International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 6 Issue: 4 220 - 226

A Review of Authentication Protocols

Miss. Pratiksha M. Matkar, Miss. Priyanka V. Bhiogade, Miss. Puja P. Patle,

Mrs. Pratik Gangewane, Prof. F.M.Shelke

P. R. Pote College of Engineering & Management, Amravati

Abstract: Authentication is a process that ensures and confirms a users identity. Authorization is the process of giving someone permissions to do or have something. There are different types of authentication methods such as local password authentication, server-based-password authentication, certificate-based authentication, two-factor authentication etc. Authentication protocol developed for Password Authentication Protocol (PAP), Challenge-Handshake Authentication Protocol (CHAP), and Extensible Authentication Protocol (EAP). There are different types of application for authentications are as follows: 1.protocols developed for PPP Point-to-Point Protocol 2. Authentication, Authorization and Accounting 3.Kerberos.

***Keywords**: Authentication, Authorization, Protocol, Kerberos, CHAP,EAPetc.*

*****

I. Introduction

Authentication can be accomplished in many ways. Importance of selecting an environment appropriate authentication protocol for the secure password. Authentication is the technique by which process verify that its communication partner is who it is supposed to be and not and imposter. Authorization is a process by which a server determines if the client has permission to use a resource are access a file. Authorization is the function of specifying access rights or privileges to resources related to information security and computer security in general and to access control in particular. Authentication is use by a server when the server when the server needs to know exactly who is accessing their information or site. For authentication we use a different protocol. The protocol is use for sending secure message from sender to receiver i.e. sender send message with encrypt format and receiver received this message using their public key in decrypt format.

II. Methods For authentication are

1.Local password authentication

The simplest authentication is based on user accounts stores locally on the FortiGate unit. for each account, a username and password is stored. The account also has a disable option so that you can suspend the account without deleting it.Local user accounts work well for a single FortiGate installation. If your network has multiple FortiGate units that will use the same accounts the use of an external authentication server can simplify account configuration and maintenance.

You can create local user accounts in the web-based manager under User &Device>User>User Definition. This page is also used to create accounts where an external authentication server stores and verifies the password.

2.Server-based-password authentication

Using external authentication servers is desirable when multiple FortiGate units need to authenticate the same users, or where the FortiGate unit is added to a network that already contains an authentication server. FortiOS supports the use of LDAP, RADIUS, TACACS+, AD or POP3 servers for authentication. When you use an external authentication server to authenticate users, the FortiGate unit sends the user's entered credentials to the external server. The password is encrypted. The server's response indicates whether the supplied credentials are valid or not.

You must configure the FortiGate unit to access the external authentication servers that you want to use. The configuration includes the parameters that authenticate the FortiGate unit to the authentication server.

You can use external authentication servers in two ways:

- Create user accounts on the FortiGate unit, but instead of storing each user's password, specify the server used to authenticate that user. As with accounts that store the password locally, you add these users to appropriate user groups.
- Add the authentication server to user groups. Any user who has an account on the server can be authenticated and have the access privileges of the FortiGate user group. Optionally, when an LDAP server is a FortiGate user group member, you can limit access to users who belong to specific groups defined on the LDAP server.

3.Certificate-based authentication

An RSA X.509 server certificate is a small file issued by a Certificate Authority (CA) that is installed on a computer or FortiGate unit to authenticate itself to other devices on the network. When one party on a network presents the certificate as authentication, the other party can validate that the certificate was issued by the CA. The identification is therefore as trustworthy as the Certificate Authority (CA) that issued the certificate.

To protect against compromised or misused certificates, CAs can revoke any certificate by adding it to a Certificate Revocation List (CRL). Certificate status can also be checked online using Online Certificate Status Protocol (OCSP).RSA X.509 certificates are based on public-key cryptography, in which there are two keys: the private key and the public key. Data encrypted with the private key can be decrypted only with the public key and vice versa. As the names suggest, the private key is never revealed to anyone and the public key can be freely distributed. Encryption with the recipient's public key creates a message that only the intended recipient can read. Encryption with the sender's private key creates a message whose authenticity is proven because it can be decrypted only with the sender's public key.

Server certificates contain a signature string encrypted with the CA's private key. The CA's public key is contained in a CA root certificate. If the signature string can be decrypted with the CA's public key, the certificate is genuine.

A certificate authority can be:

- an organization, such as VeriSign Inc., that provides certificate services
- a software application, such as Microsoft Certificate Services or OpenSSH

For a company web portal or customer-facing SSL VPN, a third-party certificate service has some advantages. The CA certificates are already included in popular web browsers and customers trust the third-party. On the other hand, third-party services have a cost.

For administrators and for employee VPN users, the local CA based on a software application provides the required security at low cost. You can generate and distribute certificates as needed. If an employee leaves the organization, you can simply revoke their certificate.

Certificates for users

FortiGate unit administrators and SSL VPN users can install certificates in their web browsers to authenticate themselves. If the FortiGate unit uses a CA-issued certificate to authenticate itself to the clients, the browser will also need the appropriate CA certificate.

FortiGate IPsec VPN users can install server and CA certificates according to the instructions for their IPsec VPN client software. The FortiClient Endpoint Security application, for example, can import and store the certificates required by VPN connections.

FortiGate units are also compatible with some Public Key Infrastructure systems. For an example of this type of system, see RSA ACE (SecurID) servers.

4.Two-factor authentication

A user can be required to provide both something they know (their username and password combination) and something they have (certificate or a random token code). Certificates are installed on the user's computer.

Two-factor authentication is available for PKI users. For more information, see Certificate.

Another type of two-factor authentication is to use a randomly generated token (multi-digit number) along with the username and password combination. One method is a FortiToken — a one time passcode (OTP) generator that generates a unique code every 60 seconds. Others use email or SMS text messaging to deliver the random token code to the user or administrator. When one of these methods is configured, the user enters this code at login after the username and password have been verified. The FortiGate unit verifies the token code after as well as the password and username For more information, see Two-factor authentication.

III. Types of authentication

3.1 Authentication protocols developed for PPP Point-to-Point Protocol

In computer networking, Point-to-Point Protocol (PPP) is a data link layer (layer 2) communications protocol used to establish a direct connection between two nodes. It connects two routers directly without any host or any other networking device in between. It can provide connection authentication, transmission encryption,[1] and compression.

PPP is used over many types of physical networks including serial cable, phone line, trunk line, cellular telephone, specialized radio links, and fiber optic links such as SONET. Internet service providers (ISPs) have used PPP for customer dial-up access to the Internet, since IP packets cannot be transmitted over a modem line on their own, without some data link protocol.

Two derivatives of PPP, Point-to-Point Protocol over Ethernet (PPPoE) and Point-to-Point Protocol over ATM (PPPoA), are used most commonly by Internet Service Providers (ISPs) to establish a Digital Subscriber Line (DSL) Internet service connection with customers.

PPP is commonly used as a data link layer protocol for connection over synchronous and asynchronous circuits, where it has largely superseded the older Serial Line Internet Protocol (SLIP) and telephone company mandated standards (such as Link Access Protocol, Balanced (LAPB) in the X.25 protocol suite). The only requirement for PPP is that the circuit provided be duplex. PPP was designed to work with numerous network layer protocols, including Internet Protocol (IP), TRILL, Novell's Internetwork Packet Exchange (IPX), NBF, DECnet and AppleTalk. Like SLIP, this is a full Internet connection over telephone lines via modem. It is more reliable than SLIP because it double checks to make sure that Internet packets arrive intact.[2] It resends any damaged packets.

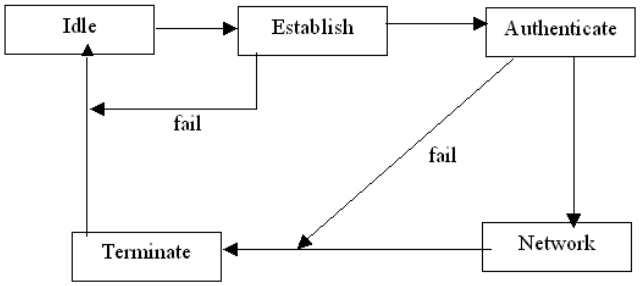

Fig: Point to Point Authentication

3.1.1Password Authentication Protocol (PAP)

Password Authentication Protocol (PAP) is a password-based authentication protocol used by Point to Point Protocol (PPP) to validate users. Almost all network operating systemremote servers support PAP.

PAP is considered a weak authentication scheme (weak schemes are simple and have lighter computational overhead but are much more vulnerable to attack; while weak schemes may have limited application in some constrained environments, they are avoided in general). Among PAP's deficiencies is the fact that it transmits unencrypted passwords over the network. PAP is therefore used only as a last resort when the remote server does not support a stronger scheme such as CHAP or EAP.

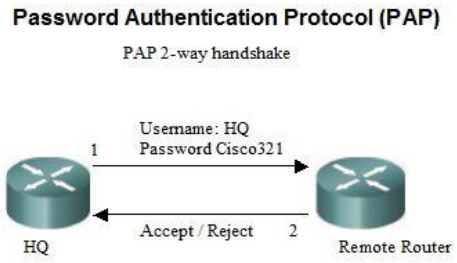

Fig: Password Authentication Protocol

Working cycle:

Client sends username and password Server sends authentication-ack (if credentials are OK) or authentication-nak PAP packet embedded in a PPP frame. The protocol field has a value of C023 (hex).

PAP Packets:

Description	1 byte	1 byte	2 bytes	1 byte	Variable	1 byte	Variable
Authentication-request	Code = 1	ID	Length	Username length	Username	Password length	Password
Authentication-ack	Code = 2	ID	Length	Message length	Message		
Authentication-nak	Code = 3	ID	Length	Message length	Message		

3.1.2 Challenge-Handshake Authentication Protocol (CHAP)

Challenge-Handshake Authentication Protocol (CHAP) authenticates a user or network host to an authenticating entity. That entity may be, for example, an Internet service provider. CHAP is specified in RFC 1994.CHAP provides protection against replay attacks by the peer through the use of an incrementally changing identifier and of a variable challenge-value. CHAP requires that both the client and server know the plaintext of the secret, although it is never sent over the network. Thus, CHAP provides better security as compared to Password Authentication Protocol(PAP) which is vulnerable for both these reasons. The MS-CHAP variant does not require either peer to know the plaintext and does not transmit it, but has been broken.

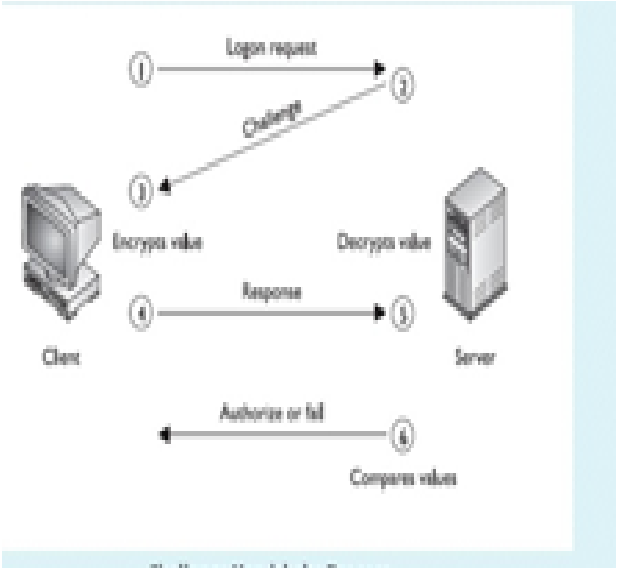

Fig:Challenge-Handshak-Authentication Protocol

Working cycle:

CHAP is an authentication scheme used by Point-to-Point Protocol (PPP) servers to validate the identity of remote clients. CHAP periodically verifies the identity of the client by using a three-way handshake. This happens at the time of establishing the initial link (LCP), and may happen again at any time afterwards. The verification is based on a shared secret(such as the client's password).[2]

1. After the completion of the link establishment phase, the authenticator sends a "challenge" message to the peer.
2. The peer responds with a value calculated using a one-way hash function on the challenge and the secret combined.
3. The authenticator checks the response against its own calculation of the expected hash value. If the values match, the authenticator acknowledges the authentication; otherwise it should terminate the connection.
4. At random intervals the authenticator sends a new challenge to the peer and repeats steps 1 through

CHAP Packets:

Description	**1 byte**	**1 byte**	**2 bytes**	**1 byte**	**Variable**	**variable**
Challenge	Code = 1	ID	Length	Challenge Length	Challenge value	Name
Response	Code = 2	ID	Length	Response Length	Response value	Name
Success	Code = 3	ID	Length		Message	
Failure	Code = 4	ID	Length		Message	

The ID chosen for the random challenge is also used in the corresponding response, success, and failure packets. A new challenge with a new ID must be different from the last challenge with another ID. If the success or failure is lost, the same response can be sent again, and it triggers the same success or failure indication. For MD5 as hash the response value is `MD5(ID||secret||challenge)`, the MD5 for the concatenation of ID, secret, and challenge.[

3.1.3 Extensible Authentication Protocol (EAP)

Extensible Authentication Protocol, or EAP, is an authentication framework frequently used in wireless networks and point-to-point connections. It is defined in RFC 3748, which made RFC 2284 obsolete, and is updated by RFC 5247.

EAP is an authentication framework for providing the transport and usage of keying material and parameters generated by EAP methods. There are many methods defined by RFCs and a number of vendor specific methods and new proposals exist. EAP is not a wire protocol; instead it only defines message formats. Each protocol that uses EAP defines a way to encapsulate EAP messages within that protocol's messages.

EAP is in wide use. For example, in IEEE 802.11 (WiFi) the WPA and WPA2 standards have adopted IEEE 802.1X with one hundred EAP Types as the official authentication mechanisms.

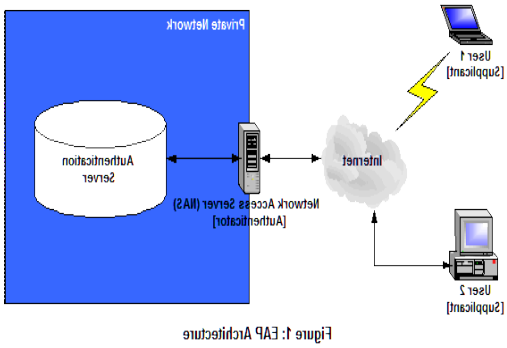

Fig: Extensible Authentication Protocol

3.2 Authentication, Authorization and Accounting.

It is used to refer to a family of protocols which mediate network access.

Two network protocols providing this functionality are particularly popular: the RADIUS protocol,[1] and its newer Diameter counterpart.[2][3]

Further explanations of Authentication, Authorization, and Accounting are available on external sites.

Uses of AAA

In some cases, the term AAA has been used to refer to protocol-specific information. For example, Diameter uses the URI scheme; **AAA**, which stands for **Authentication, Authorization and Accounting**, and **AAAS** which stands for **Authentication, Authorization and Accounting with Secure Transport**, which is a Diameter-based Protocol.[4]These protocols were defined by the Internet Engineering Task Force in RFC 6733 and are intended to provide an Authentication, Authorization, and Accounting

(AAA) framework for applications such as network access or IP mobility in both local and roaming situations.[5]

While the term AAA has been used in such a narrow context, the concept of *AAA* is more widely used within the industry. As a result, it is incorrect to refer to AAA and Diameter as being one and the same.

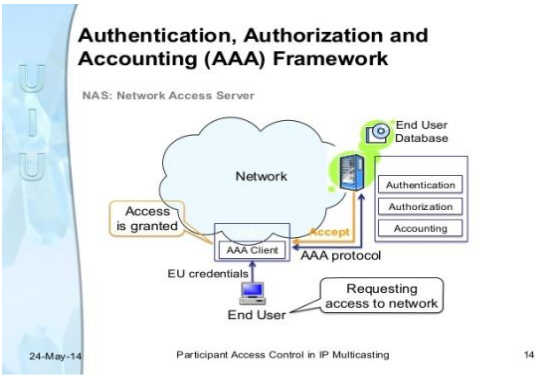

Fig:AAA protocol

3.2.1Terminal Access Controller Access-Control System(TACACS)

Terminal Access Controller Access-Control System refers to a family of related protocols handling remote authentication and related services for networked access control through a centralized server. The original **TACACS** protocol, which dates back to 1984, was used for communicating with an authentication server, common in older UNIX networks; it spawned related protocols:

3.2.2Extended TACACS (XTACACS) Extended TACACS (XTACACS) is a proprietary extension to TACACS introduced by Cisco Systems in 1990 without backwards compatibility to the original protocol. TACACS and XTACACS both allow a remote access server to communicate with an authentication server in order to determine if the user has access to the network.

3.2.3Terminal Access Controller Access-Control System Plus (TACACS+) Terminal Access Controller Access-Control System Plus (TACACS+) is a protocol developed by Cisco and released as an open standard beginning in 1993. Although derived from TACACS, TACACS+ is a separate protocol that handles authentication, authorization, and accounting (AAA) services. TACACS+ and other flexible AAA protocols have largely replaced their predecessors.

3.3 Kerberos

Kerberos is a computer network authentication protocol that works on the basis of *tickets* to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. The protocol was named after the character *Kerberos* (or *Cerberus*) from Greek mythology, the ferocious three-headed guard dog of Hades. Its designers aimed it primarily at a client–server model and it provides mutual authentication—both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks.

Kerberos builds on symmetric key cryptography and requires a trusted third party, and optionally may use public-key cryptographyduring certain phases of authentication.[1] Kerberos uses UDP port 88 by default.

The client authenticates itself to the Authentication Server (AS) which forwards the username to a key distribution center (KDC). The KDC issues a ticket-granting ticket (TGT), which is time stamped and encrypts it using the ticket-granting service's (TGS) secret key and returns the encrypted result to the user's workstation. This is done infrequently, typically at user logon; the TGT expires at some point although it may be transparently renewed by the user's session manager while they are logged in.

When the client needs to communicate with another node ("principal" in Kerberos parlance) to some service on that node the client sends the TGT to the TGS, which usually shares the same host as the KDC. Service must be registered at TGT with a Service Principal Name (SPN). The client uses the SPN to request access to this service. After verifying that the TGT is valid and that the user is permitted to access the requested service, the TGS issues ticket and session keys to the client. The client then sends the ticket to the service server (SS) along with its service request.

User Client-based Logon

1. A user enters a username and password on the client machine(s). Other credential mechanisms like pkinit (RFC 4556) allow for the use of public keys in place of a password.
2. The client transforms the password into the key of a symmetric cipher. This either uses the built-in key scheduling, or a one-way hash, depending on the cipher-suite used.

Client Authentication

1. The client sends a cleartext message of the user ID to the AS (Authentication Server) requesting services on behalf of the user. (Note: Neither the secret key nor the password is sent to the AS.)
2. The AS checks to see if the client is in its database. If it is, the AS generates the secret key by hashing the password of the user found at the database (e.g., Active Directoryin Windows Server) and sends back the following two messages to the client:
   - Message A: *Client/TGS Session Key* encrypted using the secret key of the client/user.
   - Message B: *Ticket-Granting-Ticket* (TGT, which includes the client ID, client network address, ticket validity period, and the *client/TGS session key*) encrypted using the secret key of the TGS.

3. Once the client receives messages A and B, it attempts to decrypt message A with the secret key generated from the password entered by the user. If the user entered password does not match the password in the AS database, the client's secret key will be different and thus unable to decrypt message A. With a valid password and secret key the client decrypts message A to obtain the *Client/TGS Session Key*. This session key is used for further communications with the TGS. (Note: The client cannot decrypt Message B, as it is encrypted using TGS's secret key.) At this point, the client has enough information to authenticate itself to the TGS.

Client Service Authorization

1. When requesting services, the client sends the following messages to the TGS:
   - Message C: Composed of the TGT from message B and the ID of the requested service.
   - Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the *Client/TGS Session Key*.
2. Upon receiving messages C and D, the TGS retrieves message B out of message C. It decrypts message B using the TGS secret key. This gives it the "client/TGS session key". Using this key, the TGS decrypts message D (Authenticator) and compare client ID from message C and D, if they match server sends the following two messages to the client:
   - Message E: *Client-to-server ticket* (which includes the client ID, client network address, validity period and *Client/Server Session Key*) encrypted using the service's secret key.
   - Message F: *Client/Server Session Key* encrypted with the *Client/TGS Session Key*.

Client Service Request

1. Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the Service Server (SS). The client connects to the SS and sends the following two messages:
   - Message E from the previous step (the *client-to-server ticket*, encrypted using service's secret key).
   - Message G: a new Authenticator, which includes the client ID, timestamp and is encrypted using *Client/Server Session Key*.
2. The SS decrypts the ticket (message E) using its own secret key to retrieve the *Client/Server Session Key*. Using the sessions key, SS decrypts the Authenticator and compare client ID from message E and G, if they match server sends the following message to the client to confirm its true identity and willingness to serve the client:
   - Message H: the timestamp found in client's Authenticator (plus 1 in version 4, but not necessary in version 5[3][4]), encrypted using the *Client/Server Session Key*.
3. The client decrypts the confirmation (message H) using the *Client/Server Session Key* and checks whether the timestamp is correct. If so, then the client can trust the server and can start issuing service requests to the server.
4. The server provides the requested services to the client.

Drawback and limitation

- Single point of failure: It requires continuous availability of a central server. When the Kerberos server is down, new users cannot log in. This can be mitigated by using multiple Kerberos servers and fallback authentication mechanisms.
- Kerberos has strict time requirements, which means the clocks of the involved hosts must be synchronized within configured limits. The tickets have a time availability period and if the host clock is not synchronized with the Kerberos server clock, the authentication will fail. The default configuration per MIT requires that clock times be no more than five minutes apart. In practice Network Time Protocol daemons are usually used to keep the host clocks synchronized. Note that some servers (Microsoft's implementation being one of them) may return a KRB_AP_ERR_SKEW result containing the encrypted server time in case both clocks have an offset greater than the configured maximum value. In that case, the client could retry by calculating the time using the provided server time to find the offset. This behavior is documented in RFC 4430.
- The administration protocol is not standardized and differs between server implementations. Password changes are described in RFC 3244.
- In case of symmetric cryptography adoption (Kerberos can work using symmetric or asymmetric (public-key) cryptography), since all authentications are controlled by a centralized key distribution center (KDC), compromise of this authentication infrastructure will allow an attacker to impersonate any user.
- Each network service which requires a different host name will need its own set of Kerberos keys. This complicates virtual hosting and clusters.
- Kerberos requires user accounts, user clients and the services on the server to all have a trusted relationship to the Kerberos token server (All must be in the same Kerberos domain or in domains that have a trust relationship between each other). Kerberos cannot be used in scenarios where users want to connect to services from unknown/untrusted clients as in a typical Internet or cloud computer scenario, where the authentication provider typically does not have knowledge about the users client system.

- The required client trust makes creating staged environments (e.g., separate domains for test environment, pre-production environment and production environment) difficult: Either domain trust relationships need to be created that prevent a strict separation of environment domains or additional user clients need to be provided for each environ

IV. Conclusion

Protocol designers should clearly identify the role of encryption in their protocols. If the protocol requires nonmalleable encryption, clearly state so in the specification, and ensure that the encryption scheme actually achieves this requirement. The use of unauthenticated encryption in version 4 violates the implicit requirement of nonmalleability, with dramatic results.

So,in this paper we study the different method of authentication protocol and different type of authentication protocol.

References

[1]. Ducan,Richard(23 october 2001).”an overview of different Authentication Methods and Protocols”.www.sang.org.SANSinstitude. Retrived 31 October 2015.

[2]. Shinder ,Deb (28 August 2001).”Understanding and selecting aumethode”. www.techrepublic,com. Retrieved 30 Octomber 2015.

[3]. van tilborg,henk C.A(2000)Fundamentals of Cryptology.Masschusett: Kluwer Academic Publishers.pp.66-67.ISBN 0-7923-8675-2.

[4]. Smith ,Richard E.(1997).Internet Cryptography. Massachusett:AddisonWesley Longmap.pp 1-27.ISBN 0-201-92480-3.

[5]. Halevi,Shai.”Public-key cryptography and password protocols”.citeseerx.ist.psu.edu Retrieved 31 Octomber 2015.

[6]. Vanek, Tomas.”Authentizanciprotokoly v telekomunikanich a datovychsitich”. CVUT Prague. Retrived 31 Octomber 2015.

[7]. “AAA protocl”www.cisco.com.CISCO.Retrived 31 octomber 2015.

[8]. Liu,Jeffrey(24 January 2006).”Introduction to Diameter”www.ibm.com.IBM.retrieved 31 Octomber 2015.

[9]. ”kerboros: The Network Authentication Protocol”. Web.mit.edu.MIT Kerboros.10 September 2015.Retrieved 31 Octomber 2015.

[10]. Schneier,Bruce(1997).Applied Cryptography.NewYork:John Wiley & Sons,Inc.pp.52-74.ISBN 0-471-12845-7.

[11]. ”Protocol of the past”.srp.stanford.edu.Stanford University.

[12]. Retrieved 31 Octomber 2015.

Author	method	Pros	cons
Ducan,Richard	overview of different Authentication Methods and Protocols	Strong and reliable authentication	Privacy of user Once hacker cant create duplicate account
van tilborg,henk C.A	Fundamentals of Cryptology	easy to implement	High availability Selective access control
Smith ,Richard E.	Internet Cryptography	Faster access and easy to implement	It make used as polarization nature of light.
Halevi,Shai	Public-key cryptography and password protocols	High effective and secured	Asynchronous encryption need to be implements in usable way.
Liu,Jeffrey	Introduction to Diameter	Secured, reliable and unlimited scalability	Difficult to used for conventional GC/MS. Attention required for splitless injection.
Schneier,Bruce	Applied Cryptography	Easy to use, high speed	Export of cryptography and cryptographic software and hardware