UNIT-2

Kerberos

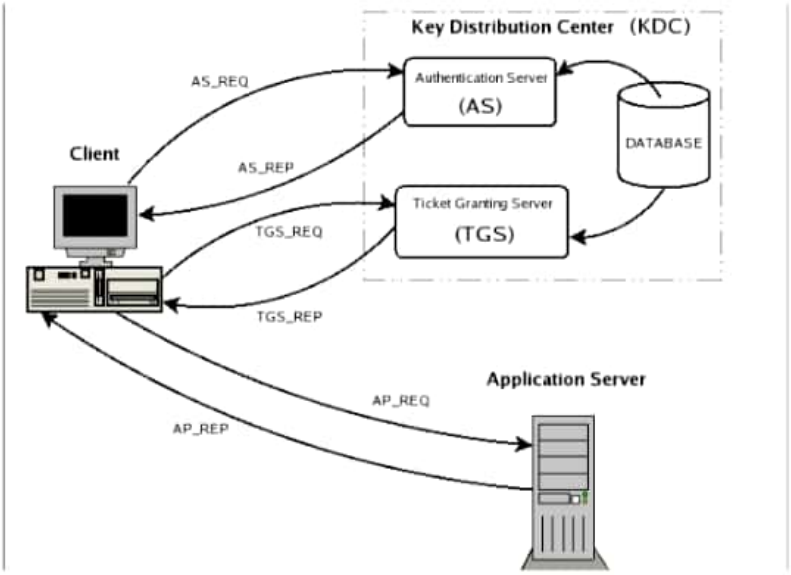
Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. Kerberos is built in to all major operating systems, including Microsoft Windows, Apple OS X, FreeBSD and Linux.

Since Windows 2000, Microsoft has incorporated the Kerberos protocol as the default authentication method in Windows, and it is an integral component of the Windows Active Directory service.

Kerberos was originally developed for Project Athena at the Massachusetts Institute of Technology (MIT). The name Kerberos was taken from Greek mythology; Kerberos (Cerberus) was a three-headed dog who guarded the gates of Hades. The three heads of the Kerberos protocol represent a client, a server and a Key Distribution Center (KDC), which acts as Kerberos' trusted third-party authentication service.

Working of Kerberos:

- The authentication service, or AS, receives the request by the client and verifies that the client is indeed the computer it claims to be. This is usually just a simple database lookup of the user's ID.
- Upon verification, a timestamp is created. This puts the current time in a user session, along with an expiration date. The default expiration date of a timestamp is 8 hours. The encryption key is then created. The timestamp ensures that when 8 hours is up, the encryption key is useless.
- The key is sent back to the client in the form of a ticket-granting ticket or TGT. This is a simple ticket that is issued by the authentication service. It is used for authentication the client for future reference.
- The client submits the ticket-granting ticket to the ticket-granting server, or TGS, to get authenticated.
- The TGS creates an encrypted key with a timestamp and grants the client a service ticket.
- The client decrypts the ticket, tells the TGS it has done so, and then sends its own encrypted key to the AS.
- The AS decrypts the key and makes sure the timestamp is still valid. If it is still valid, communication is initiated between client and server.



Pretty good privacy(PGP)

Pretty Good Privacy or PGP is a popular program used to encrypt and decrypt email over the Internet, as well as authenticate messages with digital signatures and encrypted stored files.

Previously available as freeware and now only available as a low-cost commercial version, PGP was once the most widely used privacy-ensuring program by individuals and is also used by many corporations. It was developed by Philip R. Zimmermann in 1991 and has become a de facto standard for email security.

It uses public key cryptography, symmetric key cryptography, hash function, and digital signature. It provides −

- Privacy
- Sender Authentication
- Message Integrity
- Non-repudiation

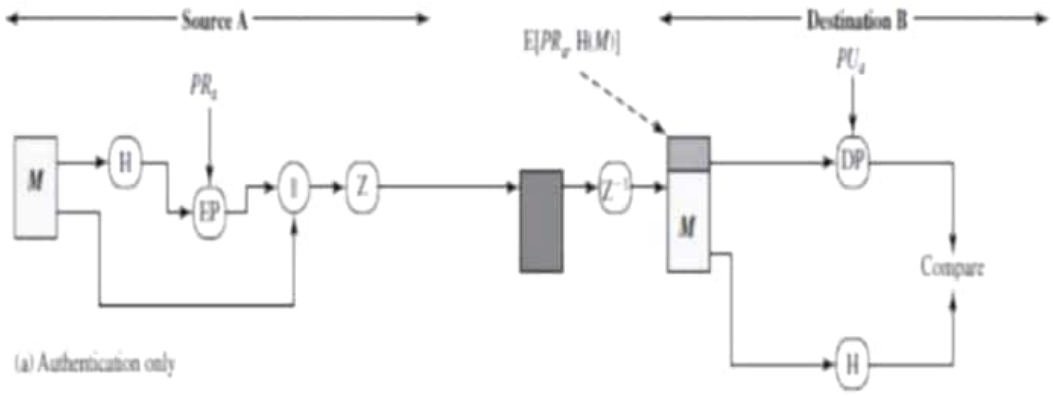
Along with these security services, it also provides data compression and key management support. PGP uses existing cryptographic algorithms such as RSA, IDEA, MD5, etc., rather than inventing the new ones.

Services provided by PGP are:

Authentication

The digital signature service provided by PGP.

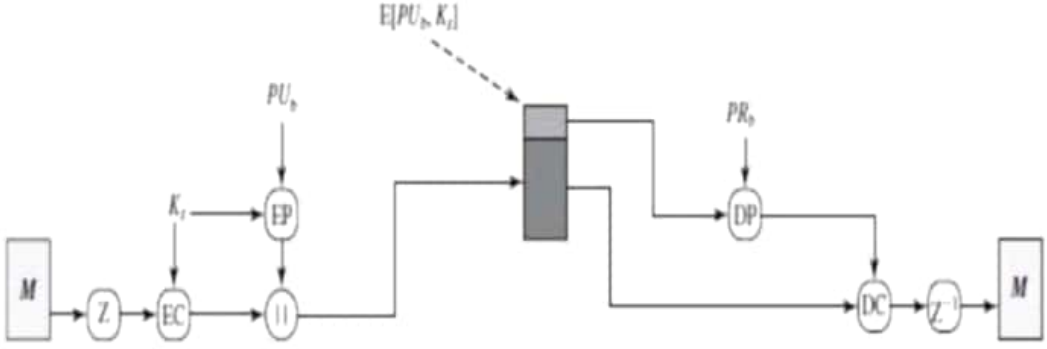
- The sender creates a message.
- SHA-1 is used to generate a 160-bit hash code of the message.
- The hash code is encrypted with RSA using the sender's private key, and the result is prepended to the message.
- The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
- The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.



Confidentiality

PGP another service is confidentiality, which is encrypting messages for transmitting or to store files locally. In both cases, the symmetric encryption algorithm CAST-128 may be used. Alternatively, IDEA or 3DES may be used. And the 64-bit cipher feedback (CFB) mode is used. In PGP, each symmetric key is used only once. The session key is bound to the message. To protect the key, it is encrypted with the receiver's public key.

- The sender generates a message and a random 128-bit number to be used as a session key for this message only.
- The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
- The session key is encrypted with RSA using the recipient's public key and is prepended to the message.
- The receiver uses RSA with its private key to decrypt and recover the session key.
- The session key is used to decrypt the message.



Compression

- PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.
- Z for compression and Z–1 for decompression.
- The signature is generated before compression for two reasons:
  - It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification.
  - If you generate signature after compression then there is a need recompression for message verification, PGP's compression algorithm presents a difficulty.
- Message encryption is applied after compression to strengthen cryptographic security. Therefore cryptanalysis is more difficult.

E-Mail-Compatibility

The resulting message block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is radix-64 conversion.

- Each group of three octets of binary data is mapped into four ASCII characters.
- This format also appends a CRC to detect transmission errors.
- The use of radix 64 expands a message by 33%. Fortunately, the session key and signature portions of the message are relatively compact, and the plaintext message has been compressed.
- In fact, the compression should be more than enough to compensate for the radix-64 expansion.

Segmentation

E-mail facilities are often restricted to a maximum message length Any message longer than that must be broken up into smaller segments, each of which is mailed separately. To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to sent via e-mail. The segmentation is done after all the other processing, including the radix-64 conversion. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block.

MIME

The MIME stands for Multi-Purpose Internet Mail Extensions. As the name indicates, it is an extension to the Internet email protocol that allows it's users to exchange different kinds of data files over the Internet such as images, audio, and video. MIME is a kind of protocol which allows non-ASCII data to be sent through SMTP. In 1991, Nathan Borenstein of Bellcore proposed to the IETF that SMTP be extended so that Internet (but mainly Web) clients and servers could recognize and handle other kinds of data than ASCII text. As a result, new file types were added to "mail" as a supported Internet Protocol file type.

MIME was invented to overcome the following limitations of SMTP:

- SMTP cannot transfer executable files and binary objects.
- SMTP cannot transmit text data of other language, e.g. French, Japanese, Chinese etc, as these are represented in 8-bit codes.
- SMTP services may reject mails having size greater than a certain size.
- SMTP cannot handle non-textual data such as pictures, images, and video/audio content.

Features of MIME –

- It is able to send multiple attachments with a single message.
- Unlimited message length.
- Binary attachments (executables, images, audio, or video files) which may be divided if needed.
- MIME provided support for varying content types and multi-part messages.

MIME Header

The five header fields defined in MIME are as follows:

MIME Header

The five header fields defined in MIME are as follows:

1. MIME-version - It indicates the MIME version being used. The current version is 1.1. It is represented as : MIME-version: 1.1.

2. Content-type - It describes the type and subtype of the data in the body of the message. The content type and content subtype are separated by slash. This field describes how the object in the body is to be interpreted. The default value is plaintext in US ASCII. Content type field is represented as: Context-type: <type/subtype; parameters>

3. Content-transfer encoding - It describes how the object within the body has been encoded to US ASCII to make it acceptable for mail transfer. Thus it specifies the method used to encode the message into 0s and 1s for transport. The content transfer encoding field is represented as :

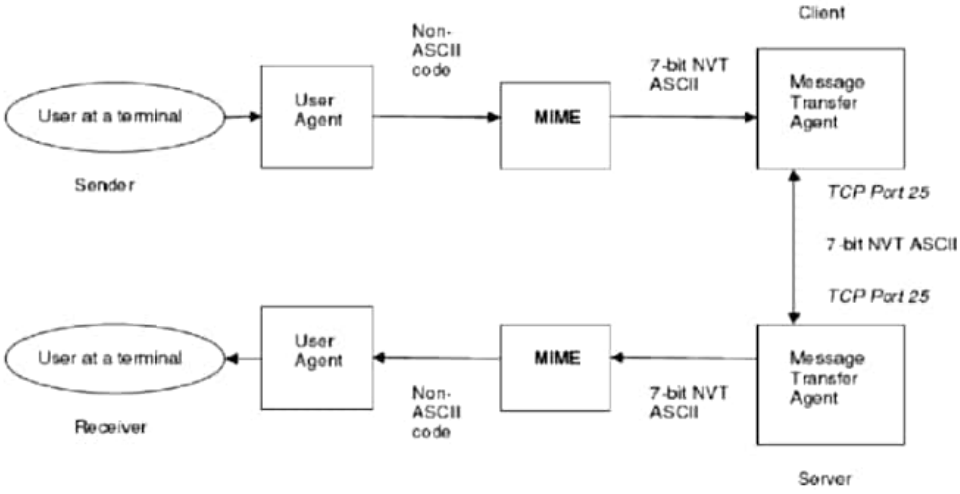
Content-transfer-encoding : <type>

4. Content-Id - It is used to uniquely identify the MIME entities in multiple contexts i.e. it uniquely identifies the whole message in a multiple message environment. This field is represented as:

Content-id : id = <content-id>

5. Content-description - It is a plaintext description of the object within the body; It specifies whether the body is image, audio or video. This field is represented as:

Content-description: <description>



S/MIME

S/MIME, the acronym of Secure/Multipurpose Internet Mail Extensions, is a universal web standard defined by the IETF. S/MIME is employed to encrypt emails. Initially developed by RSA Data Security, it's been made a standard by the IETF and has been defined in many of its documents.

S/MIME, based on Public Key Infrastructure or Asymmetric Encryption, facilitates email security by virtue of encryption, authentication, and integrity. In other words, it allows you to digitally sign your emails so that only the intended recipient can see the message and he/she also gets to know that the email really came from you. While the email is on its way, the encryption takes care of its integrity as it doesn't allow an unauthorized third party to intercept and tamper with the data.

However, this is not enough. Even if your email server has been encrypted, it still cannot stop a perpetrator from stealing emails from your inbox as the emails are stored unencrypted on the servers. Also, it doesn't protect when your emails are in transit from another server.

To implement S/MIME, you must install an S/MIME certificate—also known as "client certificate." As they work on the principles of public key infrastructure (PKI), what they do is pretty similar to SSL certificates. The only key difference is that these certs are installed on clients while SSL/TLS certs are implemented on servers.

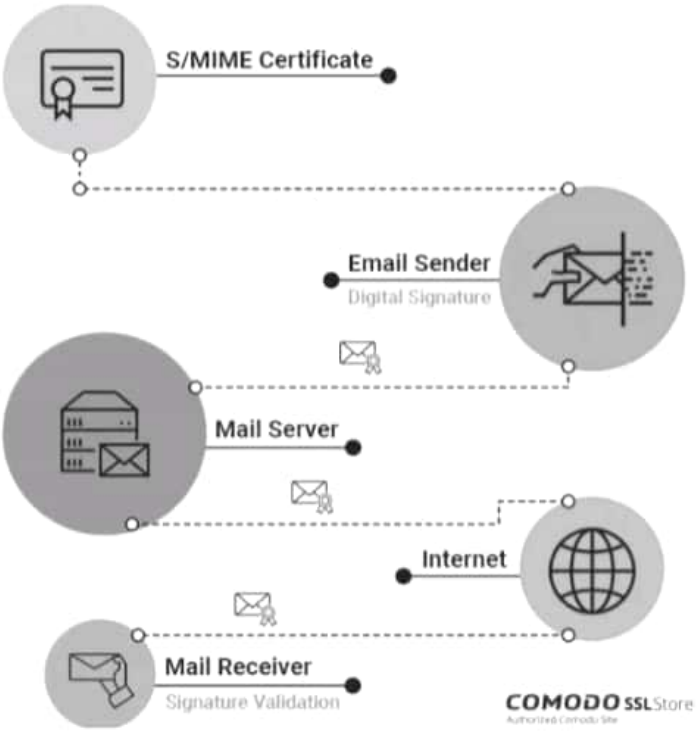
How does S/MIME (Client) certificate work?

S/MIME certs are based on asymmetric encryption. That's why they involve two distinct keys – a public key and a private key. Keep in mind that a public key and private key come in a pair, one public key can only have one private key and vice versa. This is because they're mathematically related to each other, the public key is actually derived from the private key.

While using an S/MIME certificate, a sender sends an email by encrypting it through recipient's public key. On the other side, the recipient decrypts the email using the private key he/she has. There's no scope for someone else to come in between and see or tamper with the email as it's in an encrypted format.

In simple terms, this entire process is called 'signing.' You might have heard some security experts talk about signing emails. Well, this is what signing your email means.

Signing emails, as you can see, removes two significant roadblocks in email security. First, it takes away the likelihood of a 3rd party intervention while the email is in transit. This will work even if you haven't encrypted your server (but don't use this as an excuse, you still need to encrypt your server). Another thing this does is provides authentication to the user as the signature of the sender is attached to every email.

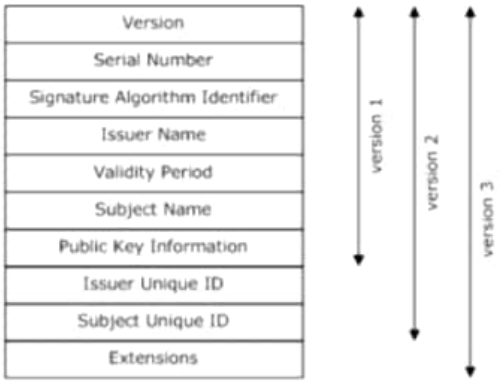


X.509 certificate

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 contains information about the identity to which a certificate is issued and the identity that issued it. Standard information in an X.509 certificate includes:

- **Version –** which X.509 version applies to the certificate (which indicates what data the certificate must include)
- **Serial number –** the identity creating the certificate must assign it a serial number that distinguishes it from other certificates
- **Algorithm information –** the algorithm used by the issuer to sign the certificate
- **Issuer name –** the name of the entity issuing the certificate (usually a certificate authority)
- **Validity period of the certificate –** start/end date and time
- **Subject name** – the name of the identity the certificate is issued to
- **public key information –** the public key associated with the identity

The first X.509 certificates were issued in 1988 as part of the International Telecommunications Union's Telecommunication Standardization Sector (ITU-T) and the X.500 Directory Services Standard. In 1993, version 2 added two fields to support directory access control. Version 3 was released in 1996 and defines the formatting used for certificate extensions.



The process of obtaining Digital Certificate by a person/entity is depicted in the following illustration.

As shown in the illustration, the CA accepts the application from a client to certify his public key. The CA, after duly verifying identity of client, issues a digital certificate to that client.