**VISION INSTITUTE OF MANAGEMENT**

**COMPUTER NETWORK SECURITY**

**BCA 3rd YEAR/6th SEM**

**UNIT 2(NETWORK SECURITY)**

**Authentication Applications**

**Networks** that are used to store, process, or transmit sensitive information must provide appropriate protection to prevent undesirable events such as compromise of information or denial of service. This document will show how to improve the **security** of a **network** through the use of **authentication applications**.

# ****Kerberos**:**The Computer Network Authentication Protocol:

# 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.

# With the technological advancements, communication over the internet is rapidly increasing day by day. Well, so does the increase of eavesdropping and replay attacks over the internet. Many authentic services have been developed to prevent malicious users from trying to access data passed over the network. Kerberos is such protocol designed to ensure the security when communicating over a non-secure network. Simply I can put it as an authentication protocol which allows only legitimate users to access the intended services over the internet.

# Kerberos was named after the ferocious three-headed guard dog of Hades appearing in Greek myths. Kerberos was developed by Massachusetts Institute of Technology (MIT) for a project called Athena.

Main entities involved in Kerberos flow:

* **Client:** Initiates the communication for a service request. Acts on behalf of the user.
* **Server:** The server with the service the user wants to access.
* **Authentication Sever (AS):** Performs client authentication. If the client is authenticated successfully the AS issues a ticket called TGT (**Ticket Granting Ticket**). TGT proves to other servers that client has been authenticated.
* **Key Distribution Centre (KDC):** In Kerberos environment authentication sever is logically separated into three parts: Database (db), Authentication Server (AS) and Ticket Granting Server (TGS). Physically these 3 parts are existing in a single server and it is called as Key Distribution centre.
* **Ticket Granting Server** (TGS): An application server which provides the issuing of service tickets as a service.



In Kerberos flow there are 3 important secret keys. A unique secret key is there for client/user, TGS and server which is shared with the AS.

* **Client/user secret key**: Hash derived by the user’s password.
* **TGS secret key**: Hash of the password used to determine TGS.
* **Server secret key**: Hash of the password used to determine the server offering the service.

Now comes the interesting part. I’ll be explaining step by step what happens in the Kerberos authentication protocol. Please note that I’ll be using several abbreviations when explaining to prevent the repetition of common terms.

First of all the user has to log into his/her account. By using the password provided by the user, client secret key is computed. Next comes the actual Kerberos flow.

# ****Step 1**** -

This is the initial authentication request. The client requests AS for a Ticket Granting Ticket (TGT). The client ID is sent in the request. Note that the password and client/user secret key is not sent.

# ****Step 2**** -

The AS checks for the availability of the client and TGS in the database (db). If either of values are not found, then an error message is sent to the client. If both values are available then the client/user secret key is generated by using the hash of the password of the user. The password of the user is available in the db. Also the TGS secret key is computed.

AS generates a session key (SK1) to be shared between the client and the TGS. SK1 is encrypted using the client secret key.

AS generates a TGT which contains client ID, client network address, lifetime, timestamp and SK1. The ticket is encrypted with the TGS secret key, so its contents can only be deciphered by the TGS.

The response message to be sent to client contains of the generated SK1 and TGT. Finally this message body is encrypted with the client/user secret key so that only the client is able to decode the message.

# ****Step 3**** -

Client decrypts the message using its client/user secret key (generated from the password entered by the user) and extracts SK1 and TGT. The authenticator which is used to validate the client to TGS is generated. The authenticator contains client ID, client network address and client machine timestamp and is encrypted using the extracted SK1.

The client sends the created authenticator and the extracted TGT to TGS, requesting a ticket from the server offering the service.

# ****Step 4 -****

The TGS decrypts the received TGT using the TGS secret key and extracts SK1. Using this key, the TGS decrypts authenticator and checks if client ID and client network address from TGT and authenticator match. A check is also performed to find if the TGT has not expired from. This is done from the timestamp extracted.

If all the checks are met then a service session key (SK2) is generated. SK2 is the secret shared between the client and the target server.

A service ticket containing client id, client network address, timestamp and SK2 is generated. This ticket is encrypted with the server secret key which is obtained from the db.

The response message body to be sent to the client contains SK2 and service ticket. Finally the message is encrypted with SK1 which is known to the client.

# ****Step 5 -****

After receiving the message, the client decrypts it using SK1 and extracts SK2. A new authenticator, which includes the client ID, client network address and timestamp is generated. This authenticator is encrypted using SK2.

The client sends the authenticator and service ticket to the target server.

# ****Step 6 -****

The target server decrypts the service ticket with the server secret key. SK2 is extracted from the service ticket. Next the authenticator is decrypted using SK2 and client ID, client network address, timestamp is extracted.

Checks are performed to verify if client ID and client network address from service ticket and authenticator match. A check is also performed to find that the service ticket has not expired.

If all checks are met the target server returns a message consisting of the time stamp plus 1, encrypted with SK2 to the client. This message proves that client and the server have completely authenticated each other. Therefore a trusted service session can now begin.

So this is how the Kerberos authentication protocol allows clients to communicate over a network in a secure manner.

# X.509 certificate:

# An X. 509 certificate is a digital certificate that uses the widely accepted international X. 509 public key infrastructure (PKI) standard to verify that a public key belongs to the user, computer or service identity contained within the certificate.

An X.509 certificate is something that can be used in software to both:

1. Verify a person’s identity so you can be sure that the person *really* is who they say they are.
2. Send the person who owns the certificate encrypted data that only they will be able to decrypt and read.

To be fair, X.509 certificates can be used to do these things for more than just people – they’re heavily used by software applications or computers to do this amongst themselves as well.  Anyway, just keep in mind when I say ‘person’ in this article, it can mean any of these.

Let’s look at these two use cases a little more closely.

## **Identity Verification**

So we’ve said that an X.509 certificate can be used to verify a person’s identity so you can trust that they really are who they say they are.

In this use case, you can think of an X.509 certificate as similar to a national passport. And as with national passports, you are very careful about who would ever have access to it – you would never give your passport away to anyone else. It uniquely identifies you and only you. Because of this uniqueness, the government uses passports to verify who you are – you present it as a way of proving that you are a citizen of your country.

X.509 certificates act kind of like a digital passport of sorts – it contains information about you and only you. And just as a national government acts as an authority for issuing and validating passports, something similar, called a Certificate Authority (CA), exists for X.509 certificates.

A Certificate Authority is a 3rd party trusted by both you and anyone who might verify your identity. That is, when you use your X.509 certificate with someone who needs to verify your identity, you both trust that a certain Certificate Authority has validated your identity. Because the 3rd party trusts that the CA verified you, they in turn trust that your X.509 certificate really represents you and only you.

There are well-known global and public Certificate Authorities, such as [Verisign](https://www.verisign.com/) and [Digicert](https://www.digicert.com/). But a Certificate Authority can also be any party that both you and the person verifying you agree to as trusted. Many companies have their own private Certificate Authorities used to verify employee identities, for example.

## **Securing your data**

In addition to verifying your identity, X.509 certificates can also be used to secure data intended for you so that prying eyes won’t be able to see it. It does this via a mathematical concept known as asymmetric key cryptography.

## **Asymmetric Key Cryptography**

Asymmetric means, well, **not** symmetric of course, but for the purpose of this discussion, it helps to think of asymmetric as ‘not equal or similar’. A key in this case is what you would think it would be – something used to lock or unlock a protective barrier. So when put together, asymmetric key cryptography basically means that one key is used to lock up data, but an entirely different key is used to unlock the data.

Asymmetric Key cryptography is also known as Public/Private Key cryptography for this reason: one of the two asymmetric keys can be freely given out to the world at large; anyone can see and use it, which is why it is called the ‘public’ key. The other of the two keys however must remain totally private to you, so no one will ever see it or be able to use it. This is naturally called the ‘private’ key because it should remain private always.

The reason Public/Private Key cryptography works is that data locked by the public key can only be unlocked by the private key and vice versa.  If someone locks data with the public key, no one else who has the public key can unlock it – not even the person that originally locked it.  Only the person with the private key can unlock the data.  That is why the public key can be given to and seen by anyone.  As long as the private key remains safe, you can rest assured the data is locked safely.

So if we had a public and private key, how would they be used?

As an example, let’s say your bank wants to send you your bank account balance. Naturally this is very sensitive information that should be for your eyes only. The bank can use your *public* key to encrypt your bank account balance. The encrypted data can be safely emailed directly to you (maybe as a file attachment), because it is all ‘jumbled up’ and no one can make sense of it (remember that public keys cannot unlock data that was previously locked with the same public key).

Because your private key is the only thing that can ‘un’ encrypt (aka ‘decrypt’) the bank’s email, you can use your private key to unlock the attached file and see your bank account balance. Because no one else should ever have your private key, you can rest assured that you’re the only one who can see your bank account balance.

## **X.509 cryptography**

X.509 certificate is composed of two chunks of information:

1. Your identifying information, such as your name and maybe address
2. Your public key

As you can see, your public key is included in the X.509 certificate. This way, anyone who gets your certificate can both verify your identity (via a Certificate Authority) and encrypt data ‘for your eyes only’ with the included public key.

 

## **Create an X.509 Certificate**

Anyone can create their own X.509 certificate. Just know that until it is verified by a Certificate Authority (like Digicert or Verisign or your company’s own CA), most people won’t trust it or use it. But we can list the process of creating your own and having a CA verify it.

Here is the high level overview of how one would create a validated X.509 certificate:

1. Using a cryptographic software tool (such as [OpenSSL](https://www.openssl.org/), the Java [key tool](https://download.oracle.com/javase/6/docs/technotes/tools/solaris/keytool.html), etc.), a user generates a cryptographic public/private key pair and of course keeps the private key very secret (known to himself only).
2. Using their software tool, the user’s public key and an X.500 hierarchical name identity, called a ‘Distinguished Name’ are bundled up together into an intermediate file called a certificate. This certificate is **not** considered a valid X.509 certificate yet.  We’ll get to that.
3. The user then encrypts this certificate using their private key which results in a new file called a ‘Certificate Signing Request’ or CSR.  If you look inside this file, you’ll see —–BEGIN CERTIFICATE REQUEST—– and —–END CERTIFICATE REQUEST—– lines sandwiching a Base 64-encoded blob.
4. The user then submits this CSR file to a Certificate Authority (Digicert, Verisign, etc.) along with his public key (so the CA knows how to decrypt the CSR).
5. The CA decrypts the CSR using the user’s public key which results in the intermediate certificate from step 2 that contains the user’s Distinguished Name and the user’s public key.
6. The CA verifies the identity of the user associated with the CSR.  They will ask for faxed identification of some sort – a national ID, a driver’s license, passport, etc.
7. Once the CA is happy that they have verified the user’s identity, they take the intermediate certificate and encrypt it with their own private key – which results in a new file.  If you look inside this file, you’ll see —–BEGIN CERTIFICATE —– and —–END CERTIFICATE—– lines sandwiching a Base 64-encoded blob.  **This file is what is known a valid X.509 certificate**.
8. The CA sends this newly created X.509 certificate back to the person who was verified.

After this final step, the user has their validated X.509 certificate that they can use as necessary.

**What is Authentication Service?**

**Authentication Service** facilitates username/password validation using your on-premises Active **Directory**/LDAP **server**. **Authentication Service** is installed as a virtual appliance and communicates with your local **directory** using LDAP over SSL.

Authentication Service facilitates username/password validation using your on-premises Active Directory/LDAP server. Authentication Service is installed as a virtual appliance and communicates with your local directory using LDAP over SSL. It can operate in the DMZ or inside the local area network (LAN), or both, based on the mode(s) of operation:

|  |  |
| --- | --- |
|  | * Desktop single sign-on (SSO). This option applies to end users using cloud or hybrid filtering to access the Internet from within your network. In this case, the user's desktop credentials are validated by Authentication Service using Kerberos tickets distributed by your Key Distribution Center (KDC) machine. Authentication Service is installed inside the LAN and acts as a federation server within your network, creating an in-network federation authority that communicates with the Websense proxy using SAML 2.0 assertions.
 |

The user authenticates with the Active Directory/LDAP server within the network (leveraging existing network security). When a user from within the corporate network accesses an external URL, they are redirected to Authentication Service, which authenticates the user with the LDAP directory and generates a SAML assertion to the Websense proxy. The user credentials never leave the corporate network.

Note that using this configuration, all user authentications happen in-network; the Websense proxy does not enforce multiple authentication factors, but simply accepts the SAML assertion from Authentication Service. Users can also use this mode from outside the network via a VPN connection.

 

* Username/Password verification. This option applies to off-site users. In this case, the users can access the Websense proxy from outside their LAN and Authentication Service needs to run in your DMZ. The user's Active Directory/LDAP credentials are collected by the Websense proxy and passed to Authentication Service to be validated against your Active Directory/LDAP server. Once authenticated, the user has full access to Web sites according to their policy settings.

 

* Hybrid (both). Here both internal desktop SSO and external username/password validation are required. Users can connect to Authentication Service internally or from outside the LAN.

**PGP (Pretty Good Privacy)**

* PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
* PGP was designed to provide all four aspects of security, i.e., privacy, integrity, authentication, and non-repudiation in the sending of email.
* PGP uses a digital signature (a combination of hashing and public key encryption) to provide integrity, authentication, and non-repudiation. PGP uses a combination of secret key encryption and public key encryption to provide privacy. Therefore, we can say that the digital signature uses one hash function, one secret key, and two private-public key pairs.
* PGP is an open source and freely available software package for email security.
* PGP provides authentication through the use of Digital Signature.
* It provides confidentiality through the use of symmetric block encryption.
* It provides compression by using the ZIP algorithm, and EMAIL compatibility using the radix-64 encoding scheme.

### **Following are the steps taken by PGP to create secure e-mail at the sender site:**

* The e-mail message is hashed by using a hashing function to create a digest.
* The digest is then encrypted to form a signed digest by using the sender's private key, and then signed digest is added to the original email message.
* The original message and signed digest are encrypted by using a one-time secret key created by the sender.
* The secret key is encrypted by using a receiver's public key.
* Both the encrypted secret key and the encrypted combination of message and digest are sent together.

### **PGP at the Sender site (A)**



### **Following are the steps taken to show how PGP uses hashing and a combination of three keys to generate the original message:**

* The receiver receives the combination of encrypted secret key and message digest is received.
* The encrypted secret key is decrypted by using the sender's private key to get the one-time secret key.
* The secret key is then used to decrypt the combination of message and digest.
* The digest is decrypted by using the sender's public key, and the original message is hashed by using a hash function to create a digest.
* Both the digests are compared if both of them are equal means that all the aspects of security are preserved.

### **PGP at the Receiver site (B)**



### **Disadvantages of PGP Encryption:**

1. **The Administration is difficult:** The different versions of PGP complicate the administration.
2. **Compatibility issues:** Both the sender and the receiver must have compatible versions of PGP. For example, if you encrypt an email by using PGP with one of the encryption technique, the receiver has a different version of PGP which cannot read the data.
3. **Complexity:** PGP is a complex technique. Other security schemes use symmetric encryption that uses one key or asymmetric encryption that uses two different keys. PGP uses a hybrid approach that implements symmetric encryption with two keys. PGP is more complex, and it is less familiar than the traditional symmetric or asymmetric methods.
4. **No Recovery:** Computer administrators face the problems of losing their passwords. In such situations, an administrator should use a special program to retrieve passwords. For example, a technician has physical access to a PC which can be used to retrieve a password. However, PGP does not offer such a special program for recovery; encryption methods are very strong so, it does not retrieve the forgotten passwords results in lost messages or lost files.

# S/MIME

# S/MIME (Secure/Multipurpose Internet Mail Extensions) is a widely accepted method (or more precisely, a protocol) for sending digitally signed and encrypted messages. S/MIME allows you to encrypt emails and digitally sign them. When you use S/MIME with an email message, it helps the people who receive that message to be certain that what they see in their inbox is the exact message that started with the sender. It will also help people who receive messages to be certain that the message came from the specific sender and not from someone pretending to be the sender. To do this, S/MIME provides for cryptographic security services such as authentication, message integrity, and non-repudiation of origin (using digital signatures). It also helps enhance privacy and data security (using encryption) for electronic messaging.

#

The steps that you follow to set up S/MIME with each of these end points is slightly different. Generally, you will need to do the following steps:

1. Install a Windows-based Certification Authority and set up a public key infrastructure to issue S/MIME certificates. Certificates issued by third-party certificate providers are also supported.
2. Publish the user certificate in an on-premises AD DS account in the **UserSMIMECertificate** and/or **UserCertificate** attributes.
3. For Exchange Online organizations, synchronize the user certificates from AD DS to Azure Active Directory by using an appropriate version of Azure AD Connect. These certificates will then get synchronized from Azure Active Directory to Exchange Online directory and will be used when encrypting a message to a recipient.
4. Set up a virtual certificate collection in order to validate S/MIME. This information is used by Outlook on the web when validating the signature of an email and ensuring that it was signed by a trusted certificate.
5. Set up the Outlook or EAS end point to use S/MIME

## **Related message encryption technologies:**

As message security becomes more important, admins need to understand the principles and concepts of secure messaging. This understanding is especially important because of the growing variety of protection-related technologies (including S/MIME) that are available.A variety of encryption technologies work together to provide protection for messages at rest and in-transit. S/MIME can work simultaneously with the following technologies but is not dependent on them:

* **Transport Layer Security (TLS)** encrypts the tunnel or the route between email servers in order to help prevent snooping and eavesdropping.
* **Secure Sockets Layer (SSL)** encrypts the connection between email clients and Office 365 servers.
* **BitLocker** encrypts the data on a hard drive in a datacentre so that if someone gets unauthorized access, they can't read it.



### **S/MIME compared with Office 365 Message Encryption:**

S/MIME requires a certificate and publishing infrastructure that is often used in business-to-business and business-to-consumer situations. The user controls the cryptographic keys in S/MIME and can choose whether to use them for each message they send. Email programs such as Outlook search a trusted root certificate authority location to perform digital signing and verification of the signature. Office 365 Message Encryption is a policy-based encryption service that can be configured by an administrator, and not an individual user, to encrypt mail sent to anyone inside or outside of the organization. It's an online service that's built on Azure Rights Management (RMS) and does not rely on a public key infrastructure. Office 365 Message Encryption also provides additional capabilities, such as the capability to customize the mail with organization's brand.

