(x\|y)$. Also a valid user who does not know the master secret x cannot compute T_i and A_i required to generate a correct login request message even if he has extracted the parameters (V_i, B_i, H_i, h(.), h(y), b) from his SC. The protocol was claimed by authors to be secure against masquerade attack. However, the security analysis of the scheme reveals that a valid malicious user can launch a forgery attack (Li, 2013) by using his own SC information and an intercepted login message (CID_i, P_{ij}, Q_i, N_i) by performing the following steps.

S1: Assume that A is a legitimate registered user and has the information (V_i, B_i, H_i, h(.), h(y), b) stored in his/her SC.

S2: When a different user U_i, interacts with server S_j, adversary A can intercept (CID_i, $P_{ij,}$ $Q_{i,}$ N_i) and calculate $T_i = P_{ij} \oplus h(h(y) \| N_i \| SID_j)$, using the h(y) values in his own SC and the values in the intercepted message. To send a login request message resembling a valid message to a service providing server S_k, following steps are performed by adversary A.

S3: A calculates $A_i^* = h(T_i \| h(y) \| N_A)$, $CID_i^* = h(b_A \oplus PW_A) \oplus h(T_i \| A_i^* \| N_A)$, $P_{ik}^* = T_i \oplus h(h(y) \| N_A \| SID_k)$, $Q_i{}^* = h(B_A \| A_i{}^* \| N_A)$, where N_A is a nonce. The adversary A then sends (CID_i*, $P_{ik}{}^*{}_,$ $Q_i{}^*{}_,$ N_A), to server S_k.

S4: On receiving message (CID_i*, $P_{ik}{}^*{}_,$ $Q_i{}^*{}_,$ N_A), calculates $T_i = P_{ik}^* \oplus h(h(y) \| N_A \| SID_k)$, $A_i^* = h(T_i \| h(y) \| N_A)$, $h(b \oplus PW_i)^* = CID_i^* \oplus h(T_i \| A_i^* \| N_A) = h(b_A \oplus PW_A)$, and $B_i^* = h(h(b \oplus PW_i)^* \| h(x\|y)) = h(h(b_A \oplus PW_A) \| h(x\|y)) = B_A$. Hence $h(B_i^* \| A_i{}^* \| N_A) = h(B_A \| A_i{}^* \| N_A) = Q_i{}^*$ and the server S_k authenticates A.

S5: S_k generate a nonce N_k and calculates $M_{ik}' = h(B_i^* \| N_A \| A_i{}^* \| SID_k)$ and sends (M_{ik}', N_k) to A.

S6: After receiving the message, A computes $h(B_A \| \| N_A \| A_i{}^* \| SID_k)$. Obviously $h(B_A \| \| N_A \| A_i{}^* \| SID_k) = h(B_i^* \| N_A \| A_i{}^* \| SID_k) = M_{ik}'$ and A successfully authenticates S_k. Then A computes $M_{ik}'' = h(B_A \| \| N_k \| A_i{}^* \| SID_k)$ and sends to S_k.

S7: S_k on receiving M_{ik}'' computes $h(B_i^* \| N_k \| A_i^* \| SID_k)$. Obviously $h(B_i^* \| N_k \| A_i^* \| SID_k) = h(B_A \| N_k \| A_i^* \| SID_k) = M_{ik}$' and S_k successfully authenticates A. then the adversary A and S_k share a common session key SK = $h(B_A \| N_A \| N_k \| A_i^* \| SID_k) = h(B_i^* \| N_A \| N_k \| A_i^* \| SID_k)$

The above analysis demonstrates that a forgery attack can be launched by an attacker using his/her own SC information (B_A, h( ), h(y)) and by computing T_i from a intercepted login request message.

Inefficient Password Change Phase: In Lee et al.'s (2011) scheme, the information $B_i = h(h(b \oplus PW_i) \| h(x\|y))$ stored in the SC of U_i is calculated by RC using its master secret x, secret key y and password PW_i of U_i. Hence, whenever U_i needs to update his/her password there should be a communication with RC and secure channel should be established to modify password which increases the communication overhead.

The scheme proposed Li et al. (2013) for multi-server environments causes user's identity to change dynamically in every login request. The various phases of the scheme proceeds as follows:

Registration Phase: Step 1: U_i selects a number b, which is random in nature and calculates $A_i = h(b \oplus PW_i)$. U_i sends ID_i, A_i to RC.

Step 2: RC calculates $B_i = h(ID_i \|x)$, $C_i = h(ID_i \| h(y) \| A_i)$, $D_i = h(B_i \| h(x\|y))$, $E_i = B_i \oplus h(x\|y)$.

Step 3: (C_i, D_i, E_i, h(.), h(y)) are stored into SC by registration center and is send to U_i who stores b into his SC which contains (C_i, D_i, E_i, b, h

(.), h(y)).

Login Phase: Step 1: U_i types in his ID_i and PW_i after inserting SC into the system. SC calculates $A_i = h(b \oplus PW_i)$, $C_i *= h(ID_i \| h(y) \| A_i)$ and checks for equality with C_i stored in SC. If there is a mismatch, then login request is rejected. Otherwise U_i is considered as a legitimate user and the login request is generated as in step 2.

Step 2: SC calculates $P_{ij} = E_i \oplus h(h(SID_j \| h(y)) \| N_i)$, $CID_i = A_i \oplus h(D_i \| SID_j \| N_i)$, $M_1 = h(P_{ij} \| CID_i \| D_i \| N_i)$, $M_2 = h(SID_j \| h(y)) \oplus N_i$, where N_i is the nonce of SC . U_i submits (P_{ij}, CID_i, M_1, M_2) to S_j as a login request.

Verification Phase: During this phase S_j and U_i mutually authenticates each other and generates a shared session key be performing the following steps.

Step 1: S_j computes $N_i = h(SID_j \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_j \| h(y)) \| N_i)$, $B_i = E_i \oplus h(x\|y)$, $D_i = h(B_i \| h(x\|y))$ and $A_i = CID_i \oplus h(D_i \| SID_j \| N_i)$.

Step 2: S_j calculates $h(P_{ij} \| CID_i \| D_i \| N_i)$ and compares with the received M_1 . Request is rejected if there is a mismatch. Otherwise S_j successfully authenticates U_i and generates a nonce N_j to compute $M_3 = h(D_i \| A_i \| N_j \| SID_j)$, $M_4 = A_i \oplus N_i \oplus N_j$. S_j sends (M_3 , M_4) to U_i .

Step 3: U_i on receiving (M_3, M_4) U_i computes $N_j = A_i \oplus N_i \oplus M_4$, $h(D_i \| A_i \| N_j \| SID_j)$ and compares with the received M_3. If there is no match, U_i fails to authenticate S_j and terminates the session. Otherwise U_i computes $M_5 = h(D_i \| A_i \| N_i \| SID_j)$ and sends (M_5) to server.

Step 4: S_j on receiving (M_5) computes $h(D_i \| A_i \| N_i \| SID_j)$ and checks for equality with M_5 sent by U_i . On equality, S_j successfully authenticates U_i.

On successful mutual authentication, the session key, $SK = h(D_i \| A_i \| N_i \| N_j \| SID_j)$ is computed simultaneously by S_j and U_i.

Password Change Phase: This phase involves only U_i and SC and proceeds as follows.

Step 1: U_i types in his ID_i and PW_i after inserting SC into the system and submits request to change password.

Step 2: SC calculates $A_i = h(b \oplus PW_i)$, $C_i{}^* = h(ID_i \| h(y) \| A_i)$ and checks whether $C_i{}^* = C_i$ stored in SC. A mismatch results in rejection of request. Otherwise U_i is considered as a legitimate user and U_i chooses a $PW_i{}^{new}$ and b_{new}.

Step 3: SC computes $A_i{}^{new} = h(b_{new} \oplus PW_i{}^{new})$ and $C_i{}^{new} = h(ID_i \| h(y) \| A_i{}^{new})$ .The value in the SC is replaced with C_{inew} .

Security Analysis of Li et al. Scheme: This section analyses the security of the above discussed scheme and weaknesses are explained.

Denial-of-Service Attack: Though *Li et al. Scheme* uses nonce values, the freshness of nonce values are not checked which can result in a DoS attack. Assume that an attacker A, intercepts a (P_{ij}, CID_i, M_1, M_2) for login and resends the message at a later point in time. The server without verifying the freshness of the nonce N_i calculates, $N_i = h(SID_j \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_j \| h(y)) \| N_i)$, $B_i = E_i \oplus h(x\|y)$, $D_i = h(B_i \| h(x\|y))$, $A_i = CID_i \oplus h(D_i \| SID_j \| N_i)$. S_j calculates $h(P_{ij} \| CID_i \| D_i \| N_i)$ and compares with M_1. If equal S_j accepts the login request and computes

$M_3 = h(D_i \| A_i \| N_j \| SID_j)$, $M_4 = A_i \oplus N_i \oplus N_j$ where N_j is a nonce generated by S_j. S_j sends (M_3, M_4) to A . Here though A will not be able to compute N_j he will be successful in blocking the computing resources of the server and many such invalid login request messages can ultimately lead to DoS attack.

A DoS attack can also be carried out by a valid user B having access to h(y) and has malicious intentions. Then B can modify a login request message $(P_{ij}, CID_i, M_1, M_2)$ send by U_i, to S_j, where M_2 is calculated using a nonce N_B whereas P_{ij}, M_1 are calculated using N_i. The server computes the nonce N_B which is used to calculate E_i and M_1. The computed value will not match the received value and server will reject the login request from the honest user U_i denying access to resources he is authorized to access.

Smart Card lost Attack: Stealing of SC of a valid user U_i by a malicious valid user A will help A to launch this attack. A in this scenario will be knowing h(y) and from an intercepted login request message $(P_{ij}, CID_i, M_1, M_2)$ send by U_i to S_j, he/she can compute $N_i = h(SID_j \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_j \| h(y)) \| N_i)$ and $A_i = CID_i \oplus h(D_i \| SID_j \| N_i)$ where D_i is obtained from SC of U_i . Then A can try to retrieve the password of U_i by an offline guessing attack. The adversary A guesses a password PW_{iguess} and calculates $A_{iguess} = h(b \oplus PW_{iguess})$ and compares with A_i until the correct password if obtained.

Masquerade Server Attack (Madhusudhan and Adireddi 2014)*:* In Li et al.'s scheme the value $h(x \| y)$ is shared among all servers which is used by servers to verify the user. Assume that malicious insider A, of a registered server has the knowledge of $h(x\|y)$. Then if A intercepts a valid message (P_{ik}, CID_i, $M_{1,}$ M_2) send by U_i for login to a registered server S_k, A can compute $N_i = h(SID_k \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_k \| h(y)) \| N_i)$, $B_i = E_i \oplus h(x\|y)$, $D_i = h(B_i \| h(x\|y))$, $A_i = CID_i \oplus h(D_i \| SID_k \| N_i)$. A can generate a nonce N_A and compute $M_3 = h(D_i \| A_i \| N_A \| SID_k)$, $M_4 = A_i \oplus N_i \oplus N_A$. A sends ($M_3$, M_4) to U_i. U_i computes $N_A = A_i \oplus N_i \oplus M_4$, $h(D_i \| A_i \| N_A \| SID_k)$ and compares with the received M_3. Then U_i computes the mutual authentication message $M_5 = h(D_i \| A_i \| N_i \| SID_k)$ and sends (M_5) to the adversary A. A verifies M_5 and mutual authentication is done. Then the adversary A and U_i calculates $SK = h(D_i \| A_i \| N_i \| N_A \| SID_k)$, which serves as the session key.

Eavesdropping Attack: Presume that attacker A has understood the SC details of a registered user $U_{i.}$ Now if A intercepts the message (P_{ij}, CID_i, $M_{1,}$ M_2) from U_i to the server S_j, then A can compute $N_i = h(SID_j \| h(y)) \oplus M_2$, $A_i = CID_i \oplus h(D_i \| SID_j \| N_i)$ where D_i is obtained from the SC of U_i. Thereafter if A intercepts the message (M_3, M_4) from S_j then A can compute $N_j = A_i \oplus N_i \oplus M_4$ and the session key $SK = h(D_i \| A_i \| N_i \| N_j \| SID_j)$. The calculated session key can be used by A to eavesdrop on the future communications between U_i and S_j.

In a distributed cloud environment, millions of clients share the same computing infrastructure at a large scale. As a consequence, cloud environment demands stronger authentication mechanisms compared to traditional client-server systems. The following section discusses the most recent and relevant works published in the area of authentication in cloud.

2.4.3 Authentication Schemes for Cloud

Shen et al. (2010) proposed a theoretical prototype system to address the security concern in cloud computing environment. In the discussed system, cloud computing and trusted platform support services (TSS) are combined to provide secure cloud computing environment. TSS has its basis on trusted platform module and requires a separate device at the user side. Authors claimed that the proposed system design can offer better authentication, access control based on roles assigned and data protection in cloud. However, Shen et al.'s scheme does not address the authentication for cloud computing users.

Celesti et al. (2010) in their work discusses Identity Management and authentication issues in a cloud federation scenario. Authors analyse the issue of identity management in an inter-cloud environment and proposed a reference architecture. The work which focuses on heterogeneous and federated clouds distinguishes clouds as home clouds and foreign clouds. Home cloud is described as a cloud provider, whose capability of virtualization infrastructure has reached the maximum capacity, preventing further instantiation, of virtual machine instances and hence forwards its requests for federation to foreign clouds who shares part of its computing capabilities for free or by charge. The work does not discuss authentication for cloud computing users.

Kang and Zhang (2010) in their work discusses an authentication scheme which uses the concept of bilinear pairing to authenticate a user who wants to share the data stored by another user in the cloud, when both the users are in the same domain. The owner will validate the request for data and will send a token with a signature to the requestor which is submitted to the cloud end, who verifies the same and permits or denies access to the user.

Chow et al. (2010) proposed an authentication model for mobile users that takes into consideration the input constraints, power limitations and practical computation capability of handsets. The proposed approach for authentication is based on a framework that supports decisions regarding authentication. This flexible framework named “Trustcube” also uses an approach based on “user behavior” referred to as “implicit authentication” where in user's past behavioral data is used to authenticate to access a service. Chow et al., describes, an authentication platform where policies and open standards are used to facilitate the integration of different authentication methods.

Pertinant observations on the scheme are as follows:

- The authentication factor is “What the user does”.
- Proposed framework for authentication is based on the past behavioral data of the users’ viz. the history of the web sites visited by the user.
- Users’ activities are tracked and stored for future reference which questions the privacy of the user.

Lee et al. (2010) proposed a Two-Factor authentication framework that prevents unauthorized access to cloud services. The proposed authentication is carried out in two steps, In the first step, the PKI authentication is used and only registered users with valid certificates will be permitted by the cloud authentication server to proceed to the next step.

The second step uses OOB (Out-of-Band) authentication in which a one-time random code is sent via SMS to the user's mobile phone, by the authentication server. The code is verified by the web server before granting access to the user.

Salient observations on the scheme are as follows:

- Susceptible to attacks on SMS based OTP's, the one-time random code being sent via SMS to the user's phone. Mulliner et al. (2013) explains in his paper that SMS-based one-time passwords (OTP) are prone to threats such as SIM Swap attack, wireless interception due to security vulnerabilities in GSM network. Mulliner also mentions in his work that mobile phone malware, particularly Trojans designed specifically for intercepting messages containing SMS-based OTP's have become a serious threat.

- A communication with the Certificate Authority is required to verify the certificate of the user.

- Implementation of Public Key Infrastructure (PKI) increases the complexity of the authentication systems.

- Password is submitted in plain text form to the registration server during the registration process, which can be captured by an attacker if the communication channel is not secure. Also the password is revelaed to the server which makes the scheme susceptible to insider attack.

- The server stores the password of user and hence the scheme is prone to stolen verifier problems.

- Provides no support for changing user's password.

- Authentication of the server is not done by the user which makes the scheme prone to man-in-the-middle attack. A Man-in-the-Middle (MITM) attack is launched by an adversary who sniffs the messages exchanged between a client and server, modifies message and inserts messages pretending to be an honest user or server (Chen and Yeh 2005)

- Registration is done directly at the service providing server. Thus to access multiple services, user needs to undergo multiple registration processes and maintain multiple accounts.

Zhu et al. (2011), proposed a novel biometric-based authentication scheme in which voice template is used for user authentication. The authentication process is carried out in two phases viz. the enrollment phase and the matching phase. The discussed approach uses homomorphic encryption to encrypt the code book and voice print biometrics. The authentication system computes distortion measurement without disclosing user's data by comparing user's encrypted biometric data with the encrypted code book.

Noteworthy observations on the scheme are as follows:

- The size of code book database increases with the number of users and hence the computational overhead increases as user's increase.

- In Voice recognition systems it is difficult to control sensor and channel variances that significantly impact capabilities. Also voice of different individuals may not be sufficiently distinctive for identification over large data bases (PBworks, 2007).

Liu et al. (2012), proposed a cloud mutual authentication scheme to solve the authentication problem between the user and the cloud server. This scheme applies Trusted Computing Technology and smart card authentication to cloud computing service platform. The server uses TPM which is in hardware architecture, to generate Public and Private Key instances and this key will be specific to hardware.

Most relevant observations on the scheme are as follows:

- ID is transmitted in the plain text form to the cloud server during registration and this can be captured by the attacker.
- During login phase, only ID is verified by the smart card. Hence even a wrongly entered password will cause a login message to be send to the server.
- Time stamps are used to resist replay attack. This can result in time concurrency issues (Gong 1992, Chang 2006).

Dinesh and Agrawal (2012), proposed a multi-level password based authentication scheme. Authentication is done at the organization, user and team levels. Authors propose the generation of passwords by concatenating passwords at different levels viz. password within the organization, password within the team and password for the particular user. At the user level, the scheme verifies the authorization of the user to access a cloud resource. Security of the scheme is solely dependent on password of the user which can be hacked by social engineering attacks.

Zwattendorfer and Tauber (2012) in their work suggests the use of national electronic IDs (eIDs) for user authentication in cloud. The work discusses the requirement for providing Single Sign-on functionality in the cloud environment. Authors point out that user’s have to undergo multiple authentication processes to access the services of different service providers, if service provider appications are bundled for example, through a web portal or a one-stop shop. Single Sign-on (SSO) provides users with the ability to authenticate once and access several secured resources in a distributed network environment.

Significant observations on the scheme are as follows:

- E-Government or e-health services have to achieve higher security and privacy requirements as they need to comply with national law or data protection requirements.

- Cloud applications dealing with sensitive areas such as e-Government and e-health sector require more reliable and secure authentication mechanism than the conventional username/password authentication.

- For achieving higher security requirements for identification and authentication in cloud, the work extends the existing eID framework STORK (Security Across Borders Linked), which is the identification and authentication framework across Europe.

- The deployment of extended STORK framework onto public cloud platforms may render the personal data such as the unqiue identifier of the user, fully visible to the cloud service provider.

Banyal et al. (2013), proposed a multi-factor authentication framework for cloud. The authentication mechanism combines traditional authentication based on ID and password with an approach that uses splitting secret value and Captcha values in encrypted form for user authentication. According to the risk involved and security required, authors categorize the cloud services and resources into low, medium and high. Depending on the type of the resource accessed, user needs to undergo one, two or three levels of authentication by sending an encrypted captcha, a one-time key and the IMEI number.

Salient observations on the scheme are as follows:

- A secret key is shared between each user and server and this value is stored by the server.

- ID and Password are sent in the clear text form to the server during registration phase.

- Password information is stored by the server and is used to verify the user during authentication phase. The scheme is susceptible to stolen verifier attack.

- One-Time key is send via SMS which makes the scheme prone to attacks on SMS-based one-time passwords (Milliner et al 2013). Abraham (2009) in his article on two-factor authentication in cloud, has mentioned that availability of SMS based OTP’s is dependent on the network coverage of the user’s mobile phone and user’s will also incur SMS costs.

- Password change requires the involvement of the server.

Saurabh et al. (2013), proposed an authentication scheme for authenticating a mobile device to a cloud service. The proposed scheme viz. Message Digest Authentication (MDA) uses ID, Password, encrypted hashed messages (message digests) for authentication and can be used by mobile devices not having IMSI chips. Since MDA is used, a loss of the mobile device will not compromise authentication information of user. This scheme requires the server to maintain a verification table and the scheme does not provide the user with the option to change password.

Since the current research, focuses on Two-factor authentication schemes using hash funtions, this section includes a detailed discussion of a few of the most recent works in this area:

Hao et al. Scheme (2011): Hao et al. proposed a time-bound ticket based mutual authentication using smart cards. Here, users are registered with cloud servers who issues digital tickets to the client. Tickets are linked to the smart card of the client. A ticket is used only once for verifying integrity of data after which the ticket becomes obsolete. The authentication scheme includes three phases which can be explained as follows:

Registration Phase: During this phase user U_i registers with the server S.

Step 1: U_i selects identity ID_i, password PW_i and a random number b. U_i computes $IPB_i = h(ID_i \| h(PW_i \oplus b))$ and sends $\{ID_i, IPB_i, t\}$ where t is the number of tickets U_i needs and U_i pays to S for the tickets issued.

Step 2: Server S, on receiving the message generates the tickets for Ui where $T_i^{(j)}$ denotes the jth ticket of U_i where j = 1, 2…. t. The ticket ID and valid period of $T_i^{(j)}$ is denoted by $TID_i^{(j)}$ and $VP_i^{(j)}$ respectively. Thus S generates $\{(TID_i^{(j)}$ and $VP_i^{(j)}, j = 1,2, \ldots.t)\}$ and computes the following:

$W_i = IPB_i \oplus h(ID_i , K_1)$, $\alpha_i^{(j)} = H_{K_2} (ID_i \| TID_i^{(j)} \| VP_i^{(j)})$,

$\beta_i^{(j)} = \alpha_i^{(j)} \oplus IPB_i$

where K_1 and K_2 are a long term secret keys of S.

$T_i^{(j)} = (T_i^{(j)1} , T_i^{(j)2})$, of which $T_i^{(j)1} = (TID_i^{(j)}, VP_i^{(j)})$, $T_i^{(j)2} = \beta_i^{(j)}$.

S computes $Z_i = H_{K_2} (ID_i) \oplus IPB_i$, which is used for changing user password.

Step 3: S includes $\{ID_i, t, W_i, Z_i, T_i^{(j)}, j = 1, 2\ldots. t.)$ into a SC and issues to U_i

Step 4: U_i stores b into the smart card.

Verification Phase: U_i can use the t tickets in the SC to perform data verification for a maximum of t times. Assuming the U_i is using the mth ticket, the verification proceeds as follows:

Step 1: U_i keys in ID_i and PW_i after inserting SC into the system.

Step 2: SC generates a nonce r_u as per system time and calculates $IPB_i = h(ID_i \| h(PW_i \oplus b))$, $H_i = W_i \oplus IPB_i$, $C_1 = r_u \oplus H_i$, $C_2 = h(r_u) \oplus T_i^{(m)}{}_2 \oplus IPB_i$

Step 3: SC sends $\{ID_i, T_i^{(m)}{}_2, C_1, C_2\}$ to S.

Mutual Authentication Phase: S performs the following steps to verify U_i:

Step 1: S verifies ID_i, and rejects the request if ID_i is invalid.

Step 2: S verifies that ticket with ID, $TID_i^{(m)}$ is already used by comparing with the used tickets published in the bulletin board. If so, the request is rejected and session is terminated.

Step 3: S checks whether the period $VP_i^{(m)}$ is within the current date. If not, the ticket is rejected.

Step 4: S then computes $D_0 = H(ID_i , K_1)$, $D_1 = C_1 \oplus D_0$ and $D_2 = H(D_1) \oplus C_2$.

Step 5: S calculates $H_{K_2} (ID_i \| TID_i^{(m)} \| VP_i^{(m)})$ and compares with D_2. If there is a mismatch, request for login is rejected. Otherwise S authenticates U_i successfully.

Step 6: S calculates $C_3 = D_0 \oplus r_s$, $C_4 = h(r_u , r_s)$, $K_S = h(D_0 , r_u \| r_s)$ where r_s is a nonce of S and K_S is the session key for subsequent sessions. S sends, (C_3, C_4) to U_i.

Step 7: SC computes $D_3 = C_3 \oplus H_i = H(ID_i , K_1)$ and compares $h(r_u , D_3)$ with C_4. If they are not equal, U_i fails to authenticate S. Otherwise after successful authentication, SC computes $K_C = h(H_i , r_u \| r_s)$ and uses K_C as the session key to communicate with S as $K_C = K_S$. The ticket $Ti^{(m)}$ is removed by U_i from SC after the Mth verification is over, and $TID_i^{(m)}$ is published by S on its bulletin board.

Password Change Phase: This phase is used by U_i to change the stored password.

Step 1: U_i keys in his ID_i and PW_i after inserting SC into the system.

Step 2: SC generates a nonce r_u as per system time and computes $IPB_i = h(ID_i \| h(PW_i \oplus b))$, $C_1 = r_u \oplus W_i \oplus IPB_{i,}$ $C_2 = h(r_u) \oplus Z_i \oplus IPB_i$

Step 3: SC sends {update, ID_i, C_1, C_2} to S wherein update indicates a password change request.

Step 4: S checks validity of ID_i, and rejects the request if ID_i is invalid

computes $D_0 = H(ID_i , K_1)$, $D_1 = C_1 \oplus D_0$ and $D_2 = h(D_1) \oplus C_2$.

Step 5: S calculates $D_1 = C_1 \oplus H(ID_i , K_1)$, $D_2 = h(D_1) \oplus C_2$ and checks whether $D_2 = H_{K_2}(ID_i)$. If there is a mismatch, S rejects the request. Otherwise S authenticates U_i successfully and request for change is accepted.

Step 6: S computes $C_3 = H(ID_i , K_1) \oplus r_{s,}$ $C_4 = h(r_u , r_s)$, where r_s is a random nonce generated by S. S sends, (C_3 , C_4) to U_i.

Step 7: SC computes $D_3 = C_3 \oplus W_i \oplus IPB_i$ and compares C_4 with $h(r_u , D_3)$. If equal, SC authenticates S and prompts U_i to enter new password.

Step 8: U_i enters new password. PW_i^{new}. SC computes $IPB_i^{new} = h(ID_i \| h(PW_i^{new} \oplus b))$, $W_i^{new} = W_i \oplus IPB_i \oplus IPB_i^{new}$, $Z_i^{new} = Z_i \oplus IPB_i \oplus IPB_i^{new}$. and replaces W_i with W_i^{new} and Z_i with Z_i^{new} in the SC. The SC also updates $T_i^{(j)2}$ with $T_i^{(j)2} \oplus IPB_i \oplus IPB_i^{new}$. for all remaining tickets in the SC.

Most relevant observations on the scheme are as follows:

- Absence of early detection of wrong password before generation of login request renders the scheme susceptible to denial-of-service attack.

- Tickets are required to access services and on expiry of the issued tickets, client needs to purchase new tickets from the server. Thus a new smart card needs to be issued or smart card contents need to be modified every time new tickets are issued.

- Password update phase requires the involvement of the server, since user authentication is done by the server before allowing the user to change his/her password.

- Registration is done by the service providing server. Hence to access multiple cloud services, user should register individually for each service and should maintain a smart card issued by that service provider. Thus multiple authentication tokens should be carried to access multiple services.

Choudhury et al. (2011) proposed a user authentication framework for cloud. Authors discussed a novel idea that provides identity management with authentication using smart card. The scheme which uses light-weight XOR and hash operations, applies a two-step verification to authenticate a user. Verifcation is done using password, smart card and out-of-band authentication in which a one-time key is send as SMS via HTTP/SMS gateway.

Registration Phase: During this phase, user U_i registers with server S. After successful registration, S issues U_i with a smart card.

Step 1: U_i selects ID, PW and a random number x. U_i calculates $h(PW \oplus x)$ and sends $\{ID , h(PW \oplus x) ,h(x)\}$ to S.

Step 2: S checks whether the ID is available. If not U_i is prompted to repeat the process from step 1. Otherwise, S calculates $J = h(ID \oplus h(PW \oplus x))$, $I = h(ID||y)$, $B = g^{h(I||J) + h(x) + h(y)} \bmod p$, where ‘y’ is a nonce. S stores {I, J, B, p, g, h(.)} into the SC and issues to U_i who stores x into the SC.

Step 3: Server S enters user ID in a table stored by the server.

Login Phase: In this phase user U_i sends a login request to the cloud server S.

Step 1: U_i keys in his ID, PW after inserting SC into the system.

Step 2: Client calculates $J_1 = h(ID \oplus h(PW \oplus x))$ and compares with the J in the SC. If there is a mismatch the session is terminated. Otherwise U_i will compute $C = h(I||J)$ and sends the login request message $M_1 = \{B, C\}$ to S.

Step 3: S generates a one-time key K and compute $B'' = g^{C+h(y)} \bmod p$, $h(B'')$, $L = h(B'' || k)$ and $h(L)$. S sends $M_2 = \{h(B''), h(L)\}$ to U_i using a public channel and the one-time key K via SMS to the user’s phone.

Step 4: U_i on receiving M_2 calculates $B' = Bg^{-h(x)} \bmod p$, $h(B')$ and $L^* = h(B'|| k)$ and $h(L^*)$. U_i compares $h(B')$ with $h(B'')$ and $h(L^*)$ with $h(L)$. If they are not equal U_i aborts the session. Otherwise computes $R = h(T||B')$ using the current time stamp T and sends $M_3 = \{I, h(R), T\}$ to S over a public channel.

Authentication Phase: This phase is carried out by the cloud server S to verify the authenticity before allowing A to login.

Step 1: S checks whether $T' - T \leq \blacktriangle T$ is satisfied or not. If not, then the session is terminated. Otherwise S proceeds to compute $I' = h(ID||y)$ and $R^* = h(T||B'')$ and checks whether $h(R^*) = h(R)$ and $I'=I$. If both the conditions are satisfied, then S proceeds to the step 2. Otherwise terminates the session.

Step 2: S generates the session key $S_k = (R \oplus L)$ and sends $M_{4} = h(S_k)$ via a public channel to U_i.

Step 3: On receiving M_4, U_i verifies S_k by computing $(R \oplus L)$.

Password Change Phase: This phase permits U_i to change the password without the intervention of S.

Step 1: U_i enters ID, PW and computes $J^* = h(ID \oplus h(PW \oplus x))$. J^* is compared with the J stored in the SC. If they are not equal the session is terminated. Otherwise U_i enters the new password, PW^* and generate x^*.

Step 2: The new value of J is computed as $J' = h(ID \oplus h(PW^* \oplus x^*))$ and J' replaces J in the SC.

Significant observations on the scheme are as follows:

- Flaw in password change phase:

The smart card contains the values {I, J, B, p, g, h(.)} where $J = h(ID \oplus h(PW \oplus x))$, $I = h(ID||y)$, $B = g^{h(I||J) + h(x) + h(y)} \bmod p$. Here J is calculated using user's password and the value of B contains J.

During password change phase, user U_i enters ID, PW. SC computes

$J^* = h(ID \oplus h(PW \oplus x))$ and checks whether $J^* =$ J stored in the SC. If they are not equal the session is terminated. Otherwise U_i enters the new password, PW_{new} and generates x_{new}. The new value of J is computed as $J_{new} = h(ID \oplus h(PW^* \oplus x^*))$ and J_{new} replaces J in the SC.

While modifiying the password during the password change phase, only the value of J is re-calculated using the new password and x values. The value of B which contains J remains the same. B should be re-calculated using the new value of J and x, which is not done. This will lead to login failures once the user changes the password.

After modifiying the password, user calculates $C = h(I||J)$ and sends login request <B, C> to the cloud server. Server computes $B'' = g^{C+h(y)} \bmod p = g^{h(I||J)+h(y)} \bmod p$ and the value h(B") is send by the server to user . U_i calculates $B' = Bg^{-h(x)} \bmod p = g^{h(I||J_{old}) + h(x_{old}) + h(y)} g^{-h(x)} \bmod p$. Since the B value was not re-calculated using the changed J and x values, h(B') will not match with h(B") and the user will not be able to login.

- One-time key is send via SMS to the user's mobile phone. Mulliner et al. (2013) explains in his paper that SMS-based one-time passwords (OTP) are prone to threats such as SIM Swap attack, wireless interception due to security vulnerabilities in GSM network. Mulliner also mentions in his work that mobile phone malware, particularly Trojans designed specifically for intercepting messages containing SMS-based OTP's have become a serious threat.

- To resist replay attack time stamps are used. However, the time-stamp based approach faces some draw backs such as variation in time zone, delivery latency etc. (Chang et al. 2006) and clocks can become unsynchronized due to faults in the synchronization mechanism (Gong 1992).

- In this scheme, registration is done by the service providing server and hence to access services of different service providing servers, a user needs to go undergo multiple registration processes and carry multiple authentication tokens.

Jaidhar (2013) in his work mentions that among the security issues of cloud computing, authentication is considered as one among the most important issues. He proposed a two-factor authentication scheme using password and smart card to address the vulnerabilities in authentication scheme which allows an intruder to gain access to cloud resources.

Major observations on the scheme are as follows:

- The proposed scheme has a flaw in the computations done during the login phase which will prevent even a valid user from successfully completing the login process and to access cloud services. This flaw can be explained by discussing the steps in the registration phase and login phase of the scheme as follows:

Registration Phase: During this phase, user registers with server S. After successful registration, S issues U_i with a smart card.

Step 1: U_i selects ID_i, PW_i, random number b and calculates $IPB_i = h(PW_i \oplus b)$. U_i sends $\{ID_i, IPB_i, t\}$ to cloud server over a secure channel.

Step 2: S generates 't' tickets for the user. S calculates $IU_i = h(ID_i{}^x \bmod p) \oplus IPB_i$, $A_i = h(IU_i \oplus IPB_i \| ID_i) = h(h(ID_i{}^x \bmod p) \| ID_i)$.

Step 3: S computes $B_i^j = h_{ki}(ID_i \| TID_i \| TID_i^{(j)} \| VP_i^{(j)})$ where j = 1,2, …t.

Step 4: S issues a SC containing $\{IU_i, A_i, B_i^j, T_i^{(j)}\}$ into the SC and issues to U_i who stores b into the SC.

Login Phase: In this phase user U_i sends a login request to the cloud server S.

Step 1: U_i enters SC and inputs ID_i, PW_i,

Step 2: SC calculates $IU_i' = IU_i \oplus h(PW_i \| b) = h(ID_i{}^x \bmod p) \oplus IPB_i \oplus h(PW_i \| b) = h(ID_i{}^x \bmod p) \oplus h(PW_i \oplus b) \oplus h(PW_i \| b)$.

$A_i' = h(IU_i' \| ID_i) = h(h(ID_i{}^x \bmod p) \oplus h(PW_i \oplus b) \oplus h(PW_i \| b) \| ID_i)$.

Step 3: A_i' is the value calculated by SC. Now the value of A_i stored in SC is $A_i = h(h(ID_i{}^x \bmod p) \| ID_i)$.

A_i' will never be equal to A_i since there is a computation mistake. Author is using $IPB_i = h(PW_i \oplus b)$ to calculate A_i in the registration phase. However, during the login & verification phase, the value used to calculate IU_i' is $h(PW_i \| b)$. This will result in a different value of A_i' which is calculated using $IU_i'$and hence the login request will fail. The same computational mistake is occurring during the mutual authentication phase and password change phase.

- Mutual authentication phase involves the calculation of a shared key K_A. However, it results in two different values at the client and server side due to a wrong computation at the client side by smart card.

- Password change phase involves the same steps as in login request phase which results in a value that do not match with the value stored in smart card. Hence password change request will always be rejected.

- Registration and issuing smart cards are done by the service providing server. In a scenario where the user needs to access different cloud services, he will need to undergo multiple registration processes and carry multiple smart card (authentication factor).

Rui Jiang (2013) proposed a scheme which uses password and SC to overcome the limitations of Choudhary et al.'s scheme. Only simple hash functions and xor operations are used in the authentication protocol and does not use OOB authentication as required in the case of Choudhary et al.'s scheme.

Registration Phase: This phase is invoked by the user U_i to register to the cloud server S. After successful registration, S issues, U_i with a smart card.

Step 1: U_i selects identity ID, password PW and a random number x. U_i computes $h(PW \oplus x)$ and sends $\{ID , h(PW \oplus x) ,h(PW)\}$ to S through a secure channel.

Step 2: S checks whether the ID is available and not issued to another user. If not S rejects request for registration. Otherwise, S calculates $I = h(ID||y)$, where y is a secret number generated by S.

$B = g^{ID + h(PW) + h(y)} \bmod p$. S issues to U_i via a secure channel, a SC containing {I, B, p, g, h(.)} and A stores x into the SC.

Step 3: ID and $h(PW \oplus x)$ are stored by S in the server.

Login Phase: When U_i wants to login into S, this phase is executed.

Step 1: U_i inserts his SC and types in ID, PW.

Step 2: The SC computes $C = h(ID \| h(PW \oplus x) \| T_u)$ where T_u denotes U_i's current time stamp. U_i sends {ID, C, T_u} to S using a public channel.

Authentication Phase: S on receiving the message {ID, C, T_u} verifies the identity of U_i by performing the following steps.

Step 1: S checks whether $T_u' - T_u \leq \blacktriangle T$ is satisfied or not. Here T_u' is the current time stamp of S and $\blacktriangle T$ is the maximum allowed delay in transmission. If the condition is not satisfied, then the login request is rejected. Otherwise S computes $I^* = h(ID\|y)$ and $C^* = h(I^* \| h(PW \oplus x) \| T_u)$. If $C^* = C$, S accepts login request of A and computes $K' = g^{ID + h(y)} \bmod p$, $h(K')$, $R = h(K' \| T_s)$, where T_u is the current time stamp of S. S generates a random number a and sends $E_{h(K')}\{R, T_s, a\}$ to U_i.

Step 2: U_i computes $K'' = Bg^{-h(PW)} \bmod p$ and $h(K'')$, $E_{h(K'')}\{R, T_s, a\}$ to obtain $\{R, T_s, a\}$. U_i checks T_s with current time stamp and terminates the session if the transmission delay is more than the allowed maximum. Else U_i computes $R' = h(K' \| T_s)$ and compares with the R received from S. If $R' = R$, U_i successfully authenticates S and sends h(a) to S.

Step 3: S checks h(a) and if correct, mutual authentication is successfully done and both U_i and S calculates the session key as $S_k = h(K' \| a) = h(k \| a)$.

Password Change Phase: This phase permits U_i to change the password in the SC.

Step 1: U_i types in ID, PW after inserting SC into the system. U_i sends to S, $E_{S_k}\{h(PW \oplus x) \| h(PW' \oplus x) \| b\}$ where b is a random number and

PW’ is the new password of U_i. The steps in the login and authentication phase are executed and after successful authentication, A sends a request for changing password to S and submits $h(PW \oplus x)$ and $h(PW' \oplus x)$.

Step 2: S verifies $h(PW \oplus x)$ and replaces it with $h(PW' \oplus x)$. S sends h(b) to A.

Step 3: A verifies h(b) and if correct SC computes $Z = Bg^{-ID-h(PW)} \bmod p$, $B' = Zg^{ID+h(PW)} \bmod p$, A replaces B with B’ in the SC.

Most relevant observations on the scheme are as follows:

- Susceptible to stolen verifier attack and Denial-of-Service (DoS) attack since variant of password is stored at the server ie. $h(PW \oplus x)$.

- Login request is created without verifying the password. So even if a wrong password is entered, login request will be created without verifying the password and hence attacker can easily launch DoS attack by either entering an incorrect password or identifier.

- Password change is done by the server and each time the password is changed, communication is required between the user and the server. This phase is also prone to DoS attack.

- Time stamps are used to resist replay attacks and this can result in time concurrency issues. Protocols using time stamps can suffer from problems due to variation in time zone, delivery latency etc. (Chang et al. 2006) and Gong (1992) mentions in his work that clocks can become unsynchronized due to faults in the synchronization mechanism.

- In this scheme, to access the services of the service provider, user needs to register directly at the service providing server. In such a scenario, to access different cloud services, a user will have to undergo multiple registration processes and will need to carry around multiple authentication tokens such as Smart Cards and crypto-tokens.

As the two-factor authentication protocols discussed in sections 3.1.4, 4.1.4 and 5.3 requires the use of mobile phones and Quick Response Code (QR-Code), a few relevant authentication schemes using mobile phones as an authentication factor is discussed in the following section.

2.4.4 Authentication Using Mobile Phone

The rapid advancements in the field of mobile communication technologies have lead to the invention of smart phones with commendable storage and processing capabilities. Hence, many authentication schemes that leverages the use of mobile phones as an authentication factor have been proposed by researchers. QR-codes (ISO/IEC 2000) or “Quick Response” codes, introduced by Denso-Wave, a Japanese company, provides a new input interface to smart phones. This two- dimensional bar code which can store a greater volume of information compared to bar codes, can be scanned and read by devices having embedded QR code scanning application. Majority of the currently available smart phones come with in-built software that can decode the scanned QR code (Falas and Kashani 2007).

Liao et al. (2009) proposed an authentication system that eliminates the usage of password verification table. Authors are discussing a practically feasible authentication solution using one-time passwords (OTP) and QR code which is used as an input interface to communicate the OTP to the user’s mobile phone. This scheme requires the user to share a secret key with each service provider and hence the user needs to store different secret keys in his mobile phone to access the services of different service providers. Time stamps are used to verify the originality of messages and protocols using time stamps are prone to issues due to differences in time zone at client and server, latency in delivery etc. (Chang et al. 2006) and clocks can become unsynchronized due to faults in the synchronization mechanism (Gong 1992).

Lee et al. (2010), proposed an authentication system for online banking using a mobile-OTP in combination with a QR code. To authenticate to the server, user needs a mobile OTP program downloaded into his mobile phone. A random value send by the server and mobile serial number is used by user to generate OTP which is used to authenticate to server. In the discussed scheme server authentication is not done and the user does not verify whether the QR code is generated by the correct server. The protocol requires public key certificates and verification of digital signatures to complete the authentication process.

Mukhopadhyay and Argles (2011) proposed a Single Sign-On (SSO) model for user authentication. The work uses QR-code based one-time passwords to address the issues of phishing attacks inherent in Single Sign-on based authentication systems. In this scheme, during registration, user name and password are submitted in plaintext form to Identity Provider (IdP) which makes the system susceptible to password guessing attack. IdP maintains a verification table to store the root password of the users, which makes the scheme susceptible to stolen verifier attack. Time stamps are used by client and IdP to verify originality of the messages and this requires the respective clocks to be synchronized in time and this can lead to time concurrency issues (Gong 1992, Chang 2006). In the discussed authentication scheme, if a user has to change the password, then it can be done only with the intervention of the server.

David (2012) proposed a proof of concept authentication system that provides two factor authentication by combining a password and a camera equipped mobile phone for authenticating the user. In this scheme which provides both online and offline mode of authentication, user passwords, IMEI are stored in the server. A public, private key pair is generated for each user which means that the users need to share a key pair with every server. Hence to access the service of multiple service providers a user needs to maintain multiple accounts and different key pairs. The server stores the password, private key/public key pair of all the users. Storing the password and key pairs of users by the server makes the scheme prone to stolen verifier problems. Also since password is stored by the server, changing of password requires server's support.

Dodson et al. (2012) proposed a phone based authentication system for making online payments. The user stores a shared secret which is a random key generated by the server and during the process of account creation. This shared secret which is unique to each user is maintained by the server and is later used to verify the authenticity of the user during login process. The secret is communicated by the server to the user via the QR code which is scanned and stored on the phone's password manager. The user needs to negotiate and manage a shared secret with each web site it visits and this means that, to access multiple services, the user needs to maintain different shared secrets. The stored shared secret is used to generate the response to the challenge that is send by the server during the login process.

2.4.5 Security Attacks on Authentication Protocols

This section discusses common security attacks applicable to authentication protocols. The contents of this section are referred from the work of Misbahuddin (2010).

Replay Attack: A Replay attack is launched by an adversary to gain unauthorized access to the system. This attack is performed by intercepting a message exchanged between two honest communication partners and retransmitting the captured message at a later point in time.

Replay attacks can be handled by changing some value in the message during every session. To achieve this, time stamps and random values are included in transmitted messages, so that, the freshness of these values are checked by the verifier to ascertain the originality of the received message. Time stamps requires time synchronization between communicating entities and this can lead to time concurreny problems especially when client and server belongs to different time zones (Chang etal. 2006). To overcome the time concurreny problem many protocols use random numbers or nonce (number used once) values, which varies with time. To check the freshness of the nonce, the verifier needs to maintain a previous nonce value for a certain period of time.

Guessing Attack: The tendency of human beings to use simple passwords make them insecure. To launch a guessing attack, the adversary understands the nature of the password, guess an arbitrary password and verifes the guessed password by logging in repeatedly until he gets the correct password. To prevent online guessing attack, many systems, block the account of the user after a certain number of login attempts. However, the probability of successful password guessing is high in an offline scenario as there is no restriction on the number of trials.

Brute Force Attack: A brute-force attack is a type of password guessing attack in which the attacker attempts to guess the correct password by trying sequentially every possible combination of numbers, upper and lower case letters, alphanumeric characters, spaces etc. The process is continued until the attacker arrives at the correct password. This attack, which requires a lot of time and computing power, is carried out using automated tools.

Dictionary Attack: A dictionary attack is a variant of password guessing attack wherein the attacker attempts to guess the password by trying out passwords from a list of most popular passwords. Such popular passwords are usually collected and traded by hackers and are available in the form of a list of commonly used passwords. To resist dictionary attack, the password of users should include a combination of numbers, upper and lower case letters and should not be a word from the dictironary.

Insider Attack: Insider attack is performed by a system administrator or an employee of the service provider who has access to the secret information of the user. From a convenience perspective, users' have a tendency to use one password to access multiple applications such as e-mail, online banking etc. If an insider with privileges of an administrator can access the password information of a registered user maintained by the Authentication Server, they can use it to impersonate the user or leak out the information to others.

Stolen Verifier Attack: Many authentication servers, store user passwords in a hashed form or a hash of the salt value combined with the user password in the database. To launch a stolen verifier attack, the adversary steals this verification table and attempts to guess the password using an offline guessing attack. The attacker compares entries in the verification table with the message digest of entries in a dictionary of passwords or he compares the entries with a rainbow table. The adversary will arrive at the right password when there is a match.

Shoulder Surfing Attack: Attacker obtaines the authentication credentials of the target by monitoring his typing of credentials without his knowledge. Even partial information about the victim gathered via this attack can pose serious threats when used to launch other attacks. For example, a password guessing attack can be launched by using the password length information gathered using a shoulder surfing attack.

Server Spoofing Attack: Server spoofing attack is launched by an attacker who impersonates a legitimate server and exchange messages with the user to attain the objective of gathering the secret credentials of the user. To thwart Server spoofing attack, the user should properly authenticate the server, before exchanging secret information.

Man-in-the-Middle Attack: This attack is an active form of eavesdropping wherein the attacker creates separate connections with the client and the server and intercepts the messages exchanged between them and replaces them with fabricated message. The victims will be made to believe that they are talking to each other directly over a secure channel while the entire conversation is being monitored and controlled by the attacker. In reality, both the victims are receiving messages injected into the channel by the man-in-the-middle whose existence is transparent to them.

Phishing Attack: Phishing is a very popular attack with cybercriminals wherein the attacker lures a legitimate user into revealing his sensitive information such as login credentials, credit card details, account information etc. by pretending to be a trustworthy entity. In most of the cases, the victim receives a mail that appears to have been sent by a reputed source or by a contact known to the victim. The victim on clicking a link in the mail will be directed to a web site that looks very similar to a valid server page where he will be prompted to divulge his personal or financial details.

Impersonation Attack: In impersonation attack, the adversary attempts to gain unauthorized access to resources hosted by the server or access the secret credentials of the user by pretending to be a legitimate entity. The attacker acts like a legitimate server or a registered user and attempts to fool the other entity to believe that he is communicating with an honest entity.

Reflection Attack: This attack is launched by an attacker who convinces the target to reveal the secret to the challenge generated by the victim. Reflection attack which is performed on mutual authentication protocols is launched by creating parallel sessions. Assume that an attacker who impersonates a legitimate user sends a login request message to the server. The server generates a challenge and sends to the attacker who is expected to generate a response. The attacker who is ignorant of the values required to generate the response, establishes another session with the server and sends the message received in the previous session. The server generates and sends to the adversary a new secret which is used by the attacker in the sesson established first. Server verifies and validates this response and resource access is permitted to the attacker. Reflection attack will succeed if challenge-response messages are symmetric in nature.

Crypto-token/ Mobile-Token Lost Attack: If an adversary gains possession of the token of a legitimate user, then he can attempt various nefarious activities such as offline password guessing, impersonate the user and gain unauthorized access to resources by logging into his account, modify the stored contents of the token etc. The authentication protocol and the contents of the token should be designed such that it is infeasible for the attacker to derive secret information from the token or launch the discussed attacks.

Denial-of-Service Attack: There are three different ways in which a denial-of-service attack can be launched. (1) Assume that an administrator who has access to the user database stored in the server, modifies the secret information used to authenticate the user. In that case, a legitimate user, who attempts to login with his valid credentials, will be denied from accessing the resouces he is authorized to access. (2) Another possible scenario where a denial-of-service attack can happen is when the crypto-token is possessed by the attacker. In this case, the attacker attempts to login to the account of the owner of the token using a random password and he will be denied access. Now if the attacker modifies the current password with his own password, then the legitimate user will be denied access in his future login attempts. (3) Generating login requests without password verification at the client side can also lead to denial-of-service attack, since the resources of the server will ultimately get blocked in verifying the received login requests. To resist denial-of-service attack in the first case, the authentication protocols without verification table at the server was designed. In the second and third case, denial-of-service can be resisted by verifying the password before permitting password update and before generating a login request.

Mutual authentication protocols facilitating session key agreement should satisfy the following properties (Misbahuddin 2010):

Key Confirmation: Key confirmation is the property whereby communicating entities are assured that both possess the same secrey key (Boyd and Mathuria 2013).

Known-Key Security (Tsai 2008) (Boyd et al. 2013): Known-key security is the property which assures that a compromise of the past session key will not enable an adversary to create a new session key. This property is satisfied when the keys generated in different sessions are computationally independent of each other.

Forward Secrecy (Li et al. 2013, Boyd et al. 2013): Forward secrecy property means that even if the master secret key 'x' is compromised then the attacker will not be able to derive the previous session keys. This property is indicative of the fact that knowledge of the master secret alone is not sufficient enough to derive the session key.

2.4.6 Scyther – An Automated Tool for Protocol Verification

A communication protocol is a set of rules and every protocol should follow the defined conventions to establish semantically correct communications between the participating entities. A regular communication protocol in which defined cryptographic mechanisms are used to secure the message exchanged is referred to as a security protocol. The mechanisms such as hashing, symmetric encryption and asymmetric encryption are used to achieve various cryptographic properties such as Confidentiality, Integrity, Authenticity, Non-Repudiation etc (stallings 2006). However, using cryptographic primitives alone will not guarantee secure operation of the protocol and resistance to attacks. Many accepted and published protocols reported to be safe were later identified to have security flaws (Needham and Schroeder 1978), (Denning and Sacco 1981) (Dalal et al. 2010) and this is due to the fact that manual analysis of such protocols are extremely difficult (Cremers 2008). There was a need to do rigorous verification of the communication protocols using mechanisms relevant to the domain and this has initiated research on formal logic for designing and formal analysis of security protocols. Though many approaches were proposed for designing security protocols, there are no effective approaches for constructing flawless and efficient protocols. The research in the area of formal logic that will help to prove that a protocol is correct and secure is still going on. Currently there are very few automated tools for verifying security protocols which includes Scyther (Cremers 2008), ProVerif (Blanchet 2001), AVISPA (Arnado 2005), Athena (Song et al. 2001). These tools differ in their input language, the way they verify the protocols and deliver the output. Among these, Scyther which is an open source tool offers a GUI, many novel features and takes only a fraction of seconds to verify a small protocol. Also research studies reveals (Cremers and Lafourcade 2007) that, Scyther offers better performance than AVISPA (Arnado 2005) and is similar in performance to ProVerif (Blanchet 2001). Considering the performance and popularity of Scyther, the research uses Scyther tool to do the formal analysis of the protocols discussed in chapters 3,4 and 5.

An overview of features of Scyther and assumptions made to verify the Security of a protocol while using Scyther is included in this section. The assumptions, justifications and explanations in this section are referred from (Cremers and Casimier 2006).

Having a security protocol about whose security we are assured of is not sufficient enough to accept that protocol as a flawless one. Instead, we want some guarantees and supporting facts about its security and the major objective of Scyther is to verify the protocol and give the guarantee. This requires both the protocol as well as network to be created based on a mathematical model and the network is assumed to be controlled by an atttcker. Dolev-Yao (Dolev and Yao 1983) introduced an idealized abstraction of cryptographic primitives known as Dolev-Yao model, to address the need to reason about the security of protocols. First, it is assumed that cryptography is perfect which means that, a message cannot be cracked by anobody other than the right owner of the key. Second assumption is that the intruder can either understand the complete message or he understands nothing. Third assumption is that the network is fully controlled by the adversary who can read, modify, delete and re-route transmitted messages and inject his own messages.

A security protocol comprises of a number of dynamic behaviors and each distinct behavior of a protocol is called a ‘role’ depicted as a sequence of events. For example, we have roles such as initiator & responder, client & server, sender & receiver in a protocol. We have a number of communicating entities called ‘agents’ in a communication system and a system executes the roles performed by agents. The execution of a role by an agent to achieve a secure exchange of message is called a run. Agents try to achieve security while adversary tries to act against them by compromising security.

A security protocol model can be described using the following components:

Protocol Specification: The protocol specification uses formal language based on abstract syntax followed by the security protocol description language to describe the behavior of role in a protocol. The specification of a protocol encompasses initial knowledge for role execution, declaration of constants, variables, functions, macros, nonces used in challenge -response mechanisms and session keys used for securing future communications.

Agent Model: Agents are communicating entities executing roles in the protocol. The agent model of Scyther assumes that honest agents behave as expected and described in the protocol specification and does not leak any important information unless explicitly specified. Under normal circumstances, agents execute the role description in a sequential manner. Thus an agent who has send a message waits for the corresponding receive event until it receives an anticipated message. This implies that an agent matches received message with the format of an expected message and ignores unanticipated message.

Threat Model: A network which is completely under the control of the adversary is to be created. This requires creating a network model in which the attacker can intercept a message, modify transmitted messages, inject messages constructed from its initial knowledge and can compromise any number of agents and learn their secret keys.

Cryptographic primitives: In this relevant properties of mathematical constructs such as encryption, decryption, hashing and other functions are modelled. Cryptographic primitives are modeled by adopting the black-box approach which assumes that without knowing the secret key, a plain text cannot be retrieved from the cipher text.

Security Requirements: Specifies the objectives of a security protocol such as maintaining the secrecy of exchanged messages, maintaining the order of the messages and the values transmitted in the messages as described by the protocol.

Scyther requires security protocols to be described using a specification language called security protocol description language and scyther files are saved using ".spdl" extension. Role terms used in the protocol specification are constructed using the following basic sets:

Var: denotes the variables used to store received messages.

Const: the fresh constants such as nonce and key values which are unique to each instance of a role and treated as local values.

Role: denotes the roles performed by various agents.

Func: denotes the function names used in the protocol description

Scyther tool provides a command line interface (CLI) which uses Python as the scripting language. The tool also provides a graphical user interface (GUI) which makes it easier to understand and verify the protocol.

Security properties to be verified are modeled as claim events in Scyther. An attack graph will be generated whenever an attack is found corresponding to a particular claim. The verification of the protocol can be done for a bounded or an unbounded number of sessions. Scyther supports verifying the security of the protocol against multiple attacks as opposed to the verification against a single attack supported by other similar tools. The tool can be used to verify user defined claims and automatic claims generated by Scyther.

CONCLUDING REMARKS

Though there are various two factor authentication schemes proposed using hash functions for cloud environment, every scheme is found to have some limitation in terms of desirable security features. Moreover, none of the schemes provide perfect security and is thus susceptible to various attacks. Also the proposed two-factor authentication schemes using smart cards for cloud environment, requires the User to directly register at the service providing servers who will then issue the smart card which serves as an authentication factor. Hence in a scenario where the User needs to access multiple cloud services, the User should undergo multiple registration processes, maintain multiple accounts and remember multiple identities. These limitations of the currently available two-factor authentication schemes, viz. susceptibility to attacks, need for multiple registration, maintaining multiple accounts and carrying different authentication tokens (smart cards/crypto-tokens) when accessing the services of multiple service providers etc. are being addressed by the proposed research work. The research proposes a hash function based, two-factor authentication scheme using crypto-tokens. Since procurement/issue of crypto-token involves cost and since the token needs to be carried around by the User, from a User friendly perspective, the research also proposes a hash function based authentication scheme using mobile-token. The proposed schemes use nonce values to resist replay attacks and does not require the server to maintain a verifier table.

Based on the authentication requirements, cloud service providers can be categorized into two. Category one includes those service providers dealing with highly sensitive data and working in a controlled environment such as those providing health-care services and financial services. These service providers need a strong, Two-Factor user authentication mechanism without any additional functionality such as Single sign-on and would prefer to directly authenticate users of its services. The second category of service providers are those dealing with secure data while working in a collaborative environment whose services are accessed by the users simultaneously during the same session, with other services. Category two providers need a strong, Two-Factor authentication mechanism that also provides the users with a Single sign-on functionality. These service providers would prefer to delegate the authentication of users of its services to a trusted third party. In the related literature, the researcher was not able to identify an authentication architecture that caters to the requirements of both the categories of service providers.

These gaps are addressed in this research which moves forward with the following objectives:

- Proposing an authentication architecture and different Two-Factor authentication protocols that uses password as the first factor and Cryptoken/Mobile-Token as the second authentication factor, which can be adapted by service providers who prefer to directly authenticate users of its services. Users to avail the services of service providers, should register at a centralized registration authority (Identity Provider), who will issue the second authentication factor viz. Crypto-Token/ Mobile-Token. While accessing the services of different service provider's, a user should authenticate individually to each service provider, using password and the second authentication factor (Crypto-token or Mobile-Token).

- Proposing an authentication architecture and different Two-Factor authentication protocols that uses password as the first factor and Cryptoken/Mobile-Token as the second authentication factor, which can be adapted by service providers who require Single Sign-on functionality and would prefer to delegate the authentication of users of its services to a trusted third party. In the proposed architecture, users can do a single registration at a centralized registration authority (Identity Provider) and be issued with a single authentication factor. To access the services of different service providers during a session, user's need to authenticate only once at the Identity Provider, using password and the second authentication factor (Crypto-token or Mobile-Token).

- An authentication framework which includes an integrated authentication architecture and a two-factor authentication protocol that facilitates the Users with the convenience of single registration and of accessing different cloud services using a single authentication token (either a Crypto-Token or a Mobile-Token). The framework comprises of

Users (of cloud services), category one service providers who prefer to directly authenticate its users using a strong Two-Factor authentication protocol, category two service providers who need a strong Two-Factor authentication protocol and require Single sign-on functionality which is achieved by delegating the authentication to an authentication broker. The service providers who are part of the framework have the flexibility to choose between direct and brokered authentication. The proposed authentication protocols using hash functions, does not require the server to maintain verifier tables.

CHAPTER 3

DIRECT AUTHENTICATION SCHEME WITHOUT VERIFIER TABLE

Verifying the identity of remote users is a necessary pre-requisite in a cloud environment before being allowed access to secure resources/services/applications. The simplest and most commonly used user authentication mechanism is password based authentication. With the proliferation of Internet enabled services, users have to manage a growing number of logins/passwords which represent their identity across different service providers (SP’s). Since management of multiple identities is cumbersome Users tend to choose low entropy, easy to remember, passwords, rendering the authentication system susceptible to various attacks. Warren (2006) in his work on passwords, observes that a responsible User, is expected to securely manage his password and update the same on a regular basis and to utilize different passwords for each new service he registers. However, this does not happen as a vast majority of users for ease of remembering often write passwords on paper, store in mobile phones, or in the worst case use the same password for multiple services. Also, Hart (2009) said that “archaic static password, one tier login “constitutes one of the biggest security risks and is not enough for cloud services. Therefore, password authentication alone is not sufficient for a secure authentication to cloud services and many works on cloud security recommends adopting a Two-factor authentication (Abraham 2009, Fernandes et al. 2014) mechanism. Subashini and Kavitha (2011) in their work has mentioned that, many a times user credentials are stored at the service providers’ databases and authentication schemes with verification table are vulnerable to various attacks (Tsai 2008).

Service providers, offering their services from a cloud environment can be broadly categorized into two, from the prespective of their authentication requirements. A set of service providers dealing with highly sensitive information and working in a controlled and regulated environment, such as those providing services for health care sector, can be grouped into one category. These service providers require a strong authentication mechanism to authenticate its Users. However, they do not require any additional functionality such as Single sign-on. Similarly, there are another category of service providers that deal with secure information but operate in a collaborative environment. Service providers whose applications are bundled through a web portal, SaaS services such as Aceproject for project management (Fenton 2011) and Assembla for code management (Media 2012) which are simultaneously accessed by organisations during a session for collaborative project management etc. can be grouped under the second category. If each of these providers has its own independent user management mechanism, then the users will have to go through multiple registrations and maintain multiple accounts. To provide the users with a seamless authentication experience, the second category of service providers prefer to have a Single Sign-on functionality by which the users can authenticate to one of the service provider and can access multiple services without re-entering the credentials during the same session. Chapter 3 and chapter 4 proposes authentication architectures and authentication protocols to cater to the requirement of the category one service providers and category two service providers respectively.

Two-factor authentication technology which requires the user to provide more than one authentication information, seeks to decrease the probability that the requestioner is presenting false evidence of its identity. The adoption of two-factor mechanisms makes it more difficult for attackers to bypass the entity authentication of cloud systems, because even if attackers could guess a customer's password correctly, they still need to acquire the specific second piece of information for authentication. Unfortunately, if different service providers set up their own two-factor authentication services, users may have to experience painful registration process repeatedly. In addition, users accessing multiple cloud services may be required to hold multiple authentication tokens associated with various service providers.

Taking cognizance of the aforesaid issues related to authentication in cloud environment, in chapter 3 and chapter 4 we propose authentication schemes where in users authenticate to different cloud services (Service Providers) using a single password and a second authentication factor which can be a crypto-token or a mobile-token. The proposed approach takes away the requirement of the user to remember multiple identities, carry multiple devices to access various services and provides the benefit of a higher level of security in the form of a second authentication factor. The scheme also addresses the issues such as stolen-verifier attack, insider attack, denial-of-service attack etc. by eliminating the requirement to maintain a verification table at the server.

Access to different services across diverse service providers using a single authentication token requires the interoperability between the providers which is now the limiting factor for deployment of such strong authentication solutions. As a result, the research intends to propose the utilization of the 2-factor authentication scheme using password as the first factor and Crypto-token/Mobile-Token as the second factor within a specific environment. The environment includes a trusted entity called an Identity Provider (IdP) with whom the users and the Service providers will be registered. The Identity Provider entity is responsible for providing and managing the second authentication factor concept to end users. The IdP is the central authority responsible for registering the users, issuing the authentication factor and distributing their profile information to the service providers. After obtaining the authentication factor from the IdP, the User who wants to access the service can be authenticated either directly by the service provider or by an authentication broker (Identity provider) to whom the users are re-directed to by the service provider, for achieving Single Sign-on functionality. The authentication factor issued by the identity provider along with the password can be used to authenticate to Service Providers. In this system, mutual association between Service Providers and the Identity Provider is necessary.

This chapter discusses the scheme proposed for direct authentication by service providers and chapter 4 discusses the authentication scheme proposed for brokered authentication by an authentication broker. Authentication protocols discussed in sections 3.1.3, 3.1.4 of current chapter, in sections 4.1.3, 4.1.4 of chapter 4 and in section 5.3 of chapter 5 are based on hash functions and xor-operations as these simple operations reduces the computational load of the authentication system (Das et al. 2004). The proposed authentication protocols do not require the server to maintain a verifier table.

3.1 DIRECT AUTHENTICATION SCHEME

As discussed in the previous paragraph, within the proposed model, there exists an Identity Provider which is responsible for registering a user, issuing him with the authentication factor and providing the profile information of the user to different Service Providers registered with the Identity Provider. The major advantage of this approach is that the Service Providers are relieved of the burden of issuing the authentication tokens and can concentrate on their core functionality of providing services. Also the users need not go through the pain of carrying multiple devices to access multiple services.

The proposed direct authentication solution is possible by the use of a Crypto-Token/ Mobile-Token as the second authentication factor. As illustrated in Figure 3.1, this factor will have, the user’s authentication parameters saved by Identity Provider/downloaded remotely from the Identity Provider and stored securely within it. Once identities are downloaded, the user can provide them to the Service Provider, totally independent of the Identity Provider. The Service Provider will exchange the two-factor authentication protocol with the user and based on the result will allow or deny access to the requested resource.

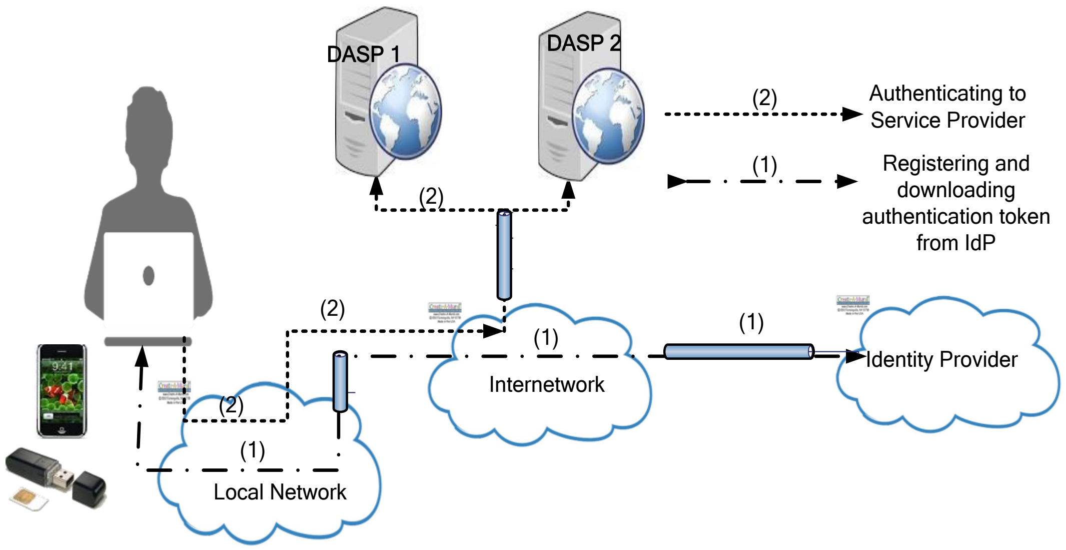

DASP 1
DASP 2
(2)
Authenticating to Service Provider
(1)
Registering and downloading authentication token from IdP
(2)
(2)
(1)
(1)
Identity Provider
Internetwork
(2)
(1)
Local Network

Figure 3. 1 Crypto-Token/Mobile-Token Deployment and Direct Authentication at SP

3.1.1 Identity Provider and Service Providers Association

The proposed scheme considers that the association between Service Providers and Identity Providers takes place in an integrated trust based environment. The participating entities in such an environment exchange information about resources and users by using a common set of practices and policies.

Association: The Service Providers need to register with the registration server of the IdP by providing a unique server ID, Service Provider URL, a short description of the service provided, and the preferred mode of authentication as “Direct Authentication”. At the end of the registration process, The IdP issues an authentication module containing the proposed Two-Factor authentication protocol to the Service Provider (SP) which can be integrated with the authentication engine of the SP. Also a secret key of the IdP is communicated in a secure manner to the Service Provider (SP). This is later used by the SP to verify an authentication parameter during the authentication of the user by using the proposed 2-factor authentication protocol.

Trust: In the proposed direct authentication model, trust should be established between the communicating entities such as users, service providers and the Identity Provider to accept and process communications from each other. As stated in the trust model guidelines of the OASIS consortium (Linn, 2004), authentication and business agreements (BA) are the two criteria to be followed for establishing trust. The proposed two-factor direct authentication solution has adopted the "Pairwise/Direct" model for the Service providers and the Identity Provider, where Service Providers and Identity Providers have a BA and authentication of each other is done by using Digital Certificates and PKI technology (Stienne et al. 2013). Service providers enter into an agreement with the Identity Provider by undergoing a registration process and they will exchange their own digital certificates in order to establish trust for future communications.

3.1.2 Proposed Direct Authentication Architecture

The proposed architecture for a Cloud environment includes four participants' viz. a Registration Server (RS), an Authentication Server (AS), Service Provider's (SP's) and users'. The RS and AS are in the same trusted domain and together they provide the functionality of the Identity Provider (IdP).

The user's and SP's comprising the proposed architecture needs to register with the registration server of the IdP. When a SP registers with the IdP, he submits his identity information and the details of the services provided. The cloud service provider's (CSP's) and IdP work in a trust based environment.

In this two-factor authentication scheme, user's password and a registered crypto-token serve as the authentication factors. When a user wants to get the service of a CSP, he is re-directed to the IdP by the SP if he is not a registered user in which case his profile information will not be available with the SP. In such a scenario, the user needs to do a single registration at IdP as illustrated in Figure 3.2, by providing the User-ID and Password. On successful registration, IdP provides the user with a Crypto-token / Mobile-Token containing the security parameters. The server Id's of all the participating service providing servers and the details of their services are also communicated to the user via an e-mail. The login and authentication phase of the proposed scheme runs independently on each SP and the service providers directly authenticate the users requesting their services as is illustrated in Figure 3.3. A user who wants to access the services of a particular SP, tries to login to the provider's web page by submitting his ID and PW. The authentication module within each SP executes the proposed protocol for mutual authentication, before providing the requested service. The second authentication factor of the proposed protocols contains only a few hashed values generated from user's ID, password and the secret key of the server. It does not contain any digital signature which is generated by encrypting the hash of a value by the sender's private key. This requires the implementation of public key infrastructure (PKI). The proposed protocols do not require the support of PKI.

The protocols do not require the server to maintain a password verification table. The registration and authentication process flow is illustrated in Figure 3.4.

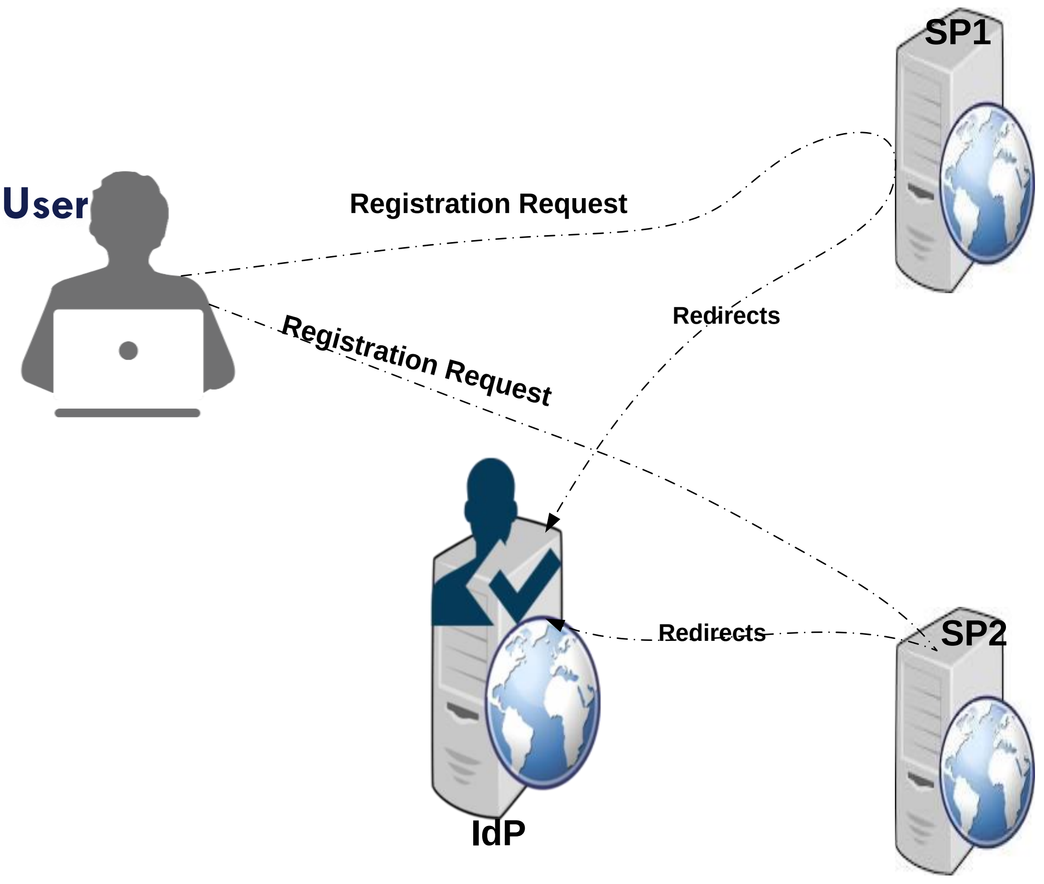


Figure 3. 2 Direct Authentication **-** Registration Redirect

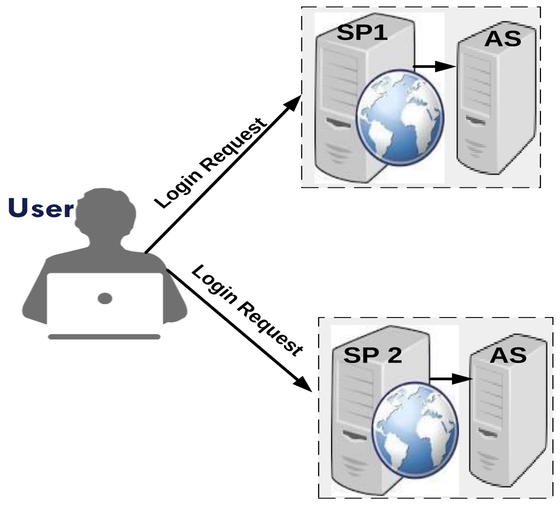


Figure 3.3 Direct Authentication – Login and Authenticate to each SP

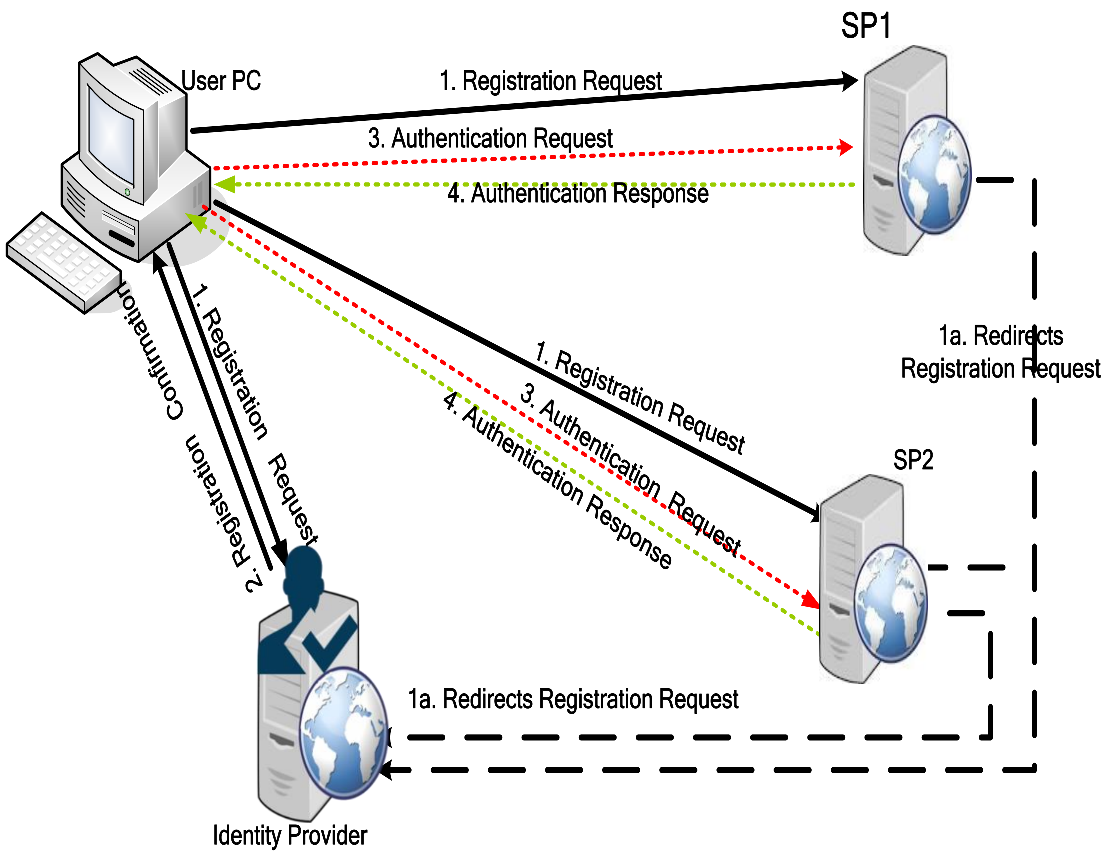

SP1
User PC
1. Registration Request
3. Authentication Request
4. Authentication Response
1a. Redirects Registration Request
1. Registration Request
3. Authentication Request
4. Authentication Response
SP2
1. Registration Request
2. Registration Confirmation
1a. Redirects Registration Request
Identity Provider

Figure 3.4 Registration and Authentication Process Flow

3.1.3 Crypto-Token Based Direct Authentication Protocol without Verifier Table

In the crypto-token based authentication protocols discussed in sections 3.1.3 ,4.1.3 and 5.3, we are ensuring the security of user's password communicated to server by adopting concepts of cyclic groups and by taking advantage of the fact that discrete logarithm problem is notoriously hard to solve in many groups. Hence this section includes a brief explanation of cyclic groups and Discrete Logarithm Problem (DLP). A group (G,.) is a set G together with a binary operation "." (multiplication or addition), which satisfies the following properties (Beachy and Blair 2005):

(a) The group G is closed under the binary operation(b)The binary operation is associative (c)The group G has an identity element and (d)Every element of group G has an inverse element.

Any group G is said to be cyclic if there exists an element $a \in G$ such that the element $b \in G$ can be written as $b = a^x$ for some $x \in Z$, where Z is the set of integers. Here 'a' is called the generator of G and we denote this as a = <G>. It is known that all multiplicative groups $G = Z_p*$ where p is a prime number is cyclic (Damgard and Nielsen 2012). Now if the integer x and the generator a is given, then the power a^x can be easily calculated by the square and multiply method (Gao 1999). The inverse problem, that is given a group G, it's generator a and element $b \in G$, finding integer x such that $a^x = b$, is the discrete logarithm problem and it is considered to be hard (Gao 1999).If $a^x = b$, then the discrete logarithm of b base a is x, and it is represented as $DL_a(b) = x$.

Example: Consider the group $G = Z_p^*$ where p is a prime number and p = 1999 (Brawley and Gao 1999). This group is cyclic under multiplication modulo p.

$G = Z_p^* = \{1,2, 3, 4, \ldots\ldots\ldots\ldots\ldots.., p\text{-}1\}$

Now the element a = 3 is a generator of G and is also known as the primitive element modulo p.

Then $G = \{a^0, a^1, a^2, a^3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, a^{p-2}\}$ mod p.

Ie. $G = \{a^0, a^1, a^2, a^3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, a^{1997}\}$ mod p.

Now it is easy to calculate 3^{789} mod 1999 = 1452. Ie. $3^{789} \equiv 1452$ mod 1999. On the other hand, it is difficult to determine that x = 789, given only that x takes a value between 0 and 1997 and satisfies the equation $3^x \equiv 1452$ mod 1999.

The discrete logarithm problem (DLP) is considered to be notoriously hard in many groups such as in Z_p^* where p is a large prime number (Damgard and Nielsen 2012).

Phases of the Proposed Protocol: The proposed protocol consists of four phases viz., Registration, Login, Mutual Authentication & Key Agreement Phase and the Password change phase. The notations used are listed in Table 3.1.

Table 3. 1 Notations Used in the Protocol (Direct&Crypto-Token)

IdP, SP	Identity Provider, Service Provider in the cloud
U_i, S_j, SID_j	i th User, j th SP, ID of the jth SP
ID_i, PW_i, g_0, p	Unique Identification of U_i, password of U_i, generator of cyclic group, Prime Number Chosen by U_i.
S	Secret key of server of IdP shared with service providers
N_i, N_j	Nonce values chosen by Crypto-token and server respectively
$h(.)$, $\oplus$, $\|\|$	One-way hash function, XOR operation, Concatenation Operation

Registration Phase

Registration Phase illustrated in Figure 3.5 can be explained as follows:

R1: U_i generates a cyclic group of prime order p and selects a generator g_0.

R2: U_i selects his identity ID_i and Password PW_i. Computes $b = h(PW_i)$,

$k = g_0^{\ b} \mod p$.

R3: U_i submits $h(ID_i)$, $h(ID_i||k)$ to IdP through a secure channel. IdP checks the availability of h(IDi). Otherwise U_i is prompted to select a new ID_i.

R4: Upon receiving $h(ID_i)$, $h(ID_i || k)$, IdP computes

$V_i = h(ID_i || k) \oplus h(ID_i)$;

$K_i = h(IDi) \oplus h(h(ID_i) \| h(S)); M_i = h(ID_i\|k) \oplus K_i$.

Here h(S) is the hash of the secret key 'S' of the IdP which is shared with all the registered service providers.

IdP sends a registration confirmation message to U_i along with the list of service providers registered under its domain. IdP stores {h(.), V_i, M_i} into crypto-token and sends to the user U_i via a secure channel such as a trusted courier. U_i stores g_0, p into the crypto-token. IdP also updates the service provider's data base with the profile information of registered users. Here the Cloud Service Provider's maintain a database of user profile information such as the unique user identity 'ID_s' = (ID_i) in hashed form ($h(ID_i)$) as one entry along with e-mail ID, firtst-name, last-name, mobile number etc.

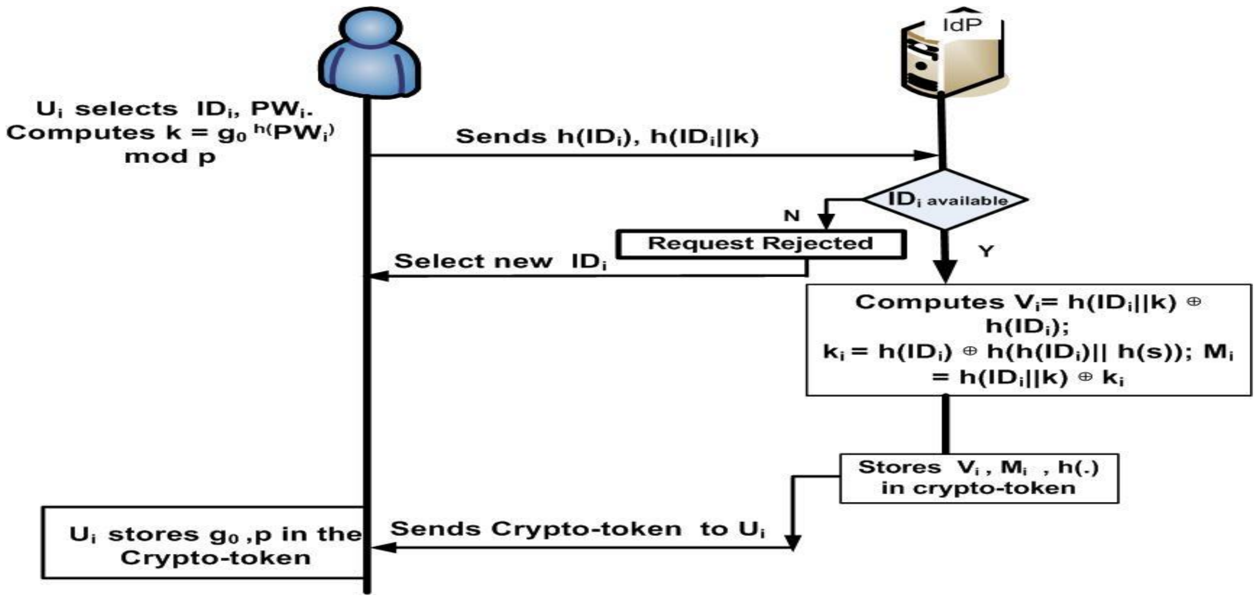

IdP
U_i selects ID_i, PW_i. Computes $k = g_0^{h(PW_i)}$ mod p
Sends $h(ID_i)$, $h(ID_i\|k)$
ID_i available
N
Request Rejected
Select new ID_i
Y
Computes $V_i = h(ID_i\|k) \oplus h(ID_i)$; $k_i = h(ID_i) \oplus h(h(ID_i)\| h(s))$; $M_i = h(ID_i\|k) \oplus k_i$
Stores V_i, M_i, h(.) in crypto-token
Sends Crypto-token to U_i
U_i stores g_0,p in the Crypto-token

Figure 3.5 Registration Phase of Direct Authentication Using Crypto-Token

Login Phase

Login Phase illustrated in Figure 3.6 can be explained as follows:

L1: U_i clicks the URL of Service Provider. In the login page U_i enters his identity ID_i. SP verifies whether $ID_s = h(ID_i)$ exists in his database. If so U_i is prompted to proceed.

L2: U_i inserts his crypto-token into the system. U_i enters the server ID 'SID_j' of the service providing server S_j and his password PW_i.

L3: Crypto-token computes $b = h(PW_i)$, $k = g_0{}^b \mod p$.

L4: Crypto-token computes $V_i' = h(ID_i \| k) \oplus h(ID_i)$ and checks whether it is equal to the V_i stored in the crypto-token. If so crypto-token generates a nonce N_i and computes the challenge $C_1 = h(ID_i) \oplus h(SID_j \| N_i)$.

L5: Crypto-token sends C_1, N_i to S_j.

L6: Sj on receiving C_1, N_i computes $h(ID_i) = C_1 \oplus h(SID_j \| N_i)$ and ensures that he is communicating with a registered user to whom he sent a message to proceed with login. Sj computes the response $C_2 = h(N_i \| 1) \oplus h(N_j)$, and where N_j is the nonce generated by S_j. S_j sends $<C_2 , N_j>$ to Crypto-token.

L7: Crypto-token computes, $T_i = h(N_i \| 1)$, $(h(N_j))' = C_2 \oplus Ti$ and checks whether $h(N_j)' = h(N_j)$. By doing so, Crypto-token ensures the freshness of nonce N_i. It ensures that the response is from the service providing server to whom its challenge was sent.

L8: Crypto-token computes $K_i = M_i \oplus h(IDi \| k)$, $B_i = K_i \oplus h(ID_i)$,

$R_i = h(ID_i \| h(k))$,$P_{ij} = h(B_i \oplus (h(N_j) + 1) \| R_i)$, $L_{ij} = B_i \oplus R_i$

L9: Crypto-token sends <P_{ij}, L_{ij}> to Sj.

Authentication and Key Agreement Phase

A1: On receiving P_{ij}, S_j computes B_i‘= h (h(ID_s) || h(S)).

A2: Sj computes, R_i’ = L_{ij} ⊕ B_i’, P_{ij}’ = h (B_i‘⊕ (h(N_j)’ + 1) || R_i). Server ensures the freshness of the nonce N_j and checks whether P_{ij}’ is equal to the received P_{ij}. If it does not hold, Sj rejects the login request. Otherwise S_j considers U_i as authenticated and sends the response J_i = h(B_i’ || R_i’ || N_i) to U_i .

A3: U_i computes J_i ‘= h(B_i || R_i || N_i) and compares with the received J_i . If equal, U_i successfully authenticates the server. After successful mutual authentication, both client and the server computes the session key as SK_{us} = h(B_i ||SID_j|| h(N_i) || h(N_j)|| R_i) and SK_{su} = h(B_i’ ||SID_j|| h(N_i) || h(N_j)|| R_i’) respectively.

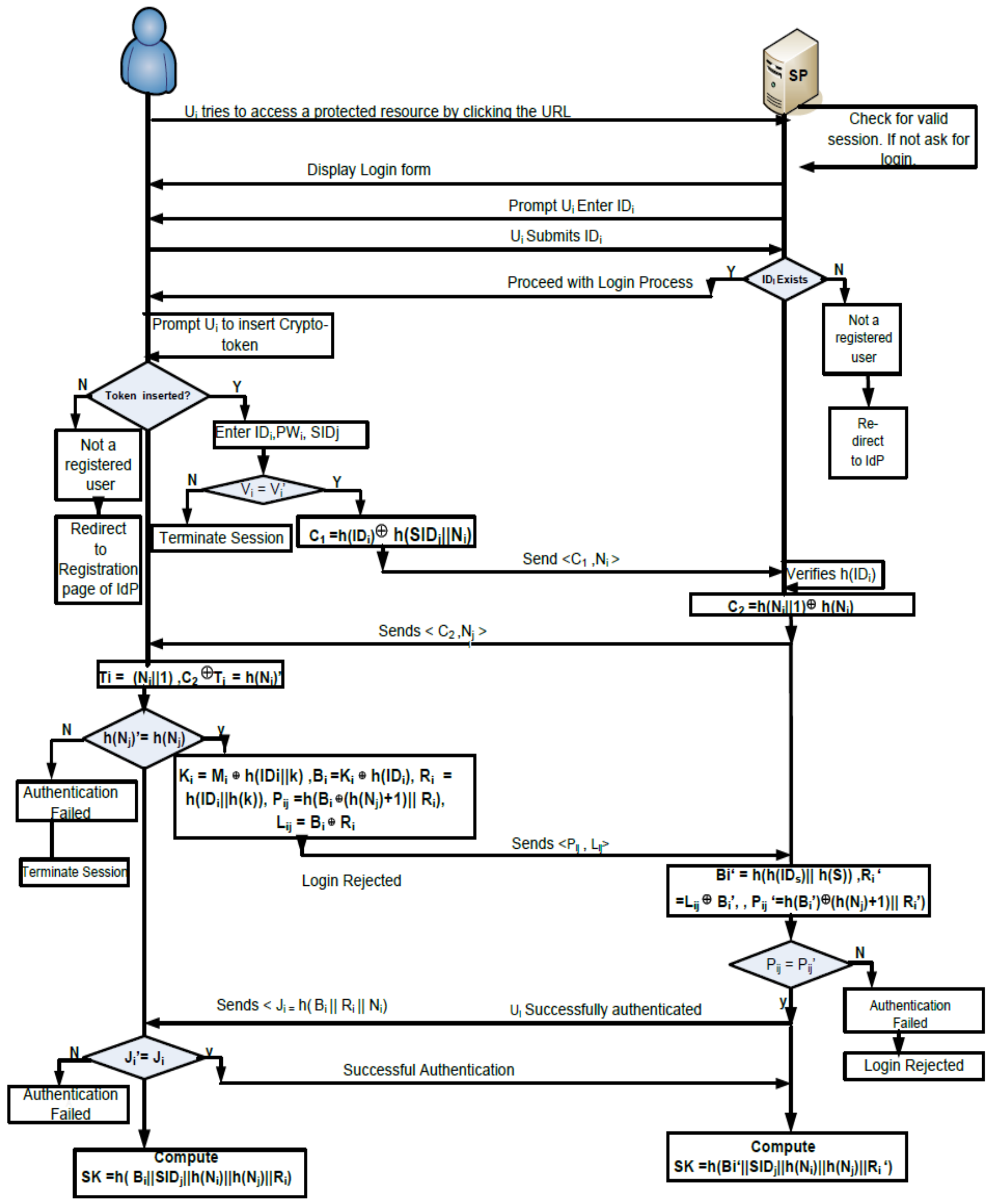

SP
U_i tries to access a protected resource by clicking the URL
Check for valid session. If not ask for login.
Display Login form
Prompt U_i Enter ID_i
U_i Submits ID_i
ID_i Exists
Y
N
Proceed with Login Process
Not a registered user
Re-direct to IdP
Prompt U_i to insert Crypto-token
Token inserted?
N
Y
Not a registered user
Redirect to Registration page of IdP
Enter ID_i,PW_i, SIDj
$V_i = V_i'$
N
Y
Terminate Session
$C_1 = h(ID_i) \oplus h(SID_j || N_i)$
Send $<C_1, N_i>$
Verifies $h(ID_i)$
$C_2 = h(N_j || 1) \oplus h(N_i)$
Sends $< C_2, N_j >$
$Ti = (N_j || 1), C_2 \oplus T_i = h(N_j)'$
$h(N_j)' = h(N_j)$
N
y
Authentication Failed
Terminate Session
$K_i = M_i \oplus h(IDi || k), B_i = K_i \oplus h(ID_i), R_i = h(ID_i || h(k)), P_{ij} = h(B_i \oplus (h(N_j)+1) || R_i), L_{ij} = B_i \oplus R_i$
Sends $<P_{ij}, L_{ij}>$
Login Rejected
$Bi' = h(h(ID_s) || h(S)), R_i' = L_{ij} \oplus B_i', , P_{ij}' = h(B_i') \oplus (h(N_j)+1) || R_i')$
$P_{ij} = P_{ij}'$
N
y
Authentication Failed
Login Rejected
Sends $< J_i = h(B_i || R_i || N_i)$
U_i Successfully authenticated
$J_i' = J_i$
N
y
Authentication Failed
Successful Authentication
Compute
$SK = h(B_i || SID_j || h(N_i) || h(N_j) || R_i)$
Compute
$SK = h(Bi' || SID_j || h(N_i) || h(N_j) || R_i')$

Figure 3.6 Login and Authentication Phase of Direct Authentication Using Crypto-Token

Password Change Phase

Password change Phase illustrated in Figure 3.7 can be explained as follows:

Ui inserts his Crypto-token and enters ID_i, his password PW_i .and requests for a password change.

P1: Crypto-token computes $b = h(PW_i)$, $k = g_0{}^{b} \bmod p$.

P2: Crypto-token computes $V_i{}' = h(ID_i \| k) \oplus h(ID_i)$ and checks whether it is equal to the V_i stored in the Crypto-token. If equal, U_i is asked to enter the new password.

P3: U_i submits PW_{inew}. Crypto-token computes $b_{new} = h(PW_{inew})$,

$k_{new} = g_0{}^{bnew} \bmod p$.

P4: Crypto-token computes $K_i = M_i \oplus h(ID_i \| k)$, $V_{inew} = h(ID_i \| k_{new}) \oplus h(ID_i)$; $M_{inew} = h(ID_i \| k_{new}) \oplus M_i \oplus h(ID_i \| k)$ .Crypto-token replaces V_i with V_{inew} and M_i with M_{inew} in the Crypto-token.

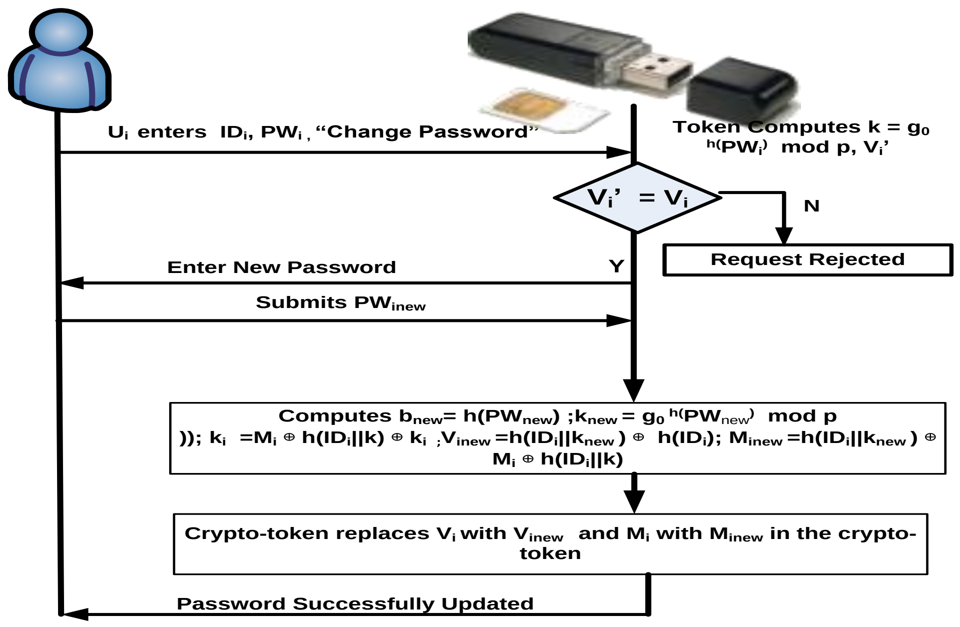

U_i enters ID_i, PW_i, "Change Password"
Token Computes $k = g_0^{h(PW_i)} \bmod p$, V_i'
$V_i' = V_i$
N
Request Rejected
Enter New Password
Y
Submits PW_{inew}
Computes $b_{new} = h(PW_{new})$; $k_{new} = g_0^{h(PW_{new})} \bmod p$)); $k_i = M_i \oplus h(ID_i||k) \oplus k_i$; $V_{inew} = h(ID_i||k_{new}) \oplus h(ID_i)$; $M_{inew} = h(ID_i||k_{new}) \oplus M_i \oplus h(ID_i||k)$
Crypto-token replaces V_i with V_{inew} and M_i with M_{inew} in the crypto-token
Password Successfully Updated

Figure 3.7 Password Change Phase of Direct Authentication Using Crypto-Token

Security Analysis

Security analysis is carried out to analyze the resistance of the protocol to various attacks. The proposed protocol is secure against the following attacks.

i. Security against Replay Attack: A replay attack involves capturing the messages exchanged between a valid user and a server and replaying the same at a later point in time. Time stamps are commonly used to resist replay attacks. However, in a distributed cloud environment, using time stamps might lead to time synchronization problems if the clocks of sender and receiver are not synchronized properly. Hence the proposed scheme uses nonce values to resist replay attacks. To successfully launch a replay attack, an adversary should be able replay a valid login request message $\{P_{ij} = h(B_i \oplus (h(N_j) + 1) \| R_i), L_{ij} = B_i \oplus R_i\}$

send by U_i or the response message $\{J_{i} = h(B_i \| R_i \| N_i)\}$ send by the server, at a later point in time. However, server and U_i verify the freshness of the nonce values before accepting the request and response. Random nonce values used in the proposed scheme viz. N_i and N_j are generated independently and their values are session dependent. Hence attackers cannot gain access to the system by replaying messages previously transmitted by legal users.

ii. **Man-in-the-Middle Attack:** In the proposed protocol, if the adversary modified any of the message exchanged between the client and the server, then the session will be terminated. For example, assume that ID_i is modified into ID_i^* in the message C_1 exchanged during the login phase. The server during the login phase checks whether an ID_s corresponding to the ID_i^* is there in its user table. If it is not there, then the login request will be rejected.

If IDi^* is some other user's ID, then B_i' is calculated as $B_i' = h(h(ID_i^*) \| h(S)) = M_i \oplus h(ID_i^* \| k) \oplus h(IDi)$ where $k = g_0^b \mod p$ corresponds to the password of IDi^*. Also to calculate $< P_{ij}, L_{ij} >$ and the session key, the adversary needs to know the password and the server's secret key. Hence, this attack will fail since the adversary will not be able to impersonate a valid user without knowing his password.

iii. **Security against Stolen Verifier Attack:** In most of the authentication schemes the server stores some verification or password table in its database to verify the legitimacy of the user. Thus the attacker may steal verification information from the server's database and attempt to impersonate valid users. In the proposed scheme, only $h(ID_s)$ and some profile information are stored in the server. Using $h(ID_s)$ alone, the attacker cannot compute values used for authentication and hence the attack will fail.

iv. Security against Server Spoofing Attack: In a server spoofing attack, an unauthorized server tries to masquerade as a valid server and attempts to obtain the credentials of a valid user. Assume that an adversary intercepts C_1, T_i and $<P_{ij,} L_{ij}>$ transmitted between the user and the server during an earlier communication. To spoof the server, the adversary should be able to generate the response $K_i = h(B_i || R_i || N_i)$. To calculate $B_i = h(h(ID_i) || h(S))$, the adversary should have the knowledge of server's secret key, which is unknown to the adversary. Again to calculate R_i, adversary should know B_i. Also, he will not be able to calculate the session key without knowing the values of $B_{i,}$ R_i which are never transmitted across the communication channel during the course of any of session.

v. Security against Guessing Attack: In the proposed scheme the password is never transmitted in the plain text form. Moreover, the password is modified into $k = g_0^{\ b} \mod p$ where $b = h$ (password) before transmitting password information to the IdP. Hence even if the attacker needs to verify the guessed password, he needs to solve the discrete logarithm problem. Also, at the client side, when the user enters a password, it is first verified by the crypto-token. The token will keep an account of the number of failed login attempts and the user will be blocked after three attempts.

vi. Security against Phishing Attack: In this attack, the adversary uses fraudulent means to obtain sensitive information such as password, credit card details etc. by pretending to be a trustworthy entity. In the proposed scheme, before sending the login request the crypto-token and server between steps L4 and L7, ensures that both are communicating with the correct and valid entity by checking the freshness of the nonce. The proposed scheme is thus resistant to phishing attack.

vii. **Security against Crypto-token lost Attack:** If the Adversary steals the Crypto-token, containing the parameters {h(.), V_i, M_i, g_o, p}, he can neither retrieve the user's password nor the IdP's master secret 'S' from the stored values.

viii. **Security against Denial-of-Service Attack: A** denial-of-service attack can be launched by an adversary by creating invalid login request messages and bombarding the server with the same or by modifying the current password in the crypto-token which prevents a valid user from accessing resources, he is authorized to access. This attack can also be launched by an adversary who has got control over the server and is able to modify the user information stored in the server's database which in turn prevents the valid user from accessing the resources. The first scenario will not work in the case of the proposed scheme, since it is impossible for the adversary to create valid login request messages without knowing the correct password. The validity of the password is checked at the client side before creating a login request as well as before allowing a user to modify the current password. The second scenario is also not applicable in the proposed scheme, since the server does not maintain a verifier/password table.

ix. **User Anonymity Preserved:** The user will send the login request P_{ij} to the cloud server S_j in each login session. To trace the user, the adversary will intercept the login message and attempt to extract ID_i from the message. The irreversibility property of one-way hash functions prevents the adversary from extracting ID_i from P_{ij}. More over each login message is made dynamic by including the nonce N_j which is unique for each login session. Therefore, an adversary cannot identify the person making a login attempt and hence the proposed scheme preserves user anonymity.

x. Security of Session Key

- **Known-Key Security:** Known-key security property ensures that a compromise of past session key will not contribute to deriving any further session key. In the proposed scheme, the session key SK is calculated using $B_{i,}$ $h(N_i)$ and $h(N_j)$**,** R_i which are never communicated across a transmission channel. Security properties of hash functions such as collision resistance and irreversibility guarantees that even if the past session key is revealed, the adversary cannot derive B_i, R_i. Furthermore, the nonce values N_i and N_j are session dependent and generated independently by U_i and S_j and they themselves will not be having the knowledge of N_i and N_j that will be generated in the future sessions. Thus the proposed scheme satisfies known-key security property.

- **Forward Secrecy:** Forward secrecy property ensures that even if the attacker manages to obtain the master secret 'S' of the registration authority, it will not result in the compromise of any previous sessions. Suppose that the adversary has the knowledge of 'S'. The adversary cannot compute the value of $B_i = M_i \oplus h(ID_i \| k) \oplus h(IDi)$ without knowing the password of the valid user. Thus he cannot derive the session key $SK = h(B_i \| SID_j \| h(N_i) \| h(N_j) \| R_i)$ which is required to decrypt the messages sent from the client to the server and vice-versa. Also the session key is calculated using the unique nonce values generated independently by the user and the service provider. Hence, even they will not be able to predict the session key values. Also, the nonce values are very large making it difficult for the attacker to guess the values to generate the session keys.

xi. Security against Denial-of-Service Attack: The scheme allows the crypto-token holder to change the password without the intervention of the IdP. The Crypto-token verifies the legitimacy of the user before changing the password to prevent unauthorized users from easily changing the password if they obtain the crypto-token of some other registered user. Thus only valid user who knows the correct ID and password, corresponding to the crypto-token can change the password.

Efficiency Analysis

This section analyzes the efficiency of the proposed scheme in terms of the computational and the communication cost. It is assumed that ID_i, PW_i, g_0, p, nonce values are 128 bits long and the output of hash function (SHA-2) is 256 bits long. Let T_h T_x, T_e and T_c denote the time complexity for hashing, XOR, exponentiation and concatenation operations respectively. In the protocol, the parameters stored in the Crypto-token are V_i, M_i, g_0, p and the memory (E1) needed in the crypto-token is 768 (2*256 +2 *128) bits. Communication cost of Login, Authentication& Key agreement (E2) includes the capacity of transmitting parameters (C_1, N_i, C_2, N_j, P_{ij}, L_{ij}) which makes E2 equal to (4*256 + 2*128) = 1280 bits. The computation cost of user registration (E3) is the total time of all operations executed in this phase by the user and Registration authority and is equal to $4T_h + 3T_x + 1T_e + 2T_c$. The computation cost of the user

(E4) and the server (E5) authentication is the total time of all operations executed by the crypto-token and Server during login, authentication and key agreement phase. During login & authentication, the crypto-token performs 9 hash functions, 6 XOR, 1 exponentiation and 9 Concatenation making E4 equal to $9T_h + 6T_x + 1T_e + 9T_c$. Similarly, E5 is $7T_h + 3T_x + 9T_c$. The computation cost of password changing (E6) is the total time of all operations executed in this phase by the user and is equal to $5T_h + 4T_x + 2T_c + 2T_e$. Comparisons with other protocols are shown in Table 3.2. Comparison results reveals that the computational efficiency of the proposed protocol for direct authentication using Crypto-token is comparable with similar other two-factor authentication protocols. In the case of the proposed crypto-token based protocol, the security of password send to the server during registration, is enhanced by obfuscating the password, by exponentiating the password to the power of the generator of a cyclic group. In this protocol, the research is exploiting the difficulty in solving discrete logarithm problem for cyclic groups of the form Z_n where 'n' is a very large odd prime number. Though these computations increase the computation cost of the protocol and affects total computational time, the protocol aids in providing enhanced security. In such a scenario, it can be mentioned in the Service Level Agreement between the IdP and the Service Providers that the authentication protocol provided by the IdP, provides secure authentication of users that requires a certain time period for execution. The authentication protocol can be adopted by those service providers to whom the time duration for execution of authentication protocol is agreeable.

Computations done during the password change by Rui Jiang's protocol is much more compared to the proposed protocol, as in Rui Jiang's protocol, the entire steps in authentication phase is executed before the password is changed by the server and user.

Table 3.2 Comparison of Computational Efficiency with Other Protocols

	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Crypto-token based Protocol	768 bits	1280 bits	$4T_h + 3T_x + 1T_e + 2T_c$	$9T_h + 6T_x + 1T_e + 9T_c$	$7T_h + 3T_x + 9T_c$.	$5T_h + 4T_x + 2T_e + 2T_c$.
Choudhary et al. [2011]	1024 bits	1920 bits	$6T_h + 3T_x + 1T_e + 2T_c$	$10T_h + 2T_x + 1T_e + 3T_c$	$8T_h + 1T_x + 1T_e + 3T_c$.	$4T_h + 4T_x$
Jaidhar [2013]	1024 bits	1664 bits	$5T_h + 5T_x + 1T_e + 5T_c$	$6T_h + 2T_x + 9T_c + 2T_s + 1T_d$	$5T_h + 1T_x + 8T_c + 2T_d + 1T_s + 1T_e$	$3T_h + 2T_x + 3T_c$
Rui Jiang [2013]	768 bits	1152 bits	$4T_h + 1T_x + 1T_e + 1T_c$	$7T_h + 1T_x + 4T_c + 1T_d + 1T_e$	$7T_h + 5T_c + 1T_s + 1T_e$	$18T_h + 3T_x + 11T_c + 2T_s + 1T_d + 4T_e$

Scyther Analysis

Scyther tool requires the optimal parameters and the protocol description to be given as input. The tool analyses the input and outputs a summary report and displays a graph for each attack.

The strength of the protocol is verified using Scyther tool which ascertains the strength by evaluating the resistance of the protocol to various attacks. Scyther uses strand space model for formalizing logic and uses Dolev-Yao model for modelling the network, which caters to the requirement of a mathematical approach for validating the protocol.

The description of a protocol is written in Security Protocol Description language (SPDL) (Cremers, 2008) and the analysis results of login phase are shown in Figure 3.8. Specification of a security protocol describes the communicating entities, events describing the protocol and order of execution of events, and initial knowledge required for communication parties such as constants used in the protocol. Events include sending and receiving of messages and security claims. $\text{Send}_{\text{Label}}$(I,R,p) corresponding to role I, denote I sending message p to R and $\text{Recv}_{\text{Label}}$(I,R,p) denotes R receive p sent by I. Labels are used to mark corresponding send and receive events. Claim events describe Security properties. "Claim(Principal, Claim,Parameter)," where Principal is the user's name, Claim is a security Property, and Parameter is the term for which the security property is checked. The proposed protocol can be modeled using SPDL as follows:

```
// Login, Authentication and Key Agreement Phase  of Protocol for Direct Authentication
const  exp: Function; const hash: Function; hashfunction h; const XOR: Function;
const h1:Function; const plus:Function;const mod:Function;
protocol Directauthlogin(I,R){
role I {
const IDi, Pi,, Bi, SIDj,k,s,b,g,p;
fresh N1: Nonce;
var N2: Nonce;
macro b = h(b);
macro k =  mod(exp(g,b),p); macro Pij =  h(XOR(XOR(h(IDi), h(h(IDi),
h(s))), plus(h(N2),1)),h(IDi,k)); macro Lij = XOR(h(N2) , h(IDi,k));
send_1(I,R, XOR(h(SIDj,N1), h(IDi)),N1);//C1
recv_2(R,I, XOR(h(N2), h(N1,1)));//C2
send_3(I,R, h(XOR(XOR(h(IDi), h(h(IDi), h(s))),
plus(h(N2),1)),h(IDi,k)));//Pij
send_4(I,R,Lij);
claim_i1(I, Secret, XOR(h(SIDj,N1), h(IDi)));//C1
claim_i2(I,Secret,XOR(h(N2), h(N1,1)));//C2
claim_i3(I,Secret,h(XOR(XOR(h(IDi), h(h(IDi), h(s))),
plus(h(N2),1)),h(IDi,k)));//Pij
claim_i4(I, Secret, h(s)); claim_i10(I,Secret,k); claim_i11(I,Secret,h(IDi));
claim_i12(I, Secret,N1); claim_i5(I, Secret, h(N2));
claim_i13(I,Secret,Lij);
claim_i6(I, Niagree); claim_i7(I,Nisynch);claim_i8(I,
Alive);claim_i9(I,Weakagree);
claim_i10(I, Commit, R,N1,N2);
}
role R{
const IDi,Pi,N2,Bi, SIDj,k,s,b,g,p;
var N1:Nonce; fresh N2: Nonce;
recv_1(I,R, XOR(h(SIDj,N1), h(IDi)),N1);//C1
send_2(R,I, XOR(h(N2), h(N1,1)));//C2
recv_3(I,R, h(XOR(XOR(h(IDi), h(h(IDi), h(s))),
plus(h(N2),1)),h(IDi,k)));//Pij
recv_4(I,R,Lij);
claim_r13(R,Secret,Lij); //Lij
claim_r1(R, Secret, XOR(h(SIDj,N1), h(IDi)));//C1
claim_r2(R, Secret, XOR(h(N2), h(N1,1)));//C2
claim_r2(R, Secret, h(XOR(XOR(h(IDi), h(h(IDi), h(s))),
plus(h(N2),1)),h(IDi,k)));//Pij
```

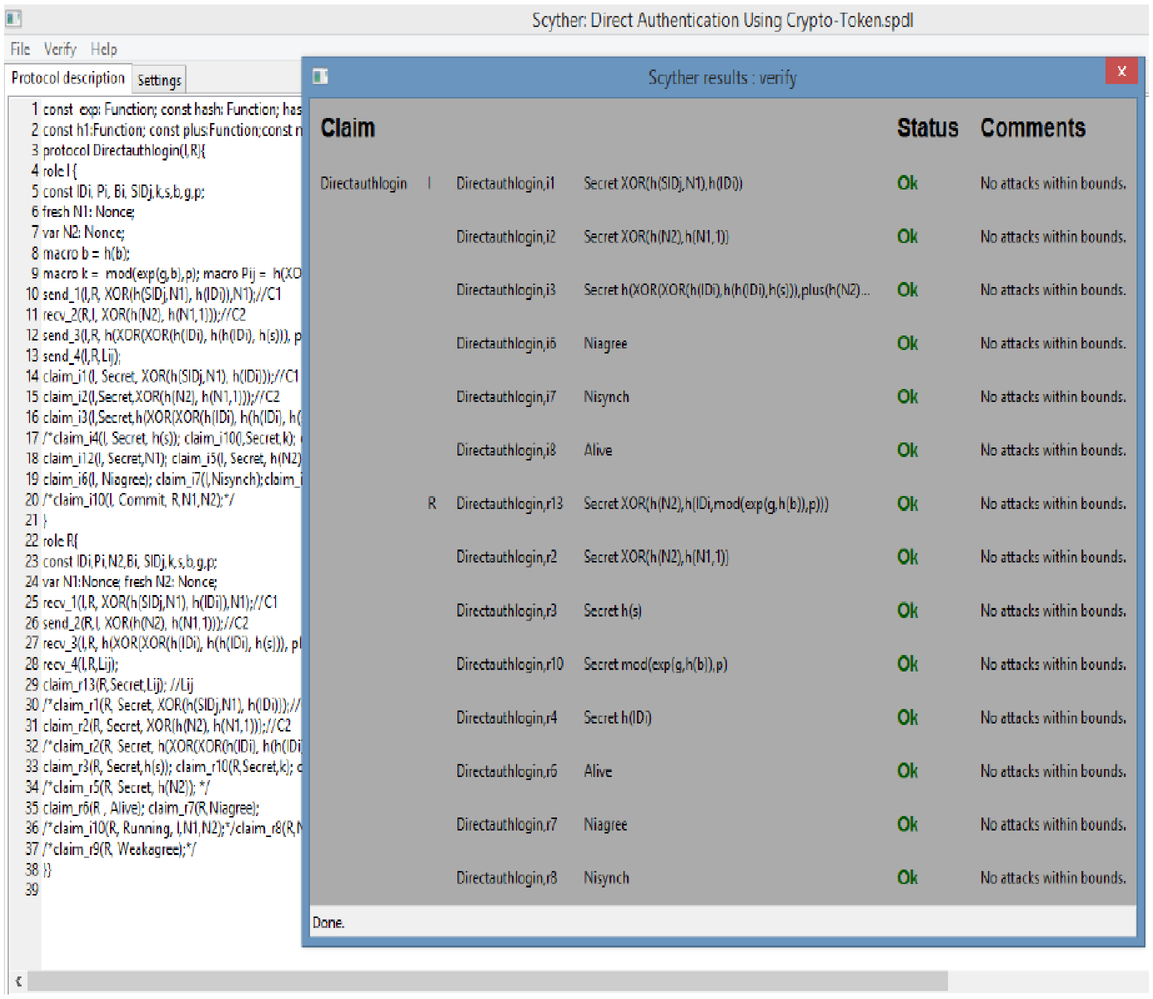
claim_r3(R, Secret,h(s)); claim_r10(R,Secret,k); claim_r4(R,Secret,h(IDi));

claim_r5(R, Secret, h(N2)); claim_r6(R , Alive); claim_r7(R,Niagree);

claim_i10(R, Running, I,N1,N2);claim_r8(R,Nisynch);

claim_r9(R, Weakagree);

}}



```
const  exp: Function; const hash: Function; has
const h1:Function; const plus:Function;const n
protocol Directauthlogin(I,R){
role I {
const IDi, Pi, Bi, SIDj,k,s,b,g,p;
fresh N1: Nonce;
var N2: Nonce;
macro b = h(b);
macro k =  mod(exp(g,b),p); macro Pij =  h(XO
send_1(I,R, XOR(h(SIDj,N1), h(IDi)),N1);//C1
recv_2(R,I, XOR(h(N2), h(N1,1)));//C2
send_3(I,R, h(XOR(XOR(h(IDi), h(h(IDi), h(s))), p
send_4(I,R,Lij);
claim_i1(I, Secret, XOR(h(SIDj,N1), h(IDi)));//C1
claim_i2(I,Secret,XOR(h(N2), h(N1,1)));//C2
claim_i3(I,Secret,h(XOR(XOR(h(IDi), h(h(IDi), h(
/*claim_i4(I, Secret, h(s)); claim_i10(I,Secret,k);
claim_i12(I, Secret,N1); claim_i5(I, Secret, h(N2)
claim_i6(I, Niagree); claim_i7(I,Nisynch);claim_i
/*claim_i10(I, Commit, R,N1,N2);*/
}
role R{
const IDi,Pi,N2,Bi, SIDj,k,s,b,g,p;
var N1:Nonce; fresh N2: Nonce;
recv_1(I,R, XOR(h(SIDj,N1), h(IDi)),N1);//C1
send_2(R,I, XOR(h(N2), h(N1,1)));//C2
recv_3(I,R, h(XOR(XOR(h(IDi), h(h(IDi), h(s))), pl
recv_4(I,R,Lij);
claim_r13(R,Secret,Lij); //Lij
/*claim_r1(R, Secret, XOR(h(SIDj,N1), h(IDi)));//
claim_r2(R, Secret, XOR(h(N2), h(N1,1)));//C2
/*claim_r2(R, Secret, h(XOR(XOR(h(IDi), h(h(IDi
claim_r3(R, Secret,h(s)); claim_r10(R,Secret,k); c
/*claim_r5(R, Secret, h(N2)); */
claim_r6(R , Alive); claim_r7(R,Niagree);
/*claim_i10(R, Running, I,N1,N2);*/claim_r8(R,N
/*claim_r9(R, Weakagree);*/
}}
```

```
Claim                                                                                  Status  Comments

Directauthlogin  I  Directauthlogin,i1   Secret XOR(h(SIDj,N1),h(IDi))                        Ok      No attacks within bounds.
                    Directauthlogin,i2   Secret XOR(h(N2),h(N1,1))                            Ok      No attacks within bounds.
                    Directauthlogin,i3   Secret h(XOR(XOR(h(IDi),h(h(IDi),h(s))),plus(h(N2)...  Ok      No attacks within bounds.
                    Directauthlogin,i6   Niagree                                              Ok      No attacks within bounds.
                    Directauthlogin,i7   Nisynch                                              Ok      No attacks within bounds.
                    Directauthlogin,i8   Alive                                                Ok      No attacks within bounds.
                 R  Directauthlogin,r13  Secret XOR(h(N2),h(IDi,mod(exp(g,h(b)),p)))          Ok      No attacks within bounds.
                    Directauthlogin,r2   Secret XOR(h(N2),h(N1,1))                            Ok      No attacks within bounds.
                    Directauthlogin,r3   Secret h(s)                                          Ok      No attacks within bounds.
                    Directauthlogin,r10  Secret mod(exp(g,h(b)),p)                            Ok      No attacks within bounds.
                    Directauthlogin,r4   Secret h(IDi)                                        Ok      No attacks within bounds.
                    Directauthlogin,r6   Alive                                                Ok      No attacks within bounds.
                    Directauthlogin,r7   Niagree                                              Ok      No attacks within bounds.
                    Directauthlogin,r8   Nisynch                                              Ok      No attacks within bounds.
```

Figure 3.8 Scyther Analysis of Direct Authentication Using Crypto-Token

Formal Analysis using Scyther

To perform the formal security analysis, this section focuses on evaluating the vulnerability of certain parameters such as h(IDi), k, S, C1, C2, N1, N2

Pij which are used in the proposed authentication scheme. If the parameters are compromised during any stage of communication between user and server during authentication, then the protocol is vulnerable to attacks which fails to justify the security of authentication scheme. The proposed protocol is coded and analyzed using the security analyzer Scyther, which checks for the vulnerability of each of the parameters used in the scheme. Scyther is configured with ten (10) runs and all possible attacks. There are various claims made as part of the security analysis and these claims are validated by executing and analyzing the proposed scheme using Scyther. The “No attack” results shown in Figure 3.8 prove that Scyther validates all the claims made as part of security analysis.

Claim 1: The proposed scheme is designed to ensure the secrecy of the user ID, throughout the registration and authentication process.

The user ID is submitted in the hashed form to the IdP during the registration process. This is used along with the password and the secret key of IdP to generate the secret parameters to be stored in the crypto-token. During the authentication process, user ID is hashed and XOR-ed with server-ID and nonce N1 to generate the challenge C1. The claim that user ID, ID_i is safe is verified by Scyther.

Claim 2: The proposed scheme is designed to ensure the secrecy of the variant of password ‘k’ throughout the registration and authentication process.

The password is never transmitted in the plaintext form either to the IdP or to the cloud server. It is converted into a modified form ‘k, by finding the hash of the password viz. ‘b’ and then raising g_0 (generator of a cyclic group) to the power of ‘b’. Now to obtain the password from ‘k’, we need to solve the discrete logarithm problem. During the authentication process, password is used to generate the login request. It is not sent to the cloud service provider, but it is used to check the stored password and to calculate the values K_i, R_i. Also the password is not stored anywhere other than in the crypto-token. Scyther results validate the claim that ‘k’ remains a secret.

Claim 3: The proposed scheme requires the S to be a secret

‘S’ is the secret key of the IdP. It is used in its hashed form to compute the parameters to be stored in the crypto-token and to verify the user during the authentication process. Scyther validated the claim that ‘S’ is safe.

Claim 4: The proposed scheme requires that the challenge C1 is secret

Challenge C1 is the hashed information containing user ID, service provider ID and nonce N1 sent by the user to the server to ensure security from replay attack. Scyther validated the claim that ‘C1’ is safe.

Claim 5: The proposed scheme requires that the response C2 remains a secret

C2 is the communication sent by the service provider in response to the challenge C1 sent by the user. The computation of C2 is done using the challenge N1 and a nonce N2 generated by the server. Scyther validated the claim that ‘C2’ is safe.

Claim 6: The proposed scheme requires that P_{ij} is secret

P_{ij} is the login request sent from the user to the server, which contains the user ID, the secret key of the IdP and the nonce sent by the server. The login request should not reveal any information, which will enable an adversary to forge a valid login request. Scyther validated the claim that P_{ij} is safe.

Claim 7: The scheme assures the user and the server remains alive and also the server is assured that the user remains alive.

The server is said to be alive, if the proposed protocol is used by the server for the initial (k-1) messages exchanged with the user, when the user sends the k^{th} message. Thus if the entity making the claim, receives atleast one message from it's honest communication partner, before it makes the claim, then the claim will be valid. The Scyther tool validates the aliveness claim.

Claim 8: The scheme guarantees Weak agree

The scheme guarantees that the user (crypto-token) is in weak agreement with the server which ascertains that both of them are interacting with each other. Hence, the user and the server are executing the proposed scheme with each other. The actions of the adversary do not affect the operation of the proposed scheme during the execution of the protocol run. The claim is validated by the Scyther results.

Claim 9: The scheme assures Niagree between the user (crypto-token) and the server

Niagree claim enforces that the sender (user) and the receiver (server) agree upon the values of variables exchanged during the running of the proposed scheme. During the operation of the proposed scheme, the user and sever can exchange data safely and the correctness of the claim is justified by the analysis results.

Claim 10: The proposed scheme holds Synchronization during the authentication process

Ni-Synch or Non-Injective Synchronization property requires that the corresponding send and receive events (1) happened in the correct order and (2) have the same contents. Ni-Synch is valid if all actions before the claim are performed as per the protocol description of the proposed scheme. The proposed protocol satisfies this claim as indicated by the result of Scyther analysis.

3.1.4 Mobile-Token Based Direct Authentication Protocol without Verifier Table

Physical tokens such as crypto-tokens and smart cards enable secure identification of users and offer the advantage of a highly secure tamper resistant environment making it difficult to misuse the contents of the memory (Scheuermann, 2002). Due to the computational capability, tamper-resistance property, and convenience in managing authentication parameters of users's, devices with cryptographic capabilities such as crypto-tokens/smart cards have been widely adopted as the second authentication factor in many remote user authentication schemes (Hwang and Li, 2000) (Chien, 2002) (Hsiang and Shi, 2009). However, carrying around a separate additional device such as a crypto-token remains a burden to users. Also it involves cost factor and hence these schemes are mostly constrained to corporate environments. This points out to the requirement of an authentication factor that can be used by laymen as well.

In the recent past, mobile phones have become more of a necessity than a luxury and hence leveraging the mobile device to serve as an authentication factor can help improve the security of authentication schemes. Phone aids in identifying the owner and can be used to store authentication information which makes it the right candidate as a second factor for user authentication. The process flows of the registration and authentication stages are as depicted in Figure 3.4.

Phases of the Proposed Protocol: The focus is on designing a mutual authentication protocol with lesser number of stored variables in the mobile token and less processor intensive operations contributing to less power consumption and heat generation for power constrained equipment’s like mobile phones**.**

The proposed protocol consists of four phases’ viz., User registration phase, Login, Authentication & key agreement phase and Password change phase. During the registration and authentication phase of this protocol, users mobile phone should have Internet connectivity. The notations used are listed in Table 3.3.

Table 3. 3 Notations Used in the Protocol (Direct&Mobile-Token)

ID_i, PW_i	Identity, Password of user U_i.
S, N_j	Secret key of IdP, Nonce of RS
N_i	Nonce of U_i.
$h(.)$, $\oplus$, $\|$	hash function, XOR operation, Concatenation Operation

Registration Phase

During the registration process, User downloads a mobile app into his mobile phone. This app enables the user to execute the registration, login& authentication and password change phase of the protocol. This phase is executed only once during which the user submits his credentials to RS of IdP. RS generates a set of security parameters using the submitted credentials and his key value. RS stores the security parameters within a secret file which is downloaded and stored in a secure location within the user's mobile phone. The secret file is encrypted using the password (PBE) of the user which ensures that only a valid user will be able to store the token into his mobile phone and use the same to avail secure access to the Cloud services.

The registration process illustrated in Figure 3.9 can be explained as follows:

R1: The user U_i clicks the "Register" link at Service Provider's (SPs) page. SP redirects U_i to the registration page of the IdP.

R2: IdP prompts U_i to submit her identity ID_i and PW_i. The user submits his $h(ID_i)$ and $h(PW_i.)$.

R3: RS checks whether $h(ID_i)$ already exists in its user table. If so U_i is prompted to select a new ID_i.

R4: RS creates a file containing the authentication parameters $K_{i,}$ M_i, J_i, h(.) and the file is encrypted using password of U_i and a salt value. The salt value is generated using a PRNG function and is concatenated with $h(PW_i)$ and the hash of the result is generated using SHA-256. ie. $h(h(PW_i) \| salt)$.

The output is a 256-bit value which is used as the key for AES encryption algorithm to encrypt the file.

The values of $K_{i,}$ M_i, J_i are generated by performing hash and XOR operations on ID_i, PW_i, S as follows:

$V_i = h(h(ID_i) \| h(S))$,

$K_i = V_i \oplus h(h(ID_i) \| h(PW_i))$,

$J_i = h(V_i)$

$M_i = h(h(ID_i) \oplus h(S)) \oplus J_i$.

Here ‘S’ is the key shared between IDP and the Service Providers.

R5: IdP generates a QR code embedding Service Provider URL, Salt and the URL for downloading the secret file.

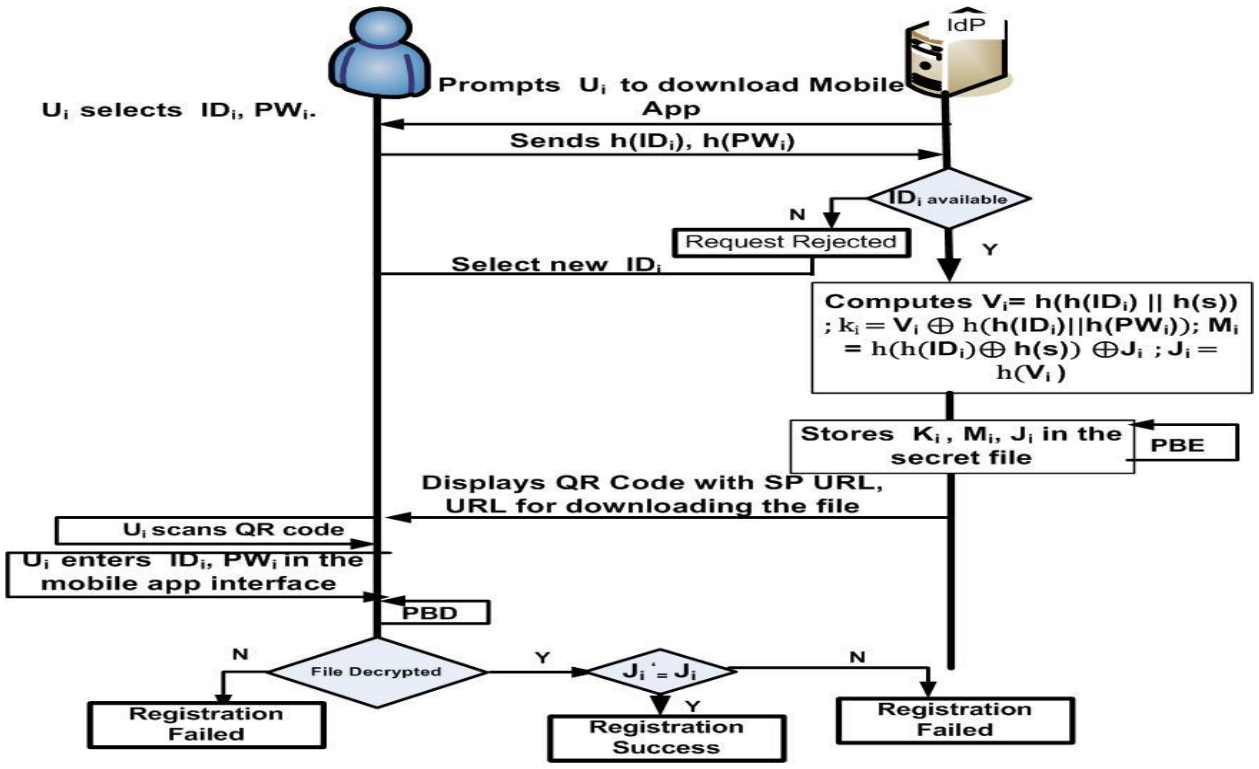
R6: The QR code will be displayed on the Service Provider’s page and the user will be prompted to scan the QR code.

R7: The mobile app, invokes the scanning application, and the user can scan the code. The user will get URL for downloading the secret file, salt, the service provider URL and the link to download the secret file.

R8: The user will be prompted to enter his ID_i, PW_i. The app attempts to decrypt the file using password given as input by the user and the salt value attached to the file. If the decryption is successful, the secret file contents will be accessed.

R9: When the user touches the register button in the mobile app, mobile app, calculates $V_i' = K_i \oplus h(h(ID_i') \| h(PW_i'))$ and $J_i' = h(V_i')$. J_i' is compared with J_i stored in the mobile token and if equal, the registration process is considered successful and user will get the "Registration Successful" message.

R10: The file will be stored in a safe location within the user's phone in the form of a mobile token. IdP stores the User ID ie. $h(ID_i)$ and other profile information in its user table. If registration is not successful, then the file will be deleted from the user's phone and user will get a "Registration Failed" message.



IdP
Prompts U_i to download Mobile App
U_i selects ID_i, PW_i.
Sends $h(ID_i)$, $h(PW_i)$
ID_i available
N
Request Rejected
Y
Select new ID_i
Computes $V_i = h(h(ID_i) || h(s))$; $k_i = V_i \oplus h(h(ID_i)||h(PW_i))$; $M_i = h(h(ID_i) \oplus h(s)) \oplus J_i$; $J_i = h(V_i)$
Stores K_i, M_i, J_i in the secret file
PBE
Displays QR Code with SP URL, URL for downloading the file
U_i scans QR code
U_i enters ID_i, PW_i in the mobile app interface
PBD
N
File Decrypted
Y
$J_i' = J_i$
N
Registration Failed
Y
Registration Success
Registration Failed

Figure 3.9 Registration Phase of Direct Authentication Using Mobile-Token

Login phase

Login Phase is executed when the user attempts to access a protected resource of a Service Provider (SP). It is assumed that, the browser at this point does not have an established session with the SP. If there is no existing session between the browser and the SP, then SP generates a login session and authenticates the user by executing the authentication phase, as illustrated in figure 3.10. The user uses his password and the parameters stored within the mobile token deployed in the mobile phone, to authenticate himself to the SP. The procedure can be explained as follows:

L1: Authentication Server (AS) displays the login page and prompts the user to enter user's identity (ID_i) and Password (PW_i). The values are sent over the communication channel as $h(ID_i)$ and $h(PW_i)$). AS calculates:

$L_j = h(h(ID_i) \| h(S))$, $M_j = h(h(ID_i) \| h(PW_i))$, $C_j = h(L_j \oplus M_j)$,

$P_j = h(L_j)$,

$T_j = h(h(ID_i) \oplus h(S)) \oplus P_j$.

AS generates a nonce N_j and computes the challenge $Q_j = h(C_j \| P_j \| N_j)$

L2: The random nonce N_j and challenge Q_j ie $< Q_j, N_j >$ is send to the user U_i, via a secure communication channel (QR code)

L3: The mobile app computes $C_j' = h(K_i)$, $P_j' = J_i$, $Q_j' = h(C_j' \| P_j' \| N_j)$ and checks whether Q_j' = Challenge Q_j, received from AS. If so, mobile app considers the message as being received from an authenticated source and continues with the following steps. This step is included to avoid the possibility of phishing attack, since only the servers which hold the shared key h(S) of IdP will be able to generate this message.

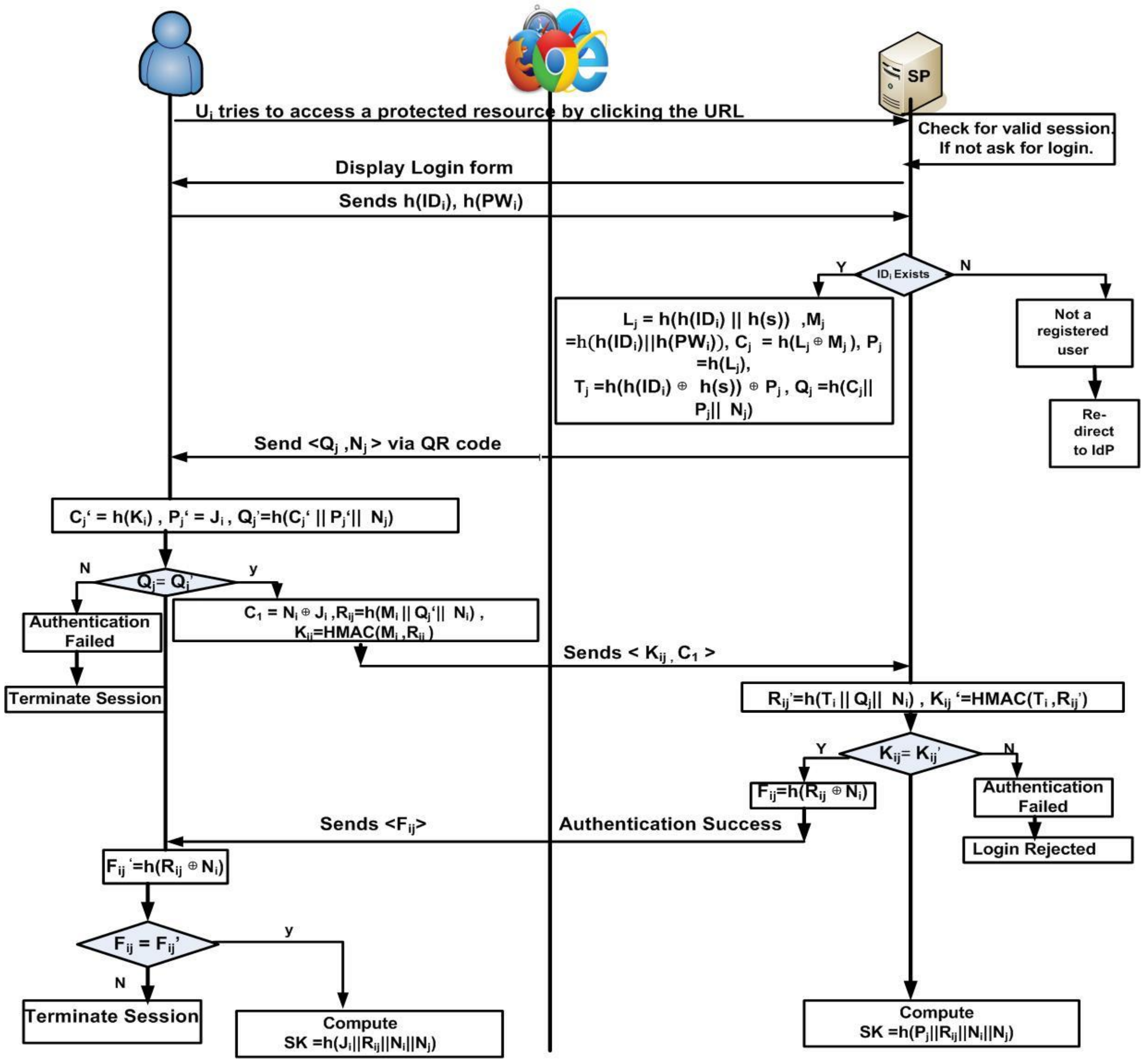
Authentication and Key Agreement Phase

A1: Mobile app on behalf of user U_i computes $R_{ij} = h(M_i \| Q_j' \| N_i)$, where N_i is a nonce generated by U_i .

A2: Computes $C_1 = N_i \oplus J_i$, K_{ij} = HMAC (M_i, R_{ij}) to AS of the SP. U_i sends $< K_{ij}, C_1 >$ to AS via wi-fi or cellular network (GSM/GPRS).

HMAC is a keyed hash function and M_i serves as the key which is used to encrypt the message R_{ij}.

A3:AS on receiving the message $<K_{ij}, C_1 >$, computes $N_i' = C_1 \oplus h(h(h(ID_i) \| h(S)))$, $R_{ij}' = h(T_j \| Q_j \| N_i')$ and K_{ij}' = HMAC (T_j, R_{ij}'). AS recalculates the HMAC value by using T_j as the key and R_{ij}' as the message. Since key M_i which is equal to T_j, is known only to the user, the value K_{ij} would have been calculated only by the user. AS assures the freshness of the nonce N_j. AS checks whether K_{ij}'is equal to the received K_{ij}. If equal SP considers the user as authenticated and that the integrity of message is maintained. Otherwise the login request is rejected.



SP
Ui tries to access a protected resource by clicking the URL
Check for valid session. If not ask for login.
Display Login form
Sends h(IDi), h(PWi)
IDi Exists
Y
N
Lj = h(h(IDi) || h(s)) ,Mj =h(h(IDi)||h(PWi)), Cj = h(Lj ⊕ Mj), Pj =h(Lj), Tj =h(h(IDi) ⊕ h(s)) ⊕ Pj , Qj =h(Cj|| Pj|| Nj)
Not a registered user
Re-direct to IdP
Send <Qj ,Nj > via QR code
Cj' = h(Ki) , Pj' = Ji , Qj'=h(Cj' || Pj'|| Nj)
Qj= Qj'
N
y
Authentication Failed
Terminate Session
C1 = Ni ⊕ Ji ,Rij=h(Mi || Qj'|| Ni) , Kij=HMAC(Mi ,Rij)
Sends < Kij , C1 >
Rij'=h(Ti || Qj|| Ni) , Kij '=HMAC(Ti ,Rij')
Kij= Kij'
Y
N
Fij=h(Rij ⊕ Ni)
Authentication Failed
Login Rejected
Sends <Fij>
Authentication Success
Fij '=h(Rij ⊕ Ni)
Fij = Fij'
y
N
Terminate Session
Compute SK =h(Ji||Rij||Ni||Nj)
Compute SK =h(Pj||Rij||Ni||Nj)

Figure 3.10 Login and Authentication Phase of Direct Authentication Using Mobile-Token

A4: The SP sends a response $F_{ij} = h(R_{ij} \oplus N_i)$ along with a successful authentication message.

A5: If the authentication is successful then SP notifies the user's browser of a successful login. The user on receiving $F_{ij,}$

computes F_{ij} ' $= h(R_{ij} \oplus N_i)$.U_i checks the freshness of the nonce N_i and verifies the authenticity of the server.

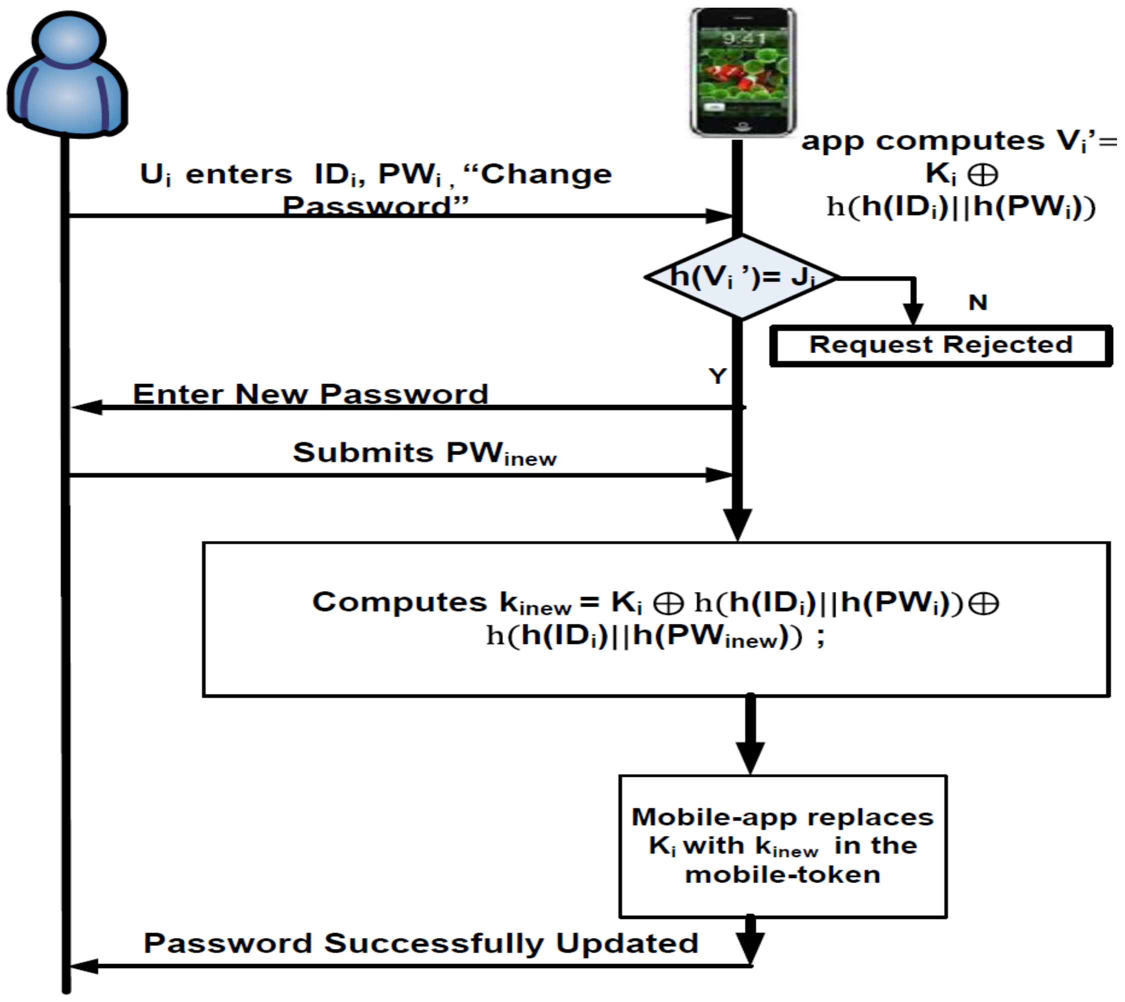
A6: Both U_i and SP computes the session key as $SK_{us}= h(J_i \| R_{ij} \| N_i \| N_j)$ and $SK_{su}= h(P_j\| R_{ij}' \| N_i \| N_j)$ respectively.

Password Change Phase

The password change phase as shown in Figure 3.11 is invoked when the user wishes to change his password without the intervention of the IdP or the SP and is carried out as follows:

P1: User enters his identity (ID_i) and Password (PW_i) and executes the "Password Change" request. The mobile app computes $V_i' = K_i \oplus h(h(ID_i)\| h(PW_i))$ and checks if $h(V_i')$ it is equal to stored J_i. If equal, the mobile app prompts the user to enter the new password 'PW_{inew}'. Otherwise the "password change" request is rejected.

P2: The app calculates $K_{inew} = K_i \oplus h(h(ID_i)\| h(PW_i)) \oplus h(h(ID_i)\| h(PW_{inew}))$ and replaces the existing K_i value in the file with K_{inew}.



U_i enters ID_i, PW_i, "Change Password"
app computes $V_i' = K_i \oplus h(h(ID_i)||h(PW_i))$
$h(V_i') = J_i$
N
Request Rejected
Y
Enter New Password
Submits PW_{inew}
Computes $k_{inew} = K_i \oplus h(h(ID_i)||h(PW_i)) \oplus h(h(ID_i)||h(PW_{inew}))$;
Mobile-app replaces K_i with k_{inew} in the mobile-token
Password Successfully Updated

Figure 3.11 Password Change Phase of Direct Authentication Using Mobile-Token

Security Analysis

i. Security against Guessing Attack: The aim of this attack is to find out the password of the user. Assume that the adversary A, manages to get the secret file containing <K_i, M_i, J_i, h(.)>. Among these parameters, K_i contains the user password and $K_i = V_i \oplus h(h(ID_i) \| h(PW_i))$. Now assume that 'A' guesses the password PW_i *. Then he can calculate $K_i = V_i \oplus h(h(ID_i) \|$

$h(PW_i^*)$). However, to check whether the guessed password is correct, the adversary should know $V_i = h(h(ID_i) \| h(S))$, which is not stored in the mobile-token. It cannot be extracted from $J_i = h(V_i)$ as hash functions are not reversible. Otherwise, to obtain V_i, he should be knowing the secret key of the server. In the case of android phones, the secret file is stored in a private location accessible only to the mobile app within the phones memory. Hence even the owner of the file will not be able to access its contents which rules out the possibility of a valid user getting h(S) using his own password and then trying to guess another user's password by stealing his mobile-token.

ii. Security against Replay Attack: The proposed authentication protocol uses nonce values to resist replay attack. The server generates the nonce N_j which is used to calculate the challenge $Q_j = h(C_j \| P_j \| N_j)$. Now the user U_i generates a response to this challenge as $K_{ij} = HMAC(M_i, R_{ij})$ where $R_{ij} = h(M_i \| Q_j \| N_i)$. Thus the response contains Q_j which inturn includes the nonce N_j generated and transmitted by the server to user. On receiving the response K_{ij} which contains the nonce N_j, uniquely generated for that particular session, server is assured that this is not a replay attack. The server then sends an authentication response message $F_{ij} = h(R_{ij} \oplus N_i)$ where N_i is the nonce of U_i and is unique to that session. On receiving F_{ij}, U_i checks the freshness of the nonce and is assured that this is not a replay attack. These nonce values which are generated independently by the server and user are unique to a particular session and are included in the messages exchanged between the user and the server. Hence an adversary cannot get unauthorized access to a system by using previous messages.

iii. Server Spoofing Attack: For an adversary to masquerade as a legitimate service provider, he must be able to generate the messages that are generated by a valid server. Thus if A is an adversary, he should be able to generate $<Q_{j,} N_j>$, $< F_{ij} >$ and the session key. However $Q_j = h(C_j \| P_j \| N_j)$, where $C_j = h(L_j \oplus M_j)$ with $L_j = h(h(ID_i) \| h(S))$, $M_j = h(h(ID_i) \| h(PW_i))$, and $P_j = h(L_j)$. Hence, to generate $Q_{j,}$ the adversary should have the knowledge of server's secret key 'S' and the password of user. Similarly $F_{ij} = h(R_{ij} \oplus N_i)$ where $R_{ij} = h(M_i \| Q_j \| N_i)$ with $M_i = h(h(ID_i) \oplus h()S)) \oplus h(h(h(ID_i) \| h(S))$. To generate a valid F_{ij}, he should have the knowledge of server's secret key, user passwords and the nonce $N_{i.}$ These values are neither send across the communication channel, nor can they be extracted from the messages between the user and the server.

iv. Insider and Stolen Verifier Attack: Insider attack is launched by an administrator who deliberately leaks secret information resulting in security flaws of the authentication scheme. In the proposed scheme both during registration and login phase, the $h(PW_i)$ is send to the server. Deriving the password from $h(PW_i)$ within a specific time interval is very difficult. The proposed scheme does not maintain any verifier table and hence it is secure against stolen verifier attack.

v. Two-Factor Security: In a scenario where, both the user's mobile token and his password are stolen, then there is no way to prevent the attacker from masquerading as the user. Hence the security of the proposed two-factor authentication scheme can be guaranteed when either the mobile-token or the password is stolen but not both. This security property is referred to as two-factor security. In the discussed scheme the secret parameters $< K_{i\,,} M_i , J_i , h(.)>$ of the mobile token are difficult to be derived if the attacker has obtained the user's password alone and not the mobile token. Now if the attacker also intercepts the challenge $Q_j = h(C_j \| P_j \| N_j)$, it is a laborious process to extract M_j (which contains PW_i) from C_j and P_j due to the irreversible property of one-way hash functions.

Again if the attacker intercepts the response < K_{ij}, C_1 > from the user, it is infeasible to derive h(S) or h(PW_i) from HMAC (M_i, R_{ij}) as they are calculated using hash functions. Irreversible property and collision-resistance property of hash functions makes its computationally infeasible for the attacker to retrieve the password within a required time interval. On the other hand, if the attacker, manages to get the mobile token and extracts the values < K_i , M_i , J_i , h(.)> using power analysis attacks suggested by Messergers et al.(1999), he still cannot obtain, PW_i directly from any of these stored values.

vi. Known-Key Security: The known key security means that even if the session key of any of the previous sessions is compromised, the attacker should not be able to derive the session key of any of the future sessions. In the proposed protocol, the session key is calculated using P_j and R_{ij} as $SK_{su} = h(P_j \| R_{ij} \| N_i \| N_j)$ which require the knowledge of password and server's secret key, which is not known to the adversary. The irreversible property of hash functions ensures that P_j and Rij cannot be derived from the past session keys, which makes it difficult for the attacker to derive the future keys. Also the session key calculation involves nonce values generated randomly and independently by both the user and the server. Hence even the valid user and the server will not able to predict the future session keys.

vii. Forward Key Secrecy: The forward key secrecy property requires that a compromise of the master key of the system should not help the adversary to calculate the previously established session keys. In the proposed protocol, even if the master key of the IdP is compromised, the adversary cannot compute any of the previous session keys without knowing the password PW_i of the user.

viii. Mutual Authentication: When the user receives the challenge Q_j from the server, it is verified as Q_j '=h(C_j ' || P_j' || N_j), where C_j ' and P_j' are calculated using parameters in the mobile-token. A response to this challenge is generated by using M_i, which is extracted from the mobile token and is not there in the challenge received from server. The server calculates $T_j = M_i$, using the user's password and its own secret key. A successful verification proves the authenticity of user. Again the response send from the server, F_{ij} is verified by the user. Thus the proposed protocol achieves the requirement of mutual authentication which is required in a multi-server environment.

Efficiency Analysis

This section analyzes the efficiency of the proposed mobile token based protocol in terms of the computational and the communication cost. It is assumed that nonce values are 128 bits long and the output of hash function (SHA-2) is 256 bits long. Let T_h, T_x and T_c denote the time complexity for hashing, XOR and concatenation respectively. In the protocol, the parameters stored in the secret file are K_i, M_i, J_i and the memory (E1) needed in the mobile is 768 (3*256) bits. Communication cost of authentication (E2) includes the capacity of transmitting parameters ($h(ID_i)$, $h(PW_i)$, Q_j, N_j, k_{ij}, C_1, F_{ij}) which makes E2 equal to 1664 (6*256 + 1 *128) bits. The computation cost of user registration (E3) is the total time of all operations executed in this phase by the user and IdP and is equal to $11T_h + 4T_x + 3T_c + 1T_s + 1T_d$. The computation cost of the user (E4) and the server (E5) is the total time of all operations executed by the mobile app and the service provider during login and authentication. During authentication, the mobile app performs 7 hash functions, 1 XOR and 6 concatenation making E4 equal to $8T_h + 4T_x + 9T_c$. Similarly, E5 is $11T_h + 7T_x + 11T_c$.The computation cost of password change (E6) is the total time of all operations executed in this phase by the user and is equal to $6T_h + 2T_x + 2T_c$. Comparison with other protocols are shown in Table 3.4.

Table 3.4 Comparison of Computational Efficiency with Other Protocols

	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Mobile-token based Protocol	768 bits	1664 bits	$11T_h + 4T_x + 3T_c + 1T_s + 1T_d$	$8T_h + 4T_x + 9T_c$	$11T_h + 7T_x + 11T_c$	$6T_h+2T_x+2T_c$
Choudhary et al. [2011]	1024 bits	1920 bits	$6T_h + 3T_x + 1T_e + 2T_c$	$10T_h +2T_x+ 1T_e + 3T_c$	$8T_h+1T_x +1T_e +3T_c$.	$4T_h+4T_x$
Jaidhar [2013]	1024 bits	1664 bits	$5T_h + 5T_x + 1T_e + 5T_c$	$6T_h + 2T_x + 9T_c+2T_s +1T_d$	$5T_h+1T_x+ 8T_c+2T_d +1T_s+1T_e$	$3T_h + 2T_x + 3T_c$
Rui Jiang [2013]	768 bits	1152 bits	$4T_h + 1T_x + 1T_e + 1T_c$	$7T_h + 1T_x + 4T_c+1T_d+1T_e$	$7T_h+ 5T_c+1T_s+1T_e$	$18T_h + 3T_x + 11T_c+2T_s +1T_d +4T_e$

Number of hash operations during the registration phase (E3) is more in the case of proposed protocol for direct authentication using mobile-token, due to the calculations done at the client side to verify the authenticity of the user, before storing the mobile-token permanently in the phone. In the case of the proposed mobile token based protocol, the authenticity of the user is verified before storing the secret file into the user’s smart phone, by attempting to decrypt the encrypted file downloaded from the Identity Provider. In addition, the protocol also ensures the integrity of the stored parameters in the secret file which is downloaded from the server, by recalculating the value of a parameter strored into the file by the server. Only after these two verifications are done, will the file be permanently stored in to user’s smart phone.

Also the proposed mobile token based protocol is using HMAC to generate the response from the phone to the user, and HMAC requires two hash, two concatenations and two XOR operations. Again the HMAC value send by the client is verified by the server to authenticate the user. All these processes are enhancing security though it leads to an increase number of computations.

Though these computations increase the computation cost of the protocol and affects total computational time and efficiency, the protocol aids in providing enhanced security. In such a scenario, it can be mentioned in the Service Level Agreement between the IdP and the Service Providers that the authentication protocol provided by the IdP, provides secure authentication of users that requires a certain time period for execution. The authentication protocol can be adopted by those service providers to whom the time duration for execution of authentication protocol is agreeable.

Computations done during the password change by Rui Jiang’s protocol is much more compared to the proposed protocol, as in Rui Jiang’s protocol, the entire steps in authentication phase is executed before the password is changed by the server and user.

Scyther Analysis

The formal analysis of protocol is done using Scyther. The strength of the protocol is verified using Scyther tool which ascertains the strength by evaluating the resistance of the protocol to various attacks. Scyther uses strand space model for formalizing logic and uses Dolev-Yao model for modelling the network, which caters to the requirement of a mathematical approach for validating the protocol.

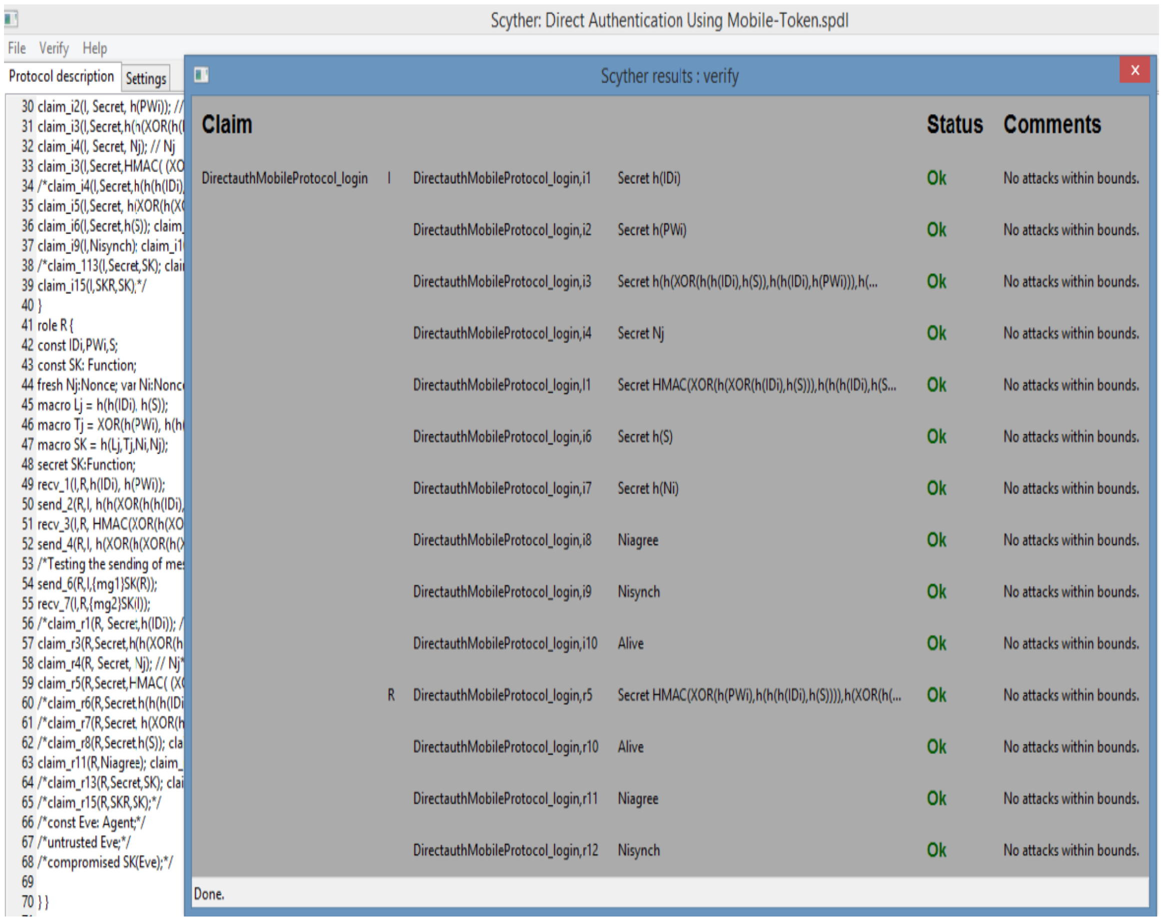
The analysis result of login phase is shown in Figure 3.12. The protocol is written in SPDL as follows:

```
//Login Phase with Symmetric Key Encryption of Message and Testing the
Compromise of Symmetric Key
const  exp: Function; const hash: Function; hashfunction h; const XOR:
Function;
const h1:Function; const HMAC:Function; usertype SessionKey;
secret SK:Function; const Fresh:Function;
protocol DirectauthMobileProtocol-login(I,R){
role I {
const IDi,PWi,S,mg1,mg2;
var Nj:Nonce; var SK: SessionKey;
fresh Ni :Nonce; send_1(I,R,h(IDi), h(PWi));
recv_2(R,I, h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi))), h(h(h(IDi), h(S))) ,
Nj)), Nj);//Qj
//Calculating Mi = XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )))
//Calculating Cj '= h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi))));
//Calculating Pj' = h(h(h(IDi), h(S)));
//Calculating Qj' = h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi),
h(S))) , Nj))
```

```
//Calculating Rij = h(Mi||Qj'||Nj) = h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni)

//Calculating Kij  = HMAC(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) ))), h(XOR(h(PWi), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni))

send_3(I,R, HMAC(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) ))), h(XOR(h(PWi), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni)), Ni); //Kij

recv_4(R,I, h(XOR(h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni), Ni)))); //Fij

//Calculating Fij'=h(XOR(h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni), Ni))

//SK= h(Ji|| Mi || Ni || Nj)

macro Ji = h(h(IDi), h(S));

macro Mi= XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) ));

macro SK = h(Ji,Mi,Ni,Nj);

secret SK:Function;

//recv_5(R,I, SK);

/*Testing the sending of messages encrypted using the generated session key */

recv_6(R,I,{mg1}SK(R));

send_7(I,R,{mg2}SK(I));

claim_i1(I, Secret, h(IDi)); // IDi

claim_i2(I, Secret, h(PWi)); // PWi

claim_i3(I,Secret,h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi))), h(h(h(IDi), h(S))) , Nj))); //Qj'

claim_i4(I, Secret, Nj); // Nj

claim_i3(I,Secret,HMAC( (XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) ))), h(XOR(h(PWi), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni)))); //Kij
```

```
claim_i4(I,Secret,h(h(h(IDi), h(S)))); //Ji

claim_i5(I,Secret, h(XOR(h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S))
)), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni),
Ni))));//Fij

claim_i6(I,Secret,h(S)); claim_i7(I,Secret,h(Ni)); claim_i8(I,Niagree);

claim_i9(I,Nisynch); claim_i10(I, Alive); claim_i11(I,Weakagree);

claim_113(I,Secret,SK); claim_i12(I,Empty,(Fresh,SK));
claim_i14(I,Commit,R,Ni,Nj);

claim_i15(I,SKR,SK);

}

role R {

const IDi,PWi,S;

const SK: Function;

fresh Nj:Nonce; var Ni:Nonce; fresh SK:SessionKey;const mg1, mg2;

macro Lj = h(h(IDi), h(S));

macro Tj = XOR(h(PWi), h(h(h(IDi), h(S)) ));

macro SK = h(Lj,Tj,Ni,Nj);

secret SK:Function;

recv_1(I,R,h(IDi), h(PWi));

//Calculating Lj = h(h(IDi), h(S));

//Calculating Mj = h(h(IDi), h(PWi));

//Calculating Cj = h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)))

//Calculating Tj = XOR(h(PWi), h(h(h(IDi), h(S)) ))

//Calculating Pj = h(h(h(IDi), h(S)));

//Calculating Qj = h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi),
h(S))) , Nj))

send_2(R,I, h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi))), h(h(h(IDi), h(S))) ,
Nj)), Nj);//Qj

recv_3(I,R, HMAC(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) ))),
h(XOR(h(PWi), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi),
h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni)), Ni); //Kij
```

```
//Calculating Rij' = h(Tj||Qj||Ni)  = h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni)

//Calculating Kij = HMAC(Tj, Rij') = HMAC(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) ))), h(XOR(h(PWi), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni))

//Calculating Fij=h(XOR(h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni), Ni))

send_4(R,I, h(XOR(h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni), Ni)))); //Fij

//h(Lj|| Tj || Ni || Nj)

//Calculating symmetrc key, SK = h(Lj,Tj,Ni,Nj);

//send_5(R,I, SK);

/*Testing the sending of messages encrypted using the generated session key*/

send_6(R,I,{mg1}SK(R));

recv_7(I,R,{mg2}SK(I));

claim_r1(R, Secret,h(IDi)); // IDi  claim_r2(R, Secret, h(PWi)); // PWi

claim_r3(R,Secret,h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi))), h(h(h(IDi), h(S))) , Nj))); //Qj

claim_r4(R, Secret, Nj); // Nj

claim_r5(R,Secret,HMAC( (XOR(h(PWi), h(h(h(IDi), h(S)) ))), h(XOR(h(PWi), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni)))); //Kij'

claim_r6(R,Secret,h(h(h(IDi), h(S)))); //Pj

claim_r7(R,Secret, h(XOR(h(XOR(h(XOR(h(IDi), h(S))), h(h(h(IDi), h(S)) )), h(h(XOR(h(h(IDi), h(S)), h(h(IDi), h(PWi)), h(h(h(IDi), h(S))) , Nj)), Ni), Ni))));//Fij

claim_r8(R,Secret,h(S)); claim_r9(R,Secret,h(Ni)); claim_r10(R,Alive);

claim_r11(R,Niagree); claim_r12(R,Nisynch); claim_r11(R,Weakagree);
claim_r13(R,Secret,SK);                claim_r12(R,Empty,(Fresh,SK));
claim_r14(R,Running,I,Ni,Nj);
claim_r15(R,SKR,SK);}}
const Eve: Agent;
untrusted Eve;
compromised SK(Eve);
```



```
claim_i2(I, Secret, h(PWi)); //
claim_i3(I,Secret,h(h(XOR(h(
claim_i4(I, Secret, Nj); // Nj
claim_i3(I,Secret,HMAC( (XO
/*claim_i4(I,Secret,h(h(h(IDi),
claim_i5(I,Secret, h(XOR(h(XO
claim_i6(I,Secret,h(S)); claim_
claim_i9(I,Nisynch); claim_i1
/*claim_113(I,Secret,SK); clai
claim_i15(I,SKR,SK);*/
}
role R {
const IDi,PWi,S;
const SK: Function;
fresh Nj:Nonce; var Ni:Nonce
macro Lj = h(h(IDi), h(S));
macro Tj = XOR(h(PWi), h(h(
macro SK = h(Lj,Tj,Ni,Nj);
secret SK:Function;
recv_1(I,R,h(IDi), h(PWi));
send_2(R,I, h(h(XOR(h(h(IDi),
recv_3(I,R, HMAC(XOR(h(XO
send_4(R,I, h(XOR(h(XOR(h(X
/*Testing the sending of mes
send_6(R,I,{mg1}SK(R));
recv_7(I,R,{mg2}SK(I));
/*claim_r1(R, Secret,h(IDi)); /
claim_r3(R,Secret,h(h(XOR(h
claim_r4(R, Secret, Nj); // Nj*
claim_r5(R,Secret,HMAC( (X
/*claim_r6(R,Secret,h(h(h(IDi
/*claim_r7(R,Secret, h(XOR(h
/*claim_r8(R,Secret,h(S)); cla
claim_r11(R,Niagree); claim_
/*claim_r13(R,Secret,SK); clai
/*claim_r15(R,SKR,SK);*/
/*const Eve: Agent;*/
/*untrusted Eve;*/
/*compromised SK(Eve);*/

}}
```

Claim				Status	Comments
DirectauthMobileProtocol_login	I	DirectauthMobileProtocol_login,i1	Secret h(IDi)	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i2	Secret h(PWi)	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i3	Secret h(h(XOR(h(h(IDi),h(S)),h(h(IDi),h(PWi))),h(...	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i4	Secret Nj	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,I1	Secret HMAC(XOR(h(XOR(h(IDi),h(S))),h(h(h(IDi),h(S...	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i6	Secret h(S)	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i7	Secret h(Ni)	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i8	Niagree	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i9	Nisynch	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,i10	Alive	Ok	No attacks within bounds.
	R	DirectauthMobileProtocol_login,r5	Secret HMAC(XOR(h(PWi),h(h(h(IDi),h(S)))),h(XOR(h(...	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,r10	Alive	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,r11	Niagree	Ok	No attacks within bounds.
		DirectauthMobileProtocol_login,r12	Nisynch	Ok	No attacks within bounds.

Done.

Figure 3.12 Scyther Analysis of Direct Authentication Using Mobile Token

Formal Analysis using Scyther

To perform the formal security analysis, this section focuses on evaluating the vulnerability of certain parameters such as IDi, PW_i, S, V_i, K_i, M_i, J_i , Q_j, N_i, N_j ,K_{ij}, P_j, F_{ij} which are used in the proposed authentication scheme.

There are various claims made as part of the security analysis and these claims are validated by executing and analyzing the proposed scheme using Scyther. The "No attack" results shown in Figure 3.12 proves that Scyther validates all the claims made as part of security analysis.

Claim 1: The proposed scheme is designed to ensure the secrecy of the user ID, throughout the registration and authentication process.

The user ID is submitted in the hashed form to the IdP during the registration process. This is used along with the password and the secret key of IdP to generate the secret parameters to be stored in the mobile-token. Claim that user ID, IDi is safe is verified and validated by Scyther.

Claim 2: The proposed scheme is designed to ensure the secrecy of the password 'PW_i' throughout the registration and authentication process.

The password is never transmitted in the plaintext form either to the IdP during the registration process or to the service provider during the authentication process. It is transmitted in a hashed form and hash operations are irreversible. During the authentication process, the hashed password is used by the server along with h(IDi) , h(S) and nonce N_j to compute the challenge $Q_{j.}$ The calculation of Q_j involves several hashing and XOR operations, which makes it very difficult to retrieve PWi or h(PWi) from Q_j. Though the response K_{ij} generated by the mobile token also includes the user password, it is hashed and XOR-ed with other parameters and nonce values and the HMAC of the result is taken to get the K_{ij} value. This makes it all the more difficult to extract PWi or h(PWi) from K_{ij}. Also the variant of the password stored in the mobile-token is in a hashed form and is combined with other parameters. Scyther results validate the claim that ‘PW_i’ remains a secret.

Claim 3: The proposed scheme requires the S to be a secret

‘S’ is the secret key of the IdP, which is shared with the registered service providers. It is used in its hashed form to compute the parameters to be stored in the mobile-token and to verify the user during the authentication process. Scyther validates the claim that ‘S’ is safe.

Claim 4: The proposed scheme requires that the parameter V_i stored remains a secret

V_i is a value used to generate the authentication parameters calculated by the IdP and stored in the mobile token. V_i is computed by hashing the concatenation of hash of user ID and hash of server’s secret key S. Scyther validates the claim that ‘V_i’ is safe.

Claim 5: The proposed scheme requires that the authentication parameter K_i stored in the mobile token remains a secret

K_i is one among the authentication parameters calculated by the IDP and stored in the mobile token. K_i is computed by performing hash and XOR operations on the hashed values of V_i, h(IDi), h(PWi). Scyther validates the claim that ‘K_i’ is safe.

Claim 6: The proposed scheme requires that the authentication parameter J_i stored in the mobile token remains a secret

J_i is one among the authentication parameters calculated by the IDP and stored in the mobile token. J_i is computed by performing hash of V_i. Scyther validates the claim that ‘J_i’ is safe.

Claim 7: The proposed scheme requires that the authentication parameter M_i stored in the mobile token remains a secret

M_i is one among the authentication parameters calculated by the IdP and stored in the mobile token. M_i is computed by performing XOR operations on the hashed values of Ji and h(PWi). M_i should not reveal any information that will enable the adversary to impersonate a valid user. Scyther validates the claim that 'M_i' is safe.

Claim 8: The proposed scheme requires that the parameter P_j is secret

P_j generated by the server is the hashed information containing hash value of hash of user ID concatenated with the hash of user PW. The value of P_j is used in computing the challenge $Q_{j.}$ P_j is not transmitted in the plain text form to the user. Scyther validated the claim that 'P_j' is safe.

Claim 9: The proposed scheme requires that the challenge Q_j is secret

Challenge Q_j generated by the server is the hashed information containing hash of user ID, hash of user PW, hash of server secret key, S and nonce N_j sent by the server to the user to ensure against replay attack and phishing attack. Q_j is re-calculated by the mobile token to verify the authenticity of the origin of communication. Q_j should not reveal any information that will enable an adversary to generate a valid challenge. Scyther validated the claim that 'Q_j' is safe.

Claim 10: The proposed scheme requires that the response K_{ij} remains a secret

K_{ij} is the communication sent by the mobile in response to the challenge Q_j sent by the server. The computation of K_{ij} is done by generating an HMAC value which uses two inputs. HMAC algorithm uses M_i as the key and hash of the concatenation of Mi, the received challenge Q_j and a nonce N_j generated by the mobile token as the message whose MAC is to be calculated. K_{ij} is re-calculated by the server using its own set of values. HMAC guarantees authenticity of the origin and integrity of the message. K_{ij} which represents the login request should not reveal any information, which will enable an adversary to forge a valid login request. Scyther validated the claim that 'K_{ij}' is safe.

Claim 11: The proposed scheme requires that F_{ij} is secret

F_{ij} is the response sent from the server to the user after the authentication of user is done. The computation of F_{ij} includes R_{ij} and nonce N_i sent by the user to the server. Fij should not reveal any information, which will enable an adversary to forge the authentication response from the server. Scyther validated the claim that F_{ij} is safe.

Claim 12: The proposed scheme requires that the session key SK is secret

Session key SK is calculated by the user (mobile-token) by using the parameters J_i, M_i, and nonce values N_i, $N_{j.}$ The same session key SK is calculated by the server using the parameters L_j, $T_{j,}$ and nonce value N_i, N_j. SK should not reveal any information that will help the adversary to derive a session key to be used for any of the future sessions. Scyther validated the claim that SK is safe.

Claim 13: The scheme assures the user that the server remains alive and also the server is assured that the user remains alive

The Scyther tool validates the aliveness claim since both the user and the server receives messages from each other before the claim is made.

Claim 14: The scheme assures Niagree between the user (mobile-token) and the server

Niagree claim enforces that the sender (user) and the receiver (server) agree upon the values of variables exchanged during the running of the proposed protocol. During the operation of the proposed protocol, the user and server can send data confidentially and the correctness of the claim is justified by the analysis results.

Claim 16: The proposed protocol holds Synchronization during the registration and authentication process

Ni-Synch or Non-Injective Synchronization property requires that the corresponding send and receive events (1) happened in the correct order and (2) have the same contents. Ni-Synch is valid if all actions before the claim are performed as per the description of the proposed scheme. The proposed protocol satisfies this claim as indicated by the result of Scyther analysis.

CONCLUDING REMARKS

This chapter elaborated an authentication scheme that can be adopted by service providers who would prefer to directly authenticate its User using a strong Two-Factor authentication mechanism and does not require Single Sign-on functionality. The proposed authentication scheme does not require the server to maintain a verifier table, which makes the scheme resistant to insider attack and stolen verifier attack. In the first section of the chapter we have proposed an authentication protocol that uses password and Crypto-token as authentication factors. However, Crypto-tokens need to be carried around and involves cost, owing to which it is suggested for use by corporate sector. Considering the fact that there is an increase in the number of Users using personal mobile devices, as part of the research work we are also proposing an authentication protocol that uses password and mobile token as the two authentication factors. Mobile phones which is proposed as the second factor (Mobile Token) is all pervasive now and is a required necessity for any User. Its proposed use as the 2nd authentication factor thus provides the convenience of using something which is readily available and no extra cost being incurred by the User. The chapter also includes the analysis of the proposed protocols which includes the security, efficiency and formal analysis.

Security analysis of the protocols are done to verify the resistance to various attacks. Efficiency analysis is done to compare the computational efficiency with similar schemes. Formal verification is done using Scyther which verifies the security claims made about the protocol.

CHAPTER 4

BROKERED AUTHENTICATION SCHEME WITHOUT VERIFIER TABLE

Direct Authentication is not always a viable solution in scenarios where users need to access different services simultaneously, in the same session without requiring to login for every service. Services provided via a web portal or services provided by service providers that are functioning in a collaborative environment can be accessed simultaneously by a user. For example, logged-in users of research analyst site Gartner are allowed access to research produced by research analyst site Forrester. Similarly, users may access e-mail service by G-Mail, CRM services by Sales Force and storage services provided by Dropbox Simultaneoulsy. In a scenario, where users are directly authenticated by individual service providers, users have to go through multiple authentication processes to acess these services. This requires redundant storage of information, repetitive exchange of credentials and repeated execution of authentication protocol.

4.1 BROKERED AUTHENTICATION SCHEME

Brokered Authentication effectively solves the problem of direct authentication by having an authentication broker who does the authentication on behalf of the rest of the service providers. By doing so, the service providers are relieved from the task of identifying and authenticating users and the users are provided with Single sign-on functionality, where in they are required to authenticate only once during a session. However, in many cases users need to use services from different domains. These services belonging to different providers need to have interoperability to accept the tokens issued by the central authentication broker/ Identity Provider.

For brokered authentication, the proposed protocols require a Security Token Service (STS) whose functionality is executed by an Identity Provider. The Authentication server of the IdP, authenticates the user by executing the two-factor authentication protocol and generates a SAML token, which is signed by the IdP and sent to the service provider (relying party). The IdP also provides a Single Sign-on (SSO) functionality using Security Assertion Markup Language (SAML) tokens (redirect -POST binding).

4.1.1 Identity Provider and Service Providers Association

The proposed scheme considers that the association between Service Providers and Identity Provider takes place in an integrated trust based environment. An established set of policies and practices are used by the participating entites to exchange information. To exchange information there is a need to establish interoperability and trust relationship between the Authentication Broker/Identity Provider and the service providers whose services (cloud services) need to be accessed by the User. To establish interoperability, the proposed authentication scheme, require Security Assertion Markup Language (SAML)). The SAML open standard provides an efficient mechanism to create and exchange authentication related information of user, between the Service Providers and the Identity Provider.

Association: The Service Providers need to register with the registration server of the IdP by providing a unique server ID, Service Provider URL, a short description of the service provided, and the preferred mode of authentication as “Brokered Authentication”. It is assumed that the IdP and the service providers verify each other’s authenticity using Digital certificates.

In brokered authentication scheme, since user authentication is done by the centralized Identity Provider (IdP), the research needs to address the concern of the Identity Provider becoming a single point of failure. The problem of single point of failure can be addressed by implementing redundant or back-up authentication brokers, although this increases the complexity of the solution.IDP application is hosted in a Cluster of servers. So even if one server goes down another server can immediately take over the services. This can also help in auto scaling. Also the data will be stored in a master database and several slave databases. However, if the entire infrastructure fails, then disaster recovery procedures will be initiated and with minimum amount of delay, the services of IdP will be supported by another data center.

Trust: In the proposed model, the service providers and Identity Providers will have to trust each other to accept and process communications from each other. In this case, when the Identity Provider, produces the identities of the user using SAML assertions, Service Providers will have to trust these assertions. Each Service provider enters into a Business Agreement (BA) with the Identity Provider, thus following a “Pairwise/Direct” trust model (Linn 2004) and both will exchange their own digital certificates issued by a trusted CA, in order to establish trust for future exchange of secure information.

4.1.2 Proposed Brokered Authentication Architecture

The proposed architecture for a Cloud environment includes four participants' viz. a Registration Server (RS), an Authentication Server (AS), Service Provider's (SP's) and users'. The RS and AS are in the same trusted domain and together they provide the functionality of the Identity Provider (IdP).

The user's and SP's comprising the proposed architecture needs to register with the registration server of the IdP. When a SP registers with the IdP, he submits his identity information and the details of the services provided. The CSP's and IdP work in a trust based environment.

In this two-factor authentication scheme, user's password and a registered crypto-token/mobile-token serve as the authentication factors. When a user wants to get the service of a CSP, he is re-directed to the IdP by the SP if he is not a registered user. In such a scenario, the user needs to do a single registration at IdP as illustrated in Figure 3.2 of section 3.1.2 of chapter 3, by providing the User-ID and Password. On successful registration, IdP provides the user with a Crypto-token/Mobile-token containing the security parameters. The server ID's of all the participating service providing servers and the details of their services are also communicated to the user via an e-mail. The login and authentication phase of the proposed scheme runs on the IdP and the service providers redirect the users requesting their services to the IdP for authentication. A user who wants to access the services of a particular SP, tries to login to the provider's web page by submitting the login request. The user is re-directed to IdP and authentication module within the IdP executes the proposed protocol. The second authentication factor of the proposed protocols contains only a few hashed values generated from user's ID, password and the secret key of the server. It does not contain any digital signature which is generated by encrypting the hash of a value by the sender's private key. This requires the implementation of public key infrastructure (PKI). The proposed protocols do not require the support of PKI.

The protocols do not require the server to maintain a password verification table. The registration and authentication process flow is illustrated in Figure 4.1.

In a Single Sign-on platform, if users are authenticated at one service, they do not have to re-enter their credentials and repeat the authentication process to log on to access another service (Hillenbrand et al. 2005). Most of the existing Single sign-on (SSO) solutions typically rely on browser cookies for maintaining state and exchanging identity information. Cookie poisoning is an authentication attack, which involves the modification of cookies of an authorized user to gain unauthorized access to resources. Hence cookies are not a reliable mechanism for sending authentication information. Browser cookies are not transferrable across DNS domains and hence the browser cookies, created from one security domain, for security reasons (same origin policy) can't be read from another one (Trosch 2008). Therefore, to solve cross domain SSO, proprietary mechanisms to pass the authentication data between security domains have been used. This solution which works fine for a single enterprise, becomes impractical when different organizations using different mechanisms collaborate. The proposed brokered authentication protocol uses SAML to exchange authentication information and the information is contained in an encrypted SAML token. To maintain information about the sessions of authenticated users, SAML protocol uses session cookies which contains only information such as session ID of the user and the domain information of the IdP

The kind of Authentication Broker required by the discussed brokered authentication scheme is a Security Token Service (STS) that issues SAML tokens. The SAML protocol is an open standard for exchanging security information between hosted SAML enabled applications (OASIS 2005). SAML enables a user who has established and verified his identity in one domain to access services hosted in another domain.

In the proposed brokered authentication scheme, IdP is representative of the STS who does the role of the authentication authority and also provides SSO functionality using the SAML tokens. Here both IdP who authenticates the user and issues the SAML assertion and the Service Providers who accepts the SAML assertions from IdP should be enabled with SAML. The IdP carries out the two-factor authentication protocol exchange with the user who is re-directed to the IdP by the Service Provider for the authentication process. If the authentication is successful, IdP generates a signed SAML token and the user is redirected to the service provider. The service provider verifies the SAML token and ascertains the origin and the content of the response, before providing the requested service.

To understand how SAML provides Single Sign-on functionality, let us assume that there are two service providers viz. **Safe-Cloud1.com** and **Safe-Cloud2.com** whose services the user Ann would like to access. The Identity Provider who provides the authentication service operates form the domain **Safe-Cloud-IdP.com**. The following scenario will explain how SSO works when Ann tries to access the services of Safe-Cloud1.com and Safe-Cloud2.com during the same session.

- Ann visits Safe-Cloud1.com and attempts to login to access the services. The Service Provider Safe-Cloud1.com generates a SAML authentication request and re-directs Ann's browser to IdP site viz. Safe-Cloud-IdP.com. Ann enters her credentials at the IdP's login page. After successful authentication, Ann is again re-directed to Safe-Cloud1.com along with the SAML assertion generated by the IdP. The assertion contains the authentication information of Ann. IdP creates a session for Ann and generates a session cookie to identify the user (Shibboleth 2015). This cookie is stored in Ann's browser. [Cookies are name-value pairs that is stored in a user's browser and it is created by the web application with which a user has communicated. Every cookie has a domain associated with it, which is the domain of the application that created the cookie and a cookie created by one domain cannot be accessed by another domain. Once a cookie corresponding to a particular domain is created and placed in the browser of a user by that domain, whenever the user's browser makes an HTTP request to the corresponding domain all the cookies associated with that domain are also sent along with that request.]
- Later, during the same session, Ann opens another tab in her browser and tries to access the services of Safe-Cloud2.com web site. Now, when Ann tries to login as in the previous case, she will be re-directed to to the IdP's site Safe-Cloud-IdP.com and with this re-direct, the cookies set by IdP's domain will also be sent. The IdP receives cookies and understands that Ann has an existing session in IdP. In such a scenario, Ann will not have to undergo the authentication process again and IdP sends an assertion and

re-directs Ann's browser back to Safe-Cloud2.com. Ann is logged into Safe-Cloud2.com without having to enter her credentials again.

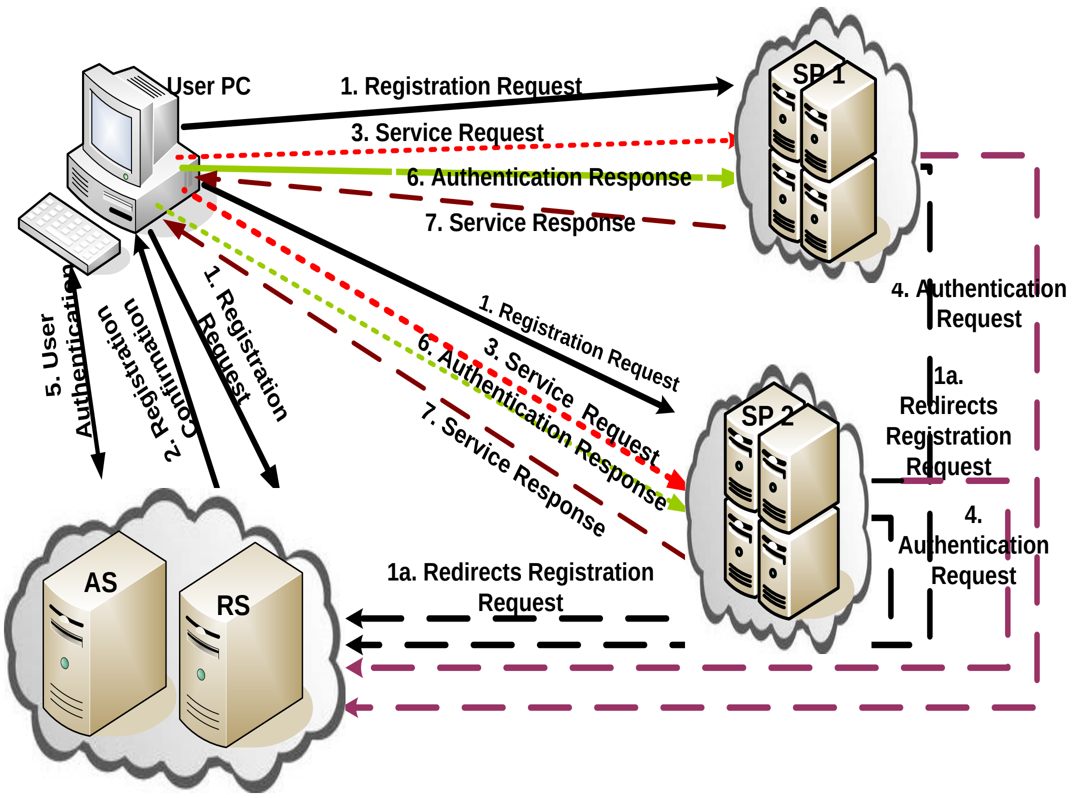

User PC
1. Registration Request
3. Service Request
6. Authentication Response
7. Service Response
SP 1
4. Authentication Request
1. Registration Request
3. Service Request
6. Authentication Response
7. Service Response
SP 2
1a. Redirects Registration Request
4. Authentication Request
5. User Authentication
2. Registration Confirmation
1. Registration Request
1a. Redirects Registration Request
AS
RS

Figure 4. 1 Brokered Authentication - Registration and Authentication Process Flow

4.1.3 A Strong Single Sign-on User Authentication Scheme for Cloud Based Services

Phases of the Proposed Protocol: The proposed protocol consists of three phases viz., Registration, Login and Authentication & and the Password change phase. The notations used are listed in Table 4.1.

Table 4.1 Notations Used in the Protocol (Brokered&Crypto-Token)

IdP, SP, Sj	Identity Provider, Service Provider in the cloud, j th SP
U_i, ID_i, PW_i	i th User, Unique Identification of U_i, password of U_i
SID, SID_j	ID of the Identity Provider, ID of the jth SP
G, g_0, n	Cyclic group, generator of cyclic group G, Prime Number Chosen by U_i which is the order of G
Y	Secret key of server of IdP
R	Random number generated by Crypto-token
h(.) , $\oplus$, ||	One-way hash function, XOR operation, Concatenation Operation
m_1, n_1	Nonce values

Registration Phase

To register for the services of a Service Provider 'SP', the user U_i clicks the "register" button at the SP's home page. SP re-directs U_i to the registration page of IdP. The registration process illustrated in Figure 4.2 can be explained as follows:

R1: U_i generates a cyclic group G of prime order n and selects a generator g_0.

R2: U_i selects his identity ID_i and Password PW_i. Computes $b = h(PW_i)$,

$k = g_0{}^{b} \mod n$.

R3: U_i submits $h(ID_i)$, k to IdP through a secure channel. RS of IdP checks the availability of $h(ID_i)$. Otherwise U_i is prompted to select a new ID_i.

R4: Upon receiving $h(ID_i)$, k, IdP computes

$B_i = h(h(ID_i)\|\ h(SID\|h(y)))$; $Ji = h(h(ID_i)\ \|h(y)) \oplus k$;

$C_i= h(h(ID_i)\|\ h(h(ID_i)\ \|h(y))\ \|k)$; $E_i = B_i \oplus h(h(ID_i)\ \|h(y))$. Here h(y) is the hash of the secret key 'y' of the IdP which is known only to the IdP.

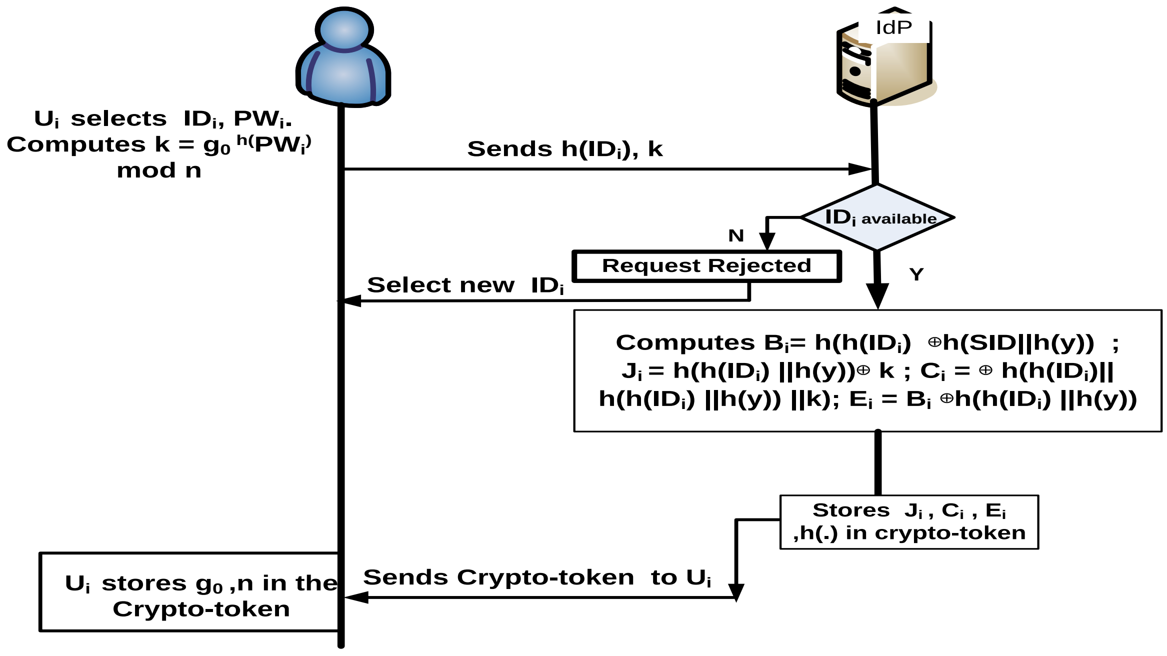

IdP
U_i selects ID_i, PW_i. Computes $k = g_0{}^{h(PW_i)}$ mod n
Sends $h(ID_i)$, k
ID_i available
N
Y
Request Rejected
Select new ID_i
Computes $B_i= h(h(ID_i)\ \oplus h(SID\|h(y))$; $J_i = h(h(ID_i)\ \|h(y))\oplus k$; $C_i = \oplus\ h(h(ID_i)\|\ h(h(ID_i)\ \|h(y))\ \|k)$; $E_i = B_i \oplus h(h(ID_i)\ \|h(y))$
Stores J_i, C_i, E_i ,h(.) in crypto-token
Sends Crypto-token to U_i
U_i stores g_0 ,n in the Crypto-token

Figure 4.2 Registration Phase of Brokered-Authentication Using Crypto-Token

IdP sends a registration confirmation message to U_i along with the list of service providers registered under its domain. IdP stores $\{J_i, C_i, E_i, h(.)\}$ into crypto-token and sends to the user U_i via a secure channel such as a trusted courier. U_i stores g_o, n in the crypto-token which now contains $\{J_i, C_i, E_i, h(.), g_o, n\}$

Login and Authentication Phase

Login and authentication phase as shown in Figure 4.3 can be explained as follows:

LA1: U_i requests for login to the Service Provider. It is assumed that there is no existing session between SP and U_i. SP redirects U_i to the IdP with a SAML assertion containing an authentication request and redirects the user (HTTP Redirect) to the IdP. The authentication request contains the information regarding the SP who initiated the request. The request also contains the Assertion Consumer Service URL (ACS) to which the response should be send.

LA2: IdP displays the login page and prompts U_i to enter his identity. U_i enters ID_i and IdP verifies whether $ID_s = h(ID_i)$ exists in his database. If so IdP generates a random nonce m_1 and sends $<h(h(ID_i), m_1)>$ to U_i. U_i is prompted to proceed with the step LA3. Otherwise U_i is considered to be not registered.

LA3: IdP prompts U_i to insert the crypto-token. U_i inserts the crypto-token and keys in his password PW_i and the server ID, 'SID_j' of the service providing server Sj.

LA4: Crypto-token computes $b = h(PW_i)$, $k^* = g_0^{\ b} \bmod n$.

LA5: Crypto-token computes $h(h(ID_i) \,||h(y))^* = J_i \oplus k^*$ and

$C_i^* = h(h(ID_i)|| h(h(ID_i) ||h(y))^* || K^*)$ and checks whether it is equal to the C_i stored in the crypto-token. If equal, the crypto-token generates the login message as follows. Otherwise the session is terminated.

LA6: Crypto-token selects a large random value ‘r’ and generates a nonce $n_1 = g_0{}^r$.

LA7: Crypto-token computes $P_{ij} = E_i \oplus h(h(h(ID_i) ||h(y))^* || n_1)$;

$B_i = E_i \oplus h(h(ID_i) ||h(y))^*$; $CID_i = C_i \oplus h(B_i || n_1|| SID_j)$; $M_i = h(P_{ij} || C_i || B_i || n_1|| m_1)$; $t = g_0 \oplus h(h(ID_i) ||h(y))^*$;

$Z_i = (r- CIDi) \oplus h(h(ID_i) ||h(y))^* \oplus m_1$.

LA8: Crypto-token sends the login message, $(CIDi, P_{ij}, M_i, t, Z_i)$ to IdP

LA9: On receiving $(CID_i, P_{ij}, M, t, Z_i)$, IdP computes

$r = (Z_i + CIDi) \oplus h(h(ID_i) ||h(y)) \oplus m_1$ where $h(ID_i)$ is the value submitted by U_i in the login page of IdP and h(y) is the hash of IdP’s secret key. IdP also ensures the freshness of nonce m_1.

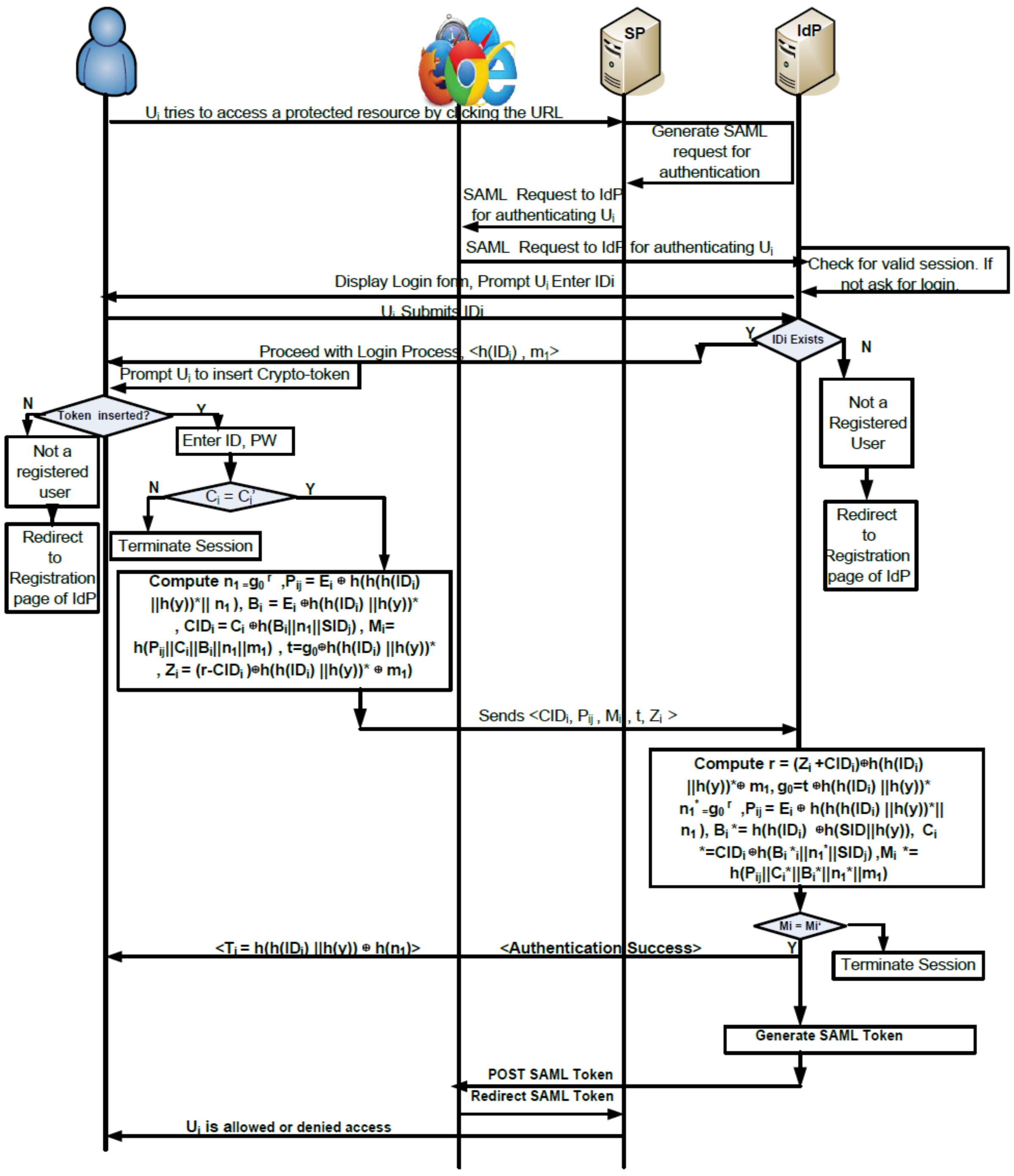
LA10: IdP computes $g_0 = t \oplus h(h(ID_i) ||h(y))^*$; $n_1^* = g_0^r$;

$B_i^*= h(h(IDi)||h(SID||h(y))$; $C_i^* = CID_i \oplus h(B_i^* || n_1^*|| SID_j)$;

LA11: IdP computes $M_i{}^*= h(P_{ij} || C_i^*|| B_i^*|| n_1^*|| m_1)$ and compares with the M_i received in the login message. If equal, IdP considers the authentication of U_i as successful.

LA12: IdP sends a response to the user along with the message

$T_i = h(h(ID_i) || h(y)) \oplus h(n_1)$. T_i is verified by the user to ascertain the freshness of the nonce n_1 and to ensure that it is receiving a communication from its honest communication partner. On successful authentication, server generates a SAML assertion containing the authentication response. IdP sends the token (SAML assertion) via HTTP POST to the ACS URL mentioned in the authentication request sent by the SP. The assertion is verified by the service provider to ascertain that the assertion was issued by the IdP. If so the SAML token is accepted and the user is allowed to access the resources. Otherwise the login request is rejected.

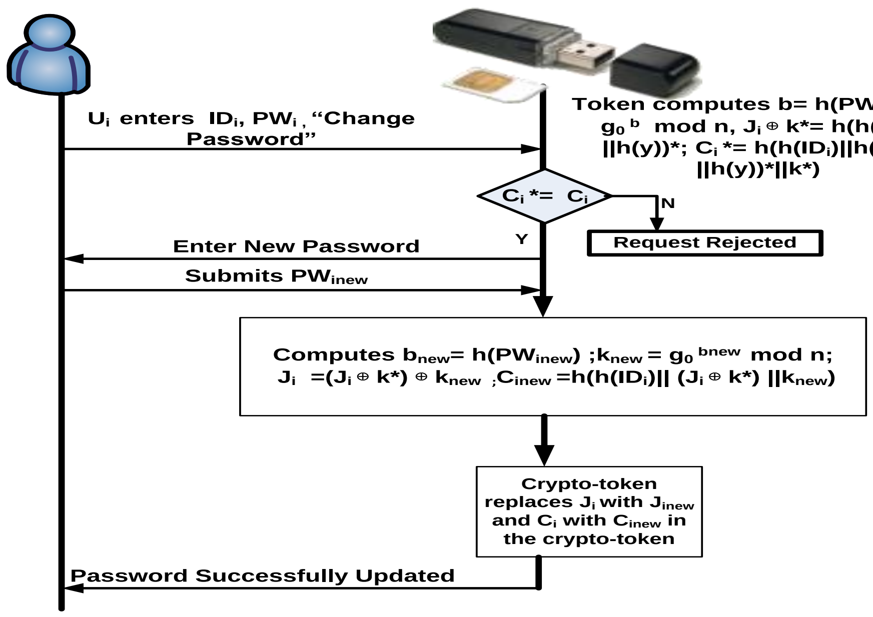


SP
IdP
U_i tries to access a protected resource by clicking the URL
Generate SAML request for authentication
SAML Request to IdP for authenticating U_i
SAML Request to IdP for authenticating U_i
Check for valid session. If not ask for login.
Display Login form, Prompt U_i Enter IDi
U_i Submits IDi
IDi Exists
Y
N
Proceed with Login Process, $<h(ID_i), m_1>$
Not a Registered User
Redirect to Registration page of IdP
Prompt U_i to insert Crypto-token
Token inserted?
N
Y
Not a registered user
Redirect to Registration page of IdP
Enter ID, PW
$C_i = C_i'$
N
Y
Terminate Session
Compute $n_1 = g_0^r$, $P_{ij} = E_i \oplus h(h(h(ID_i) || h(y))^* || n_1)$, $B_i = E_i \oplus h(h(ID_i) || h(y))^*$, $CID_i = C_i \oplus h(B_i || n_1 || SID_j)$, $M_i = h(P_{ij} || C_i || B_i || n_1 || m_1)$, $t = g_0 \oplus h(h(ID_i) || h(y))^*$, $Z_i = (r - CID_i) \oplus h(h(ID_i) || h(y))^* \oplus m_1$
Sends $<CID_i, P_{ij}, M_i, t, Z_i>$
Compute $r = (Z_i + CID_i) \oplus h(h(ID_i) || h(y))^* \oplus m_1$, $g_0 = t \oplus h(h(ID_i) || h(y))^*$ $n_1^* = g_0^r$, $P_{ij} = E_i \oplus h(h(h(ID_i) || h(y))^* || n_1)$, $B_i^* = h(h(ID_i) \oplus h(SID || h(y))$, $C_i^* = CID_i \oplus h(B_i{}^*{}_i || n_1^* || SID_j)$, $M_i^* = h(P_{ij} || C_i^* || B_i^* || n_1^* || m_1)$
Mi = Mi'
Y
Terminate Session
$<T_i = h(h(ID_i) || h(y)) \oplus h(n_1)>$
<Authentication Success>
Generate SAML Token
POST SAML Token
Redirect SAML Token
U_i is allowed or denied access

Figure 4.3 Login and Authentication Phase of Brokered Authentication Using Crypto-Token

Password Change Phase

Password change phase as shown in Figure 4.4 can be explained as follows. U_i inserts his Crypto-token into the system and enters his ID_i, PW_i. U_i requests for a password change.



U_i enters ID_i, PW_i, "Change Password"
Token computes $b= h(PW_i)$; $k^* = g_0^{\,b}$ mod n, $J_i \oplus k^*= h(h(ID_i)||h(y))^*$; $C_i^* = h(h(ID_i)||h(ID_i)||h(y))^*||k^*)$
$C_i^* = C_i$
N
Y
Request Rejected
Enter New Password
Submits PW_{inew}
Computes $b_{new}= h(PW_{inew})$;$k_{new} = g_0^{\,bnew}$ mod n; $J_i = (J_i \oplus k^*) \oplus k_{new}$;$C_{inew} = h(h(ID_i)|| (J_i \oplus k^*) ||k_{new})$
Crypto-token replaces J_i with J_{inew} and C_i with C_{inew} in the crypto-token
Password Successfully Updated

Figure 4.4 Password Change Phase of Brokered Authentication Using Crypto-Token

P1: Crypto-token computes $b = h(PW_i)$, $k^*= g_0^{\,b}$ mod n.

P2: Crypto-token computes $J_i \oplus k^* = h(h(ID_i)||h(y))^*$; Crypto-token computes $C_i^*= h(h(ID_i)|| h(h(ID_i)||h(y))^*|| k^*)$ and checks whether it is equal to the C_i stored in the Crypto-token. If equal, U_i is asked to enter the new password.

P3: U_i submits PW_{inew}. Crypto-token computes $b_{new} = h(PW_{inew})$,

$k_{new}= g_0^{\,bnew}$ mod n.

P4: Crypto-token computes $J_{inew} = J_i \oplus k^* \oplus k_{new}$;

$C_{inew} = h(h(ID_i)|| h(h(ID_i)||h(y))^*|| k_{new})$

P5: Crypto-token replaces J_i with J_{inew} and C_i with C_{inew} in the crypto-token.

Security Analysis

Security analysis is carried out to analyze the resistance of the protocol to various attacks. The proposed protocol is secure against the following attacks.

i. **Security against Replay Attack:** The proposed protocol uses nonce values to resist replay attacks. To successfully launch a replay attack, an adversary should be able to replay a previous login message {CID_i, M_i, P_{ij}, t, Z_i}. Here $Z_i = (r- CIDi) \oplus h(h(ID_i) || h(y)) * \oplus m_1$, contains the nonce $m_{1,}$ which is generated by the server and is unique to that session. Thus for every session, a unique nonce will be generated by the server and the server will be expecting that nonce in the message that is send by the user. This nonce m_1 retained by the server, will be used to calculate M_i. Assume that an adversary sends a previous login request containing Z_{ip} as $Z_{ip} = (r- CIDi) \oplus h(h(ID_i) || h(y)) * \oplus m_{1p}$. Now the nonce generated for the current session is m_{1c} and 'r*' = $(Z_i + CIDi) \oplus h(h(ID_i) || h(y)) * \oplus m_1$. The nonce n_1 will be calculated by the server as $n_1{*}= g_0^{r\,*}$ and this value will vary from the nonce send by Ui Also M_i will be calculated by the server as $M_i {*}= h(P_{ij} \| C_i{*} \| B_i{*} \| n_1{*} \| m_{1c})$ which will not match with the M_i in the reveiced replayed login request and the server will reject the request. Similarly, the response from the server ie. $T_i = h(h(ID_i) \| h(y)) \oplus h(n_1)$, contains the nonce generated by U_i, the freshness of which will be checked by U_i. These nonce values n_1 and m_1 are unique to each session. Also n_1 is not send across the communication channel in the plain text form and hence the possibility of capturing the session dependent nonce value and creating a valid login request message is also ruled out.

ii. **Security against Guessing Attack:** The values stored in the crypto-token includes $\{C_i, E_i, J_i, h(.), g_0, n\}$. Assume that an adversary guesses a password PW_{guess}. Then he computes $b_{guess} = h(PW_{guess})$ and $K_{guess} = g_0{}^{bguess}$ mod p. Now he computes $C_i = h(h(ID_i) \| h(h(ID_i) \| h(y)) \| K_{guess}$, $Ji = h(h(ID_i) \| h(y)) \oplus K_{guess}$. However, to check the guessed password, the adversary should know the server's secret key h(y), which is not stored in the crypto-token. Neither can it be extracted from the login request send across the communication channel nor can it be extracted from any of the parameters in the crypto-token. Similarly, even if the stored information E_i is revealed, $h(h(ID_i) \| h(y))$ is secure, since the adversary needs to know $B_i = h(h(ID_i) \| h(SID \| h(y)))$ to retrieve the value of $h(h(ID_i) \| h(y))$. Obviously the values g_0, n will not provide information required to generate a valid login request. Hence, from the values stored in the crypto-token, the adversary is not able to retrieve any information required to generate a valid login request.

Now, assume that the login message $\{CID_i, M_i, P_{ij}, t, Z_i\}$ send by U_i across the communication channel is listened to by the adversary. In the message P_{ij}, CID_i, M_i are all randomized by using nonce values n_1 and m_1 which is unique to each session. Also to generate n_1, the adversary should have the knowledge of g_0 and r which is not transmitted through the communication channel. Even if the adversary extracts g_0 from the crypto-token, he will not be able to calculate n_1 without knowing 'r'.

iii. **Security against Stolen Verifier Attack:** Server does not maintain a password verifier table which makes the protocol resistant to stolen verification attack.

iv. **Security against IdP Spoofing Attack:** To impersonate the IdP and to generate authentication response $T_i = h(h(ID_i) \| h(y)) \oplus h(n_1)$ on behalf of the IdP, the adversary should know the secret key 'y' of the IdP. This is not known to the adversary and cannot be extracted either from the crypto-token parameters $\{C_i, E_i, J_i, h(.), g_0, n\}$ or from the login request $\{CID_i, M_i, P_{ij}, t, Z_i\}$ message.

v. **Security against Malicious Insider Attack:** In the proposed scheme, user submits $b= h(PW_i)$, $k = g_0^b \bmod n$ to IdP instead of password in the plain text form. This prevents the IdP from knowing the correct password and hence even if the same password is used by the user to login to other servers, her credentials will be protected from an insider attack.

vi. **Security against User Impersonation Attack:** If an attacker tries to impersonate a legitimate user, he should be able to generate a valid login request on behalf of the user**.** In the discussed protocol if an attacker intercepts the login message $(CID_i, M_i, P_{ij}, t, Z_i)$ and tries to create a modified message $(CID_i^*, M_i^*, P_{ij}^*, t^*, Z_i^*)$ that is similar to a valid message, he will not succeed since the value of nonce 'n_1' and 'y'the secret key of server is not known to him. Also to calculate the values of CID_i^*, M_i^*, P_{ij}^* he should know $B_i = h(h(ID_i) \| h(SID \| h(y)))$. Even if he manages to extract the parameters stored in the crypto-token, to extract B_i from E_i, he should have the knowledge of user's password. The values stored in the crypto-token are created in such a way that even a valid user will not be able to impersonate another user by extracting values from his crypto-token.

vii. **Security against Crypto-token lost Attack:** If the adversary steals the crypto-token containing the parameters $(C_i, E_i, J_i, h(.), g_0, n)$, he cannot retrieve IdP's master secret 'y' or user's password from the stored value. To obtain the password, the adversary should be able to extract ‘k’ from Ji = $h(h(ID_i) || h(y)) \oplus k$, which is not possible without the knowledge of $h(h(ID_i) || h(y))$. It is not possible to extract $h(h(ID_i) || h(y))$ without the knowledge of B_i, which is not stored within the crypto-token. In addition, to retireve the password from $k = g_0^b$, where $b = h(PW_i)$, the adverasary should solve the discrete logarithm problem (DLP) as $b = DL_{g0}(k)$. Again the password is used in the hashed form which is irreversible.

viii. **Security against Denial-of-Service Attack:** Bombarding the server with invalid login request messages created by an adversary results in a denial-of-service attack (DoS). A DoS to a valid user can happen if an adversary who has got control over the server’s database, modifies the password or authentication credentials of the user stored by the server. Thus a DoS attack will prevent legitimate users from accessing resources they are authorized to access. In the proposed protocol, it is not possible for the attacker to create login requests without knowing the correct password and hence the first scenario will not work here. At the client side, password is checked for its correctness prior to creating a login request and before modifying the password. In the proposed protocol, a password/verifier table is not maintained by the server and hence the possibility of attacker modifying the password of a valid user is also ruled out.

ix. **User Anonymity Preserved:** The user will send the login request P_{ij} to the server S_j in each login session. To trace the user, the adversary will intercept the login message and attempt to extract ID_i from the message. The irreversibility property of one-way hash functions prevents the adversary from extracting ID_i from P_{ij}. More over each login message is made dynamic by including the nonce n_1 which is unique for each login session. Therefore, an adversary cannot identify the person making a login attempt and hence the proposed scheme preserves user anonymity.

x. **Independently Change Password:** The scheme allows the crypto-token holder to change the password without the mediation of the Service Provider or the Identity Provider. The Crypto-token verifies the legitimacy of the user before changing the password to prevent unauthorized users from easily changing the password if they obtain the crypto-token of some other registered user. Thus only valid user who knows the correct ID and password, corresponding to the crypto-token can change the password.

Efficiency Analysis

This section analyzes the efficiency of the proposed scheme in terms of the computational and the communication cost. It is assumed that ID_i, PW_i, g_o, n and nonce values are 128 bits long and the output of hash function is 256 bits long (SHA-2). Let T_h T_x, T_e and T_c denote the time complexity for hashing, XOR, exponentiation and concatenation operations respectively. In the protocol, the parameters stored in the crypto-token are C_i, E_i, J_i, g_0, n and the memory (E1) needed in the crypto-token is 1024 (3*256 +2 *128) bits. Communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication. The capacity of transmitting message (CID_i, M_i, P_{ij}, t, Z_i, M_1, T_j) is 1664 (6*256 + 1*128) bits. Total time taken by the user and IdP for executing all operations during registration is considered as the computation cost during registration phase (E3) and is equal to $7T_h + 2T_x + 1T_e + 5T_c$.Computation cost of the user (E4) and the IdP (E5) during login and authentication is the total time of all operations executed by the crypto-token and IdP during this phase. During authentication, the crypto-token performs 7 hash functions, 7 XOR, 2 exponentiations and 9 concatenations making E4 equal to $7T_h + 7T_x + 2T_e + 9T_c$. Similarly, E5 is $7T_h + 6T_x + 1T_e + 8T_c$. The computation cost of password changing phase (E6) which is the total time of all operations executed in this phase by the user is equal to $6T_h + 2T_x + 2T_e + 3T_c$. Comparison with other protocols are shown in Table 4.2.

Table 4.2 Comparison of Computational Efficiency with Other Protocols

	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Crypto-token based Protocol	1024 bits	1664 bits	$7T_h + 2T_x + 1T_e + 5T_c$	$7T_h + 7T_x + 2T_e + 9T_c$	$7T_h + 6T_x + 1T_e + 8T_c$	$6T_h + 2T_x + 2T_e + 3T_c$.
Choudhary et al. [2011]	1024 bits	1920 bits	$6T_h + 3T_x + 1T_e + 2T_c$	$10T_h + 2T_x + 1T_e + 3T_c$	$8T_h + 1T_x + 1T_e + 3T_c$.	$4T_h + 4T_x$
Jaidhar [2013]	1024 bits	1664 bits	$5T_h + 5T_x + 1T_e + 5T_c$	$6T_h + 2T_x + 9T_c + 2T_s + 1T_d$	$5T_h + 1T_x + 8T_c + 2T_d + 1T_s + 1T_e$	$3T_h + 2T_x + 3T_c$
Rui Jiang [2013]	768 bits	1152 bits	$4T_h + 1T_x + 1T_e + 1T_c$	$7T_h + 1T_x + 4T_c + 1T_d + 1T_e$	$7T_h + 5T_c + 1T_s + 1T_e$	$18T_h + 3T_x + 11T_c + 2T_s + 1T_d + 4T_e$

Results of comparison of computational efficiency demonstrate that the proposed protocol for brokered authentication using Crypto-token is comparable with similar protocols in terms of computation costs during registration, login & authentication and password change phase. In the case of the proposed crypto-token based protocol, the security of password send to the server during registration, is enhanced by obfuscating the password, by exponentiating the password to the power of the generator of a cyclic group. In this protocol, the research is exploiting the difficulty in solving discrete logarithm problem for cyclic groups of the form Z_n where 'n' is a very large odd prime number. Though these computations increase the computation cost of the protocol and affects total computational time, the protocol aids in providing enhanced security. In such a scenario, it can be mentioned in the Service Level Agreement between the IdP and the Service Providers that the authentication protocol provided by the IdP, provides secure authentication of users that requires a certain time period for execution. The authentication protocol can be adopted by those service providers to whom the time duration for execution of authentication protocol is agreeable.

Scyther Analysis

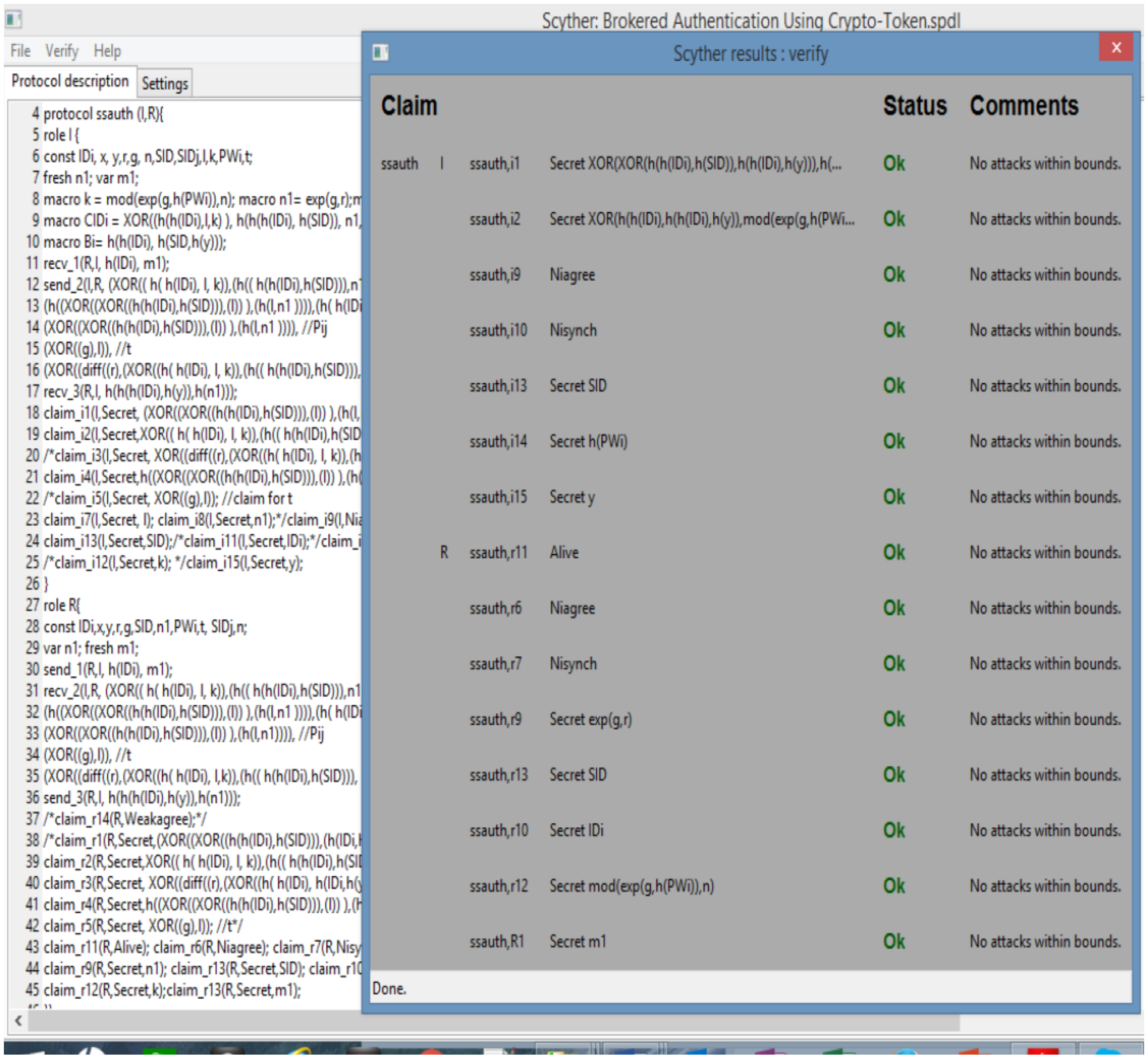
The formal analysis of protocol is done using Scyther tool. The strength of the protocol is verified using Scyther tool which ascertains the strength by evaluating the resistance of the protocol to various attacks. Scyther uses strand space model for formalizing logic and uses Dolev-Yao model for modelling the network, which caters to the requirement of a mathematical approach for validating the protocol. The analysis results of login phase are shown in Figure 4.5**.**

The protocol is written in SPDL as follows:

```
//Login & Authentication Phase of Brokered Authentication Protocol Using
   Crypto-Token
const  exp: Function;  hashfunction h;
const XOR: Function; const h1: Function; const diff: Function; const
   mod:Function;
protocol ssauth (I,R){
```

```
role I {
const IDi, x, y,r,g, n,SID,SIDj,l,k,PWi,t;
fresh n1; var m1;
macro k = mod(exp(g,h(PWi)),n); macro n1= exp(g,r);macro l
   =h(h(IDi),h(y));
macro CIDi = XOR((h(h(IDi),l,k) ), h(h(h(IDi), h(SID)), n1, SIDj));
macro Bi= h(h(IDi), h(SID,h(y)));
recv_1(R,I, h(IDi), m1);
send_2(I,R, (XOR(( h( h(IDi), l, k)),(h(( h(h(IDi),h(SID))),n1, (SIDj), //CIDi
(h((XOR((XOR((h(h(IDi),h(SID))),(l)) ),(h(l,n1 )))),(h( h(IDi), l, k)),
   (h(h(IDi),h(SID))),n1), //Mi
(XOR((XOR((h(h(IDi),h(SID))),(l)) ),(h(l,n1 )))), //Pij
(XOR((g),l)), //t
(XOR((diff((r),(XOR((h( h(IDi), l, k)),(h(( h(h(IDi),h(SID))), n1, (SIDj)
   )))))),l),m1)) ))))); //Zi
recv_3(R,I, h(h(h(IDi),h(y)),h(n1)));
claim_i1(I,Secret, (XOR((XOR((h(h(IDi),h(SID))),(l)) ),(h(l,n1 ))))); //claim
   for pij
claim_i2(I,Secret,XOR(( h( h(IDi), l, k)),(h(( h(h(IDi),h(SID))),n1, (SIDj)
   )))); //claim for CIDi
claim_i3(I,Secret, XOR((diff((r),(XOR((h( h(IDi), l, k)),(h((
   h(h(IDi),h(SID))), n1, (SIDj) )))))),l,m1)); //claim for Zi
claim_i4(I,Secret,h((XOR((XOR((h(h(IDi),h(SID))),(l)) ),(h(l,n1)))),(h(
   h(IDi), l, k)), (h(h(IDi),h(SID))), n1)); //claim for Mi
claim_i5(I,Secret, XOR((g),l)); //claim for t
claim_i7(I,Secret, l);
   claim_i8(I,Secret,n1);claim_i9(I,Niagree);claim_i10(I,Nisynch);
claim_i13(I,Secret,SID);claim_i11(I,Secret,IDi);claim_i14(I,Secret,h(PWi));
claim_i12(I,Secret,k); claim_i15(I,Secret,y);
}
role R{
```

```
const IDi,x,y,r,g,SID,n1,PWi,t, SIDj,n;
var n1; fresh m1;
send_1(R,I, h(IDi), m1);
recv_2(I,R, (XOR(( h( h(IDi), l, k)),(h(( h(h(IDi),h(SID))),n1, (SIDj), //CIDi
(h((XOR((XOR((h(h(IDi),h(SID))),(l))  ),(h(l,n1  )))),(h(  h(IDi),  l,  k)),
   (h(h(IDi),h(SID))), n1), //Mi
(XOR((XOR((h(h(IDi),h(SID))),(l)) ),(h(l,n1)))), //Pij
(XOR((g),l)), //t
(XOR((diff((r),(XOR((h(  h(IDi),  l,k)),(h((  h(h(IDi),h(SID))),  n1,  (SIDj)
   )))))),l),m1)) ))))); //Zi
send_3(R,I, h(h(h(IDi),h(y)),h(n1)));
claim_r14(R,Weakagree);
claim_r1(R,Secret,(XOR((XOR((h(h(IDi),h(SID))),(h(IDi,h(y))))),(h(h(IDi,h
   (y)),n1))))); //claim for pij
claim_r2(R,Secret,XOR((  h(  h(IDi),  l,  k)),(h((  h(h(IDi),h(SID))),n1,  (SIDj)
   )))); //claim for CIDi
claim_r3(R,Secret,   XOR((diff((r),(XOR((h(   h(IDi),   h(IDi,h(y)),   k)),(h((
   h(h(IDi),h(SID))), n1, (SIDj) )))))),h(IDi,h(y),m1))); //claim for Zi
claim_r4(R,Secret,h((XOR((XOR((h(h(IDi),h(SID))),(l))        ),(h(l,n1)))),(h(
   h(IDi), l, k)), (h(h(IDi),h(SID))), n1)); //claim for Mi
claim_r5(R,Secret, XOR((g),l)); //t
claim_r11(R,Alive); claim_r6(R,Niagree); claim_r7(R,Nisynch);
claim_r9(R,Secret,n1); claim_r13(R,Secret,SID); claim_r10(R,Secret,IDi);
claim_r12(R,Secret,k);claim_r13(R,Secret,m1);
                              }}
```



```
protocol ssauth (I,R){
role I {
const IDi, x, y,r,g, n,SID,SIDj,I,k,PWi,t;
fresh n1; var m1;
macro k = mod(exp(g,h(PWi)),n); macro n1= exp(g,r);m
macro CIDi = XOR((h(h(IDi),I,k) ), h(h(h(IDi), h(SID)), n1,
macro Bi= h(h(IDi), h(SID,h(y)));
recv_1(R,I, h(IDi), m1);
send_2(I,R, (XOR(( h( h(IDi), I, k)),(h(( h(h(IDi),h(SID))),n1
(h((XOR((XOR((h(h(IDi),h(SID))),(I)) ),(h(I,n1 )))),(h( h(IDi
(XOR((XOR((h(h(IDi),h(SID))),(I)) ),(h(I,n1 )))), //Pij
(XOR((g),I)), //t
(XOR((diff((r),(XOR((h( h(IDi), I, k)),(h(( h(h(IDi),h(SID))),
recv_3(R,I, h(h(h(IDi),h(y)),h(n1)));
claim_i1(I,Secret, (XOR((XOR((h(h(IDi),h(SID))),(I)) ),(h(I,
claim_i2(I,Secret,XOR(( h( h(IDi), I, k)),(h(( h(h(IDi),h(SID
/*claim_i3(I,Secret, XOR((diff((r),(XOR((h( h(IDi), I, k)),(h
claim_i4(I,Secret,h((XOR((XOR((h(h(IDi),h(SID))),(I)) ),(h(
/*claim_i5(I,Secret, XOR((g),I)); //claim for t
claim_i7(I,Secret, I); claim_i8(I,Secret,n1);*/claim_i9(I,Nia
claim_i13(I,Secret,SID);/*claim_i11(I,Secret,IDi);*/claim_i
/*claim_i12(I,Secret,k); */claim_i15(I,Secret,y);
}
role R{
const IDi,x,y,r,g,SID,n1,PWi,t, SIDj,n;
var n1; fresh m1;
send_1(R,I, h(IDi), m1);
recv_2(I,R, (XOR(( h( h(IDi), I, k)),(h(( h(h(IDi),h(SID))),n1
(h((XOR((XOR((h(h(IDi),h(SID))),(I)) ),(h(I,n1 )))),(h( h(IDi
(XOR((XOR((h(h(IDi),h(SID))),(I)) ),(h(I,n1)))), //Pij
(XOR((g),I)), //t
(XOR((diff((r),(XOR((h( h(IDi), I,k)),(h(( h(h(IDi),h(SID))),
send_3(R,I, h(h(h(IDi),h(y)),h(n1)));
/*claim_r14(R,Weakagree);*/
/*claim_r1(R,Secret,(XOR((XOR((h(h(IDi),h(SID))),(h(IDi,h
claim_r2(R,Secret,XOR(( h( h(IDi), I, k)),(h(( h(h(IDi),h(SID
claim_r3(R,Secret, XOR((diff((r),(XOR((h( h(IDi), h(IDi,h(y
claim_r4(R,Secret,h((XOR((XOR((h(h(IDi),h(SID))),(I)) ),(h
claim_r5(R,Secret, XOR((g),I)); //t*/
claim_r11(R,Alive); claim_r6(R,Niagree); claim_r7(R,Nisy
claim_r9(R,Secret,n1); claim_r13(R,Secret,SID); claim_r10
claim_r12(R,Secret,k);claim_r13(R,Secret,m1);
```

Claim				Status	Comments
ssauth	I	ssauth,i1	Secret XOR(XOR(h(h(IDi),h(SID)),h(h(IDi),h(y))),h(...	Ok	No attacks within bounds.
		ssauth,i2	Secret XOR(h(h(IDi),h(h(IDi),h(y)),mod(exp(g,h(PWi...	Ok	No attacks within bounds.
		ssauth,i9	Niagree	Ok	No attacks within bounds.
		ssauth,i10	Nisynch	Ok	No attacks within bounds.
		ssauth,i13	Secret SID	Ok	No attacks within bounds.
		ssauth,i14	Secret h(PWi)	Ok	No attacks within bounds.
		ssauth,i15	Secret y	Ok	No attacks within bounds.
	R	ssauth,r11	Alive	Ok	No attacks within bounds.
		ssauth,r6	Niagree	Ok	No attacks within bounds.
		ssauth,r7	Nisynch	Ok	No attacks within bounds.
		ssauth,r9	Secret exp(g,r)	Ok	No attacks within bounds.
		ssauth,r13	Secret SID	Ok	No attacks within bounds.
		ssauth,r10	Secret IDi	Ok	No attacks within bounds.
		ssauth,r12	Secret mod(exp(g,h(PWi)),n)	Ok	No attacks within bounds.
		ssauth,R1	Secret m1	Ok	No attacks within bounds.

Done.

Figure 4.5 Scyther Analysis of Brokered Authentication Using Crypto-Token

Formal Analysis using Scyther

To perform the formal security analysis, this section focuses on analyzing the parameters IDi, PW_i, k, y, SID, P_{ij}, CID_i, Z_i, M_i, t, l, n_1 which are used in the proposed authentication scheme, to check their vulnerability to attacks. There are various claims made as part of the security analysis and these claims are validated by executing and analyzing the proposed scheme using

Scyther. The “No attack” results shown in Figure 4.5 proves that Scyther validates all the claims made as part of security analysis.

Claim 1: The proposed scheme is designed to ensure the secrecy of the user ID viz. ID_i, throughout the registration and authentication process.

The user ID, ID_i is submitted during registration in the hashed form to IdP. This is used along with password and secret key ‘y’ of IdP to generate the secret parameters to be stored in the Crypto-token. During the authentication process, IDi in hashed form is submitted to server and server verifies whether the user is a registered user. If so the user is prompted to proceed with the login process and the crypto-token generates the login request (CID_i, M_i, P_{ij}, t, Z_i). In the login request, all the parameters include the hash of IDi which is concatenated and XOR-ed with other parameters, which makes it difficult to extract IDi. The claim that ID_i is safe is verified by Scyther. A “No Attacks within Bounds” results for the Secrecy claim, claim_i11(I, Secret, IDi) is indicative of the fact that whenever a run of the I role is completed with an honest communication partner, the value ID_i transmitted by I in the run will not be revealed to the adversary. Therefore, it can be said that the claim secret ID_i of the role I holds.

Claim 2: The proposed scheme is designed to ensure the secrecy of the variant of password ‘k’ throughout the registration and authentication process.

The password is never transmitted in the plaintext form either to the IdP or to the cloud server. It is converted into a modified form ‘k’, by finding the hash of the password viz. ‘b’ and then raising g_0 (generator of a cyclic group) to the power of ‘b’. Now to obtain the password from ‘k’, we need to solve the discrete logarithm problem. During authentication process, password is used to generate login request. It is not sent to cloud service provider, but it is used to check the stored password in the crypto-token and to calculate the value h(h(IDi)||h(y)) which is required to generated the login request (CID_i, M_i, P_{ij}, t, Z_i). Also the password is not stored anywhere other than in the crypto-token. Scyther results validate the claim that ‘k’ remains a secret.

Claim 3: The proposed scheme requires ‘y’ to be a secret

‘y’ is the secret key of the IdP which is used in its hashed form to compute the parameters to be stored in the crypto-token . ‘y’ is also used to generate the values r, g_0, B_i* which are used to verify the login request send by the user during the login and authentication phase. During the transmission of the parameters from the IdP to the user as well as during the transmission of login request from the user to the IdP, ‘y’ should not be revealed to the adversary and the secrecy claim for ‘y’ is validated by Scyther.

Claim 4: The proposed scheme requires that P_{ij} is secret

Pij is one among the parameters in the login request sent from the user to the cloud server, which contains the user ID, the secret key of the IdP and the nonce generated by the user (crypto-token). Pij should remain a secret during transmission. A “No Attacks within Bounds” results for the Secrecy claim, claim_i1(I, Secret, (XOR((XOR((h(h(IDi),h(SID))),(l))),(h(l,n1))))) is indicative of the fact that whenever a run of the I role is completed with a honest communication partner, the value P_{ij} transmitted by I in the run will not be revealed to the adversary. Therefore, it can be said that the claim for Secrecy of P_{ij} by the role I holds.

Claim 5: The proposed scheme requires that CID_i is secret

CID_i is one among the parameters in the login request sent from the user to the server, which is required to generate the value M_i * by the server. M_i * is used to verify the authenticity of the user. CID_i contains the user ID, the secret key of the IdP the nonce n_1 generated by the user (crypto-token) and the service provider Id, SIDj. CID_i should remain a secret during transmission. The claim that CID_i is safe is verified and Scyther validates that Secrecy of CID_i holds.

Claim 6: The proposed scheme requires that Z_i is secret

Z_i is a parameter in the login request sent from the user to the server and it is used by the server to retrieve the value of random number 'r', which is used to generate the nonce 'n_1*'. 'n_1' in turn is required to verify the authenticity of the user. Z_i should not be revealed to the adversary during transmission. The claim that Z_i is secret is verified and validated by Scyther.

Claim 7: The proposed scheme requires that M_i is secret

M_i is a parameter in the login request, which is used by the server to verify the authenticity of the user. The server re-calculates the value of M_i as $M_i^* = h(Pij||C_i^*||B_i^*||n_1^*|| m_1)$ and compares with the received M_i and an equality of M_i and M_i^* ascertains the authenticity of the user. M_i should not be revealed to the adversary during transmission. The secrecy claim for M_i is verified and validated by Scyther.

Claim 8: The proposed scheme requires that l is secret

l is written as a macro since it is repeatedly used in the calculation of various parameters. l is calculated as h(h(IDi)||h(y)) and it is used in calculating the values (J_i, C_i, E_i) stored in the crypto-token as well as in the calculation of the parameters (P_{ij}, B_i, CID_i, Z_i, t, M_i) required to generate the login request to be send to the server. l should remain safe during transmission and a “No Attacks within Bounds” results for the Secrecy claim, claim_i7(I, Secret, l) is indicative of the fact that the claim for Secrecy of l holds.

Claim 9 The proposed scheme requires that the nonce n_1 is secret

n_1 is the nonce generated by the crypto-token and is unique to each session. n_1 is used to calculate the parameters P_{ij}, CID_i, M_i included in the login request. Since the value of n_1 is unique to each session, the login request will also be different for each session. n_1 should not be revealed to the adversary and Scyther validates the claim that n_1 is safe.

Claim 10: The scheme assures Niagree between the user (crypto-token) and the cloud server

Niagree claim enforces that the sender and the receiver agree upon the values of variables exchanged during the running of the proposed scheme. During the operation of the proposed scheme, the user and server can send data safely and the correctness of the claim is justified by the analysis results.

Claim 11: The proposed scheme holds Synchronization during the authentication process

Ni-Synch or Non-Injective Synchronization property requires that the corresponding send and receive events (1) happened in the correct order and (2) have the same contents. Ni-Synch is valid if all actions before the claim are performed as per the description of the proposed scheme. The proposed protocol satisfies this claim as indicated by the result of Scyther analysis.

4.1.4 A Mobile Based User Authentication Scheme without Verifier Table for Cloud Based Services

The process flow of the registration and authentication stages of the mobile-token based protocol are depicted in Figure 4.1. The proposed architecture for brokered authentication includes four participants' viz. a Registration Server (RS), an Authentication Server (AS), Service Provider's (SP's) and users'. The RS and AS are in the same trusted domain and together they provide the functionality of the Identity Provider (IdP). It is assumed that all the service providers are registered with the IdP and they are reliable. The users and SPs, needs to register with the IdP. A user who attempts to access the services of a SP without registering at IdP, will be redirected by the SP to the IdP as shown in Figure 3.2 in section 3.1.2 of chapter 3. After registration, the user can access the services of different Service providers (SP).

Phases of the Proposed Protocol: The registration and the authentication phase of this protocol is executed by the IdP and the authentication response is send as a SAML assertion to the service provider who re-directed the user to the IdP**.** The proposed protocol consists of three phase' s viz., User registration phase, Login & Authentication phase and Password change phase. The mobile should have Internet connectivity during registration and authentication phase. The notations used are listed in Table 4.3.

Table 4.3 Notations Used in the Protocol (Brokered & Mobile-Token)

ID_i, PW_i	Identity, Password of user U_i.
S	Secret key of IdP
Rand	Random number
r_1, r_2	Nonce values
$h(.)$, $\oplus$, $\|$	hash function, XOR operation, Concatenation Operation

Registration Phase

During the registration process, user submits his credentials to IdP. IdP generates a set of security parameters using the submitted the credentials and his key value. IdP stores the security parameters within a secret file which is downloaded and stored in to a secure location within the user's mobile phone. The secret file is encrypted using the password (PBE) of the user which ensures that only a valid user will be able to store the token into his mobile phone and use the same to avail secure access to the cloud services.

The registration process illustrated in Figure 4.6 can be explained as follows:

R1: The user U_i clicks the "Register" button at the SP's web site. SP redirects U_i to the registration page of the IdP.

R2: IdP prompts U_i to download a mobile app into his smart phone from the URL specified by IdP. The app contains a scanning application to scan the QR codes generated by the IdP. The mobile app provides support for completing the registration and authentication process and for changing the user password.

R3: IdP prompts U_i to submit her identity ID_i and PW_i. The user submits $h(ID_i)$ and $h(PW_i.)$ to IdP through a secure communication channel.

R4: IdP checks whether $h(ID_i)$ already exists in its user table. If so U_i is prompted to select a new ID_i.

R5: IdP generates a random number ‘rand’ and computes the parameters $V_{i,}$ $K_{i,}$ M_i as,

$V_i = h(h(ID_i)\|\ h(PW_i)) \oplus$ rand, $K_i = h(PW_i) \oplus h(h(ID_i)\|\ h(S))$,

$M_i = K_i \oplus h(h(ID_i)\ ||\ h(PW_i))$

Here ‘S’ is the server’s (IdP) secret key which is a unique value and ‘rand’ is the random value generated by the IdP for each user and is used only during the registration phase of the user.

R6: IdP creates a file containing the authentication parameters $V_{i,}$ $K_{i,}$ M_i and the file is encrypted by Password Based Encryption (PBE) using password of U_i and a salt value.

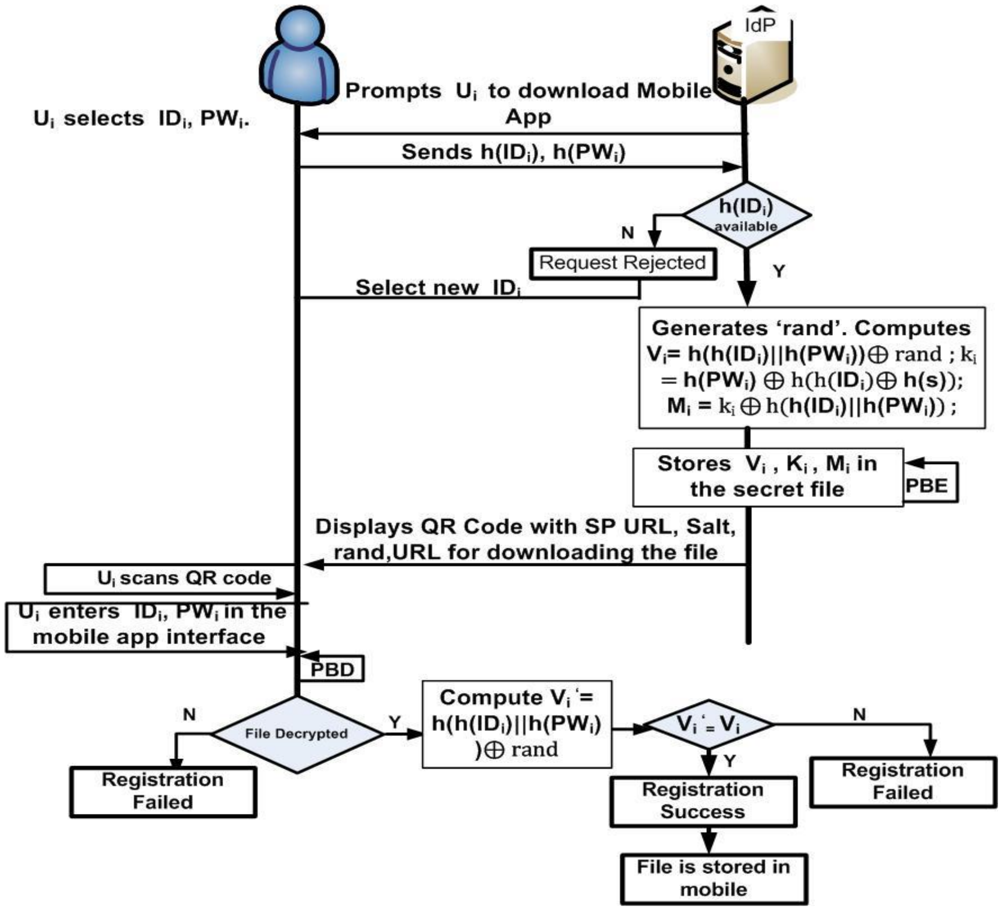
R7: IdP generates a QR code embedding ‘rand’, Service Provider URL, Salt and the link for downloading the secret file.

R8: The QR code will be displayed on the client PC and the user will have to continue the registration process using her mobile phone.

R9: The mobile part of the registration phase commences from this step. During the execution of this phase, the mobile should have internet connection. The app, invokes the scanning application, and the user can scan the QR code. The app will retrieve ‘rand’, salt, the service provider URL and the link to download the secret file.

R10: The secret file, which is in an encrypted form, will be downloaded. The user will be prompted to enter ID_i, PW_i. The app attempts to decrypt the file using password given as input by the user and the salt value read from the QR code. If the password does not match, then the registration is considered unsuccessful and the mobile token will be removed from the phone. If the decryption is successful, the secret file contents can be accessed.

R12: When the user touches the register button in the interface provided in the mobile app, mobile app, calculates $V_i' = h(h(ID_i') \mid\mid h(PW_i')) \oplus$ rand where ID_i and PW_i are values entered by the user U_i via the mobile interface and 'rand' is the value read from the QR code. The calculated V_i' is compared with the V_i stored in the mobile token and if equal, the registration is considered successful. The secret file will be permanently stored into a secure location within the phone's storage and will serve as a mobile token. A communication regarding successful verification is transmitted from the mobile phone to the IdP and U_i will get the "Registration Successful" message in both the phone and the IdP's page. Once the registration is successfully done, the user U_i can access the services of different service providers. IdP stores the user identity $h(ID_i)$ and profile information such as first name, last name and email address in its user table. If registration is not successful, the downloaded secret file will be deleted from the phone.



IdP
Prompts U_i to download Mobile App
U_i selects ID_i, PW_i.
Sends $h(ID_i)$, $h(PW_i)$
$h(ID_i)$ available
N
Y
Request Rejected
Select new ID_i
Generates 'rand'. Computes $V_i = h(h(ID_i)||h(PW_i)) \oplus rand$; $k_i = h(PW_i) \oplus h(h(ID_i) \oplus h(s))$; $M_i = k_i \oplus h(h(ID_i)||h(PW_i))$;
Stores V_i , K_i , M_i in the secret file
PBE
Displays QR Code with SP URL, Salt, rand,URL for downloading the file
U_i scans QR code
U_i enters ID_i, PW_i in the mobile app interface
PBD
File Decrypted
N
Y
Registration Failed
Compute $V_i' = h(h(ID_i)||h(PW_i)) \oplus rand$
$V_i' = V_i$
N
Y
Registration Success
Registration Failed
File is stored in mobile

Figure 4.6 Registration Phase of Brokered-Authentication Using Mobile-Token

Login and Authentication Phase

As shown in Figure 4.7, the user U_i via his browser attempts to access a protected resource of a Service Provider (SP). It is assumed that, the browser at this point does not have an established session with the SP. On receiving the request from U_i, SP generates a SAML assertion containing an authentication request and redirects the user (HTTP Redirect) to the IdP. The authentication request contains the information regarding the SP who initiated the request and the ID of the SSO service (IdP). The request also contains the assertion Consumer service (ACS) URL to which the response should be sent. IdP checks for a valid session with the browser by verifying whether a session cookie created in the IdP's domain is available in the browser or not. If there is no existing session between the browser and the IdP, then IdP generates a login session and authenticates the user by executing the authentication phase, as illustrated in Figure 4.7. The user uses his password and the parameters stored within the mobile token deployed in the mobile phone, to authenticate himself to the IdP. The procedure can be explained as follows:

L1: Identity Provider (IdP) displays the login page and prompts the user to enter user's identity (ID_i) and Password (PW_i). The values are sent over the communication channel as $h(ID_i)$ and $h(PW_i)$). IdP checks whether $h(ID_i)$ exists in its database or not. If there is no entry, then U_i is not a registered user. Otherwise IdP calculates:

$K_{idp} = h(PW_i) \oplus h(h(ID_i) \| h(S))$, $C_1 = h(h(ID_i) \| h(PW_i))$,

$B_1 = h\ (K_{idp} \| h(C_1 \oplus r_1)\ )$ where 'r_1' is a nonce value generated by IdP.

L2: The random nonce r_1 and challenge B_1 ie $< B_1, r_1>$ is send to the user U_i, via a QR code. The mobile app invokes the scanning application and scans the QR code to retrieve $< B_1, r_1>$.

L3: The mobile app computes $C_2 = M_i \oplus K_i$, where M_i and K_i are values stored in the mobile token. App computes $B_1' = h(K_i \| h(C_2 \oplus r_1))$ where 'r_1' is retrieved from the QR code and checks whether

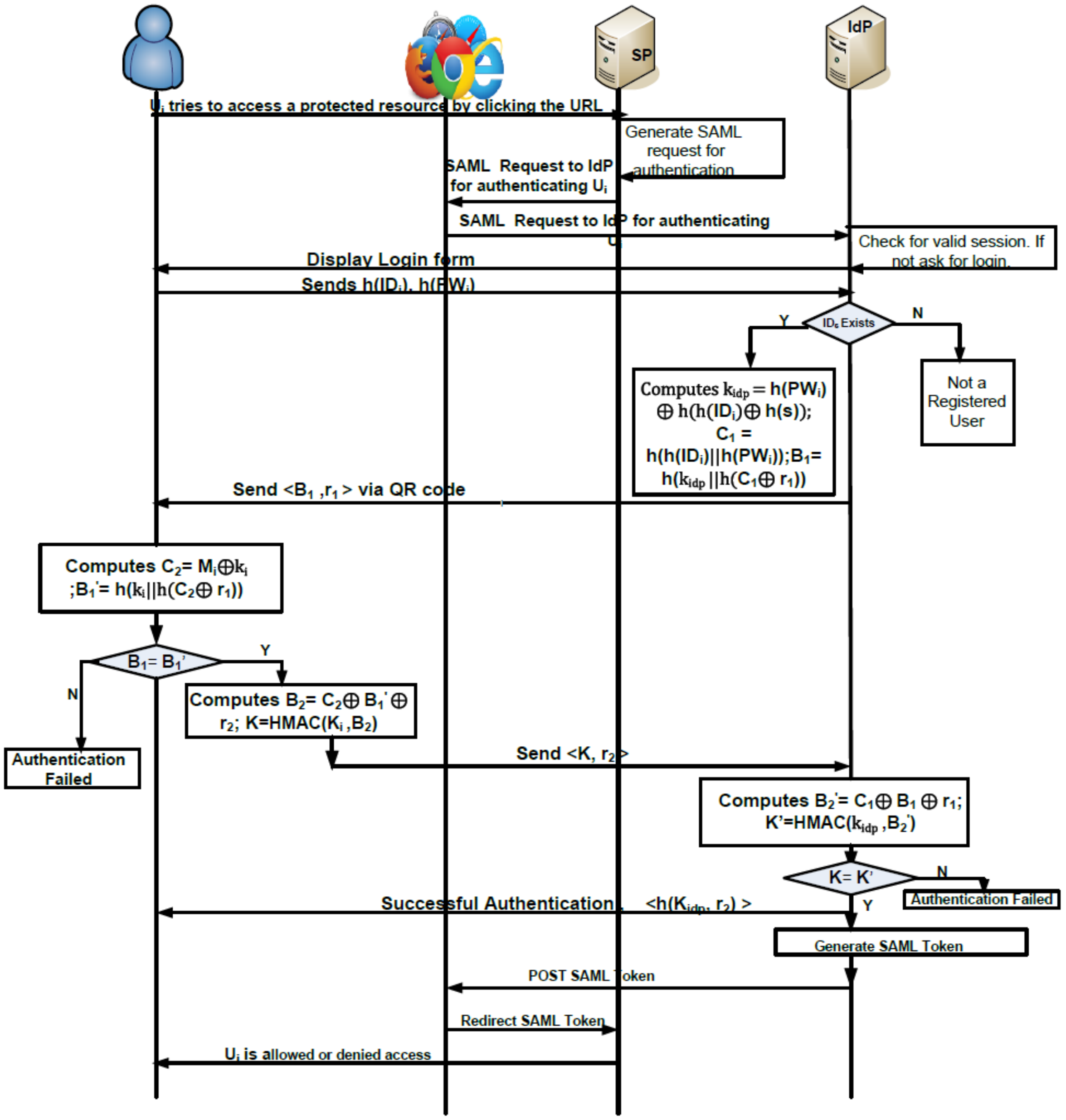

SP
IdP
U_i tries to access a protected resource by clicking the URL
Generate SAML request for authentication
SAML Request to IdP for authenticating U_i
SAML Request to IdP for authenticating U_i
Check for valid session. If not ask for login.
Display Login form
Sends h(ID_i), h(PW_i)
ID_i Exists
Y
N
Not a Registered User
Computes k_idp = h(PW_i) ⊕ h(h(ID_i)⊕ h(s)); C_1 = h(h(ID_i)||h(PW_i));B_1= h(k_idp ||h(C_1⊕ r_1))
Send <B_1 ,r_1 > via QR code
Computes C_2= M_i⊕k_i ;B_1'= h(k_i||h(C_2⊕ r_1))
B_1= B_1'
Y
N
Authentication Failed
Computes B_2= C_2⊕ B_1'⊕ r_2; K=HMAC(K_i ,B_2)
Send <K, r_2>
Computes B_2'= C_1⊕ B_1 ⊕ r_1; K'=HMAC(k_idp ,B_2')
K= K'
N
Y
Authentication Failed
Successful Authentication , <h(K_idp, r_2) >
Generate SAML Token
POST SAML Token
Redirect SAML Token
U_i is allowed or denied access

Figure 4.7 Login and Authentication Phase of Brokered Authentication Using Mobile-Token

B_1'= Challenge B_1, received from IdP. If so, mobile app considers the message as being received from an authenticated source and continues with the following steps. This step is included to avoid the possibility of phishing attack, since only the server which holds the secret key h(S) of IdP will be able to generate this message.

L4: Mobile app generates a nonce r_2 and computes $B_2 = C_2 \oplus B_1' \oplus r_2$, $K = $ HMAC (K_i, B_2) where K_i is the value stored in the mobile token. Mobile app sends <K, r_2>to IdP.

HMAC is a keyed hash function and K_i serves as the key which is used to encrypt the message B_2.

L5: IdP on receiving the message K, computes $B_2' = C_1 \oplus B_1 \oplus r_2$ and
$K' = $ HMAC (K_{idp}, B_2').

IdP recalculates the HMAC value by using K_{idp} as the key and B_2' as the message. Since key K_i which is equal to K_{idp}, is known only to the user, the value K would have been calculated only by the user.

L6: IdP checks whether K'is equal to the received K. If equal IdP considers the user as authenticated and that the integrity of message is maintained. Otherwise the login request is rejected. IdP sends a successful authentication message with < $h(K_{idp} \| r_2)$> to U_i . U_i verifies the freshness of the nonce and compares $h(K_i \| r_2)$ with the received < $h(K_{idp} \| r_2)$> and ascertains that the response message is from a honest server.

L7: IdP sends a response to the user and on successful authentication generates a SAML assertion containing the authentication response. IdP sends the token (SAML assertion) via HTTP POST to the ACS URL mentioned in the authentication request sent by the SP. The signed assertion which is received by the service provider is verified to ascertain that the assertion was issued by the IdP. If so the SAML token is accepted and the user is allowed to access the resources. Otherwise the login request is rejected.

Password Change Phase

The password change phase illustrated in Figure 4.8 is invoked when the user wishes to change his password without the intervention of the IdP or the SP and is carried out as follows:

P1: User enters his identity (ID_i) and Password (PW_i) and executes the "Password Change" request. The mobile app computes $M_i' = K_i \oplus h(h(ID_i)||h(PW_i))$ and checks if it is equal to stored M_i in the mobile-token. If equal, the mobile app prompts the user to enter the new password 'PW_{inew}'. Otherwise the "password change" request is rejected.

P2: The app calculates $K_{inew} = h(PW_{inew}) \oplus K_i \oplus h(PW_i)$. Then the app computes $M_{i\ new} = K_{inew} \oplus h(h(ID_i)||h(PW_i))$, $V_{i\ new} = h(h(ID_i)||h(PW_{inew})) \oplus V_i \oplus h(h(ID_i)||h(PW_i))$ and the app replaces the existing values in the mobile-token with the new values.

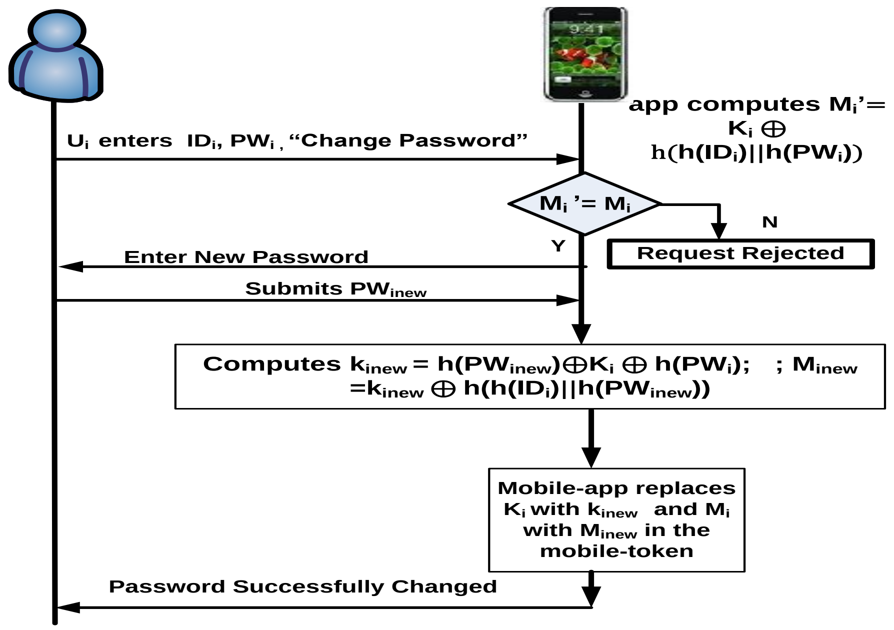

app computes $M_i' = K_i \oplus h(h(ID_i)||h(PW_i))$
U_i enters ID_i, PW_i, "Change Password"
$M_i' = M_i$
N
Y
Request Rejected
Enter New Password
Submits PW_{inew}
Computes $k_{inew} = h(PW_{inew}) \oplus K_i \oplus h(PW_i)$; ; $M_{inew} = k_{inew} \oplus h(h(ID_i)||h(PW_{inew}))$
Mobile-app replaces K_i with k_{inew} and M_i with M_{inew} in the mobile-token
Password Successfully Changed

Figure 4.8 Password Change Phase of Brokered Authentication Using Mobile-Token

Security Analysis

i. Security against Guessing Attack: The aim of this attack is to find out the password of the user. One way in which this attack can be launched by an adversary is by stealing the verifier table at the server and trying to guess the passwords corresponding to the stored hashes. This scenario is not possible in the proposed scheme, since the adversary does not maintain a verifier table.

Another way of launching the dictionary attack is by trying to guess the password from the parameters stored in the mobile token. Assume that the adversary manages to get the mobile token containing $<V_i, K_{i,} M_i, h(.)>$. Now he guesses a password PW_{guess} and computes $K_i = h(PW_{guess.}) \oplus h(h(ID_i)||$ $h(S))$. Then he should have the knowledge of the server's secret key 'S' to verify whether he is getting the K_i stored in the token. Again, if the adversary computes $V_i' = h(h(ID_i)|| h(PW_{guess.})) \oplus rand$, without the knowledge of 'rand' which is not stored within the mobile-token, he will not be able to verify the guessed password.

ii. Security against Replay Attack: The proposed authentication protocol uses nonce values 'r_1' and 'r_2'generated by the server and user to address the issue of replay attack. The nonce values are unique to a particular session and is included in the messages exchanged between the user and the server. Assume that an adversary intercepts the message B_1, K exchanged between the user and the server and tries to replay it a later time. However the attack will fail because $B_1 = h(K_{idp} || h(C_1 \oplus r_1))$ and $K = HMAC(K_i , B_2)$ are computed using the nonce values 'r_1' and 'r_2' which are checked for freshness by the receiver.Similarly, the nonce 'r_2' generated by U_i is send back in the response message $h(K_{idp} \| r_2)$, the freshness of which is verified by U_i.

iii. **Security against Server Spoofing Attack:** For an adversary to masquerade as a legal service provider, he must be able to generate the challenge $B_1 = h(K_{idp} \| h(C_1 \oplus r_1)))$. To generate K_{idp} in the challenge he should have the knowledge of user's password and server's secret 'S' which is not known to the adversary.

iv. **Security against Insider and Stolen Verifier Attack:** Insider attack is launched by an administrator who deliberately leaks secret information resulting in security flaws of the authentication scheme. In the proposed scheme both during registration and login phase, the $h(PW_i)$ is send to the server. Deriving the password from $h(PW_i)$ is infeasible due to irreversibility property of hash functions. The proposed scheme does not maintain any verifier table and hence it is secure against stolen verifier attack.

v. **Security against User Impersonation Attack:** If an adversary attempts to impersonate a valid user, he should be able to generate $K = HMAC(K_i , B_2)$. K_i is the value contained in the user's mobile token and the generation of K_i requires the knowledge of the server's secret key 'S'. Again the computation of B_2 requires the knowledge of C_2 which is calculated using the parameters M_i and K_i in the mobile token, and is never transitted across the communication channel in the plain text form. Without access to the contents of the mobile token and the knowledge of the user's password, the adversary will not be able to impersonate a valid user.

Assume that the adversary attempts to capture the link to download the secret file containing the authentication parameters of a valid user U_i, by standing behind the user during the registration process. He should be able to do this within a few seconds for which the QR code will be displayed. If the adversary has the mobile app in his phone, he will be able to capture the link and download the secret file. However, to decrypt the file and to complete the registration, he should know the password of the user U_i. Therefore, the adversary will be able to generate a response to pass the authentication of IdP only if he knows the user's password and he is able to access the contents of the mobile token, and both these conditions being satisfied simultaneously is fairly impossible within the required time limit.

vi. Security against Man-in-the-Middle Attack: In the proposed scheme, during the login phase, U_i submits $h(ID_i)$ and $h(PW_i)$ to the IdP. The IdP computes a challenge B_1 using the submitted values and his own secret key 'S'. The challenge $B_1 = h(K_{idp} \| h(C_1 \oplus r_1)))$ is verified by U_i by computing $B_1' = h(K_i \| h(C_2 \oplus r_1)))$ and comparing B_1' with the received B_1. Only if they match will the user U_i consider the communication as coming from a valid server and will proceed with the authentication process. Otherwise the session will be terminated. Now if $B_1' = B_1$, then the app will generate a response $K = HMAC(K_i, B_2)$ and the calculation of K requires the knowledge of K_i and C_2. K_i is a value stored within the mobile-token and C_2 is calculated using the challenge received from the server and using the values stored in the mobile-token. Thus only a valid user knowing the correct password and having the right mobile-token will be able to generate a valid response. Again this response is verified by the server using values generated and known only by the server. Thus both the server and the user mutually authenticate each other during the authentication phase, which makes the scheme resistant to man-in-the-middle-attack.

vii. **Two-Factor Security:** In a scenario where, both the user's mobile token and his password are stolen, there is no way to prevent the attacker from masquerading as the user. Hence the security of the proposed two-factor authentication scheme can be guaranteed when either the mobile-token or the password is stolen but not both. This security property is referred to as two-factor security. In the discussed scheme the secret parameters $< V_{i,} K_{i\,,} M_i, h(.)>$ in the mobile token are difficult to be derived if the attacker has obtained the user's password alone and not the mobile token. Now if the attacker also intercepts the challenge $B_1 = h(K_{idp} \| h(C_1 \oplus r_1)\ ))$, it is a laborious process to extract PWi from C_1 and K_{idp} due to the irreversible property of one-way hash functions. Again if the attacker intercepts the response $K = HMAC(K_i,\ B_2)$ from the user, it is infeasible to derive $h(S)$ or $h(PW_i)$ from HMAC $(K_i,\ B_2)$ as they are based on irreversible hash functions.

viii. **Mutual Authentication:** When the user receives the challenge B_1 from the server, it is verified as $B_1‘= h(K_i \| h(C_2 \oplus r_1))$, where $C_2 = M_i \oplus K_i$ is calculated using parameters M_i and K_i in the mobile token. The app compares the calculated B_1‘with the received B_1 and if equal the user U_i assumes that it is communicating with the server to whom it had sent a communication at the start of the login phase. A response to this challenge is generated by using C_2 which is extracted from the mobile token. The server calculates $B_2’ = C_1 \oplus B_1 \oplus r_2$ and $K’ = HMAC(K_{idp}, B_2’)$. The IdP compares K' with the K received from the user and a successful verification proves the authenticity of the user.

Efficiency Analysis

This section analyzes the efficiency of the proposed protocol in terms of the computational and the communication cost. It is assumed that ID_i, PW_i , nonce values are 128 bits long and the output of hash function(SHA-2) is 256 bits long. Let T_h, T_x , T_c and T_s denote the time complexity for hashing, XOR , concatenation and symmetric key encryption respectively. In the protocol, the parameters stored in the secret file are V_i, K_i, M_i and the memory (E1) needed in the mobile is 768 (3*256) bits. Communication cost of authentication (E2) includes the capacity of transmitting parameters (ID_i, PW_i, B_1, r_1, r_2, k, $h(k_{idp}||r2)$) which makes E2 equal to 1536 (5*256 + 2* 128) bits. The computation cost of user registration (E3) is the total time of all operations executed in this phase by the user and IdP and is equal to $7T_h + 4T_x + 3T_c + 1T_s +1T_d$. The computation cost of the user (E4) and the IdP (E5) is the total time of all operations executed by the user and IdP during login and authentication. During authentication, the user performs 7 hash functions, 6 XOR, 4 concatenation making E4 equal to $7T_h + 6T_x + 4T_c$. Similarly, E5 is $7T_h + 6T_x + 6T_c$. The computation cost of password changing phase (E6) is the total time of all operations executed in this phase by the user and is equal to $6T_h + 3T_x + 2T_c$. Comparison of efficiency with other protocols are shown in table 4.4.

Table 4.4 Comparison of Computational Efficiency with Other Protocols

	E1	E2	E3	E4	E5	E6
Mobile-token based protocol	768 bits	1536 bits	$7T_h + 3T_c + 4T_x + 1T_s + 1T_d$	$7T_h + 6T_x + 4T_c$	$7T_h + 6T_x + 6T_c$	$6T_h + 3T_x + 2T_c$.
Choudhary et al. [2011]	1024 bits	1920 bits	$6T_h + 3T_x + 1T_e + 2T_c$	$10T_h + 2T_x + 1T_e + 3T_c$	$8T_h + 1T_x + 1T_e + 3T_c$.	$4T_h + 4T_x$
Jaidhar [2013]	1024 bits	1664 bits	$5T_h + 5T_x + 1T_e + 5T_c$	$6T_h + 2T_x + 9T_c + 2T_s + 1T_d$	$5T_h + 1T_x + 8T_c + 2T_d + 1T_s + 1T_e$	$3T_h + 2T_x + 3T_c$
Rui Jiang [2013]	768 bits	1152 bits	$4T_h + 1T_x + 1T_e + 1T_c$	$7T_h + 1T_x + 4T_c + 1T_d + 1T_e$	$7T_h + 5T_c + 1T_s + 1T_e$	$18T_h + 3T_x + 11T_c + 2T_s + 1T_d + 4T_e$

Results of comparison of computational efficiency reveals that the proposed protocol for brokered authentication using mobile-token are comparable with other similar protocols in terms of memory needed in the token, communication cost during authentication, and computation costs during registration, login& authentication and password change phase.

In the case of the proposed mobile token based protocol, the authenticity of the user is verified before storing the secrte file into the user's smart phone, by attempting to decrypt the encrypted file downloaded from the Identity Provider. In addition, the protocol also ensures the integrity of the stored parameters in the secret file which is downloaded from the server, by recalculating the value of a parameter strored into the file by the server. Only after these two verifications are done, will the file be permanently stored in to user's smart phone.

Also the proposed mobile token based protocol is using HMAC to generate the response from the phone to the user, and HMAC requires two hash, two concatenations and two XOR operations. Again the HMAC value send by the client is verified by the server to authenticate the user. All these processes are enhancing security though it leads to an increase number of computations.

Though these computations increase the computation cost of the protocol and affects total computational time and efficiency, the protocol aids in providing enhanced security. In such a scenario, it can be mentioned in the Service Level Agreement between the IdP and the Service Providers that the authentication protocol provided by the IdP, provides secure authentication of users that requires a certain time period for execution. The authentication protocol can be adopted by those service providers to whom the time duration for execution of authentication protocol is agreeable.

Scyther Analysis

The formal analysis of protocol is done using Scyther tool. The strength of the protocol is verified using Scyther tool which ascertains the strength by evaluating the resistance of the protocol to various attacks. Scyther uses strand space model for formalizing logic and uses Dolev-Yao model for modelling the network, which caters to the requirement of a mathematical approach for validating the protocol. The analysis results of login phase are shown in Figure 4.9**.** The protocol is written in SPDL as follows**:**

```
//Login And Authentication Phase of Brokered Authentication Protocol using Mobile Token
const exp: Function; const hash: Function; hashfunction h; const XOR: Function;
const h1: Function; const HMAC: Function; protocol mobileauth(I,R){
```

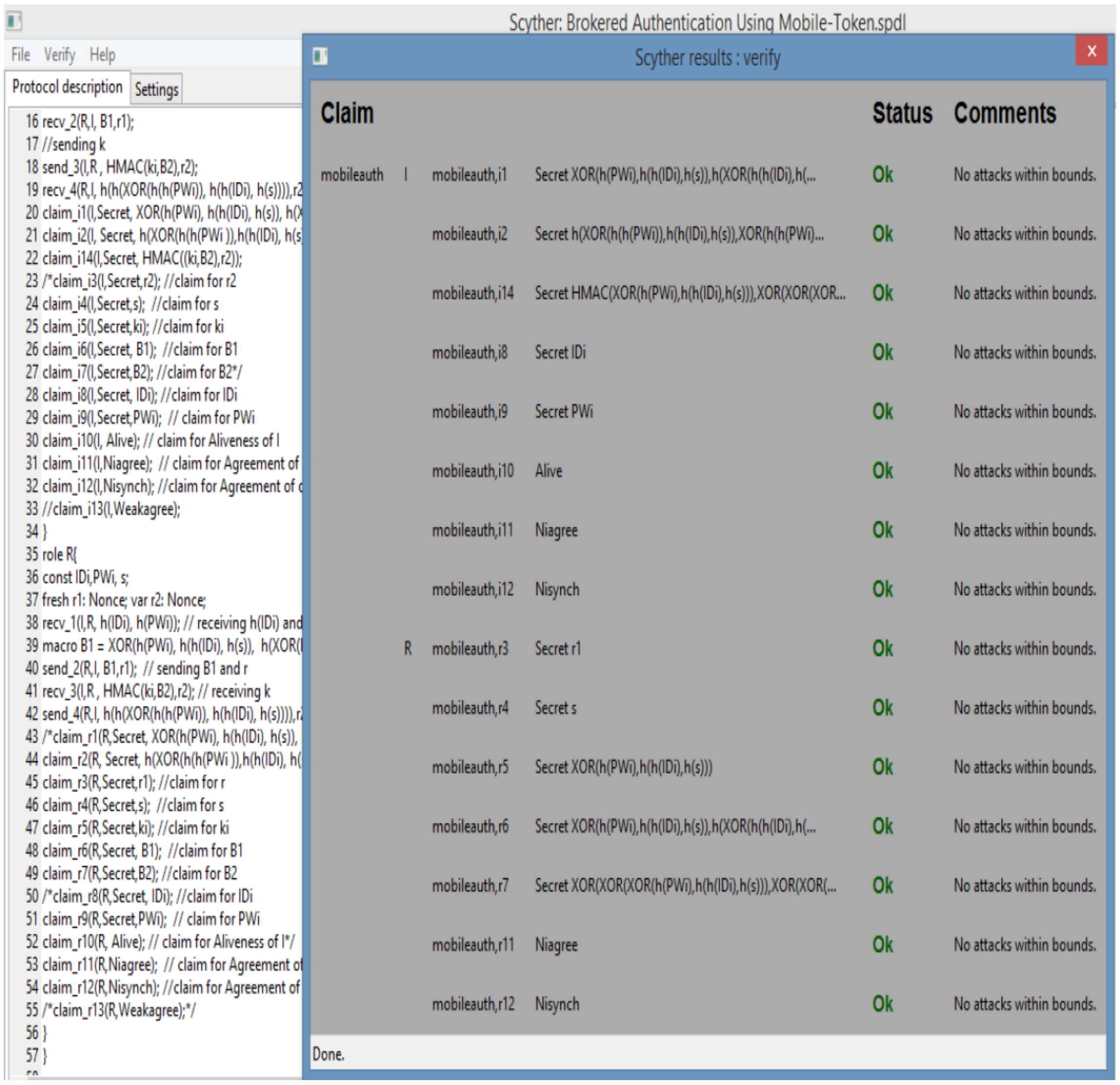
```
role I{
const IDi, PWi, s;
var r1: Nonce; fresh r2: Nonce;
macro ki = XOR(h(PWi), h(h(IDi),h(s)));
macro mi = XOR (ki, h(h(IDi),h(PWi)));
macro C2 = XOR (ki,mi);
macro B1 = XOR(h(PWi), h(h(IDi), h(s)) ,  h(XOR(h(h(IDi),h(PWi)),r1)));
macro B2 = XOR(C2, B1,r2);
//Sending Identity and Password of User
send_1(I,R, h(IDi), h(PWi));
//receiving B1 and r1
//recv_2(R,I, (h(XOR(h(PWi), h(h(IDi), h(s)))) , h(XOR(h(h(IDi),h(PWi)),r))), r);
recv_2(R,I, B1,r1);
//sending k
send_3(I,R , HMAC(ki,B2),r2);
recv_4(R,I, h(h(XOR(h(h(PWi)), h(h(IDi), h(s)))),r2));
claim_i1(I,Secret, XOR(h(PWi), h(h(IDi), h(s)), h(XOR(h(h(IDi),h(PWi)),r1)),r1));  // claim for B1
claim_i2(I, Secret, h(XOR(h(h(PWi )),h(h(IDi), h(s)), ( (XOR(h(h(PWi)), h(h(IDi), h(s))) , h(XOR(h(h(IDi),h(PWi)),r2))))))); // claim for k
claim_i3(I,Secret,r2); //claim for r2
claim_i4(I,Secret,s);  //claim for s
claim_i5(I,Secret,ki); //claim for ki
claim_i6(I,Secret, B1);  //claim for B1
claim_i7(I,Secret,B2); //claim for B2
claim_i8(I,Secret, IDi); //claim for IDi
claim_i9(I,Secret,PWi);  // claim for PWi
claim_i10(I, Alive); // claim for Aliveness of I
```

```
claim_i11(I,Niagree);   // claim for Agreement of variables and values exchanged
claim_i12(I,Nisynch); //claim for Agreement of order of execution of events & variables and values exchanged
//claim_i13(I,Weakagree);
}
role R{
const IDi,PWi, s;
fresh r1: Nonce; var r2: Nonce;
recv_1(I,R, h(IDi), h(PWi)); // receiving h(IDi) and h(PWi)
macro B1 = XOR(h(PWi), h(h(IDi), h(s)),  h(XOR(h(h(IDi),h(PWi)),r1)));
send_2(R,I, B1,r1);  // sending B1 and r
recv_3(I,R , HMAC(ki,B2),r2); // receiving k
send_4(R,I, h(h(XOR(h(h(PWi)), h(h(IDi), h(s)))),r2));
claim_r1(R,Secret, XOR(h(PWi), h(h(IDi), h(s)), h(XOR(h(h(IDi),h(PWi)),r1)),r1));  // claim for B1
claim_r2(R, Secret, h(XOR(h(h(PWi )),h(h(IDi), h(s)), ( (XOR(h(h(PWi)), h(h(IDi), h(s))) , h(XOR(h(h(IDi),h(PWi)),r2))))))); // claim for k
claim_r3(R,Secret,r1); //claim for r
claim_r4(R,Secret,s);  //claim for s
claim_r5(R,Secret,ki); //claim for ki
claim_r6(R,Secret, B1);  //claim for B1
claim_r7(R,Secret,B2); //claim for B2
claim_r8(R,Secret, IDi); //claim for IDi
claim_r9(R,Secret,PWi);  // claim for PWi
claim_r10(R, Alive); // claim for Aliveness of I
claim_r11(R,Niagree);   // claim for Agreement of variables and values exchanged
claim_r12(R,Nisynch); //claim for Agreement of order of execution of events & variables and values exchanged
```

```
claim_r13(R,Weakagree);
}

}
```

Formal Analysis using Scyther

To perform the formal security analysis, this section focuses on evaluating the vulnerability of certain parameters such as IDi, PW_i, s, V_i, K_i, M_i, B_1, B_2, K which are used in the proposed authentication scheme. The proposed protocol coded using SPDL is analyzed using the security analyzer Scyther, which checks for the vulnerability of each of the parameters used in the scheme. Scyther is configured with ten (10) runs and all possible attacks. There are various claims made as part of the security analysis and these claims are validated by executing and analyzing the proposed scheme using Scyther.



```
recv_2(R,I, B1,r1);
//sending k
send_3(I,R , HMAC(ki,B2),r2);
recv_4(R,I, h(h(XOR(h(h(PWi)), h(h(IDi), h(s)))),r2
claim_i1(I,Secret, XOR(h(PWi), h(h(IDi), h(s)), h(X
claim_i2(I, Secret, h(XOR(h(h(PWi )),h(h(IDi), h(s)
claim_i14(I,Secret, HMAC((ki,B2),r2));
/*claim_i3(I,Secret,r2); //claim for r2
claim_i4(I,Secret,s);  //claim for s
claim_i5(I,Secret,ki); //claim for ki
claim_i6(I,Secret, B1);  //claim for B1
claim_i7(I,Secret,B2); //claim for B2*/
claim_i8(I,Secret, IDi); //claim for IDi
claim_i9(I,Secret,PWi);  // claim for PWi
claim_i10(I, Alive); // claim for Aliveness of I
claim_i11(I,Niagree);  // claim for Agreement of
claim_i12(I,Nisynch); //claim for Agreement of c
//claim_i13(I,Weakagree);
}
role R{
const IDi,PWi, s;
fresh r1: Nonce; var r2: Nonce;
recv_1(I,R, h(IDi), h(PWi)); // receiving h(IDi) and
macro B1 = XOR(h(PWi), h(h(IDi), h(s)),  h(XOR(
send_2(R,I, B1,r1);  // sending B1 and r
recv_3(I,R , HMAC(ki,B2),r2); // receiving k
send_4(R,I, h(h(XOR(h(h(PWi)), h(h(IDi), h(s)))),r
/*claim_r1(R,Secret, XOR(h(PWi), h(h(IDi), h(s)),
claim_r2(R, Secret, h(XOR(h(h(PWi )),h(h(IDi), h(
claim_r3(R,Secret,r1); //claim for r
claim_r4(R,Secret,s);  //claim for s
claim_r5(R,Secret,ki); //claim for ki
claim_r6(R,Secret, B1);  //claim for B1
claim_r7(R,Secret,B2); //claim for B2
/*claim_r8(R,Secret, IDi); //claim for IDi
claim_r9(R,Secret,PWi);  // claim for PWi
claim_r10(R, Alive); // claim for Aliveness of I*/
claim_r11(R,Niagree);  // claim for Agreement of
claim_r12(R,Nisynch); //claim for Agreement of
/*claim_r13(R,Weakagree);*/
}
}
```

```
Claim                                                                                Status  Comments

mobileauth  I  mobileauth,i1   Secret XOR(h(PWi),h(h(IDi),h(s)),h(XOR(h(h(IDi),h(...        Ok      No attacks within bounds.
               mobileauth,i2   Secret h(XOR(h(h(PWi)),h(h(IDi),h(s)),XOR(h(h(PWi)...        Ok      No attacks within bounds.
               mobileauth,i14  Secret HMAC(XOR(h(PWi),h(h(IDi),h(s))),XOR(XOR(XOR...      Ok      No attacks within bounds.
               mobileauth,i8   Secret IDi                                                   Ok      No attacks within bounds.
               mobileauth,i9   Secret PWi                                                   Ok      No attacks within bounds.
               mobileauth,i10  Alive                                                        Ok      No attacks within bounds.
               mobileauth,i11  Niagree                                                      Ok      No attacks within bounds.
               mobileauth,i12  Nisynch                                                      Ok      No attacks within bounds.
            R  mobileauth,r3   Secret r1                                                    Ok      No attacks within bounds.
               mobileauth,r4   Secret s                                                     Ok      No attacks within bounds.
               mobileauth,r5   Secret XOR(h(PWi),h(h(IDi),h(s)))                            Ok      No attacks within bounds.
               mobileauth,r6   Secret XOR(h(PWi),h(h(IDi),h(s)),h(XOR(h(h(IDi),h(...        Ok      No attacks within bounds.
               mobileauth,r7   Secret XOR(XOR(XOR(h(PWi),h(h(IDi),h(s))),XOR(XOR(...       Ok      No attacks within bounds.
               mobileauth,r11  Niagree                                                      Ok      No attacks within bounds.
               mobileauth,r12  Nisynch                                                      Ok      No attacks within bounds.
Done.
```

Figure 4.9 Scyther Analysis of Brokered Authentication Using Mobile Token

The "No attack" results shown in Figure 4.9 proves that Scyther validates all the claims made as part of security analysis.

Claim 1: The proposed scheme is designed to ensure the secrecy of the user ID viz. ID_i, throughout the registration and authentication process.

The user ID, ID_i is submitted in the hashed form to the registration authority during the registration process. A "No Attacks within Bounds" results for the Secrecy claim, claim_i8(I, Secret, ID_i) is indicative of the fact that whenever a run of the I role is completed with an honest communication partner, the value ID_i transmitted by I in the run will not be revealed to the adversary. Therefore, it can be said that claim secret ID_i of the role I holds.

Claim 2: The proposed scheme is designed to ensure the secrecy of the password 'PW_i' throughout the registration and authentication process.

The password is never transmitted in the plaintext form either to the IdP during the registration process or to the service provider during the authentication process. It is transmitted in a hashed form and as we know hash operations are irreversible. During the authentication process, the hashed password is used by the server along with h(IDi), h(PWi), h(S) and nonce r_1 to compute the challenge $B_{1.}$ The calculation of B_1 involves several hashing, XOR and conactenation operations, which makes it very difficult to retrieve PW_i or $h(PW_i)$ from B_1. Though the response K generated by the mobile token also includes the user password, it is hashed and XOR-ed with other parameters and nonce value 'r_2'. After performing all these operations, the HMAC of the result is taken to get the K value. This makes it all the more difficult to extract PW_i or $h(PW_i)$ from K. Also the password is not stored anywhere other than in the mobile-token that too in a hashed form combined with other parameters. Scyther results validate the claim that 'PW_i' remains a secret.

Claim 3: The proposed scheme requires the secret key 'S' of the server to be a secret

‘S’ is the secret key of the IdP, which is used for computing the parameters in the mobile token. It is used in its hashed form to compute the parameters to be stored in the mobile-token and to verify the user during the authentication process. Scyther validates the claim that ‘S’ is safe.

Claim 4: The proposed scheme requires that the parameter V_i stored in the mobile-token remains a secret

V_i is a value used to generate the authentication parameters calculated by the IdP and stored in the mobile token. V_i is computed by hashing the concatenation of hash of user ID and hash of user’s password PW_i and the result is XOR-ed with a nonce generated by the IdP. Scyther validates the claim that ‘V_i’ is safe.

Claim 5: The proposed scheme requires that the authentication parameter K_i stored in the mobile token remains a secret

K_i is one among the authentication parameters calculated by the IdP and stored in the mobile token. K_i is computed by performing hash and XOR operations on the hash of PW_i and the hashed values of h(IDi), h(S). Scyther validates the claim that ‘K_i’ is safe.

Claim 6: The proposed scheme requires that the authentication parameter M_i stored in the mobile token remains a secret

M_i is one among the authentication parameters calculated by the IdP and stored in the mobile token. M_i is computed by performing XOR operations on K_i and the hashed values of h(IDi) and h(PWi). M_i should remain a secret and should not reveal any information that will enable the adversary to impersonate a valid user. Scyther validates the claim that ‘M_i’ is safe.

Claim 7: The proposed scheme requires that the challenge B_1 transmitted by the server is secret

Challenge B_1 generated by the server is the hashed information containing hash of user ID, hash of user PW, hash of server secret key, S and nonce r_1 sent by the server to the user to ensure against replay attack and phishing attack. B_1 is re-calculated by the mobile token using the stored parameters M_i and K_i to verify the authenticity of the origin of communication. B_1 should not reveal any information that will enable an adversary to generate a valid challenge. Scyther validates the claim that 'B_1' is safe.

Claim 8: The proposed scheme requires that the response B_2 generated by the user remains secret

B_2 generated by the mobile-token is the XOR of C_2 and B_1. Here C_2 is calculated using the M_i and K_i values stored in the mobile-token and B_1 is the value received from the server. B_2 is used by the mobile-token to generate the response K corresponding to the challenge B_1 received from the server. B_2 should not reveal any information that will enable an adversary to impersonate a valid user. Scyther validated the claim that 'B_2' is safe.

Claim 9: The proposed scheme requires that K is secret

K is the communication sent by the mobile in response to the challenge B_1 sent by the server (IdP). The computation of K is done by generating an HMAC value which uses two inputs viz. a key value and a message. In the proposed protocol, HMAC algorithm uses K_i as the key. The XOR of B_2 generated by the mobile token and the received challenge B_1 is taken as the message whose MAC is to be calculated. K is re-calculated by the server using its own set of values. HMAC guarantees authenticity of the origin and integrity of the message. K which is representative of the login request should not reveal any information that will enable an adversary to forge a valid login request. Scyther validated the claim that ‘K’ is secret.

Claim 10: The scheme assures the user that the server remains alive and also the server is assured that the user remains alive. If the proposed protocol is used by the server for the initial (i-1) messages exchanged with the user, when the user sends the i^{th} message, then the server is said to be alive. The Scyther tool validates the aliveness claim.

Claim 11: The scheme assures Niagree between the user (mobile-token) and the server

Niagree claim enforces that the sender (user) and the receiver (server) agree upon the values of variables exchanged during the running of the proposed scheme. During the operation of the proposed scheme, the user and sever can send data safely and the correctness of the claim is justified by the analysis results.

Claim 13: The proposed scheme holds Synchronization during the registration and authentication process

Ni-Synch or Non-Injective Synchronization property requires that the corresponding send and receive events (1) happened in the correct order and (2) have the same contents. Ni-Synch is valid if all actions before the claim are performed as per the description of the proposed scheme. The proposed protocol satisfies this claim as indicated by the result of Scyther analysis.

CONCLUDING REMARKS

This chapter elaborated an authentication scheme that can be adopted by service providers who would prefer to have a strong Two-Factor authentication mechanism to authenticate users of it’s services. The proposed scheme can be adapted by those service providers who work in a collaborative environment and by service providers who offer their services via a web portal. To provide the users with a seamless authentication experience to users who access different services during the same session, these service providers prefer to have Single Sign-on functionality. Hence in the proposed scheme for brokered authentication, users are authenticated by a third party Identity Provider, who does the authentication of the users re-directed to it by the service providers. Security Assertion Markup Language (SAML) protocol which is used to exchange authentication related information about users between the Identity Provider and Service Providers is required to provide Single Sign-on functionality.

The Proposed Brokered Authentication scheme is different from “Crypt DB” where the authentication is done by a proxy. Crypt DB provides confidentiality of relational databases by supporting computations on encrypted data at database server side. In Crypt DB an intermediate proxy is trusted for connection to the database server and proxy uses secret keys to encrypt all the user data stored in the database. The encryption keys are encrypted using the password of the user and stored by the proxy. Since Crypt DB stores the encryption keys, there are certain security concerns. The concerns are (i) If an attacker manages to get hold of the password of the user, then he can use the password to decrypt the keys stored in the proxy database. (ii) since the proxy is storing the user’s keys an adversary may use crypto analysis to break the encryption and retrieve the encryption keys from the proxy. (iii) if the user’s password is lost then the proxy is not able to retrieve the original encryption key as the key can be decrypted only using the user’s password.

In the proposed brokered authentication protocol, the authentication broker stores only the profile information of users (eg.user ID, first name, last name, email address etc.). The authentication broker does not maintain the password information or the user keys. Also the proposed brokered authentication protocols use two-factor authentication which requires the user to provide both his password and the parameters stored within the crypto-token/mobile-token to prove his identity to the authentication server.

The proposed authentication protocols do not require the server to maintain a verifier table. First half of the chapter discusses an authentication protocol that uses Crypto-token as an authentication factor and second half discusses an authentication protocol that uses a mobile token as an authentication factor. The chapter also includes Security analysis of proposed protocols to validate the resistance to various common attacks on authentication protocols. In addition to security analysis, efficiency analysis is done to compare the computational efficiency of the proposed protocol with similar two-factor authentication schemes for cloud. Formal verification is done using Scyther which verifies the validity of security claims made with respect to the protocol.

CHAPTER 5

SECURE INTEGRATED FRAMEWORK FOR AUTHENTICATION IN CLOUD

Cloud service providers can be of two categories based on their authentication requirements;

- Category one includes those service providers dealing with highly sensitive data and working in a controlled environment such as those providing health-care services, financial services etc. These service providers need a strong user authentication mechanism without any additional functionality such as Single Sign-On.
- The second category of service providers are those dealing with secure data while working in a collaborative environment wherein the contents are accessed by the Users simultaneously with associated services of a different service provider. Hence category 2 providers need a strong authentication mechanism that also provides the Users with a Single sign-on functionality

In Chapter 3, under section 3.1, *Direct* authentication architecture and protocols that can be adopted by service providers who prefer to authenticate its users on their own (directly) using a strong authentication mechanism was discussed. Chapter 4, section 4.1 elaborated on *Brokered* authentication architecture and protocols that can be adopted by service providers who require a Single sign-on functionality and hence prefer to delegate the authentication to a third party.

However, both types of authentication architecture and protocols (*Direct* as well as *Brokered*) are very specific in usage to service providers who adopt the corresponding mode of authentication – viz Direct and Brokered.

A major objective of this research is to propose an authentication model that can be adopted by the two categories of service providers mentioned above. To achieve this objective, this research proposes an authentication framework for Cloud which supports an integrated authentication architecture that provides the service providers with the flexibility to choose between *Direct* and *Brokered* authentication. The integrated two-factor authentication protocol, which does not require the server to maintain a verifier table, supported by the frame work allows users to do a single registration and access services of both *Direct* authentication service providers (DASP) and *Brokered* authentication service providers (BASP) using the same crypto-token/mobile-token. This chapter discusses the Architecture and components of the framework, an Integrated Authentication model and a Two-Factor authentication protocol that can be adopted by both the service providers preferring "*Direct Authentication*" or "*Brokered Authentication*" of their Users. This proposed authentication model provides users with the convenience of not having to remember different identities and carry multiple authentication devices to access multiple services. Also, Service Providers have the flexibility to either directly authenticate its users or to redirect the users to a third party for Brokered Authentication thus providing its users with a seamless authentication experience through Single sign-on functionality. Users can access the services of both Direct & Brokered Service Providers (DASP & BASP) by authenticating themselves using a single password and a Crypto-token or a Mobile-token issued by the Identity Provider.

5.1 FRAMEWORK ARCHITECTURE

The architecture of the proposed integrated framework is as depicted in Figure 5.1.

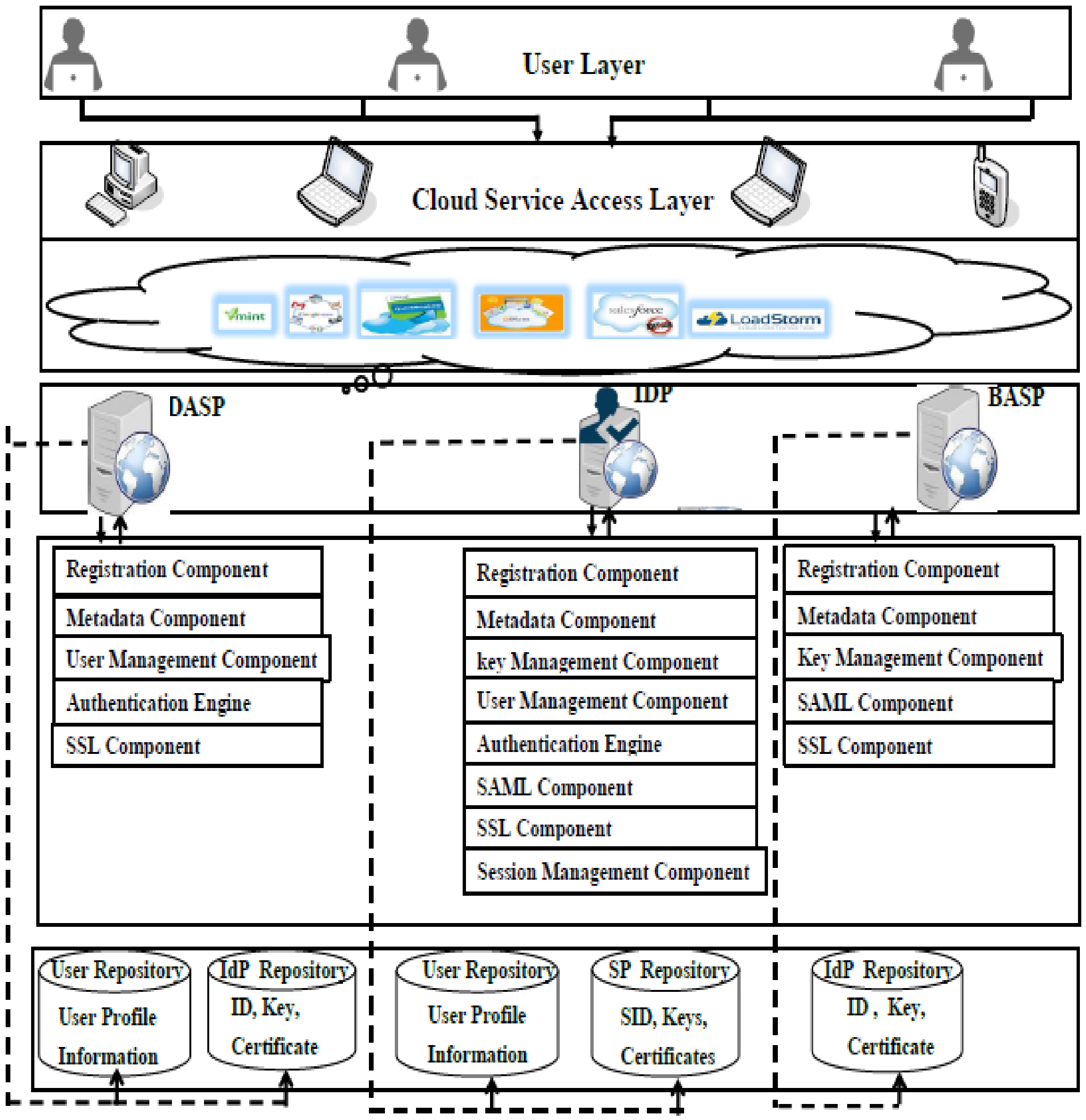


Figure 5.1 Framework Architecture

The role of the participants of the framework and the functionality of the components are explained in the following paragraphs.

Users: Users access the services provided by various Service Providers after registration and authentication.

To avail the services of registered service providers, a user needs to register with the registration server of the Identity Provider (IdP), by submitting user name, password and other profile information. After registration at the IdP, users will be issued with a Crypto-token or mobile-token containing the security parameters generated using user's credentials and IdP's secret key. Also a list of service providers whose services are accessible to the user will be provided. User password is neither stored within user's system nor at the end of Identity Provider or Service Provider. A variant of the password is stored inside the Crypto-token or the mobile-token. Access to services are provided after verifying the security parameters stored within the crypto-token/mobile-token produced by the user during authentication process. In this integrated framework, user's password and crypto-token/mobile-token serves as the two authentication factors. Accessing services of service providers adopting direct authentication require the user to authenticate individually to each service provider. In the case of service provider's adopting brokered authentication, the user needs to authenticate only once during a session.

Service Providers: Service Providers, who are part of the framework should be registered with the registration server of the Identity Provider(IdP). Service Provider's who directly authenticate its users are referred to as Direct Authentication Service Provider's (DASP's) and service provider's who delegate the authentication to the authentication broker (IdP) are referred to as Brokered Authentication Service Providers (BASP’s).

Registration Component of DASP: Registration of DASP’s at the IdP is managed by the registration component. DASP’s should undergo a registration process by submitting a unique server ID, URL of service provider, a short description of service provided, and the mode of authentication preferred as “Direct Authentication”. At the end of the registration process, Identity Provider communicates its master key to DASP in a secure manner. This is later used by the DASP to verify an authentication parameter during authentication of the user using the proposed 2-factor authentication protocol. Authentication module containing the proposed 2-factor authentication protocol is issued to DASP by the IdP and this can be integrated with the SP’s authentication engine.

Registration Component of BASP: This component manages the registration of service providers adopting brokered authentication. The BASP’s need to undergo a registration task by submitting a unique server ID, URL of service provider, a short description of service provided, and the mode of authentication preferred as “Brokered Authentication”. At the end of the registration process, IdP’s master key is conveyed to BASP in a secure manner. This key is used by IdP and BASP to generate a shared key. This shared key is used to communicate to BASP, the session key generated by IdP and user during each authentication session.

Metadata Component of DASP: This component manages metadata information of IdP which includes a unique ID, URL of IdP etc. This information is required for establishing a communication with IdP.

Metadata Component of BASP: This component manages metadata information of IdP which includes a unique ID, URL of IdP etc. This information is required by the BASP to send a SAML authentication request to the IdP and to verify the corresponding authentication response from the IdP.

User Management Component of DASP: Users requiring an access to the services of DASP, should first register with the registration server of the identity provider (IdP). The IdP will then update the database of all the service providers who prefer direct authentication, with the profile information of users. User management component of every DASP maintains a database of user profile information such as h(ID), email-ID, mobile number, address etc. When a user tries to authenticate to a DASP, the user management component will check whether the user-ID provided by the user exists in the database. Otherwise the user is redirected to the registration server of the IdP.

SAML Component of BASP: During authentication process, authentication information of user is exchanged between Identity Provider and BASP using Security Assertion MarkUp Language (SAML). Generating SAML requests and verifying SAML responses is the responsibility of SAML component of BASP. The request includes information about the SP (unique ID) who generated the request (which is verified against the metadata information maintained about the SP's by the IdP), Unique ID of IdP, assertion Consumer Service (ACS) URL which is the location to which IdP's SAML authentication response should be send.

The SAML assertions (responses) sent by the IdP will contain data such as authentication statement, authorization statements, a unique assertion identifier, an issue instant (the time at which the assertion was created), the issuer name (the IdP name), which is verified against the metadata information maintained about the IdP by the SP. The SAML responses are verified by SP and on successful verification, user will be allowed access to resources/services.

Authentication Engine of DASP: The authentication engine of DASP executes the two-factor authentication protocol. User's trying to access the services of these SP's will be authenticated by the authentication server of these service providers. Authentication is done by the authentication module implemented in the authentication server. To start with, the authentication component will communicate with the user management component to verify whether the user is registered or not. The authentication module runs the proposed authentication protocol and crypto-token/mobile-token as the two authentication factors. The execution of the mutual authentication protocol culminates with the generation of a session key which is used to secure communications occurring thereafter.

Key Management Component of BASP: Securely managing the master key of IdP shared with the service provider's is done by the key management component of BASP. This component also does the task of securely managing the session keys generated between user and IdP during the brokered authentication process and communicated to BASP by IdP during the end of the session.

Secure Sockets Layer (SSL) Component of DASP/BASP: Credentials of user should be securely communicated to the Identity Provider during authentication process. Confidentiality of information flowing through the network is ensured by transmitting the information over an SSL connection. SSL component is responsible for setting up an SSL connection and managing the same.

User Repository of DASP: This repository of DASP stores the profile information of all the registered users.

Identity Provider Repository of DASP/BASP: This repository maintains the information pertaining to the Identity Provider, which includes unique ID, Domain name, Digital Certificate of IdP, Business Agreement (BA) terms and policies etc.

Identity Provider: Identity Provider is a trusted entity providing Registration service and Authentication Service on request. IdP should have its Digital certificate and the SAML protocol configured to facilitate Single Sign-On.

Registration Service: Service Providers who are providing cloud applications/services will become participants of the proposed framework, after registering with the Identity provider. DASP’s and BASP’s should register with IdP’s registration server by submitting a unique server ID, URL of service provider, description of provided service and the mode of authentication preferred as “Direct” or “Brokered”. After registration, an authentication module will be provided by IdP to DASP. At the end of the registration process, the master key of the IdP is communicated securely to DASP which is later used to verify an authentication parameter during the execution of the 2-factor authentication protocol with the user. In the case of BASP’s, this key is used to generate a key which is shared between IdP and BASP.

Users who want to be a part of the system to access the services of the DASP’s and BASP’s need to undergo a registration process. During registration process, users submit their user name, password etc. to the RS of IdP. This information along with the IdP’s master secret key is used to compute a set of secret parameters which is later used by the user for authenticating to service providers. User is prompted to download and store these parameters into a crypto-token or into a mobile-token depending on whether a crypto-token or mobile phone is used as the authentication factor. If the second factor is a mobile phone user is required to download a mobile application from the IdP’s site into his mobile phone.

User Management Component: Profile information of users registered with IdP are managed by the user management component of IdP Users.

Metadata Component: Metadata information of all DASP’s and BASP’s registered with IdP are managed by the metadata component. During registration of users, metadata component provides support in identifying the DASP's to whom the profile information of registered users should be communicated. During Brokered Authentication, the metadata component enables the IdP to veify the source of authentication request.

Authentication Service: IdP authenticates users re-directed to it by BASP's and thus provides authentication as a service. After the exchange of two-factor authentication protocol between the authentication server of IdP and user, the result of authentication process is communicated as a SAML response to IdP.

SAML component: The SAML component of the IdP which is invoked only during Brokered Authentication should have, the SAML v2.0 configured for verifying SAML requests received from BASP's and for generating SAML response.

Key Management Component: Generating and managing keys shared with the BASP's is done by this component of IdP. Also, this component is responsible for securely communicating to BASP, the session keys generated between IdP and user during authentication process.

SSL Component: Communication between user and server should be secure during login and authentication process. Confidentiality of information flowing through the network is ensured by transmitting the information over an SSL connection. SSL component is responsible for setting up a secure

SSL connection and managing the same during login and authentication phase.

Session Management Component: This component of IdP ensures that a user authenticates once by providing his credentials, during a session, to access multiple services (SSO).

Service Provider Repository: Information pertaining to SP's which includes Unique ID, Domain name of SP, Digital Certificates, SP's preferred mode of authentication, type of services provided by SP, Business Agreement (BA) terms and policies etc. are maintained by this repository.

5.2 INTEGRATED AUTHENTICATION MODEL FOR CLOUD

In the integrated model for authentication of users, direct authentication service providers and Brokered authentication service providers should be registered with the registration server of IdP. Authentication of user will be done by the Authentication Server (AS) of the DASP in the case of direct authentication, and Authentication Server (AS) of the IdP will authenticate the user in the case of brokered authentication. The registration and authentication process flow are as illustrated in Figure 5.2.

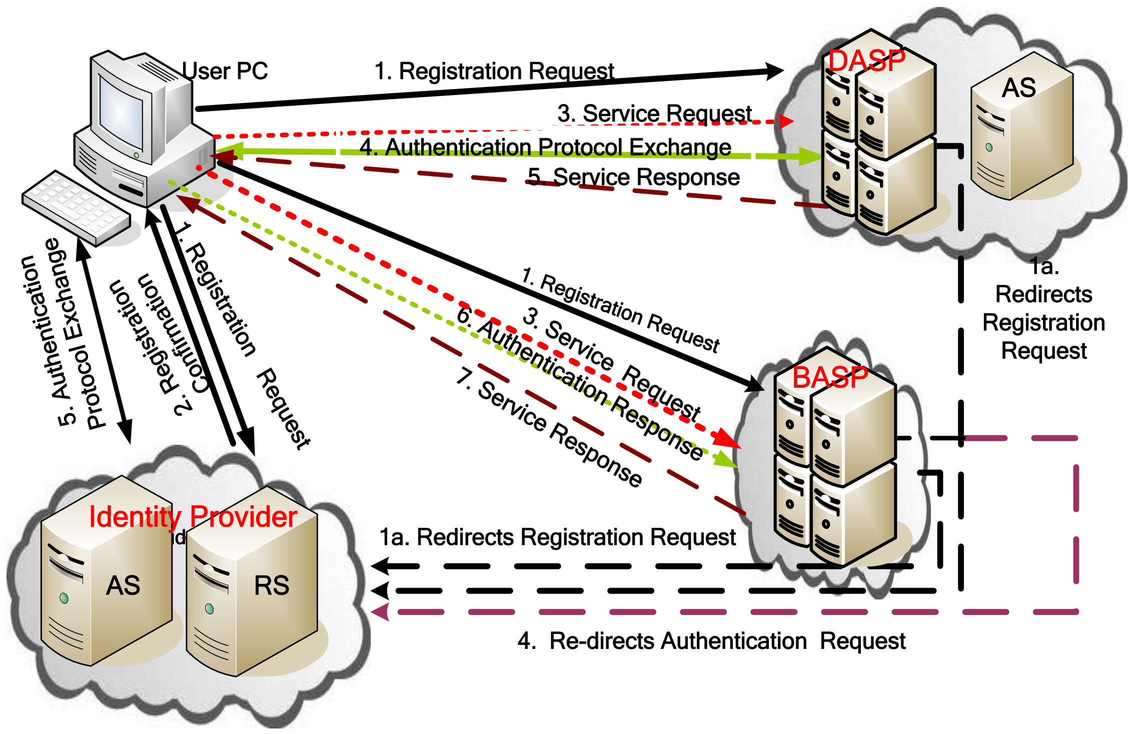

User PC
1. Registration Request
3. Service Request
4. Authentication Protocol Exchange
5. Service Response
DASP
AS
1a. Redirects Registration Request
1. Registration Request
3. Service Request
6. Authentication Response
7. Service Response
BASP
5. Authentication Protocol Exchange
2. Registration Confirmation
1. Registration Request
Identity Provider
AS
RS
1a. Redirects Registration Request
4. Re-directs Authentication Request

Figure 5.2 Registration and Authentication Process Flow for Framework

5.3 PROPOSED INTEGRATED-FRAMEWORK PROTOCOL

Phases of the Proposed protocol: The proposed protocol has four phase's viz., Registration Phase of SP, Registration phase of user, Login & Authentication Phase and the Password change phase. Registration phase of user is shown in Figure 5.3, login and authentication in Figure 5.4 and Password change in Figure 5.5. The notations used in the protocol are shown in Table 5.1.

Table 5.1 Notations Used in Proposed Integrated-Framework Protocol

IdP, SP	Identity Provider, Service Provider in the cloud
U_i, S_j, SID_j, rand	i th User, j th SP, ID of the jth SP, random number of IdP
ID_i, PW_i, g_0, p	Unique Identification of U_i, password of U_i, generator of cyclic group, Prime Number Chosen by U_i.
S	Secret key of server of IdP shared with service providers
N_1, N_2	Nonce values chosen by Server and user respectively
h(.) , $\oplus$, $\|\|$	One-way hash function, XOR operation, Concatenation Operation

5.3.1 Registration Phase - Service Provider

Service providers (SP) who would like to be a part of the proposed framework should register with the RS of the IdP. During registration phase, each SP submits his unique ID viz. SIDj, the URL of the service provider, a short description of the service and the preferred mode of authentication as either “Direct Authentication” or “Brokered Authentication”. If the preferred mode of authentication is “Direct Authentication”, then the RS of the IdP will securely communicate IdP’s master secret ‘S’ to the service provider. During this phase, the key management component of the IdP will calculate a shared key SK_j for each registered service provider supporting brokered authentication as $h(h(S)||h(SID_j))$ where h(S) is the hash of the master secret of IdP and $h(SID_j)$ is the hash of the unique ID of the service provider S_j. This shared key can be calculated by S_j at his end since he knows the master secret ‘S’ of IdP and his own unique ID, SID_j.

5.3.2 Registration Phase - User

This phase illustrated in Figure 5.3, is invoked when the user needs to register and obtain the second authentication factor (crypto-token/mobile-token) using which he can authenticate to gain access to the services of registered service providers. User $U_{i,}$ generates a cyclic group Z_p where ‘p’ is a prime number. Element ‘g_0’ of Z_p is selected as the generator of Z_p.

R1: U_i selects his identity ID_i and password ‘PW_i’. Calculates

$$b = h(PW_i),\ k = g_0{}^{b} \bmod p.$$

R2: U_i submits $h(ID_i)$, k to the registration server (RS) of IdP through a secure communication channel.

R3: Upon receiving $<h(ID_i), k>$, the RS checks with the user management component whether $h(ID_i)$ already exists in the server’s ID table. If so, user has to choose a new identity value. Otherwise, RS proceeds to the next step.

R4: RS generates a random number ‘rand’ and computes:

$I = h(S)$; $V_i = h(h(ID_i)||k) \oplus rand$; $Key_i = h(k) \oplus h(h(ID_i)||I)$;

$M_i = h(h(ID_i)||I) \oplus h(h(ID_i)||k|| rand)$

R5: RS stores $(V_i, Key_i, M_i), h(.))$ in a secret file.

R6: If the second authentication factor is a Crypto-token, the IdP will display the download link from which the file is to be downloaded and stored into the Crypto-token. U_i stores g_0 and ‘p’ into the Crypto-token.

If the second authentication factor is a mobile phone, then users phone should have internet connection during registration process. U_i will be prompted to download and install a mobile app in his phone. Then, IdP will display a QR code which will contain the link to download the secret file.

When the QR code is scanned using the mobile phone, the contents of the secret file are stored into a secure location within the mobile phone. The secret file is subjected to Password Based Encryption (PBE) and hence U_i needs to provide the password while attempting to store the secret file into his phone's internal storage. U_i stores g_0 and 'p' into the secret file.

R7: RS sends a registration confirmation message to U_i along with the list of service providers registered under its domain. RS updates the service providers preferring direct authentication with profile information of all the registered users. Thus the service providers providing direct authentication will maintain a database of user profile information which includes the unique user identity $ID_s = ID_i$ in the hashed form ie. $h(ID_i)$ as one entry along with other profile information such as first name, last name, email-ID etc. RS communicates with the metadata component of the IdP and identifies the registered Service Providers with the preferred mode of authentication as "Direct Authentication". RS updates the user management component of all these service providers with the $h(ID_i)$ and the profile information of the registered user U_i.

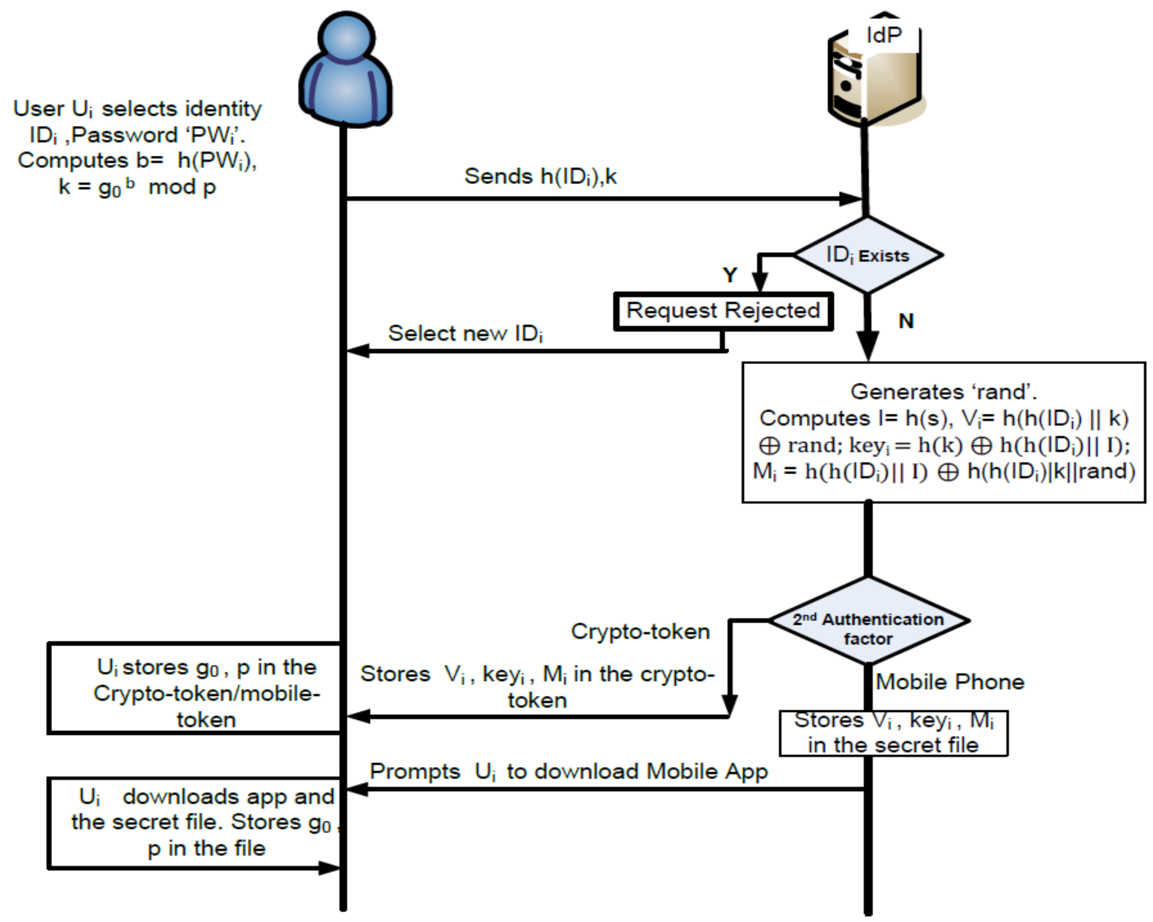

User U_i selects identity ID_i ,Password 'PW$_i$'. Computes b= h(PW$_i$), k = g_0^b mod p
IdP
Sends h(ID$_i$),k
ID$_i$ Exists
Y
N
Request Rejected
Select new ID$_i$
Generates 'rand'. Computes I= h(s), V$_i$= h(h(ID$_i$) || k) ⊕ rand; key$_i$ = h(k) ⊕ h(h(ID$_i$)|| I); M$_i$ = h(h(ID$_i$)|| I) ⊕ h(h(ID$_i$)|k||rand)
2nd Authentication factor
Crypto-token
Mobile Phone
Stores V$_i$, key$_i$, M$_i$ in the crypto-token
U$_i$ stores g_0 , p in the Crypto-token/mobile-token
Stores V$_i$, key$_i$, M$_i$ in the secret file
Prompts U$_i$ to download Mobile App
U$_i$ downloads app and the secret file. Stores g_0 , p in the file

Figure 5.3 User Registration Phase of Integrated-Framework Protocol

5.3.3 Login and Authentication Phase

This phase illustrated in Figure 5.4, is invoked whenever the user attempts to login to access the services of a registered service provider. Login and Authentication phase supports two use case scenarios (i) Authentication of user is done by the service provider (ii) Authentication of user is done by the Identity Provider.

i. Direct Authentication by the Service Provider

L1: U_i attempts to access the services of service provider (SP). SP prompts U_i to enter his identity ID_i. SP checks whether h(ID_i) exists in his database. If not U_i is re-directed for registration to IdP. Otherwise, SP generates a nonce N_1 and computes $C_1 = h(h(ID_i) \parallel I) \oplus N_1$. U_i is prompted to select his authentication factor from the given choices of Crypto-Token and Mobile-Token. If U_i selects Crypto-token, SP sends $< h(ID_i), C_1>$ to U_i and then he is prompted to insert Crypto-token and enter his password 'PW_i'. Crypto-token proceeds to step L2 to generate the authentication request.

or

If U_i selects the Mobile-token, then the server generates a QR code embedded with <$h(ID_i)$, C_1, URL of SP> and prompts U_i to scan the QR code. The QR code scanning application of the mobile app, captures the QR code and U_i is prompted to enter his password 'PW_i' in the mobile interface. The mobile app proceeds to step L2 to generate the authentication request.

L2: The Crypto-token / Mobile app calculates $b = h(PW_i)$, $k = g_0{}^{b} \bmod p$, $rand = V_i \oplus h(h(ID_i)\|k)$, $h(h(ID_i)\|I) = Key_i \oplus h(k)$ and

$M_i' = h(h(ID_i)\|I) \oplus h(h(ID_i)\|k\| rand)$. Crypto-token / Mobile app compares M_i' with Mi stored in the file. On equality, U_i is considered as a valid user by the Crypto-token / Mobile app and proceeds to step L3 to generate the authentication request. If there is a mismatch, request for login is rejected.

The checking of the password at client side prevents unauthorized users from submitting invalid login requests to the server and thus eliminates the chances of DOS attack.

L3: U_i generates 'N_2', a random nonce. U_i calculates

$t = Key_i \oplus h(k) = h(h(ID_i)\|I)$,

$N_1' = t\oplus C_1$, $l = t\oplus N_2$, $j = h(k) \oplus(l\| N_2)$, $CID_i = h((j\|t) \| h(k)\| N_1')$. U_i sends $< h(ID_i), l, j, CID_i>$ to Service provider Sj.

L4: S_j on receiving $< h(ID_i), l, j, CID_i>$ proceeds to compute the following:

$I' = h(S)$, $t' = h(h(ID_i) \| I')$, $N_2' = l \oplus t'$, $h(k)' = j \oplus (l \| N_2')$,

$CID_i' = h((j\|t) \| h(k)' \| N_1)$.

SP checks the freshness of the nonce and compares CID_i' with the received CID_i and if equal, successfully authenticates U_i and executes step L5**.** Else request for login is rejected.

L5: Service provider S_j computes $F = h(h(k)' \| t')$, $B = h(h(CID_i') \| F\| N_2')$, $C_2 = N_2' \oplus N_1$ and sends $\{B, C_2\}$ to U_i.

L6: On receiving the response, U_i calculates $C_2' = N_2 \oplus N_1$ where N_2 is its own nonce generated during this session and N_1 is the nonce received from SP during this session. Thus U_i checks the freshness of the nonce values to avoid the possibility of a replay attack. U_i computes

$B' = h(h(CID_i)\| F' \| N_2')$. B' is compared with B and if equal U_i authenticates Service provider Sj. The freshness of the nonce 'N_2', in the response 'B' from SP, assures U_i that the message is not a replay. Thus mutual authentication is done successfully upon which U_i and S_j calculate the session key, $SK = h(h(ID_i) \| N_2 \| j \| t \| N_1)$. This session key SK is used to secure the communications exchanged thereafter between U_i and S_j.

ii. Brokered Authentication by the Identity Provider

L1: U_i attempts to login to SP. SP generates a SAML authentication request and redirects U_i to IdP. Authentication server (AS) of the IdP does the authentication process. AS of IdP prompts U_i to enter his identity ID_i. AS checks whether $h(ID_i)$ exists in the database. If not U_i is re-directed for registration. Otherwise, authentication server of IdP, generates a nonce N_1 and computes $C_1 = h(h(ID_i) \| I) \oplus N_1$. U_i is prompted to select his authentication factor from the given choices of Crypto-Token and Mobile-Token. If U_i selects Crypto-token, IdP sends < $h(ID_i)$, C_1> to U_i and then he is prompted to insert Crypto-token and enter his password 'PW_i'. Crypto-token proceeds to step L2 to generate the authentication request.

or

If U_i selects the Mobile-token, then the AS generates a QR code embedded with <$h(ID_i)$, C_1, URL of SP> and prompts U_i to scan the QR code. The QR code scanning application of the mobile app, captures the QR code and U_i is prompted to enter his password 'PW_i' in the mobile interface. The mobile app proceeds to step L2 to generate the authentication request.

L2: The Crypto-token / Mobile app calculates $b = h(PW_i)$, $k = g_0{}^{b} \bmod p$, $rand = V_i \oplus h(h(ID_i)\|k)$, $h(h(ID_i)\|I) = Keyi \oplus h(k)$ and

$M_i' = h(h(ID_i)\|I) \oplus h(h(ID_i)\|k\| rand)$. Crypto-token / Mobile app compares M_i' with M_i stored in the file. If equal, U_i is considered as a valid user by the Crypto-token / Mobile app and proceeds to step L3 to generate the authentication request. If there is a mismatch, request is rejected.

L3: U_i generates 'N_2', a random nonce. U_i computes

$t = Key_i \oplus h(k) = h(h(ID_i)\|I)$, $N_1' = t \oplus C_1$, $l = t \oplus N_2$,

$j = h(k) \oplus (l\| N_2)$, $CID_i = h((j\|t) \| h(k)\| N_1')$.

U_i sends < $h(ID_i)$, l, j, CID_i> to AS.

L4: AS on receiving < $h(ID_i)$, l, j, CID_i> proceeds to compute the following:

$I' = h(S)$, $t' = h(h(ID_i) \| I')$, $N_2' = l \oplus t$, $h(k)' = j \oplus (l \| N_2')$,

$CID_i' = h((j\|t') \| h(k)' \| N_1)$.

AS checks the freshness of the nonce and compares CID_i' with the received CID_i and if equal, successfully authenticates U_i and executes from step L5**.** If there is no match, request is rejected.

L5: AS computes $F = h(h(k)' \| t')$, $B = h(h(CID_i') \| F \| N_2')$, $C_2 = N_2' \oplus N_1$ and sends $\{B, C_2\}$ to U_i.

L6: On receiving the response, U_i computes $C_2' = N_2 \oplus N_1$ where N_2 is its own nonce generated during this session and N_1 is the nonce received from AS of IdP during this session. Thus U_i checks the freshness of the nonce values to avoid the possibility of a replay attack. U_i computes $B' = h(h(CID_i) \| F' \| N_2')$ where $F' = h(h(k) \| t)$. B' is compared with B and if equal U_i authenticates IdP. The freshness of the nonce 'N_2', in the response 'B' from IdP, assures U_i that the message is not a replay. Thus mutual authentication is done successfully upon which U_i and AS of IdP agree upon the session key, $SK = h(h(ID_i) \| N_2 \| j \| t \| N_1)$.

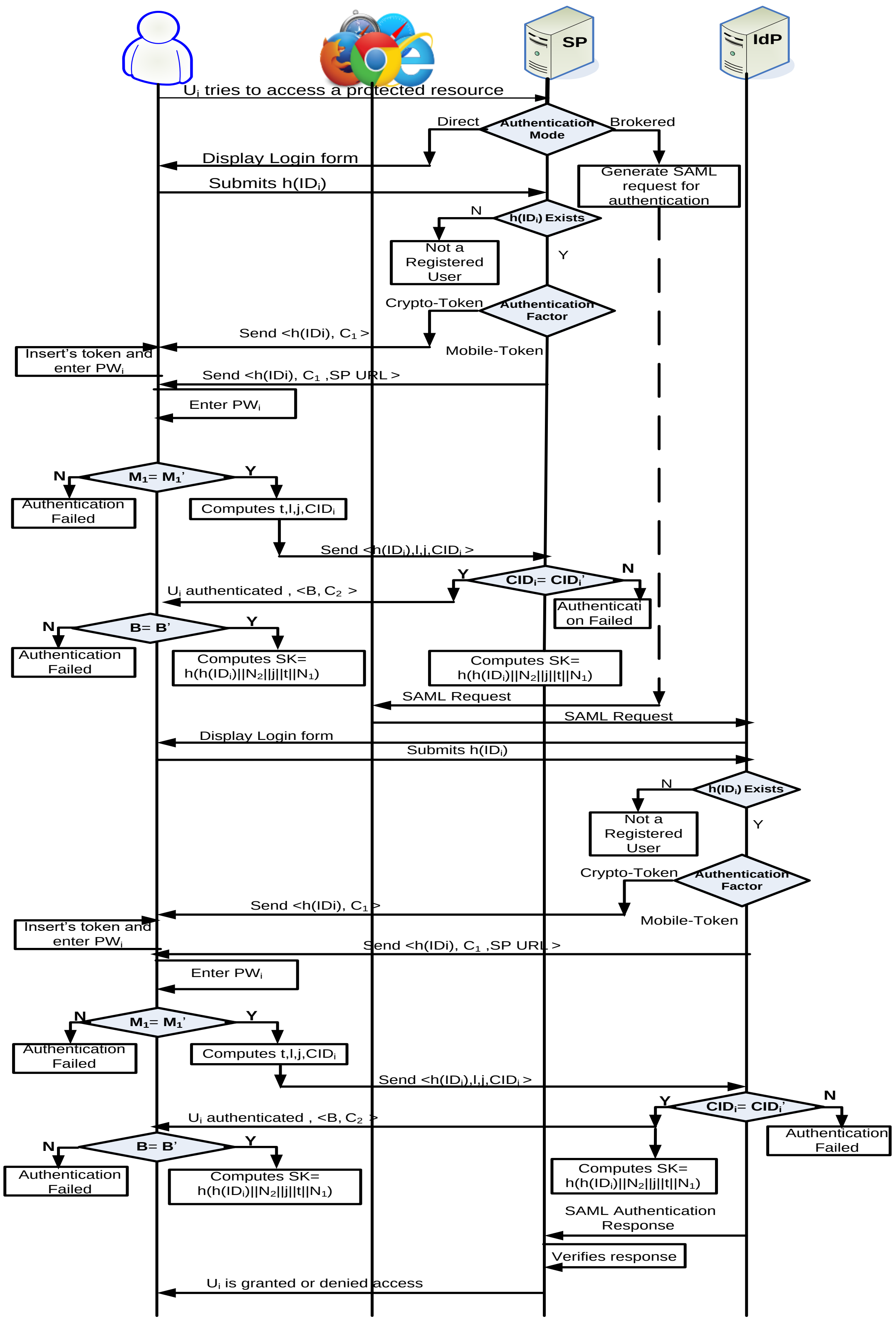


Figure 5.4 Login and Authentication Phase of Integrated-Framework Protocol

L7: SAML authentication response generated by IdP and session key generated between U_i and AS of IdP is communicated to the service provider S_j. Session key is encrypted using the key that is shared between IdP and service provider S_j. This key is maintained by the key management component of IdP corresponding to unique server ID, SID_j of each service provider S_j. The unique server ID, SID_j will be included in the SAML authentication request generated by S_j which is sent to the IdP when the user is re-directed to the IdP for authentication.

5.3.4 Password Change Phase

This phase illustrated in Figure 5.5 is invoked when the user wants to change the password without the intervention of the service provider or the Identity Provider.

P1: If the second authentication factor is a Crypto-token, U_i inserts the Crypto-token into the system and sends the password change request. Crypto-token prompts U_i to enter his identity ID_i and password 'PW_i'. Then the Crypto-token proceeds to P2 to modify the password in the device. If the second authentication factor is the mobile phone, U_i selects the password change option in the mobile app. U_i is prompted to enter his identity ID_i and password 'PW_i'.

P2: The Crypto-token / Mobile app calculates $b = h(PW_i)$, $k = g_0{}^b \bmod p$, $rand' = V_i \oplus h(h(ID_i)||k)$, $h(h(ID_i)||I) = Key_i \oplus h(k)$ and

$M_i' = Key_i \oplus h(k) \oplus h(h(ID_i)||k|| rand)$. Crypto-token / Mobile app checks whether it is equal to the M_i stored in the token. If so, U_i is considered as a valid user by the Crypto-token / Mobile app and prompts the user U_i to enter the new password PW_{inew}. Otherwise the password change request is rejected.

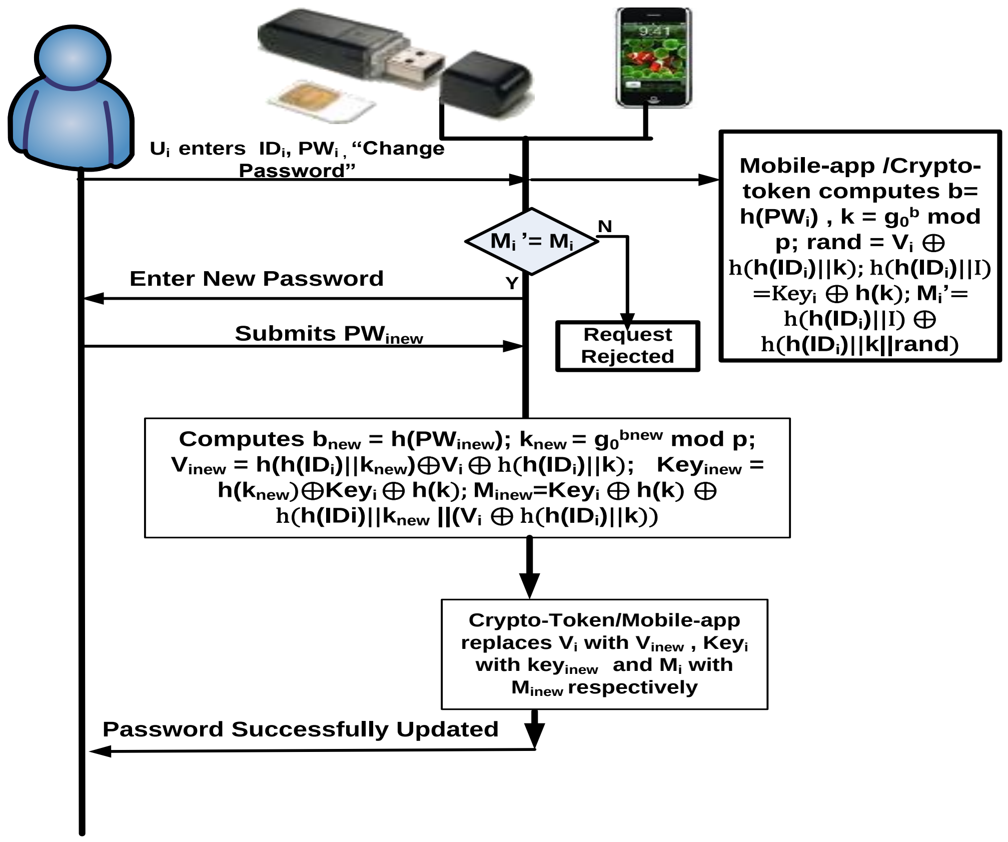
P3: The Crypto-token / Mobile app calculates $b_{new} = h(PW_{inew})$,

$k_{new} = g_0^{\ bnew} \bmod p$, $V_{inew} = h(h(ID_i) \| k_{new}) \oplus V_i \oplus h(h(ID_i) \| k)$,

$Key_{inew} = h(k_{new}) \oplus Key_i \oplus h(k)$

and $M_{inew} = (Key_i \oplus h(k)) \oplus h(h(ID_i) \| k_{new} \| (V_i \oplus h(h(ID_i) \| k)))$.

The Crypto-token / Mobile app replaces V_i, Key_i M_i with V_{inew} , Key_{inew} and M_{inew} respectively.



U_i enters ID_i, PW_i , "Change Password"
Mobile-app /Crypto-token computes $b = h(PW_i)$, $k = g_0^b \bmod p$; $rand = V_i \oplus h(h(ID_i)||k)$; $h(h(ID_i)||I) = Key_i \oplus h(k)$; $M_i' = h(h(ID_i)||I) \oplus h(h(ID_i)||k||rand)$
$M_i' = M_i$
N
Y
Enter New Password
Request Rejected
Submits PW_{inew}
Computes $b_{new} = h(PW_{inew})$; $k_{new} = g_0^{bnew} \bmod p$; $V_{inew} = h(h(ID_i)||k_{new}) \oplus V_i \oplus h(h(ID_i)||k)$; $Key_{inew} = h(k_{new}) \oplus Key_i \oplus h(k)$; $M_{inew} = Key_i \oplus h(k) \oplus h(h(IDi)||k_{new} ||(V_i \oplus h(h(ID_i)||k))$
Crypto-Token/Mobile-app replaces V_i with V_{inew} , Key_i with key_{inew} and M_i with M_{inew} respectively
Password Successfully Updated

Figure 5.5 Password Change Phase of Integrated-Framework Protocol

5.4 ANALYSIS OF PROPOSED INTEGRATED-FRAMEWORK PROTOCOL

This section briefly discusses the security, efficiency and formal analysis of the proposed protocol.

5.4.1 Security Analysis

Security analysis of the protocol verifies the resistance of proposed protocol to different attacks.

i. **Mutual Authentication:** In the case of direct authentication, this phase is executed between the SP and the user, U_i. In the case of brokered authentication, this phase is executed between the Identity Provider (IdP) and the user, U_i.

In authentication phase, the SP/IdP calculates, $CID_i' = h((j||t) \,||\, h(k)' \,||\, N_1)$ and checks with CID_i received from U_i. If equal, SP/IdP successfully authenticates U_i and sends the response $\{B, C_2\}$ to U_i. U_i computes $B' = h(h((CID_i') \,||\, F' \,||\, N_2)$ and compares with B received from SP/IdP. Again U_i computes $C_2' = N_2 \oplus N_1$ where N_2 is its own nonce generated during this session and N_1 is the nonce received from AS of IdP during this session. If $B' = B$ and $C_2' = C_2$, U_i successfully authenticates SP/IdP. Thus, the proposed integrated protocol acheieves mutual authentication, which is one among the requirements of a strong two-factor authentication protocol.

ii. **User Impersonation Attack:** To impersonate a valid user, an adversary should be able to generate a fresh CID_i to pass SP/IdP's authentication. Computing CID_i requires h(k) and N_1. Here h(k) is the hash value of a user's password in a modified form ($k = g_0^{\ b} \bmod p$) and N_1 is the session dependent IdP generated nonce value. The h(k) can neither be extracted from the Crypto-token/Mobile -token nor can it be retrieved from any authentication related data transmitted during execution of protocol.

iii. **Server Impersonation Attack:** To impersonate the SP/IdP to fool the requesting user, an adversary should forge the message {B, C_2} to respond to authentication request {h(ID_i), l, j, CID_i} sent by U_i. However, to compute B, the adversary should know the session dependent nonce N_1 generated by SP/IdP and the value t. Even if he attempts to compute N_1 from C_1, he should know the value of h(h(ID_i) || I). Both the nonce values N_1 and N_2 are never send across the communication channel in the plain text form. Also to calculate the value ‘t’ which is never transmitted across the communication channel, the adversary should know server’s secret key ‘S’. Thus the adversary will not be able to generate a valid response to impersonate the SP/IdP.

iv. **Replay Attack:** This involves capturing messages exchanged between a valid user and a server and retransmitting the same later. Time stamps are commonly used to resist replay attacks. However, in a distributed cloud environment, using time stamps might lead to time synchronization problems if the clocks of sender and receiver are not synchronized properly. Hence the proposed integrated protocol uses nonce values to overcome replay attacks. In each session, the SP/IdP and U_i generate different nonce values N_1 and N_2 respectively. The attacker must send a fresh message {h(ID_i), l, j, CID_i} to pass authentication by the SP/IdP or send fresh message {B, C_2} to be authenticated by user. This mechanism involving challenges and responses using session dependent nonce values can overcome replay attack.

v. **Key Secrecy:** In the proposed protocol, session key is derived from the session dependent nonce values N_1 and N_2 generated by SP/IdP and U_i

respectively. Adversary cannot calculate the shared secret SK = $h(h(ID_i) \| N_2 \| j \| t \| N_1)$ from the eavesdropped authentication messages exchanged between U_i and SP/IdP.

vi. Known-Key security: This property ensures that a future session key cannot be generated based on a compromised past session key. For every request for login, the protocol generates session varying nonce values N_1 and N_2 to calculate the session key SK. Hence session keys of different runs of protocol are independent and one compromised session key will not reveal information required to calculate other session keys.

vii. Forward Secrecy: Forward secrecy property ensures that even if the attacker manages to obtain the master secret 'S' of the Identity Provider, it will not contribute to the compromise of any previous session. In the integrated Protocol, the random values N_1 and N_2 are independent among every protocol execution. Hence, the compromise of the user's password 'PW_i' or the IdP's master secret 'S' will not result in a compromise of past session keys. The protocol thus achieves forward secrecy.

viii. Privileged administrator resilience: In the proposed scheme, a user can choose his password 'PW_i' on his own for registration. The user obfuscates PW_i by computing $b = h(PW_i)$, $k = g_0{}^{b} \bmod p$ and sends 'k' to the IdP. Without solving the discrete logarithm problem and by reversing the hash value in polynomial time, the IdP has no option to retrieve or guess the password 'PW_i'. Proposed protocol is thus resistant to attacks by the attacker with administrator's privilege.

Proposed protocol does not require password or verification table. This eliminates maintenance cost of password information and avoids the probability of stolen-verifier attacks.

ix. Key Control Resilience: The authentication phase of the proposed integrated protocol provides mutual authentication and generation of the session key SK = $h(h(ID_i) \| N_2 \| j \| t \| N_1)$ to secure the transmitted messages. Since N_1 and N_2 used for session key generation, created independently and randomly by U_i and SP/IdP for each session, even the user U_i and the server cannot speculate the session key value in advance.

x. Independent Password Selection: In the proposed protocol, user can select his password during registration phase and he can change his password, independent of IdP or SP, in the password change phase. The scheme allows the crypto-token/mobile-token holder to modify password without the assistance of Identity Provider or service provider. Crypto-token verifies the current password of user before changing password so as to prevent unauthorized users from easily changing the password if they obtain the crypto-token/mobile-phone of some other registered user. Thus only a valid user who knows the correct ID and password, corresponding to the crypto-token/mobile-token can change the password.

xi. Two-Factor Security: Assume that an adversary understands a valid user's password 'PW_i'. If he desires to impersonate the user to login to the server, he should generate the login request, ($h(ID_i)$, l, j, CID_i). However, without knowing the server's secret key 'S' and random nonce value 'N_2' he cannot compute the parameter 'l'. Again without knowing g_0 and p stored in the Crypto-token/Mobile-token, the adversary cannot create

$CID_i = h((j||t)||h(k)|| N_1)$.

Assume that an adversary gets a valid user’s Crypto-token/Mobile-phone. In such a scenario, we discuss the possibility of two of the most common attacks.

Case 1: Offline-guessing Attack: If a Crypto-token/Mobile-phone is lost the adversary gets $V_i = h(h(ID_i)||k) \oplus rand$; $Key_i = h(k) \oplus h(h(ID_i)||I)$;

$M_i = h(h(ID_i)||I) \oplus h(h(ID_i)||k|| rand)$, where $I = h(S)$. When the adversary attempts to check the correctness of a guessed password, he will not be able to verify without knowing the secret key ’S’ of the server or the ‘rand’ value.

Case 2: *Impersonation Attack:* Consider a scenario where an attacker has stolen a Crypto-token/Mobile-phone of a valid user, but does not know the password. If the attacker attempts to impersonate user ‘U_i’ to gain access to server, he cannot create the valid request $\{h(ID_i), l, j, CID_i\}$ without knowing the user’s password ‘PW_i’, and the nonce values ‘N_1’, ‘N_2’.

From the above analysis, it is evident that the proposed protocol offers two-factor security.

xii. Availability: When user attempts to change his password, by giving a request for change, he is verified by the Crypto-token/Mobile-token before the request is accepted. If the adversary obtains the user’s Crypto-token/Mobile-phone temporarily, without knowledge of current password of the user, adversary will not be able to modify the password. Also without knowing the correct password, it is not possible to generate the login request which makes the protocol resistant to Denial-of-Service attack. Again, since the SP/IdP does not maintain password/verification table, there is no need to synchronize the password with user.

xiii. Man-in-the-Middle Attack: In the proposed protocol, if the adversary modified any of the message exchanged between the client and the server, then the session will be terminated. For example, assume that ID_i is modified into ID_i^* in the authentication message $\{h(ID_i), l, j, CID_i\}$ exchanged during the login phase. The server during the login phase checks whether an ID_s corresponding to the ID_i^* is there in its user table. If it is not there, then the login request will be rejected.

If IDi^* is some other user's ID, then CID_i' calculated using the following computations, $I' = h(S)$, $t' = h(h(ID_i) \| I')$, $N_2' = l \oplus t$, $h(k)' = j \oplus (l \| N_2')$, $CID_i' = h((j\|t) \| h(k)' \| N_1)$ where $k = g_0{}^b \mod p$ corresponds to the password of IDi^*. Hence, this attack will fail since the adversary will not be able to impersonate a valid user without knowing his password.

xiv. Security against Stolen Verifier Attack: In the proposed scheme, only $h(ID_s)$ and profile information are stored in the server. Anyways, using $h(ID_s)$ alone, the attacker cannot compute values used for authentication and hence the attack will fail.

5.4.2 Efficiency Analysis

This section analyzes the efficiency of the proposed protocol in terms of the computational and the communication cost. It is assumed that ID_i, PW_i, nonce values are 128 bits long and the output of hash function (SHA-2) is 256 bits long. The variable 'rand' is a 256-bit hexadecimal value which is uniquely generated for each user. Let T_h T_x, T_e and T_c denote the time complexity for hashing, XOR, exponentiation and concatenation operations respectively. In the protocol, the parameters stored in the crypto-token/mobile-token are V_i, Key_i, M_i, g_0, p and the memory (E1) needed in the crypto-token/mobile-token is 1024 (3*256 +2 *128) bits. Communication cost of Login and Authentication phase (E2) includes the capacity of transmitting parameters {h(ID_i), l, j, CID_i, C_1, B, C_2} which makes E2 equal to 1664 (6*256 + 128) bits. The computation cost of user registration (E3) is the total time of all operations executed in this phase by the user and Registration server and is equal to $6T_h + 3T_x + 1T_e + 3T_c + 1T_s + 1T_d$. The computation cost of the user (E4) and the server (E5) authentication is the total time of all operations executed by the crypto-token/mobile-token and Server during login and authentication phase. During login & authentication, the crypto-token/mobile-token performs 7 hash functions, 8 concatenation operations, 6 XOR and 1 exponentiation making E4 equal to $7T_h + 8T_c + 6T_x + 1T_e$. Similarly, E5 is equal to $5T_h + 10T_c + 4T_x$. The computation cost of password changing (E6) is the total time of all operations executed in this phase by the user and is equal to $8T_h + 6T_x + 5T_c + 2T_e$. A comparison of computational efficiency with other protocols is shown in Table 5.2.

Table 5.2 Comparison of Computational Efficiency with Other Protocols

	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Proposed Integrated Protocol	1024 bits	1664 bits	$6T_h + 3T_x + 1T_e + 3T_c + 1T_s + 1T_d$	$7T_h + 6T_x + 8T_c + 1T_e$	$5T_h + 4T_x + 10T_c$	$8T_h + 6T_x + 5T_c + 2T_e$.
Choudhary et al. [2011]	1024 bits	1920 bits	$6T_h + 3T_x + 1T_e + 2T_c$	$10T_h + 2T_x + 1T_e + 3T_c$	$8T_h + 1T_x + 1T_e + 3T_c$.	$4T_h + 4T_x$
Jaidhar [2013]	1024 bits	1664 bits	$5T_h + 5T_x + 1T_e + 5T_c$	$6T_h + 2T_x + 9T_c + 2T_s + 1T_d$	$5T_h + 1T_x + 8T_c + 2T_d + 1T_s + 1T_e$	$3T_h + 2T_x + 3T_c$
Rui Jiang [2013]	768 bits	1152 bits	$4T_h + 1T_x + 1T_e + 1T_c$	$7T_h + 1T_x + 4T_c + 1T_d + 1T_e$	$7T_h + 5T_c + 1T_s + 1T_e$	$18T_h + 3T_x + 11T_c + 2T_s + 1T_d + 4T_e$

The analysis results of computational efficiency, demonstrate that the proposed integrated-framework protocol is comparable with other similar protocols, in terms of memory needed to store the parameters, communication cost during authentication and computation cost of registration, login and password change phase.

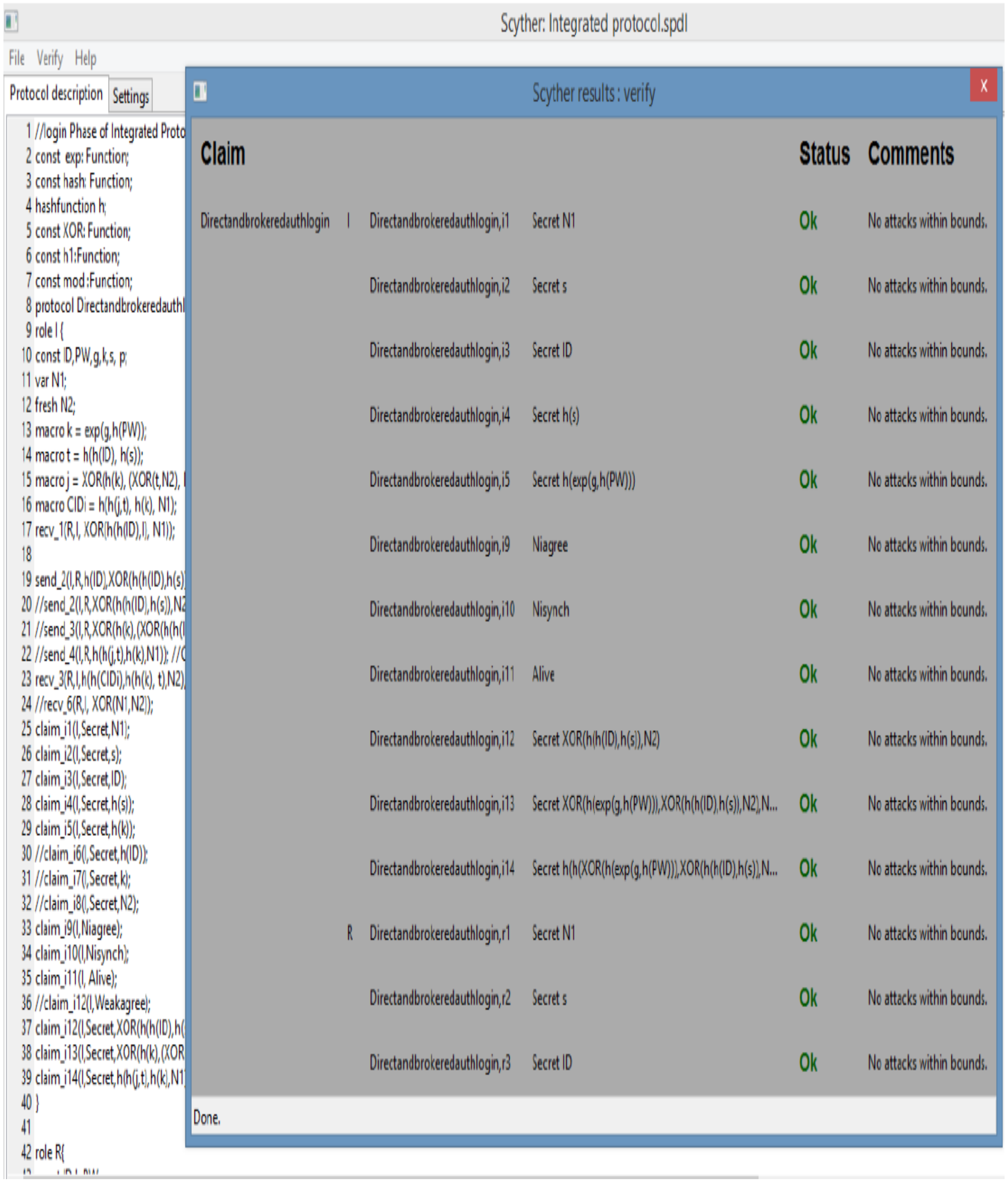
5.4.3 Formal Analysis

The formal analysis of protocol is done using Scyther tool. The strength of the protocol is verified using Scyther tool which ascertains the strength by evaluating the resistance of the protocol to various attacks. Scyther uses strand space model for formalizing logic and uses Dolev-Yao model for modelling the network, which caters to the requirement of a mathematical approach for validating the protocol. The analysis results of login phase are shown in Figure 5.6**.** The protocol is written in SPDL as follows**:**

```
//login Phase of Integrated Protocol
const  exp: Function;
const hash: Function;
hashfunction h;
const XOR: Function;
const h1:Function;
const mod :Function;
protocol Directandbrokeredauthlogin(I,R){
role I {
const ID,PW,g,k,s, p;
var N1;
fresh N2;
macro k = exp(g,h(PW));
macro t = h(h(ID), h(s));
macro j = XOR(h(k), (XOR(t,N2), N2));
macro CIDi = h(h(j,t), h(k), N1);
recv_1(R,I, XOR(h(h(ID),I), N1));

send_2(I,R,h(ID),XOR(h(h(ID),h(s)),N2),XOR(h(k),(XOR(h(h(ID),h(s)),N2
),N2)),h(h(j,t),h(k),N1)); //h(ID) , l, j , CIDi
//send_2(I,R,XOR(h(h(ID),h(s)),N2)); //l
//send_3(I,R,XOR(h(k),(XOR(h(h(ID),h(s)),N2),N2))); //j
//send_4(I,R,h(h(j,t),h(k),N1)); //CIDi
```

```
recv_3(R,I,h(h(CIDi),h(h(k), t),N2),XOR(N1,N2)); //B and C
//recv_6(R,I, XOR(N1,N2));
claim_i1(I,Secret,N1);
claim_i2(I,Secret,s);
claim_i3(I,Secret,ID);
claim_i4(I,Secret,h(s));
claim_i5(I,Secret,h(k));
//claim_i6(I,Secret,h(ID));
//claim_i7(I,Secret,k);
//claim_i8(I,Secret,N2);
//claim_i9(I,Niagree);
claim_i10(I,Nisynch);
claim_i11(I, Alive);
claim_i12(I,Weakagree);
claim_i12(I,Secret,XOR(h(h(ID),h(s)),N2)); // claim for l
claim_i13(I,Secret,XOR(h(k),(XOR(h(h(ID),h(s)),N2),N2))); //claim for j
claim_i14(I,Secret,h(h(j,t),h(k),N1));  //claim for CIDi
}

role R{
const ID,k,PW,g,s;
fresh N1;
fresh N2;
send_1(R,I, XOR(h(h(ID),I), N1));
recv_2(I,R,h(ID),XOR(h(h(ID),h(s)),N2),XOR(h(k),(XOR(h(h(ID),h(s)),N2)
,N2)),h(h(j,t),h(k),N1)); //h(ID) , l, j , CIDi
send_3(R,I,h(h(CIDi),h(h(k), t),N2),XOR(N1,N2)); //B and C

claim_r1(R,Secret,N1);
```

```
claim_r2(R,Secret,s);
claim_r3(R,Secret,ID);
//claim_r4(R,Secret,h(s));
//claim_r5(R,Secret,h(k));
//claim_r6(R,Secret,h(ID));
//claim_r7(R,Secret,N2);
//claim_r8(R,Alive);
//claim_r9(R,Niagree);
//claim_r10(R,Nisynch);
//claim_r11(R,Weakagree);
//claim_r12(R,Secret,XOR(h(h(ID),h(s)),N2)); //claim for l
//claim_r13(R,Secret,XOR(h(k),(XOR(h(h(ID),h(s)),N2),N2))); //claim for j
//claim_r14(R,Secret,h(h(ID),h(k),N1));  //claim for CIDi
}
}
```



```
//login Phase of Integrated Proto
const  exp: Function;
const hash: Function;
hashfunction h;
const XOR: Function;
const h1:Function;
const mod :Function;
protocol Directandbrokeredauthl
role I {
const ID,PW,g,k,s, p;
var N1;
fresh N2;
macro k = exp(g,h(PW));
macro t = h(h(ID), h(s));
macro j = XOR(h(k), (XOR(t,N2), I
macro CIDi = h(h(j,t), h(k), N1);
recv_1(R,I, XOR(h(h(ID),I), N1));

send_2(I,R,h(ID),XOR(h(h(ID),h(s))
//send_2(I,R,XOR(h(h(ID),h(s)),N2
//send_3(I,R,XOR(h(k),(XOR(h(h(I
//send_4(I,R,h(h(j,t),h(k),N1)); //C
recv_3(R,I,h(h(CIDi),h(h(k), t),N2),
//recv_6(R,I, XOR(N1,N2));
claim_i1(I,Secret,N1);
claim_i2(I,Secret,s);
claim_i3(I,Secret,ID);
claim_i4(I,Secret,h(s));
claim_i5(I,Secret,h(k));
//claim_i6(I,Secret,h(ID));
//claim_i7(I,Secret,k);
//claim_i8(I,Secret,N2);
claim_i9(I,Niagree);
claim_i10(I,Nisynch);
claim_i11(I, Alive);
//claim_i12(I,Weakagree);
claim_i12(I,Secret,XOR(h(h(ID),h(
claim_i13(I,Secret,XOR(h(k),(XOR
claim_i14(I,Secret,h(h(j,t),h(k),N1
}

role R{
```

```
Claim                                                                            Status  Comments

Directandbrokeredauthlogin  I  Directandbrokeredauthlogin,i1   Secret N1                                  Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i2   Secret s                                   Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i3   Secret ID                                  Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i4   Secret h(s)                                Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i5   Secret h(exp(g,h(PW)))                     Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i9   Niagree                                    Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i10  Nisynch                                    Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i11  Alive                                      Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i12  Secret XOR(h(h(ID),h(s)),N2)               Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i13  Secret XOR(h(exp(g,h(PW))),XOR(h(h(ID),h(s)),N2),N...  Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,i14  Secret h(h(XOR(h(exp(g,h(PW))),XOR(h(h(ID),h(s)),N...  Ok   No attacks within bounds.
                            R  Directandbrokeredauthlogin,r1   Secret N1                                  Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,r2   Secret s                                   Ok   No attacks within bounds.
                               Directandbrokeredauthlogin,r3   Secret ID                                  Ok   No attacks within bounds.

Done.
```

Figure 5.6 Formal Analysis of Integrated-Framework Protocol

To carry out formal security analysis, this section focuses on verifying the vulnerability of a few parameters such as IDi, K, S, l, j, N_1, N_2, CID_i which are used in the proposed authentication scheme. If there is a high vulnerability, then the argument that the proposed authentication scheme is secure will not be justifiable. There are various claims made as part of the security analysis and these claims are validated by executing and analyzing the proposed scheme using Scyther. The “No attack” results shown in Figure 5.6 proves that Scyther validates all the claims made as part of security analysis.

Claim 1: The proposed scheme is designed to ensure the secrecy of the user ID, throughout the registration and authentication process.

The user ID is submitted in the hashed form to the registration authority during the registration process. This is used along with the password and the secret key of IdP to generate the secret parameters to be stored in the crypto-token/mobile-token. During the authentication process, user ID is hashed and sent to the SP/IDP along with other parameter’s in the authentication request. Security claim that user ID, ID_i is safe is verified by Scyther.

Claim 2: The proposed scheme is designed to ensure the secrecy of the variant of password ‘k’ throughout the registration and authentication process.

The password is never transmitted in the plaintext form either to the registration authority or to the service providing server. It is converted into a modified form ‘k’, by finding the hash of the password viz. ‘b’ and then finding exponentiation of g_0 (generator of a cyclic group) to the power of ‘b’. Now to obtain the password from ‘k’, the DLP should be solved. During the authentication process, password is used to generate login request. It is not sent to service provider, but it is used to calculate the parameters in the authentication request. Also the password is not stored anywhere other than the crypto-token/mobile-token which is in the possession of the owner of the password. Scyther results validate the claim that ‘k’ remains a secret.

Claim 3: The proposed scheme requires the master key ‘S’ of the Identity Provider to be a secret

‘S’ is the secret key of the Identity Provider. It is used in its hashed form to compute the parameters to be stored in the crypto-token/mobile-token and to verify the user during the authentication process. Scyther validated the claim that ‘S’ is safe.

Claim 4: The proposed scheme requires that the authentication parameter ‘l’ is secret

The parameter ‘l’ is the information containing hash of user ID concatenated with the IdP’s master secret ‘S’, XOR-ed with the nonce N_2 generated by the user. ‘l’ is one of the authentication parameters sent by the user to the SP/IdP which is verified by the SP/IdP to ensure the authenticity of the user. Scyther validated the claim that ‘l’ is safe.

Claim 5: The proposed scheme requires that the authentication parameter ‘j’ is secret

The parameter ‘j’ is the information containing hash of ‘k’ (variant of user’s password) XOR-ed with ‘l’ concatenated with the nonce N_2 generated by the user. ‘j’ is one of the authentication parameters sent by the user to the SP/IdP which is verified to ensure the authenticity of the user and Scyther validated the claim that ‘j’ is safe.

Claim 6: The proposed scheme requires that the nonce ‘N_1’ is secret

The nonce ‘N_1’ is randomly generated by the SP/IdP during each session. ‘N_1’ is unique for each session which makes each authentication request unique and thus eliminates the probability of a replay attack. Scyther validated the claim that ‘N_1’ is safe.

Claim 7: The proposed scheme requires that the nonce 'N_2' is secret

The nonce 'N_2 is randomly generated by the U_i during each session to calculate the parameters in the authentication request sent by the user to SP/IdP. 'N_2' is unique for each session which makes each authentication request unique and thus eliminates the probability of a replay attack. Scyther validated the claim that 'N_2' is safe.

Claim 8: The proposed scheme requires that the authentication parameter 'CID_i' is secret

The parameter 'CID_i' is the one of the parameter's in the authentication request sent from the user to the server, which contains the user ID, hash of the obfuscated password of the user and the nonce N_1 generated by the user. The parameters in the authentication request should not reveal any information, which will enable an adversary to forge a valid authentication request. Scyther validated the claim that 'CIDi' is safe.

Claim 9: The scheme assures the user that the server remains alive and also the server is assured that the user remains alive, since both the user and server receives messages from each other prior to making the claim. The Scyther tool validates the aliveness claim.

Claim 10: The scheme assures Niagree between the user (crypto-token/mobile-token) and the server

Niagree claim enforces that the sender (user) and the receiver (server) agree upon the values of variables exchanged during the running of the proposed scheme. During the operation of the proposed scheme, the user and sever can send data without being modified by the adversary and the correctness of the claim is justified by the analysis results.

Claim 11: The proposed scheme holds Synchronization during the authentication process

Ni-Synch or Non-Injective Synchronization property requires that the corresponding send and receive events (1) happened in the correct order and (2) have the same contents. The proposed protocol satisfies this claim as indicated by the result of Scyther analysis.

CONCLUDING REMARKS

This chapter discussed an authentication framework for cloud which provides the service providers with the flexibility to choose between Direct Authentication and Brokered Authentication. The users who are part of the framework can access the service of both the categories of service providers viz. those supporting Direct Authentication and those providing Brokered Authentication along with Single Sign-on functionality. Users can do a one-time registration at the Identity Provider, be issued with an authentication factor such as crypto-token /Mobile token and then authenticate using the same token to access the services of multiple service providers.

CHAPTER 6

CONCLUSIONS

Most of the cloud service providers use password to authenticate its users necessitating the server to maintain a verification table. This requirement of storing verification information of User undermines the security of the authentication system. Though there are Two-Factor authentication schemes used by service providers to overcome the limitation of password based authentication, most of them are based on one-time-passwords (OTP) which require a shared secret seed to be stored by the server. Also, there is no authentication scheme that simultaneously caters to the authentication requirement of various categories of service providers.

The research proposes an authentication framework that addresses the concern of providing a secure and flexible authentication mechanism. This research aims at achieving security for User authentication using Crypto-Token/Mobile-Token as the second factor and providing flexibility of authentication mode to service providers by allowing them to choose between brokered and direct authentication.

6.1 PRESENT WORK

The research work, aimed at designing a secure Authentication Framework by enhancing and mitigating the concerns of the typical and most prevalent password based authentication in the cloud environment**.** The framework is also conceived such that it could be seamlessly applied by different categories of cloud based Service Providers and will also cater to requirement of users that avail the services of multiple categories of Service

Providers. As the first step, the research identified two categories of Service providers in cloud as Direct Authentication Service Providers (DASP) who directly authenticate its users and Brokered Authentication Service Providers (BASP) who use the authentication service of a trusted third party. To cater to the authentication requirements of both the Service Providers, the research proposed two separate authentication architectures comprising of a centralized Identity Provider (IdP), Service Providers and users. In the case of Direct Authentication, users are authenticated by the DASP using the authentication module provided by the IdP and in the case of Brokered Authentication, Users are authenticated by the IdP who also provides the Users with the Single Sign-on functionality.

Next level of the research, proposes user authentication schemes for both DASPs and BASPs. To overcome the limitations of the password based authentication and to address the concern of storing verification information by server, the research proposes Two-Factor Authentication Protocols without Verifier table. The authentication protocols proposed in the research work provides the users with the flexibility to authenticate using either a Crypto-Token or a Mobile-Token and access the services of the Service Providers. At this level, the research work had taken into cognizance issue of the Service Providers having to issue the tokens and the users managing multiple tokens to access different services. The research work addresses this concern by entrusting the IdP with the responsibility of both registering the users and also issuing the Crypto-Token / Mobile-Token. Considering the mode of authentication as Direct or Brokered and considering the authentication factor as Crypto-Token or Mobile-Token, the research has proposed four separate authentication protocols viz. Crypto-token Based

Direct Authentication Protocol without Verifier Table, Mobile-token Based Direct Authentication Protocol without Verifier Table, A Strong Single Sign-on User Authentication Scheme without Verifier Table for Cloud Based Services and A Mobile Based Remote User Authentication Scheme without Verifier Table for Cloud Based Services. In the case of Mobile-Token based protocols, parameters generated by the server process are embedded in QR-Codes which are scanned by the mobile phone to complete the verification process during the Registration and Authentication.

The protocols proposed for Direct Authentication provides mutual authentication and supports session key generation which can be used to secure future communication between the users and the Service Providers. The Direct Authentication protocols are executed by the Authentication Server of the DASP to authenticate users requesting its services. The Brokered Authentication Protocols are executed by the Authentication Server of the IdP to authenticate users re-directed to it by the BASPs. To facilitate the exchange of authentication information about users between the IdP and service providers during the authentication process, proposed protocols require Security Assertion Markup Language (SAML), which provides Single Sign-on functionality.

The authentication architecture and protocols proposed for Direct Authentication can be adopted only by the DASPs to authenticate its Users. Similarly, the authentication architecture and protocols proposed for Brokered Authentication can be adopted only by the BASPs to authenticate Users of its services. The next level of requirement envisaged as part of the research was an authentication model that can be adopted by both the DASPs and the BASPs along with a single two-factor authentication protocol that can be used to authenticate Users of both the categories of Service Providers. The research addressed this requirement by proposing an authentication framework which includes, Framework Architecture, Integrated Authentication Model that provides the flexibility to either directly authenticate the Users or re-direct them to the Centralized Identity Provider, Integrated Two-Factor Mutual Authentication Protocol without Verifier table that allows the User to authenticate to both the DASPs and the BASPs by using a Crypto-Token or a Mobile-Token and a set of components that enables the Identity Provider and the Service Providers to carry out their functionalities.

The strength of the proposed protocols is analyzed by verifying its resistance to common attacks on authentication and it is observed that the protocols are resistant to guessing attack, user impersonation attack, stolen verifier attack, replay attack etc. To verify the efficiency of the proposed protocols the communication and computation costs are compared with similar schemes and it is seen that the costs are comparable. The formal analysis of the proposed protocols is done using the security protocol analyzer “Scyther” and the “No Attacks” results prove that the security claims made as part of the analysis are valid.

6.2 LIMITATIONS OF STUDY AND FUTURE ENHANCEMENTS

This research has resulted in a secure and usable authentication framework that can be adopted by service providers in a Public cloud environment. However, there are many areas that can still be explored to enhance the adoption of the proposed work in a practical business environment. These areas are discussed in this section.

I. *Implementation of Integrated-Framework:* A detailed proof-of-Concept implementation of the proposed authentication framework will be a further improvement to this research considering the following:

- Majority of the existing authentication systems for cloud are based on password authentication and requires a password verification table to be maintained by the server.
- Single sign-on functionality is provided by many of the service providers using proprietary mechanisms which fail to support cross domain Single sign-on**.**
- Though there are many works related to authentication in cloud, there is a lack of a two-factor authentication system that provides the service providers with the flexibility to choose between brokered and direct authentication**.**

Considering the above mentioned gaps, implementation of Integrated Framework will further help to**:**

▪ Evaluate the practical feasibility of implementing a Two-Factor authentication protocol, that provides the user with the flexibility of authenticating using either a Crypto-token or a Mobile-Token**.**

▪ Understand the security benefits of an authentication scheme that does not require the server to store password verification information of user**.**

▪ Achieve cross-domain Single sign-on functionality using Security Assertion Markup Language **(**SAML**).**

▪ Understand the convenience and usability of a flexible authentication system, that allows the service providers to choose between directly authenticating its users or delegating the authentication to a third party**.**

II. *Analyzing the resistance of Protocols to Attacks:* The security analysis of the protocols is done to identify the vulnerability to attacks. Nevertheless, a strong and fool proof analysis of possible attacks and vulnerability evaluation of protocols is to be done for testing their resistance to common attacks such as Cross-Site Request Forgery, Phishing, Cross Site Scripting, Password Guessing, SQL Injection etc**.**

III. *Improving Computational Efficiency***:** Proposed authentication protocols for direct and brokered authentication using Crypto-token and Mobile-Token can be improved to achieve better computational efficiency than the currently achieved level of efficiency**.** This can be made possible by reducing the number of hash and XOR operations**.**

IV. *Zero Knowledge Proof (ZKP)**:*** Furthermore, the feasibility of a usable authentication mechanism without verifier table that provides security using Zero Knowledge Proof protocols can be explored as an enhancement of the current work**.**

V. *EID as an Authentication Factor**:*** As the proposed Integrated Framework for Authentication is flexible enough to accept any type of authentication information of users, electronic ID's **(**EID's**)** equivalent to Aadhar numbers can be used as an authentication factor**.** EID is a knowledge factor, known to the user and aids in providing one more level of verification thereby achieving greater efficiency and security**.**

BIBLIOGRAPHY

1. Abraham D, “Why 2FA in the Cloud?” September 2009, Network Security.
2. Armando A, Basin D, Boichut Y, Chevalier Y, Compagna L, Ceullar J, Hankes P D, Heam P C, Kouchneranko O, Mantovani J, Modersheim S, Oheimb D V, Rusinowitch M, Santiago J, Turuani M, Vigano L and Vigneron L, “The AVISPA Tool for The Automated Validation of Internet Security Protocols and Applications,” in *Proceedings of Computer Aided Verification (CAV)*, Vol. 3576 of LNCS. Springer, pp. 281- 285, 2005.
3. Ashraf U, “Securing Cloud Using Two-Factor Authentication,” 2013, M. Sc Infotech Thesis, University of Stuttgart, [Online], Available: ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/MSTR-3452/MSTR-3452.pdf.
4. Asoke T and Manish C, “Architecting Secure Software Systems”, CRC Press, Taylor and Francis Group, 2009, ISBN:978-1-4200-8785-7, pp.11.
5. Atreya M, “Password Based Encryption,” [Online], Available:https://web.cs.ship.edu/~cdgira/courses/CSC434/Fall2004/docs/course_docs/Article3-PBE.pdf.
6. Aguiar E, Zhang Y and Blanton M, “An overview of Issues and Recent Developments in Cloud Computing and Storage Security”, pp.1-31, Springer, Berlin, 2013.

7. Ahuja S P and Komathukattil D, "A Survey of the State of Cloud Security", *Network and Communication Technologies*, Vol.1, No.2, pp. 66-75, 2012.

8. Alexander S, Marc S, Jacob A, Arjen L, David M, Dag A O and Benne W, "MD5 considered harmful today: Creating a rogue CA certificate," December 30, 2008, [Online], Available: http://www.win.tue.nl/hashclash/rogue-ca/.

9. Allison M, "Global Insurance Assurance Certification Paper," March 29, 2000, [Online], Available: https://www.giac.org/paper/gsec/16/risks-biometric-based-authentication-schemes/100271.

10. Armbrust M, Fox A, Griffith R, Joseph A.D, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I and Zaharia M, "Above the Clouds: A Berkeley View of Cloud Computing," 2009, Technical report UCB/EECS-2009-28. Electrical Engineering and Computer Sciences University of California.

11. Banyal R K, Jain P and Jain V K, "Multi-factor Authentication Framework for Cloud Computing," *in Proceedings of Fifth International conference on Computational Intelligence, Modelling and Simulation*, pp. 105-110, 2013.

12. Barker E, Barker W, Burr W, Polk W and Smid M, "NIST Special Publication 800-57, Recommendation for Key Management-Part 1: General (revision 3)," [Online], Available: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf, 2012.

13. Beachy J, A and Blair W D , “Definition of a Group,” Abstract Algebra, 2nd edition, Chapter 3, Waveland Press, Inc. Illinois, 2005.

14. Bellare M, Canetti R and Krawczyk H, “keying Hash Functions for Message Authentication,” in: Advances in Cryptology – Crypto’96, Lecture Notes in Computer Science, Vol. 1109, ed. N. Koblitz, pp.1-15, Springer, 1996.

15. Bellowin S M and Merritt M, “Augmented Encrypted Key Exchange: A Password Based Protocol Secure Against Dictionary Attacks and Password File in *Proceedings of First ACM Conference on Computer and Communications Security*, pp.244-250, 1993.

16. Bhadauria R and Sanyal S, “Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques,” *International Journal of Computer Applications*, pp. 47-66, 2012.

17. Blanchet B, “An efficient Cryptographic protocol verifier based on Prolog rules,” *in Proceedings of 14th IEEE Computer Security Foundations Workshop (CSFW)*, pp. 82 -96,2001.

18. Blumenthal M, “Encryption: Strengths and Weaknesses of Public Key Cryptography,” Villanova University, Villanova, Computing Research Topics, CSC 3990, 2007, [Online], Available: http://www.csc. villanova.edu/~tway/courses/csc3990/f2007/csrs2007/01-pp1-7-MattBlumenthal.pdf.

19. Boyd C and Mathuria A, “Protocols for Authentication and Key Establishment,” *Information Security and Cryptography*, Springer Science and Business Media, 2013, ISBN: 3662095270,

9783662095270., pp.42.

20. Brian P, “Code Hosting Service Shuts down after Cyber Attack”, June, 2014, http://www.darkreading.com/attacks-breaches/code-hosting-service-shuts-down-after-cyber-attack/d/d-id/1278743.

21. Brawley J and Gao S, “Mathematical Models in Public Key Cryptography,” 1999, [Online], Available: http://www.math.clemson.edu/~sgao/papers/crypto_mod.pdf.

22. Buchanan R T, “Dropbox Passwords Leak: Hundreds of Accounts Leak After Third Party Security Breach,” 2014, INDEPENDENT, [online], Available: http://www.independent.co.uk/life-style/gadgets-and-tech/nearly-seven-million-dropbox-passwords-hacked-pictures-and-videos-leaked-in-latest-third-party-9792690.html.

23. CardLogix, “Smart Card and Security Basics,” 2009, [Online], Available:http://www.smartcardbasics.com/pdf/7100030_BKL_Smart-Card-Security-Basics.pdf.

24. Cryptomate64, Cryptographic USB(Token), Advanced Card Systems Holdings Limited, 2016 [Online], Available: http://www.acs.com/hk/en/products/18/cryptomate64-cryptographic-usb-tokens.

25. Crypt-token, The Right Answer to All Your Authentication Needs Marx, Last Updated 25 April 2016 [Online], Available: https://www.cryptoken.com.

26. Carlin S and Curran C,” Cloud Computing Security,” *International Journal of Ambient Computing and Intelligence*, Vol. 3, pp, 14-19,

2011.

27. Carlson C, “Side-Channel Attacks Threaten Data in the Cloud,” May 30, 2012, [Online], Available: http://www.fiercecio.com/storey/side-channel-attacks-threaten-data-cloud/2012-05-30.

28. Celesti A, Tusa F, Villari M and Puliafito A, “Security and Cloud Computing, “Workshops *on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp.263-265, 2010.

29. Chang C and Lee J, “An Efficient and Secure Multi-Server Password Authentication Scheme Using Smart Cards,” in *Proceedings of the 2004 International Conference on Cyberworlds (CW’04)*, pp.6/1 – 6/6, 2004.

30. Chang Y, Chang C, Su Y, “A Secure Improvement on the User-Friendly Remote User authentication Scheme with No Time Concurrency Mechanism,” in *the Proceedings of 20th International Conference on Advanced Information Networking and Applications*, AINA,” Vol.2, pp.18-20, 2006.

31. Chein H and Jan J, “Robust and Simple Authentication Protocol,” *The Computer Journal*, Vol.46, No.2,2003.

32. Chen Y, Pascon V and Katz R H, “What’s New about Cloud Computing Security?” Technical Report, 2010 [Online], Available: http://www.eecs.berkeley.edu/pubs/Techrpts/2010/EECS.2020-5.pdf.

33. Chen Y and Yeh h, “An Efficient Nonce-Based Authentication Scheme with Key Agreement,” *Science Direct, Applied Mathematics and Computation*, Vol.169, pp.982-994, 2005.

34. Chien H Y and Chen C H, "A Remote Authentication Scheme Preserving User Anonymity," in *Proceedings of Advanced Information Networking and Applications*, Vol.2, pp. 245-248, 2005.

35. Chien H Y, Jan J K, and Tseng Y M, "An Efficient and Practical Solution to Remote Authentication: Smart Card," *Computers & Security*, Vol. 21, No. 4, pp. 372– 375, 2002.

36. Choudhury A J, Kumar P, Sain M, Lim H and Jae-Lee H, "A Strong User Authentication Framework for Cloud Computing," *in Proceedings of IEEE Asia-Pacific Services Computing Conference,* pp.110-115, 2011.

37. Chow, Jakobsson M, Masuoka R, Molina J, Niu Y, Shi E and Song Z, "Authentication in the Clouds: A Framework and its Application to Mobile Users," *CCSW'10*, Chicago, Illinois, USA, 2010.

38. Chung C and Wu T, "Remote Password Authentication with Smart Cards," in *IEE Proceedings-E*, Vol.138, No.3, 1991.

39. Cisco, "Cisco Global Cloud Index: Forecast and Methodology, 2014-2019," 2015, [Online], Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf.

40. Coviello A, "Open Letter to Customers", 2011, [Online], Available: https://www.sec.gov/Archives/edgar/data/790070/000119312511070159/dex991.htm

41. Cremers C and Casimier J F, "Scyther - Semantics and Verification of Security Protocols," 2006, PhD Thesis, [Online], Available:

http://alexandria.tue.nl/extra2/200612074.pdf

42. Cremers C and Lafourcade P, “Comparing State Spaces in Automatic Protocol Verification,” in *Proceedings of the 7th International Workshop on Automated Verification of Critical Systems (AVoCS' 07). ENTCS*, 2007, [Online], Available, https://www.cs.ox.ac.uk/people/cas.cremers/downloads/papers/CrLa2007-Comparing.pdf

43. Cremers C, “The Scyther Tool Verification, Falsification, and Analysis of Security Protocols,” Tool Paper, 2008, [Online], Available: https://www.cs.ox.ac.uk/people/cas.cremers/downloads/papers/Cr2008-Scyther_tool.pdf

44. Cremers C: “The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols?,” in *Proceedings of the 20th International Conference on Computer Aided Verification (CAV 2008)*, Department of Computer Science, ETH Zurich, Switzerland Princeton, USA, 2008.

45. Cristofaro C E, Hongle D, Freudiger J F and Norcie Greg, “A Comparative Study of Two Factor Authentication,” *in. Proceedings on the Workshop on Usable Security USEC'14*, San Diego, CA, USA, 2014.

46. CSA, “Security guidance for critical areas of focus in Cloud Computing V2.1,” 2009, Prepared by the Cloud Security Alliance.

47. Damgard I and Nielsen J B, “Discrete Logarithms,” Introduction to

Cryptography, 2012.

48. Darren P, “Google App Engine has THIRTY flaws, says researcher,” The Register, December 2014, [Online]. Available: http://www.theregister.co.uk/2014/12/09/google_app_engine_has_thirty_flaws_says_researcher/.

49. Das M L, Saxena A and Gulati V P, “A Dynamic-ID Based Remote User Authentication Scheme,” *IEEE Transactions on Consumer Electronics*, Vol. 50, No.2, pp.629-631, 2004.

50. Dictionary Attack, Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/Dictionary_attack.

51. Diffie W and Hellman M, “New Direction in Cryptography”, *IEEE Transactions on Information Theory*, Vol. 22, No.6, pp. 644-654., 1976.

52. Dalal N, Shah J, Hisaria K and Jinwala D, “A comparative Analysis of Tools for Verification of Security Protocols,” *International Journal of Communications, Network and System Sciences*, Vol.3, pp. 779-787, 2010.

53. Das M L, Saxena A and Gulati V P,” A Dynamic ID-Based Remote User Authentication Scheme,” *IEEE Transactions on Consumer Electronics*, Vol. 50, No.2, pp. 629-631, 2004.

54. David P M, “QRP: An Improved Secure User Authentication Method Using QR codes,” [Online], Available: https://www.grc.com/sqrl/files/QRP-secure-authentication.pdf, 2012.

55. Denning D and Sacco G, “Timestamps in Key Distribution Protocols,”

Communications of the ACM, vol. 24, No.8, pp. 533-536, 1981.

56. Dinesha H A and Agrawal V K, "Multi-level Authentication Technique for Accessing Cloud Services," in *Proceedings of International Conference on Computing, Communication and Applications (ICCCA)*, IEEE, pp. 1-4, 2010.

57. Dodson B, Sengupta D, Monica D B and Lam S, "Snap2Pass: Consumer-Friendly Challenge-Response Authentication with a Phone or Secure, Consumer-Friendly Web Authentication and Payments with a Phone," *Mobile Computing, Applications, and Services*, pp.17-38, 2012.

58. Dolev D and Yao A C, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, Vol. 29, No. 12, pp.198-208,1983.

59. Entrust, Entrust in Authentication, Data Breach, "CRM Provider Salesforce Hit with Malware Attack", September, 2014, [Online]. Available: http://www.entrust.com/crm-provider-salesforce-hit-malware-attack/.

60. Falas T and Kashani H, "Two-Dimensional Bar-code, Decoding with Camera-Equipped Mobile Phones," in *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 597-600, 2007.

61. Fernandes D A B, Soares L F B, Gomes J V, Freire M M, Mário, Inácio P R M, "Security Issues in Cloud Environments: A Survey," *International Journal of Information Security*, Vol.13, No.2, pp.113-

170.

62. Fenton W, “AceProject(Free)”, 2011, [Online], Available: http://www.pcmag.com/article2/0,2817,2374923,00.asp.

63. Gao S, “Cryptosystems Based on Discrete Logarithms,” 1999, [Online], Available: http://www.math.clemson.edu/~sgao/crypto_mod/node4.html.

64. Gartner Inc., “Forecast: Public Cloud Services, Worldwide 2010-2016, 2Q12 Update,” 2012, report by Gartner, Inc.

65. Gens F, “IT Cloud Services User Survey, pt.2: Top Benefits and Challenges,” 2008, IDC, [Online], Available: http://blogs.idc.com/ie/?p=210.

66. Gens F, “New IDC IT Cloud Services Survey: Top Benefits and Challenges,” IDC Exchange, 2009, [Online]. Available: http://blogs.idc.com/ie/?p=730.

67. Gong L, “A Security Risk of Depending on Synchronized Clocks,” in *ACM Operating Systems Review*, Vol.26, No.1. pp.49-53, 1992.

68. Gong L, Lomas M A, Needham R M and Saltzer J H, “Protecting Poorly Chosen Secrets from Guessing Attacks,” *IEEE Journal on selected Areas in Communications*, Vol.11, No.5, 1993.

69. Google Authenticator, Wikipedia, May 2016, [Online], Available: https://en.wikipedia.org/wiki/Google Authenticator.

70. Google Inc. “Google Authenticator Project – Two-Step verification. 2016,[Online]. Available: http://code.google.com/p/google-authenticator/.

71. Gowrie C, “Session Hijacking and the Cloud,” Comp 116, Final Project, 2014, [Online], Available: http://www.cs.tufts.edu/comp/116/archive/fall2014/cgowrie.pdf.

72. Granneman J, “Password-based Authentication: A Weak Link in Cloud Authentication,” August 2012, [Online], Available: http://searchcloudsecurity.techtarget.com/tip/Password-based-authentication-A- weak-link-in-cloud-authentication.

73. Halpert B, “Auditing Cloud Computing*:* A Security and Privacy Guide*,*” Hoboken, NJ: John Wiley & Sons, Inc., 2011.

74. Hamdan O A, Zaidan B B, Hamid A J, Shabbir M, Al-Nabhani Y, “New Comparative Study between DES, 3DES” and AES within Nineteen Factors, Journal of Computing, Vol.2, No. 3, March 2010.

75. Hao Z, Zhong S and Yu N, “A Time Bound Ticket-Based Mutual Authentication Scheme for Cloud Computing,” *International Journal of Computers, Communications & Control*, ISSN 1841-9836, Vol. 6, No.2, pp. 227-235, 2011.

76. Hardesty L, “Thwarting the Cleverest attack”, May 1, 2012, [Online], Available: http://news.mit.edu/2012/thwarting-eavesdropping-data-0501

77. Hart J, “Remote Working: Managing the Balancing Act Between Network Access and Data Security,” *Computer Fraud & Security,* Vol.11, pp.14-17, 2009.

78. Hash Function, “Cryptographic Hash Function,” Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function.

79. Hillenbrand M, Gotze J, Muller J, Muller P, "A Single Sign-on Framework for Web-Services-based Distributed Applications," in *Proceedings of 8th International Conference on Telecommunications*, ConTEL 2005, Vol 1, pp. 273-279, 2005.

80. HMAC, "Keyed–Hash Message Authentication Code," 2002, FIPS PUB 198, [Online], Available: http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf.

81. Housley R, Ford W, Polk W, Solo D, "Internet X.509 Public Key Infrastructure and CRL Profile," RFC 2459, January 1999, http://www. ietf.org/rfc/rfc2459.txt.

82. Hsiang H C and Shih W K, "Improvement of the Secure Dynamic ID Based Remote User Authentication Scheme for Multi-Server Environment," *Computer Standards and Interfaces*, Vol.31, pp.1118-1123, 2009.

83. Hsiang H C, Shih W K, "Weaknesses and improvements of the Yoon–Ryu–Yoo Remote User Authentication Scheme Using Smart Cards," *Computer Communications*, Vol. 32, No. 4, pp. 649-652, 2009.

84. Hsu C L, "Security of Chien et al.'s Authentication scheme using Smart Cards," *Computer Standards and Interfaces*, Vol.26, pp.167-169, 2004.

85. Hwang M S and Li L H, "A New Remote User Authentication Scheme Using Smart Cards," *IEEE Transactions on Consumer Electronics*, Vol.6, No. 1, pp.28–30, 2000.

86. Imperva, "Cookie Poisoning,",2013 [Online], Available: http://www.imperva.com/resources/glossary/cookie_ poisoning.html.

87. ISO/IEC 18004:2000. Information technology-Automatic identification and data capture techniques-Bar code Symbology-QR Code, 2000.

88. Jaidhar C D, "Enhance Mutual Authentication Scheme for Cloud Architecture," *in Proceedings of 3rd IEEE International Advanced Computing Conference (IACC)*, pp. 70-75, 2013.

89. Jensen M, Schwenk J, Gruscka N and Iacono L L," On Technical Security Issues in Cloud Computing," *in Proceedings of the IEEE International Conference on Cloud Computing*, pp.109-116, 2009.

90. Jin H, Sunghwan M, "A comparison of Cryptanalytic Tradeoff Algorithms," *Journal of Cryptology*, Vol. 26, pp.559-637, 2013.

91. Joan D, Vincent R, "AES Proposal: Rijndael ", 2003, National Institute of Standards and Technology, [Online], Available: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf#page=1.

92. Juang W S, "Efficient Password Authenticated Key Agreement using Smart Cards.", *Computers and Security*, Vol.23, pp.167-173, 2004.

93. Juang W S, "Efficient Multi-Server password authenticated Key Agreement using Smart Cards," *IEEE Transactions on Consumer Electronics*, Vol. 50, No. 1, pp.251-255, 2004.

94. Kang and Zhang, "Identity-Based Authentication in Cloud storage sharing," in *Proceedings of International Conference on Multimedia Information Networking and Security*, pp.851-855, 2010.

95. Krutz R L and Vine R D, "Cloud Security: A Comprehensive Guide to Secure Cloud Computing," Chapter 5, Wiley Publishing, Inc, 2010.

96. Kulshrestha A, Dubey S K, “A literature Review on Sniffing Attacks in Computer Networks”, *International Journal of Advanced Engineering Research and Science,* Vol.1, No.2, July 2014.

97. Lamport L, “Password Authentication with Insecure Communication”, *Communications of the ACM*, Vol. 24, No.11, pp.770-772, 1981.

98. Lee C, Lin T and Chang R, “A Secure Dynamic ID Based Remote User Authentication Scheme for Multi-Server Environment Using Smart Cards,” *Expert Systems with Applications*, Vol. 38, pp. 13863–13870, 2011.

99. Lee S, Ong I, Lim H T and Lee H J, “Two factor Authentication for Cloud Computing,” *International Journal of KIMICS*, Vol.8, pp. 427-432, 2010.

100. Lee Y S, Kim N H, Lim H, Jo H and Lee H J, “Online Banking Authentication System Using Mobile-OTP with QR code,” in *Proceedings of 5th International Conference on Computer Sciences and Convergence Information Technology*, pp.644-648, 2010.

101. Lewis G, “Basics About Cloud Computing,” September 2010, Software Engineering Institute, Carnegie Mellon, [Online], Available: http://resources.sei.cmu.edu/asset_files/WhitePaper/2010_019_001_28877.pdf.

102. Li L, Lin I, and Hwang M, “A Remote Password Authentication Scheme for Multi Server Architecture Using Neural Networks,” *IEEE Transactions on Neural Networks*, Vol. 12, No. 6, pp.1498-1504, 2001.

103. Li X, Ma J, Wang W, Xiong Y and Zhang J, “A Novel Smart Card and Dynamic ID Based Remote User Authentication Scheme for Multi-

Server Environments," *Mathematical and Computer Modelling*, Vol.58, pp.85-95, 2013.

104. Liang C, "The Five Major Authentication Issues in the Current Cloud Computing Environment," 2011, [Online], Available: https://chenliangblog.wordpress.com/tag/e-commerce/.

105. Liao I E, Lee C C and Hwang M S, "Security Enhancement for a Dynamic ID-Based Remote User Authentication Scheme," in *Proceedings of Conference on Next Generation on Web Services Practice*, pp. 437-440, 2005.

106. Liao K, Lee W, Sung M and Lin T, A One-Time Password Scheme with QR-Code Based on Mobile Phone," in *Proceedings of Fifth International Joint Conference on INC, IMS and IDC*, IEEE, pp. 2069-2071, 2009.

107. Liao Y P and Wang S S, "A Secure Dynamic ID Based Remote User Authentication Scheme for Multi Server Environment," *Computer Standards and Interfaces*, Vol.31, No. 1, pp.24-29, 2009.

108. Lin I, Hwang M and Li L, "A New Remote User Authentication Scheme for Multi-Server Architecture," *Future Generation Computer Systems*, Vol 19, pp. 13-22, 2003.

109. Linn J, "OASIS, Trust Model Guidelines," 2004, [Online], Available: https://www.oasis-open.org/committees/download.php/6158/sstc-saml-trustmodels-2.0-draft-01.pdf.

110. Liou Y P, Lin J and Wang S S," A New Dynamic ID-Based Remote User Authentication Scheme Using Smart Cards," in *Proceedings of 16th Information Security Conference*, Taiwan, pp. 198-205, 2006.

111. Liu Z, Wu F, Shang K, and Shai W, "C-MAS: The Cloud Mutual Authentication Scheme," in *Proceedings of the 2012 2nd International conference on Computer and Information Application (ICCIA 2012),* pp.0769-0772, 2012.

112. Macmillan D and Yadron D, "Dropbox Blames Security Breach on Password Reuse," 2014, The Wall Street Journal, [Online], Available: http://blogs.wsj.com/digits/2014/10/14/dropbox-blames-security-breach-on-password-reuse/.

113. Madhusudhan and Adireddi, "Weaknesses of Dynamic ID Based Remote User Authentication Protocol for Multi-Server Environments," *Journal of Computer and Communications,* Vol.2, 196-200, 2014.

114. Media O, "Assembla Software Review: Overview-Features-Pricing," 2012, [Online], Available: http://project-management.com/assembla-software- review/.

115. Meena B and Challa K A," Cloud Computing Security Issues with Possible solutions," *International Journal of Computer Science and Technology,* Vol.2, No. 1, 2012.

116. Meier J D, Mackman A, Dunner M, Vasireddy S, Escamilla R and Murukan A, "Improving Web Application Security: Threats and Counter Measures," Microsoft Corporation, January 2006, [Online], Available: https://msdn.microsoft.com/en- us/library/ff648641.aspx.

117. Mell P and Grance T, "The NIST Definition of Cloud Computing," 2011, NIST Special Publication 800-145, [Online] Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

118. Meyer R, "Secure Authentication on The Internet," SANS Institute Infosec Reading Room, 2007, [Online], Available:

https://www.sans.org/reading- room/whitepapers/securecode/secure-authentication-internet-2084.

119. Microsoft, Brokered Authentication, 2005, https://msdn.microsoft.com/en-us/library/aa480560.aspx.
120. Microsoft, Direct Authentication, December 2005, https://msdn.microsoft.com/en-us/library/ff647715.asp.
121. Microsoft, Web Service Security, Scenarios, Patterns and Implementation Guidance for Web Services Enhancements (WSE) 3.0, 2005, [Online], Available: https://msdn.microsoft.com/en-us/library/ff648183.aspx.
122. Misbahuddin M, "Secure Image Based Multi-Factor Authentication (SIMFA): A Novel Approach for Web Based Services", 2010, PhD Thesis, Jawaharlal Nehru Technological University [Online], Available: http://shodhganga.inflibnet.ac.in/handle/10603/3473.
123. Misbahuddin M, Aijaz A M, Shastri M H, "A Simple and Efficient Solution to Remote User Authentication Using Smart Cards", in *Proceedings of IEEE Innovations in Information Technology Conference (IIT 06)*, Dubai, 2006.
124. Misbahuddin M, Premchand P and Govardhan A, "A User Friendly Password Authenticated Key Agreement for Multi Server Environment," in *Proceedings of the International Conference on Advances in Computing, Communication and Control*, pp. 113-119, 2009.
125. Moskowitz R," Are Biometrics Too Good?", *Network Computing*, pp.85, No.1002,1999.

126. Mukhopadhyay S and Argles D, "An Anti-Phishing mechanism for Single Sign-On," *in Proceedings of International Conference on Information Society*, IEEE, pp.505-508, 2011.

127. Mulliner C, Burgaonkar R, Stewin P, Siefert J P, "SMS Based One Time Passwords: Attack and Defense," *Lecture Notes in Computer Science*, Vol. 7967, pp. 150-159, 2013.

128. MULTOS and JavaCard, WhitePaper, Jan Kremer Consulting Services, [Online], Available: http://jkremer.com/White%20Papers/MULTOS%20and%20JAVACARD%20White%20Paper%20JKCS.pdf.

129. Needham R and Schroeder M, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, Vol. 21, 1978. pp. 993-999, 1978.

130. Nicole Lewis, **"**Utah's Medicaid Data Breach Worse than Expected**"**, 2012, [Online]. Available: http://www.darkreading.com/risk-management/utahs-medicaid-data-breach-worse-than-expected/d/d-id/1103823.

131. NIST, "FIPS-PUB:186-4 Digital Signature Standard (DSS), July2013, [Online], Available: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

132. NIST, "*SHA-3 standardization,*" 2013, [Online], Available: http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_standardization.html.

133. NIST, "Standards for Security Categorization of Federal Information and Information Systems," FIPS PUB 199, 2004, [Online], Available" http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf.

134. NIST,” NIST Cloud Computing Program,” 2012, [Online], Available: http://www.nist.gov/itl/cloud/.

135. NIST,” Verifier Impersonation Attack,” Electronic Authentication Guideline, NIST Special Publication 800-63, Version 1.0.2, April 2006.

136. OASIS, “Security Assertion Markup Language (SAML) 2.0 Technical overview, working draft 03,” 20 February 2005. https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf.

137. Ocrho, RSA SecurID Token (Model SID700), Uploaded on 19 December 2008, Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/RSA_SecurID#/media/File:SecureID_token_new.JPG.

138. Password Cracking, Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/Password_cracking.

139. PBworks, Authentication Technologies, 2007, [Online], Available: http://biometrics.pbworks.com/w/page/14811351/Authentication%20technologies.

140. Pearce M, Zeadally S, Hunt R, “Virtualization: Issues, Security Threats, and Solutions,” *ACM Computing Surveys (CSUR)*, Vol. 45, No. 2, pp. 1:71-1:739 2013.

141. Pearson S, “Privacy, Security and Trust in Cloud Computing”, *in Pearson S, Yee G (editors) Privacy and Security for Cloud Computing*, pp. 3-42. Springer, London 2013.

142. Perez-Botero D, Szefer J and Lee R B, “Characterizing Hypervisor Vulnerabilities in Cloud Computing Servers”, *in Proceedings of the*

2013 International Workshop on Security in Cloud Computing (SCC), pp. 3-10. ACM, New York, NY, USA, 2013.

143. Public Key Infrastructure, Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/ Public_key_infrastructure.
144. Rainbow Table, Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/Rainbow_table.
145. Raza M, Iqbal M, Sharif M and Haider W, "A Survey of Password Attacks and Comparative Analysis on Methods for Secure Authentication," *World Applied Sciences Journal*, Vol.19, No.4, pp. 439 - 444, 2012.
146. Rick wash, "Lecture Notes on Stream Ciphers and RC4", http://rickwash.com/papers/stream.pdf, Un-Published.
147. Rivest R L, Shamir A, Adleman L, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Communications of the ACM* Vol. 21, No. 2, pp. 120–126., 1978.
148. Rodero-Morieno L, Vaquero L M, Caron E, Desprez F, and Muresan A, "Building Safe PaaS Clouds: A Survey on Security in Multi-Tenant Software Platforms," *Computers & Security*, Vol.31, No. 1, pp.96-108, 2012.
149. Rouse M, Single-factor Authentication (SFA), March 2015, [Online], Available: http://searchsecurity.techtarget.com/definition/single-factor-authentication-SFA.
150. RSA Inc., "RSA SecurID Hardware Authenticators," 2015, [Online]. Available: http://www.emc.com/security/rsa-securID/rsa-securID-hardware-authenticators.htm.

151. RSA SecurID, Wikipedia, May 2016, [Online], Available: https://en.wikipedia.org/wiki/RSA_SecurID.

152. Rui Jiang, “Advanced Secure User Authentication Framework for Cloud Computing,” *International Journal of Smart Sensing and Intelligent System*s, Vol. 6, No.4, 2013.

153. Saurabh D, Srinivas S and Qiang Y, “Message Digest as Authentication Entity for Mobile Cloud Computing,” in *proceedings of Performance computing and communications conference (IPCCC)*, IEEE 32nd International Conference, 2013.

154. Scheuermann D, “The Smartcard as a Mobile Security Device”, *Electronics & Communication Engineering Journal*, Vol. 14, No. 5, pp.205-210, 2002.

155. Schneier B, Applied Cryptography, Second Edition: protocols, Algorithms and Source Code in C, Wiley Computer Publishing, 1996.

156. Schneier, B. "Cryptanalysis of MD5 and SHA: Time for a New Standard". *Computerworld*, August, 19, 2004, [Online], Available: https://www.schneier.com/essays/archives/2004/08/cryptanalysis_of_md5.html.

157. Shen Z, Li L, Yan F and Wu X, “Cloud Computing System based on Trusted Computing Platform,” in *Proceedings of International Conference on Intelligent Computation and Trusted Computing Platform*, pp.942-945, 2010.

158. Shibboleth, “Identity Provider and Service Provider Single Log Out,” 2015, [Online], Available: http://www.utexas.edu/its/help/shibboleth/2299.

159. Shieh W G and Wang J M, "Efficient remote Mutual Authentication and Key Agreement," *Computers and Security*, Vol. 25. pp.72-77, 2006.

160. Shih H C, "Cryptanalysis on Two Password Authentication Schemes," Laboratory *of Cryptography and Information Security*, National Central University, Taiwan, 2008.

161. Song D, Berezin S and Perrig A, "Athena: A Novel Approach to Efficient Automatic Security Protocol Analysis," *Journal of Computer Security*, Vol. 9, pp. 47-74, 2001.

162. Stallings W, "Cryptography and Network Security, Principles and Practices," Fifth edition, Pearson Publications, 2011.

163. Stallings W, "Cryptography and Network Security: Principles and Practices," 4th edition, Pearson Edition, ISBN-10, 0131873164 ISBN-13: 9780131873162, 2006.

164. Stienne D S, Nathan C and Paul R, "Strong Authentication for Web Services Using Smart Cards," in *the Proceedings of Australian Information Security Management Conference*, 2013.

165. Subashini S and Kavitha V, "A Survey on Security Issues in Service Delivery Models of Cloud Computing," *Journal of Network and Computer Applications*, Vol. 34, No.1, pp. 1 -11, 2011.

166. Sun, "An Efficient Remote Use Authentication Scheme Using Smart Cards," *IEEE Transactions on Consumer electronics*, Vol. 46, No.4, 2000.

167. Takabi H, Joshi J B D and Ahn G, "SecureCloud: Towards a Comprehensive Security Framework for Cloud Computing

Environments," *in Proceedings of the IEEE 34th Annual Computer Software and Application Conference Workshops*, pp. 393-398, 2010.

168. Thawte, "History of Cryptography," [Online], Available: http://book.itep.ru/depository/crypto/Cryptography_history.pdf, 2013.
169. Trosch J, "Identity Federation with SAML 2.0," 2008, [Online], Available: http://security.hsr.ch/theses/DA_2008_IdentityFederation_with_SAML_20.pdf.
170. Tsai H, Siebenhaar M, Miede A, Yulun H and Steinmetz R, "Threat as A Service? The Impact of Virtualization on Cloud Security," *IT Professional*, Vol. 14, No.1, pp.32-37, 2011.
171. Tsai J, "Efficient Multi-Server Authentication Scheme Based on One-way Hash Function without Verification Table," *Computers &* Security, Vol.27, pp. 115-121, 2008.
172. Tsaur W J, Wu C C and Lee W B, "A Smart-Card Based Remote Scheme for Password Authentication in Multi-Server Internet Services," *Computer Standards and Interfaces*, No. 27, pp. 39-51, 2004.
173. Oracle, "Understanding Web Service Security Concepts," copyright @ 2016, Oracle® Fusion Middle ware understanding oracle web services manager," [online], Available:https://docs.oracle.com/middleware/1212/owsm/OWSMC/owsm-security-concepts.htm#OWSMC116.
174. Vaquero L M, Rodero-Morieno L and Moran D," Locking The Sky: A Survey on IaaS Cloud Security", Computing Vol.91, No.1, 93-118, 2011.

175. Vmware, “How to Choose the Right Virtual Data Center for Your Needs,” 2012, [Online], Available: http://ww2.frost.com/files/8714/2113/2896/Whitepaper-Choosing_the_Right_Virtual_Datacenter.pdf.

176. Walter S, “Smart Card Icons,”, 2011, [Online], Available: http://stef.thewalter.net/smart-card-icons.html

177. Warren H, “Passwords and Passion”, *Software, IEEE,* Vol.23, No. 4, pp.5-7, 2006.

178. Whittaker Z, “Microsoft Admits Patriot Act Can Access EU Based Cloud Data,” 2011, [Online], Available: http://www.zdnet.com/article/microsoft-admits-patriot- act-can-access-eu-based-cloud-data/.

179. Woodward J D, "Believing in Biometrics". *Information Security Magazine*, 21 March 2000, [Online], Available: http://www.infosecuritymag.com/biometrics.htm.

180. Xiao Z and Xiao Y,” Security and Privacy in Cloud Computing,” *IEEE. Communications Surveys and Tutorials*, Vol.15, No. 2, pp.843-859, 2013.

181. Xiaoyun W, Dengguo F, Xuejia L, Hongbu Y, “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD,” *in the Crypto 2004 conference,* August 2004, [Online], Available: http://eprint.iacr.org/2004/199.pdf.

182. Xiaoyun W, Yiqun L Y, and Hongbo Y, “Finding Collisions in the Full SHA-1, 2005,” *Advances in Cryptology, -CRYPTO 2005,* Vol. 3621, Lecture Notes in Computer Science, pp.17-36.

183. Xiong L, Yongping X, Jian M, and Wendong W, "An Efficient and Security Dynamic Identity Based Authentication Protocol for Multi-Server architecture Using Smart Cards", *Journal of Network and Computer Applications* Vol.35, pp. 763-769, 2012.

184. Yoon E J and Yoo K Y," Improving the Dynamic ID-Based Remote Mutual Authentication Scheme," in *Proceedings of OTM Workshops* 2006, LNCS 4277, pp. 499-507, 2006.

185. Zarandioon S, "Improving the Security and Usability of Cloud Services with User-Centric Security Models," May 2012, Ph.D Thesis, [Online], Available: http://www.cs.rutgers.edu/~vinodg/students/samanzarandioon_phdthesis.pdf.

186. Zetter K, "FBI defends Disruptive Raid on Texas data Centers", April 2009, [Online], Available: http://www.wired.com/threatlevel/2009/04/data-centers-ra/.

187. Zhang Y, Juels A, Opera A and Reiter M K, "Home Alone: Co-Residency Detection in The Cloud Via Side-Channel Analysis,*" in Proc. IEEE Symposium on Security and Privacy*, pp. 313-328, 2011.

188. Zhou M, Zhang R, Xie W, Qian W and Zhou A," Security and Privacy in Cloud Computing: A Survey," in Proceedings of the 6th International Conference on Semantics Knowledge and Grid, IEEE Computer Society, Washington, DC, USA, pp. 105-112, 2010.

189. Zhu B, Fan X and Gong G, "Loxin – A Solution to Password-Less Universal Login," in *Proceedings of 2014 IEEE INFOCOM Workshop on Security and Privacy in Big Data*, pp. 488-493, Toronto, ON, 2013.

190. Zhu H, He Q, Tang H and Cao W, “Voiceprint-Biometric Template Design and Authentication Based on Cloud Computing Security,” in *Proceedings of International Conference on Cloud and Service Computing*, pp. 302-308, 2011.

191. Zunnurhain K, Vrbsky S, “Security Attacks and Solutions in Clouds,” Poster from CloudCom 2010, 2010, [Online], Available: http://salsahpc.indiana.edu/CloudCom2010/Poster/cloudcom2010_sub mission_98.pdf.

192. Zwattendorfer B and Tauber A, “Secure Cross-Cloud Single Sign-on (SSO) Using eIDs,” in *Proceedings of the 7th International conference for Internet Technology and Secured Transactions (ICITST 2012)*, pp. 1501-155, London, 2012.

PUBLICATIONS AND PROCEEDINGS

I. Papers in Journals

1. An Enhanced Secure Remote User Authentication Scheme without Verification Table

   Sumitra Binu, Pethuru Raj and Mohammed Misbahuddin

   International Journal of Computer Applications (IJCA), 114(10), pp.1-5, 2015.

2. A Survey of Cloud Authentication Attacks and Solution Approaches

   Sumitra Binu, Mohammed Misbahuddin and Pethuru Raj

   International Journal of Innovative Research in Computer and Communication Engineering (IJRCCE), 114(10), pp.1-5, 2014.

3. A Security Framework for an Enterprise System on Cloud

   Sumitra Binu, Meena Kumari

   International Journal of Computer Science and Engineering (IJCSE), 3(4), pp.548-552, 2012.

II. Conference Proceedings

1 A Mobile Based Remote User Authentication System without Verifier Table for Cloud Based Services.
Sumitra Binu, Mohammed Misbahuddin and Pethuru Raj

Proceedings of International Symposium on Women in Computing and Informatics(WCI-2015), held on 10-13 August, 2015, Kochi, Kerala, India.

http://dl.acm.org/citation.cfm?id=2791487&dl=ACM&coll=DL&CFID=744472768&CFTOKEN=25120408.

2 A Strong Single Sign-on User Authentication Scheme without Verifier Table for Cloud Based Services.

Sumitra Binu, Mohammed Misbahuddin and Pethuru Raj

Proceedings of International Conference on Security and Management(SAM'15), held on 27-30 July, 2015, Las Vegas, Nevada, USA.

http://worldcomp-proceedings.com/proc/p2015/SAM9742.pdf

3 A Secure Authentication Framework for Cloud Environment.
Sumitra Binu, Pethuru Raj and Mohammed Misbahuddin

Doctoral Colloquium & Demos Sessions(SAM'15), the 2015 World Congress in Computer Science, Computer Engineering and Applied Computing (WORLDCOMP'15) held on 27-30 July, 2015, Las Vegas, Nevada, USA.

4 SAFE-CLOUD: A Secure and Usable Authentication Framework for Cloud Environment.
Sumitra Binu, Pethuru Raj and Mohammed Misbahuddin

Proceedings of 2nd International Doctoral Symposium on Applied Computation and Security Systems (ACSS), held on 23-25 May, 2015, Kolkata, India.

http://link.springer.com/chapter/10.1007%2F978-81-322-2650-5_12

5 A Proof of Concept Implementation of a Mobile Based Authentication Scheme without Password Tabel for Cloud Environment.
Sumitra Binu, Archana Mohna, Deepak K.T, Manohar S, Mohammed Misbahuddin, Pethuru Raj

Proceedings of fifth IEEE International Advanced Comuputing Conference, held on 12-13 June, 2015, BMSCE, Bangalore, India.

http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7154897&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D7154897

6 An Enhanced Dynamic Identity Based Remote User Authentication Scheme Using Smart Card without Verification Table.
Sumitra Binu, Pethuru Raj, Mohammed Misbahuddin

Proceedings of Security and Privacy Symposium 2015, held on 14 February, 2015, IIIT, Delhi, India.

7 A Single Sign-on Based Secure Remote User Authentication Scheme for Multi-Server Enviroments.
Sumitra Binu, Mohammed Misbahuddin, Pethuru Raj.

Proceedings of International Conference on Computing and Communication Technologies, held on 11-13 December, 2014, Organized by Osmania University, Hyderabad, India.

8 A Survey of Traditional and Cloud Specific Security Issues.
Sumitra Binu, Mohammed Misbahuddin, Pethuru Raj.

Proceedings of International Symposium on security in computing and Communication(SSCC'13), held on 22-24 August 2013, at SJCE, Mysore, India.

http://link.springer.com/chapter/10.1007%2F978-3-642-40576-1_12