NETWORK SECURITY UNIT 4,5 AND 6

TYBSC(IT) SEM 5

COMPILED BY : SIDDHESH ZELE

302 PARANJPE UDYOG BHAVAN, NEAR KHANDELWAL SWEETS, NEAR THANE STATION , THANE (WEST)

PHONE NO: 8097071144 / 8097071155 / 8655081002

UNIT 4

Digital Certificates and Public Key Infrastructure (PKI)

Digital Certificate

- To solve the man-in-the-middle attack, Digital Certificates were introduced.
- Digital certificate is similar to a passport, which give information of the issuer's name, serial number, public key, validity period.
- Digital Certificate is issued by a trusted agency called as CA (Certification Authority).
- Another third party called as RA(Registration Authority) acts as a intermediate entity between CA and end user.
- Satisfies the principle of Authentication, non-repudiation.

Concept of Digital Certificates

A digital certificate is simply a small computer file. For example, my digital certificate would actually be a computer file with a file name such as name.cer

a digital certificate establishes the relation between a user and his/her public key. Therefore, a digital certificate must contain the user name and the user's public key. This will prove that a particular public key belongs to a particular user.

the digital certificate is actually quite similar to a passport. Just as every passport has a unique passport number, every digital certificate has a unique serial number. As we know, no two passports issued by the same issuer (i.e. government) can have the same passport number. Similarly, no two digital certificates issued by the same issuer can have the same serial number.

Certification Authority (CA)

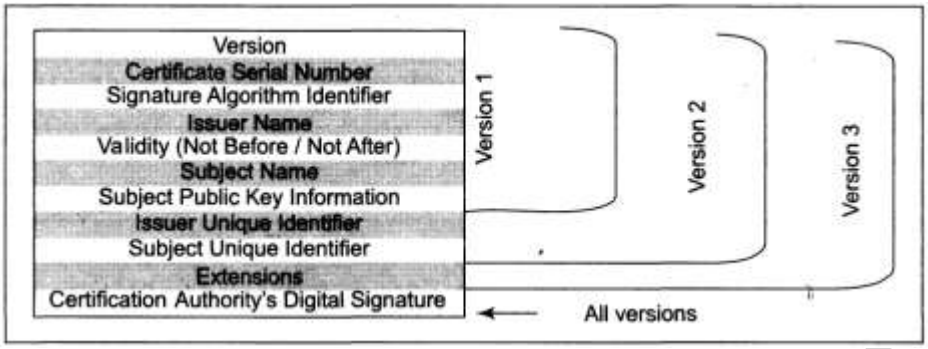
A **Certification Authority (CA)** is a trusted agency that can issue digital certificates.

Who can be a CA?

- The authority of acting as a CA has to be with someone who everybody trusts. Consequently, the governments in various countries decide who can and who cannot be a CA. (It is another matter that not everybody trusts the government in the first place!)
- Usually, a CA is a reputed organization, such as a post office, financial institution, software company, etc. Two of the world's most famous CAs are VeriSign and Entrust.
- Safescrypt Limited, a subsidiary of Satyam Infoway Limited, became the first Indian CA in February 2002.
- Thus, a CA has the authority to issue digital certificates to individuals and organizations, who want to use those certificates in asymmetric-key cryptographic applications.

Technical Details of a Digital Certificate

A standard called **X.509** defines the structure of a digital certificate. The International Telecommunication Union (ITU) came up with this standard in 1988. At that time, it was a part of another standard called **X.500.** Since then, X.509 was revised twice (in 1993 and again in 1995). The current version of the standard is Version 3, called **X.509V3.**



Fields of Version1	**Description**
Version	Version of X.509 protocol. Version can be 1,2 or 3
Certificate Serial No.	Contains unique integer which is generated by CA
Signature Algorithm Identifier	Identifies the algorithm used by CA to sign the certificate.
Issuer Name	Identifies the Distinguished Name that created & signed the certificate
Validity (not before/not after)	Contains two date-time value. This value generally specify the date & time up to seconds or milliseconds.
Subject name	Distinguished Name of the end user (user or organization)
Subject Public key info.	This field can never be blank. Contains public key & algorithm related.

Fields of Version 2	**Description**
Issuer Unique Identifier	Helps identify a CA uniquely if two or more CAs have used the same *IssuerName* over time.
Subject Unique Identifier	Helps identify a subject uniquely if two or more subjects have used the same *SubjectName* over time.

Fields of Version 3	**Description**
Authority Key Identifier	A CA may have multiple private-public key pairs. This field defines which key pairs is used to sign this certificate.
Subject Key Identifier	A subject have multiple private-public pairs. This field defines which key pairs is used to sign this certificate.
Key Usage	Defines the *scope of operations* of the public key of this particular certificate. For eg: key is used for encryption, Diffie Hellman Key exchange, digital signature, combination of these or all operations.

Extended key Usage	In addition to above operations, it also specifies which protocol this certificate can interoperate with. For eg: TLS, client or server authentication.
Private Key Usage period	Defines different usage period limits for the private & public keys
Certificate Policies	Defines the policies information that the CA associates with given certificate
Policy Mapping	Used when the subject of a given certificate is also a CA
Subject Alternative Name	Optionally defines one or more alternative names for subject.
Issuer Alternative Name	Optionally defines one or more alternative names for issuer.
Subject Directory Attributes	Can be used to provide additional information about the subject such as subject phone/fax numbers, email ids etc.
Basic Constraints	Indentifies whether the subject in this certificate may act as a CA.
Name Constraints	Specifies the name space.
Policy Constraints	Used only for CA certificates.

Digital-Certificate Creation

1. Parties Involved

- end user,
- issuer (CA),
- third party is also (optionally) called a **Registration Authority (RA).** involved in the certificate creation and management.

The RA commonly provides the following services

- Accepting and verifying registration information about new users
- Generating keys on behalf of the end users
- Accepting and authorizing requests for key backups and recovery
- Accepting and authorizing the requests for certificate revocation
- RA is mainly set up for facilitating the interaction between the end users and the CA
- The RA cannot issue digital certificates.
- The CA must handle this. Additionally, after a certificate is issued, the CA is responsible for all the certificate management aspects, such as tracking its status, issuing revocation notices if the certificate needs to be invalidated for some reason, etc.

2. *Certificate Creation Steps*

Step 1: Key Generation The action begins with the subject (i.e. the user/organization) who wants to obtain a certificate. There are two different approaches for this purpose:

The subject can create a private key and public key pair using some software. This software is usually a part of the Web browser or Web server. Alternatively, special software programs can be used for this. The subject must keep the private key thus generated, secret. The subject then sends the public key along with other information and evidences about herself to the RA.

Alternatively, the RA can generate a key pair on the subject's (user's) behalf. This can happen in cases where either the user is not aware of the technicalities involved in the generation of a key pair, or if a particular requirement demands that all the keys must be centrally generated and distributed by the RA for the ease of enforcing security policies and key management. Of course, the major disadvantages of this approach are the possibility of the RA knowing the private key of the user, as well as the scope for this key to be exposed to others while in transit after it is generated and sent to the appropriate user.

Step 2: Registration This step is required only if the user generates the key pair in the first step. If the RA generates the key pair on the user's behalf, this step will also be a part of the first step itself.

Assuming that the user has generated the key pair, the user now sends the public key and the associated registration information (e.g. subject name, as it is desired to appear in the digital certificate) and all the evidence about herself to the RA.

For this, the software provides a wizard in which the user enters data and when all data is correct, submits it. This data then travels over the network/Internet to the RA. The format for the certificate requests has been standardized, and is called **Certificate Signing Request (CSR).** This is one of the **Public Key Cryptography Standards (PKCS),**

Note that the user must not send the private key to the RA—the user must retain it securely. In fact, as far as possible, the private key must not leave the user's computer at all.

Step 3: Verification After the registration process is complete, the RA has to verify the user's credentials. This verification is in two respects, as follows.

1. Firstly, the RA needs to verify the user's credentials such as the evidences provided are correct, and that they are acceptable. If the user were actually an organization then the RA would perhaps

like to check the business records, historical documents and credibility proofs. If it is an individual user then simpler checks are in call, such as verifying the postal address, email id, phone number, passport or driving-license details can be sufficient.

2. The second check is to ensure that the user who is requesting for the certificate does indeed possess the private key corresponding to the public key that is sent as a part of the certificate request to the RA. This is very important, because there must be a record that the user possesses the private key corresponding to the given public key. Otherwise, this can create legal problems (what if a user claims that he/she never possessed the private key, when a document signed with him/her private key causes some legal problems?) This check is called checking the **Proof Of Possession (POP)** of the private key. How can the RA perform this check? There are many approaches to this, the chief ones being as follows.

   (i) The RA can demand that the user must digitally sign his/her Certificate Signing Request (CSR) using his/her private key. If the RA can verify the signature (i.e. de-sign the CSR) correctly using the public key of the user, the RA can believe that the user indeed possesses the private key.
   (ii) Alternatively, at this stage, the RA can create a random number challenge, encrypt it with the user's public key and send the encrypted challenge to the user. If the user can successfully decrypt the challenge using his/her private key, the RA can assume that the user possesses the right private key.
   (iii) Thirdly, the RA can actually generate a dummy certificate for the user, encrypt it using the user's public key and send it to the user. The user can decrypt it only if he/she can decrypt the encrypted certificate, and obtain the plain-text certificate.

Step 4: Certificate Creation Assuming that all the steps so far have been successful, the RA passes on all the details of the user to the CA. The CA does its own verification (if required) and creates a digital certificate for the user. There are programs for creating certificates in the X.509 standard format. The CA sends the certificate to the user, and also retains a copy of the certificate for its own record. The CA's copy of the certificate is maintained in a **certificate directory.** This is a central storage location maintained by the CA.

Why Should We Trust Digital Certificates

Consequently, a CA always signs a digital certificate with its private key. In effect, the CA says:

I have signed this certificate to guarantee that this user possesses the specified public key: *Trust me!*

How a CA Signs a Certificate

we have a digital certificate, which we want to verify. What should we do? Since the CA has signed the certificate with its private key, we would need to first verify the CA's signature. For this, we will use the CA's public key and check if it can de-sign the certificate correctly (we shall shortly see what this means). If the de-signing works successfully, we can consider the certificate to be a valid one.

every digital certificate not only contains the user's information (such as subject name, public key, etc.) but also the CA's digital signature. Like a passport, therefore, a digital certificate is always signed or certified. The way CA signs a certificate

How a Digital Certificate can be Verified

The verification of a digital certificate consists of the following steps.

(a) The user passes all fields except the last one of the received digital certificate to a message-digest algorithm This algorithm should be the same as the one used by the CA while signing the certificate. The CA mentions the algorithm used for signing along with the signature in the certificate, so the user here knows which algorithm is to be used. The message-digest algorithm calculates a message digest (hash) of all fields of the certificate, except for the last one. Let us call this message digest as MD1. The user now extracts the digital signature of the C A from the certificate
(b) The user de-signs the CA's signature (i.e. the user decrypts the signature with the CA's public key).
(c) This produces another message digest, which we shall call MD2. Note that MD2 is the same message digest as would have been calculated by the CA during the signing of the certificate (i.e. before it encrypted the message digest with its private key to create its digital signature over the certificate).
(d) Now, the user compares the message digest it calculated (MD1) with the one, which is the result of de-signing the CA's signature (MD2). If the two match, i.e. if MD1 = MD2, the user is convinced that the digital certificate was indeed signed by the CA with its private key. If this comparison fails, the user will not trust the certificate, and reject it.

Certificate Hierarchies and Self-signed Digital Certificates

Certificate hierarchy relieves the root CA from having to manage all the possible digital certificates. Instead, the root CA can delegate this job to the second-level CAs. This delegation can happen region-wise (e.g. one second level CA could be responsible for the Western region, another for the Eastern region, a third one for the Northern region, and a fourth one for the Southern region, etc.). Each of these second-level CAs could appoint third-level CAs state-wise within that region. Each third-level CA could delegate its responsibilities to a fourth-level CA city-wise, and so on.

The root CA (and many times, even the second or the third-level CAs) are automatically considered trusted CAs. How is this achieved? For this, Alice's software (usually the Web browser, but this can be any other piece of software that is able to store and verify certificates) contains a pre-programmed, hard-coded certificate of the root CA. Also, this certificate of the root CA is a **self-signed certificate,** i.e. the root CA signs its own certificate.

Cross-Certification

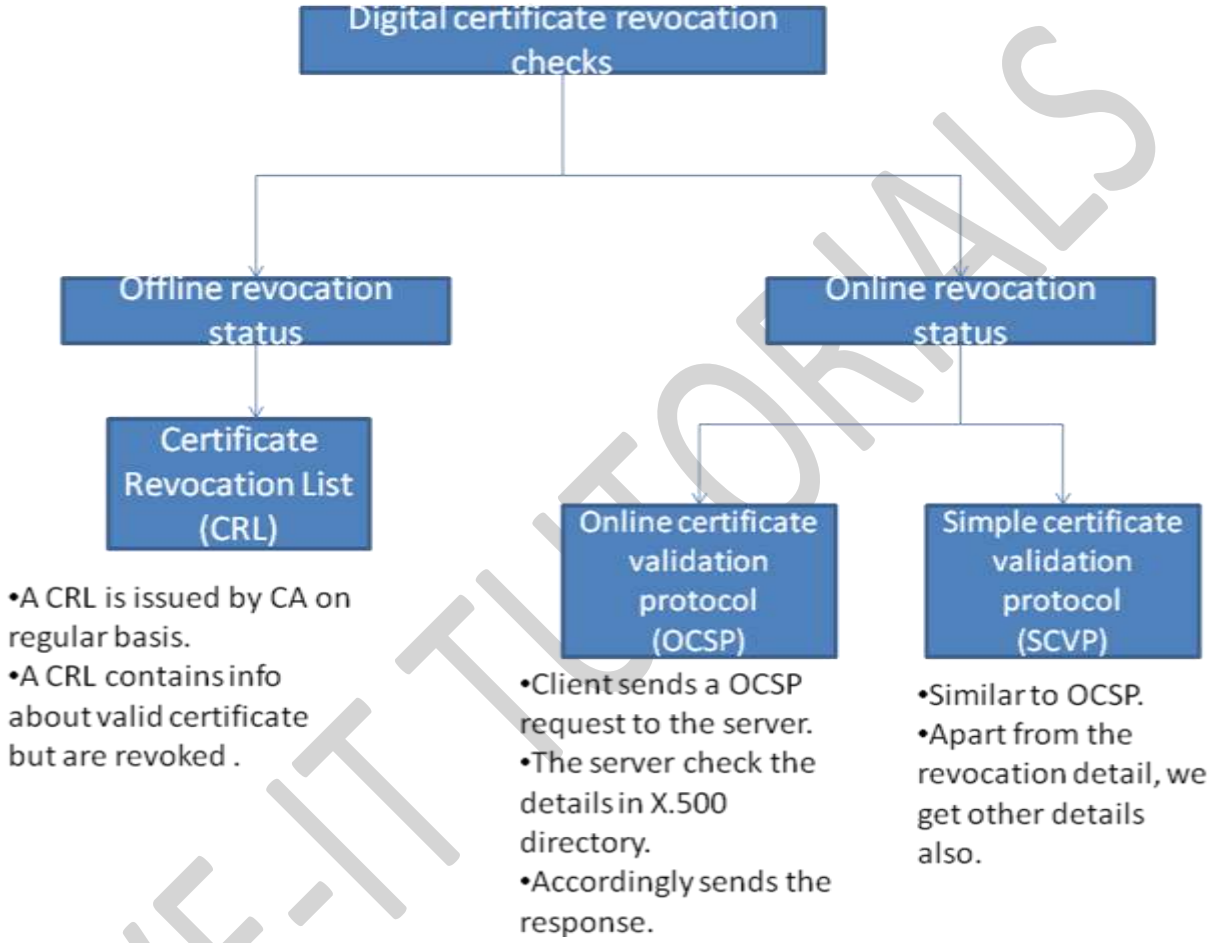
It is quite possible that Alice and Bob live in different countries. This would mean that their root CAs themselves could be different. This is because, generally each country appoints its own root CA. In fact, one country can have multiple root CAs as well. For instance, the root CAs in the US are VeriSign, Thawte, and the US Postal Service. In such cases, there is no *single* root CA, which can be trusted by all the concerned parties. In our example, why should Alice—a Japanese national, trust Bob's root CA—a US-based organization?

cross-certification allows CAs and end users from different PKI domains to interact called **cross certification**

Certificate Revocation

- Reasons for revocation:
- If the private key corresponding to the public key is stolen.
- The CA realizes that it had made mistake while issuing the certificate.
- The certificate holder leaves a job and the certificate was issued specifically while the person was employed in that job.

- If Alice wants to send message to Bob:-
  - Then Alice arises some questions as:
  - Does this certificate really belong to Bob?
  - Is this certificate valid, or is it revoked?



Certificate Revocation List (CRL) is the primary means of checking the status of a digital certificate offline.

entity performing check should do the following, in the given sequence:

1. *Certificate Expiry Check* Compare the current date with the validity period of the certificate to ensure that the certificate has not expired
2. *Signature Check:* Check that entity(x) certificate can be verified in terms of the signature by his CA
3. *Certificate Revocation: Check* Consult the latest CRL issued by entity(x) CA to ensure that entity(x) certificate is not listed there as a revoked certificate

Online Certificate Status Protocol (OCSP) The **Online Certificate Status Protocol (OCSP)** can be used to check if a given digital certificate is valid at a particular moment. Thus, it is an *online check*.

OCSP allows the certificate validators to check for the status of certificates in real time, thus providing for a quicker, simpler and more efficient mechanism for digital certificate validations.

steps:

- Client sends a OCSP request to the server.
- The server check the details in X.500 directory.
- Accordingly sends the response.

Simple Certificate Validation Protocol (SCVP) The **Simple Certificate Validation Protocol (SCVP)** is in the draft stage as of the current writing. SCVP is an online certificate status reporting protocol, designed to deal with the drawbacks of OCSP. Since SCVP is conceptually similar to OSCP, we shall simply point out the differences between the two

Point	OCSP	SCVP
Client request	The client sends just the certificate serial number to the server.	The client sends the entire certificate to the server. Consequently, the server can perform many more checks.
Chain of trust	Only the given certificate is checked.	The client can provide a collection of the intermediate certificates, which the server can check.
Checks	The only check is whether the certificate is revoked or not.	The client can request for additional checks (e.g. please check the full chain of trust), type of revocation information to be considered (i.e. whether the server should, in turn, use CRL or OCSP for revocation checks), etc.
Returned information	Only the status of the certificate is returned by the server.	The client can specify what additional information it is interested in (e.g. proof of the revocation status should be returned by the server, or the chain of certificates used for chain of trust validation should be returned by the server, etc.).
Additional features	None	The client can request for a certificate to be checked for a backdated event. For instance, suppose Bob has sent Alice a signed document along with his certificate. Then, Alice can use SCVP to check if Bob's certificate was valid at the time of signing(and not at the time of verification of the signature).

Certificate Types

(a) Email Certificates Email certificates include the user's email id. This is used to verify that the signer of an email message has an email id that is the same as it appears in that user's certificate.

(b) Server-side SSL Certificates These certificates are useful for merchants who want to allow buyers to purchase goods or services from their online Web site.

(c) Client-side SSL Certificates These certificates allow a merchant (or any other server-side entity) to verify a client (browser-side entity).

(d) Code-signing Certificates Many people do not like to download client-side code such as Java applets or ActiveX controls, because of the inherent risks associated with them. In order to alleviate these concerns, the code (i.e. the Java applets or ActiveX controls) can be signed by the signer. When a user hits a Web page that contains such code, the browser displays a warning message, indicating that the page contains such pieces of code, signed by the appropriate developer/organization, and whether the user would like to trust that developer/organization. If the user responds affirmatively, the Java applets or ActiveX controls are downloaded and get executed on the browser. However, if the user rejects the offer, the processing ends there. It must be noted that mere signing of code does not make it safe—the code could cause havoc. It simply specifies where the code originates.

Private Key Management

- To protect the private key by means:-
    - Password protection
    - PCMCIA cards (Personal Computer Memory Card International Association)
    - Tokens
    - Biometrics
    - Smart Cards
    - Apart from these, the private key used for digital signing must be destroyed. In contrast, the private key used for encryption/decryption must be archived.
        - In case of certificate expiration, the user need to update its key.
        - The CA should maintain history of certificates & keys to prevent any legal problems.

The PKIX (Public Key Infrastructure X.509) model

(a) Registration It is the process where an end-entity (subject) makes itself known to a CA. Usually, this is via an RA.

(b) Initialization This deals with the basic problems, such as the methodology of verifying that the end-entity is is talking to the right CA. We have seen how this can be tackled.

(c) Certification In this step, the CA creates a digital certificate for the end-entity and returns it to the end-entity, maintains a copy for its own records, and also copies it in public directories, if required.

(d) Key-Pair Recovery Keys used for encryption may be required to be recovered at a later date for decrypting some old documents. Key archival and recovery services can be provided by a CA or by an independent key-recovery system.

(e) Key Generation PKIX specifies that the end-entity should be able to generate private-and public-key pairs, or the CA/RA should be able to do this for the end-entity (and then distribute these keys securely to the end-entity).

(f) Key Update This allows a smooth transition from one expiring key pair to a fresh one, by the automatic renewal of digital certificates. However, there is a provision for manual digital certificate renewal request and response.

(g) Cross-certification Helps in establishing trust models, so that end-entities that are certified by different CAs can cross-verify each other.

(h) Revocation PKIX provides support for the checking of the certificate status in two modes: online (using OCSP) or offline (using CRL).

PKIX Architectural Model

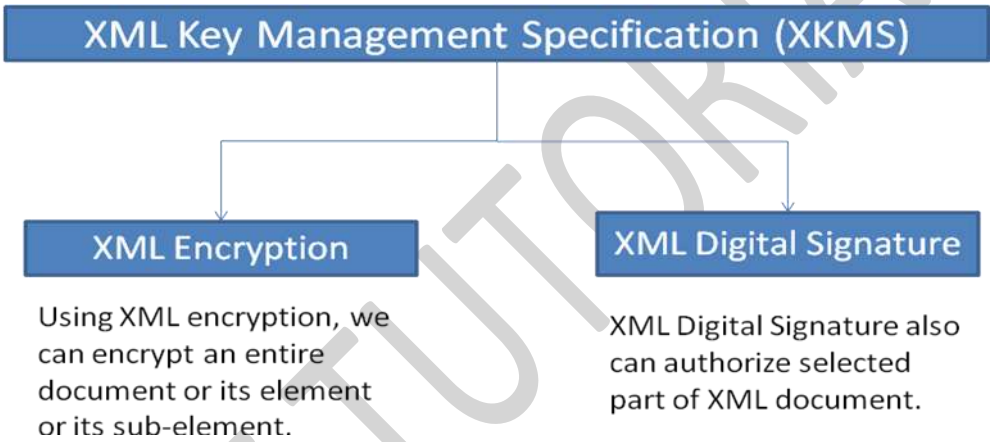
- **X.509 v3 Certificate & v2 Certificate Revocation List profiles**: (lists the use of various options while describing extensions of a digital certificate).
- **Operational Protocol**: (defines the underlying protocols that provide the transport mechanism).
- **Management Protocol**: (enables exchange of information between the various PKI entities. Specifies the structure & details of PKI messages).
- **Policy outlines**: (defines policies for the creation of Certificate Policies & Certificate Practice Statements.)
- **Timestamp & Data Certification Services**: (both are the trusted third parties that provide services to guarantee the existence of certificate & DCS verifies the correctness of data that it receives).

PKCS (Public Key Cryptography Standards)

Standard	Description
PKCS#1	RSA Encryption Standard. Defines rules for calculating digital certificate.
PKCS#2	RSA Encryption Standard for Message Digest.
PKCS#3	Diffie-Hellman Key Agreement Standard.
PKCS#4	NA. merged with PKCS#1
PKCS#5	Password Based Encryption(PBE). Defines KEK method to encrypt symmetric key.
PKCS#6	Extended Certificate Syntax Standard. Defines syntax for extending the basic attribute of an X.509 digital certificate.
PKCS#7	Cryptographic Message Syntax Standard.
PKCS#8	Private Key Information Standard.
PKCS#9	Selected Attribute Types. Defines selected attribute for use in PKCS#6 extended certificates.
PKCS#10	Certificate Request Syntax Standard
PKCS#11	Cryptographic Token Interface Standard. Specifies interfaces for single user
PKCS#12	Personal Information Exchange Syntax Standard.

PKCS#13	Elliptic Curve Cryptography Standard
PKCS#14	Pseudo –Random Number Generation Standard.
PKCS#15	Cryptographic Token Information Syntax standard

XML, PKI AND SECURITY



XML Encryption

The most interesting part about **XML encryption** is that we can encrypt an entire document, or its selected portions. This is very difficult to achieve in the non-XML world. We can encrypt one or all of the following portions of an XML document:

- The entire XML document
- An element and all its sub-elements
- The content portion of an XML document
- A reference to a resource outside of an XML document

XML Digital Signature

in a purchase request, the purchase manager may want to authorize only the quantity portion, whereas the accounting manager may want to sign only the rate portion. In such cases, **XML digital signatures** can be used.

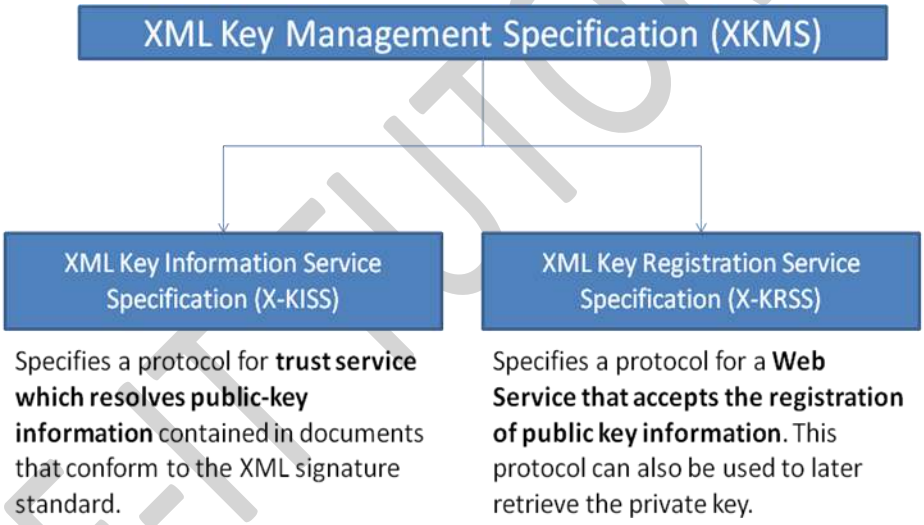
contents of the digital signature

- **<Signature> ... </Signature>** - This block identifies the start and end of the XML digital signature.
- **<SignedInfo> ... </SignedInfo>** - This block specifies the algorithm used: firstly for calculating the message digest (which is SHA-1, in this case) and then for preparing the XML digital signature. (which is RSA, in this case).
- **<SignatureValue> ... </SignatureValue>** - This block contains the actual XML digital signature.

XML digital signatures can be classified into three types: **enveloped, enveloping,** and **detached**

- In the *enveloped* XML digital signatures, the signature is inside the original document (which is being digitally signed).
- In the *enveloping* XML digital signatures, the original document is inside the signature.
- A *detached* digital signature has no enveloping concept at all, it is separate from the original document.

XML Key Management Specification (XKMS)



The **XML Key Information Service Specification (X-KISS)** specifies a protocol for a trust service, which resolves the public-key information contained in documents that conform to the XML signature standard. This protocol allows a client of such a service to delegate some or all of the tasks needed to process an XML signature element. The underlying PKI can be based on different specifications, such as X.509 or Pretty Good Privacy (PGP), and yet X-KISS shields the application from these differences.

The **XML Key Registration Service Specification (X-KRSS)** defones a protocol for a Web service, that accepts the registration of public-key information. Once registered, the public key can be used in relation with other Web services, including X-KISS. This protocol can also be used to later retrieve the private key. The protocol has provisions for authentication of the applicant and proof of possession of the private key.

The concept of Hash (Message Digest):

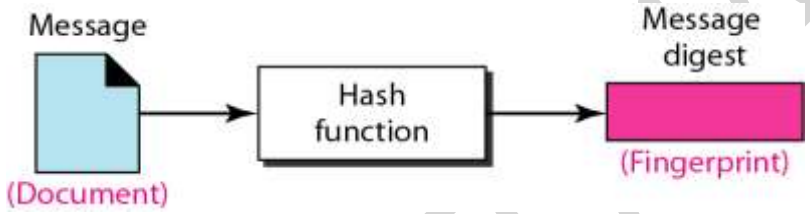
Signing the Digest

public-key encryption is efficient if the message is short.

Using a public key to sign the entire message is very inefficient if the message is very long.

The solution is to let the sender sign a digest of the document instead of the whole document. The sender creates a miniature version or digest of the document and signs it; the receiver then checks the signature on the miniature.

To create a digest of the message, we use a hash function. The hash function creates a fixed-size digest from a variable-length message, as shown in Figure



Hash function:

- The signature schemes allow only "small" messages to be signed.
- For example, when using the **DSS, a 160-bit message is signed with a 320-bit** signature.
- In general, we will want to sign much longer messages. A legal document, for example, might be many megabytes in size.
- To solve this problem would be to break a long message into 160-bit chunks, and then to sign each chunk independently.
- This is analogous to encrypting a long string of plaintext by encrypting each plaintext character independently using the same key. there are several problems with this approach in creating digital signatures.
- First of all, for a long message, we will end up with an enormous signature.
- Another disadvantage is that most "secure" signature schemes are slow since they typically use complicated arithmetic operations such as modular exponentiation.
- even more serious problem with this approach is that the various chunks of a signed message could be rearranged, or some of them removed, and the resulting message would still be verified.
- We need to protect the integrity of the entire message, and this cannot be accomplished by independently signing.
- The solution to all of these problems is to use a very fast public *cryptographic hash function, which will* take a message of arbitrary length and produce a *message digest of a specified size (160 bits if the* ***DSS*** is to be used). The message digest will then be signed.
- When Bob wants to sign a message x, he first constructs the message digest $\boldsymbol{z = h(x)}$, and then computes the signature $\boldsymbol{y = sig_K(z)}$. He transmits the ordered pair (x, y) over the channel.
- Now the verification can be performed (by anyone) by first reconstructing the message **digest** $\boldsymbol{z = h(x)}$ using the public hash function h, and then checking that $ver_K(z, y) = \text{true}$.

Collision – free hash function:

- When two records generate same hash key, it is called collision because there is already a record with hash key. When collision occurs move forward until you find empty slots.
- A function F that maps an arbitrary length message M to a fixed length message digest MD is a collision – free hash function.

- It is a one – way hash function. A hash function F is said to be one – way if it is hard to invert, where "hard to invert" means that given a hash value h, it is computationally infeasible to find some input x such that F(x) = h.

The Birthday Attack:

- we determine a necessary security condition for hash functions that depends only on the cardinality of the set *Z* (equivalently, on the size of the message digest).
- This necessary condition results from a simple method of finding collisions which is informally known as the *birthday attack*.
- This terminology arises from the so-called *birthday paradox*, which says that in a group of 23 random people, at least two will share a birthday with probability at least 1/2.
- Birthday attacks are used in an attempt to solve a class of cryptographic hash function problems.
- By repeatedly evaluating the function for different inputs the same output is expected to be obtained.
- Birthday attack is used to find collisions of hash functions.
- $1.2\sqrt{k}$ where k is the number of day in year.
- let us suppose that $h : X \rightarrow Z$ is a hash function, *X* and *Z* are finite, and $|X| \geq 2|Z|$. Denote $|X| = m$ and $|Z| = n$. It is not hard to see that there are at least *n* collisions — the question is how to find them.
- A very naive approach is to choose *k* random distinct elements $x_1,...,x_k \in X$, **compute $z_i = h(xi)$, $1 \leq i \leq k$**, and then determine if a collision has taken place.
- This process is analogous to throwing *k* balls randomly into *n* bins and then checking to see if some bin contains at least two balls. (The *k* balls correspond to the *k* random *xi*'s, and the *n* bins correspond to the *n* possible elements of *Z*.)

Key Predistribution :

- The idea of key predistribution for every pair of users {U, V}, the TA chooses a random key $K_{U,V} = K_{V,U}$ and transmits it "off-band" to U and V over a secure channel. (That is, the transmission of keys does not take place over the network, since the network is not secure).
- This approach is unconditionally secure, but it requires a secure channel between the TA and every user in the network. But, of possibly even more significance is the fact that each user must store *n* - 1 keys, and the TA needs to transmit a total of n^2 keys securely (this is sometimes called the "n^2 problem"). Even for relatively small networks.
- In on-line key distribution by TA, the TA acts as a *key server*. The TA shares a secret key *K*U with every user U in the network. When U wishes to communicate with V, she requests a *session key* from the TA.
- The TA generates a session key *K* and sends it in encrypted form for U and V to decrypt. The well-known **Kerberos** system.

- In the basic method, the TA generates keys, and gives each key to a unique pair of users in a network of *n* users. we require a secure channel between the TA and each user to transmit these keys.
- This is a significant improvement over each pair of users independently exchanging keys over a secure channel, since the number of secure channels required has been reduced from to *n*.

BLOM'S Scheme:

- Blom's scheme is a symmetric key exchange protocol in cryptography.
- The scheme was proposed by the Swedish cryptographer Rolf Blom in early **1980s**
- A trusted party gives each participant a secret key and public identifier, which enables any two participants to independently create a shared key for communicating.
- The key exchange protocol involves a trusted party(Trent) and a group of users. Let Alice and Bob be two users of the group.
- Blom's scheme used public matrix G and private matrix D(Symmetric).

Protocol Setup :

- Trent Choose a random and secret symmetric matrix $D_{K,K}$ over the finite field GF(p) where p is a prime number.
- D is required when a new user is to be added to the key sharing group.

$$K = 3$$

$$P = 17$$

$$D = \begin{bmatrix} 1 & 6 & 2 \\ 6 & 3 & 8 \\ 2 & 8 & 2 \end{bmatrix} \quad \text{MOD } 17$$

- Now new user Alice and Bob want to join the key exchanging group.
- $I_{alice}, I_{bob} \in GF(p)$

$$I_{Alice} = \begin{bmatrix} 3 \\ 10 \\ 11 \end{bmatrix}$$

$$I_{Bob} = \begin{bmatrix} 1 \\ 3 \\ 15 \end{bmatrix}$$

- Then compute their private keys
- $g_{alice} = D\ I_{alice}$
- $g_{bob} = D\ I_{bob}$

$$g_{alice} = \begin{bmatrix} 1 & 6 & 2 \\ 6 & 3 & 8 \\ 2 & 8 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 10 \\ 11 \end{bmatrix} \quad \text{MOD } 17 = \begin{bmatrix} 85 \\ 136 \\ 108 \end{bmatrix} \quad \text{mod } 17 = \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}$$

$$g_{bob} = \begin{pmatrix} 1 & 6 & 2 \\ 6 & 3 & 8 \\ 2 & 8 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 15 \end{pmatrix} \text{ MOD } 17 = \begin{pmatrix} 49 \\ 135 \\ 56 \end{pmatrix} \text{ mod } 17 = \begin{pmatrix} 15 \\ 16 \\ 5 \end{pmatrix}$$

- Each will use their private key to compute shared keys with other participants of the group.
- Now Alice and Bob wish to communicate with one another.
- Alice has Bob's identifier and her private key.
- She computes the shared key, where denotes matrix transpose. Bob does the same, using his private key and her identifier, giving the same result.

$$K_{Alice/Bob} = \begin{pmatrix} 0 \\ 0 \\ 6 \end{pmatrix}^t \begin{pmatrix} 1 \\ 3 \\ 15 \end{pmatrix} = 0 * 1 + 0 * 3 + 6 * 15 = 90 \bmod 17 = 5$$

$$K_{Bob/Alice} = \begin{pmatrix} 15 \\ 16 \\ 5 \end{pmatrix}^t \begin{pmatrix} 3 \\ 10 \\ 11 \end{pmatrix} = 15 * 3 + 16 * 10 + 5 * 11 = 90 \bmod 17 = 5$$

What is Kerberos?

- It is a Network authentication protocol.
- It provides for strong authentication for client-server applications.
- Kerberos is an authentication service developed as part of Project Athena at MIT in the mid 1980s.
- It is available as open source or in supported commercial software.
- It uses secret – key cryptography to provide strong authentication.
- It provide strong security on physically insecure network.
- A centralized authentication server which authenticates users to servers and servers to users.

Motivation/ Requirement of Kerberos:

Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.

- Security – A network eavesdropper should not be able to obtain the necessary information to impersonate a user i.e. secure against eavesdropping. Kerberos should be strong enough that a potential opponent does not find it to be the weak link.

- Reliable – For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.
- Transparent – Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password i.e. almost invisible to user.
- Scalable – The system should be capable of supporting large numbers of users and servers.

To support these requirements, the overall scheme of Kerberos is that of a trusted third-party authentication service that uses a protocol.

It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. Assuming the Kerberos protocol is well designed, then the authentication service is secure if the Kerberos server itself is secure.

Need of Kerberos:

an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. The servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services. In particular, the following three threats exist:

- A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access.

Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

Design Requirement:

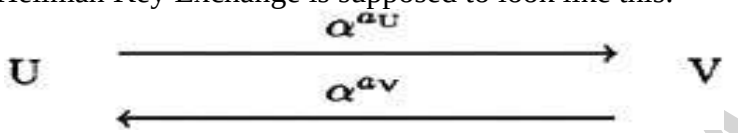
In a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers. In this environment, three approaches to security can be provided:

1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).

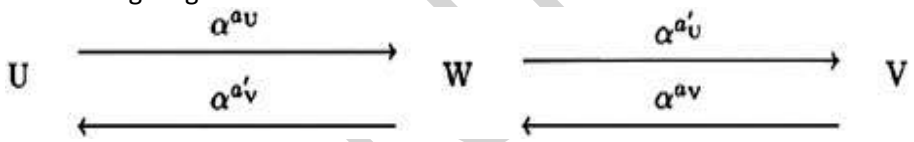
2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

Station to station protocol:

- Diffie-Hellman Key Exchange is supposed to look like this:



- Unfortunately, the protocol is vulnerable to an active adversary who uses an *intruder-in-the-middle* attack.
- There is an episode of *The Lucy Show in which Vivian Vance is having dinner in a restaurant* with a date, and Lucille Ball is hiding under the table. Vivian and her date decide to hold hands under the table. Lucy, trying to avoid detection, holds hands with each of them and they think they are holding hands with each other.
- An intruder-in-the-middle attack on the Diffie-Hellman Key Exchange protocol works in the same way. W will intercept messages between U and V and substitute his own messages, as indicated in the following diagram:



- At the end of the protocol, U has actually established the secret key $a^{aua'v}$ with W, and V has established a secret key with W $a^{a'uav}$ When U tries to encrypt a message to send to V, W will be able to decrypt it but V will not. (A similar situation holds if V sends a message to U.)

- Clearly, it is essential for U and V to make sure that they are exchanging messages with each other and not with W.
- Before exchanging keys, U and V might carry out a separate protocol to establish each other's identity.
- But this offers no protection against an intruder-in-the-middle attack if W simply remains inactive until after U and V have proved their identities to each other.
- Hence, the key agreement protocol should itself authenticate the participants' identities at the same time as the key is being established.
- Such a protocol will be called *authenticated key agreement.*
- an authenticated key agreement protocol which is a modification of **Diffie-Hellman Key Exchange.**
- The protocol assumes a publicly known prime *p and a primitive element a, and it* makes use of certificates.

- Each user U will have a signature scheme with verification algorithm Ver_U and signing algorithm Sig_U
- The TA also has a signature scheme with public verification algorithm Ver_{TA}
- Each user U has a certificate

$$\mathbf{C}(\mathrm{U}) = (\mathrm{ID}(\mathrm{U}), ver_{\mathrm{U}}, sig_{\mathrm{TA}}(\mathrm{ID}(\mathrm{U}), ver_{\mathrm{U}})),$$

- where ID(U) is identification information for U.
- The authenticated key agreement known as the **Station-to-station Protocol**

UNIT 5

What is firewall-

- A system designed to prevent unauthorized access to or from private networks.
- Can be implemented in both hardware and software, or a combination of both.
- Frequently used to prevent unauthorized internet users from accessing private networks connected to the internet, especially intranets.
- All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.
- A firewall is essentially a barrier between two networks that evaluates all incoming or outgoing traffic to determine whether or not it should be permitted to pass to the other network.

Firewall Characteristics :

A firewall is defined as collection of components placed between two networks that collectively have following characteristics:

1. All traffic from inside to outside and vice versa, must be pass through the firewall.
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass.
3. The firewall itself to immune to penetration. i.e. the use of trusted system with a secure operating system.

Limitations of firewall:

1. A firewall does not protect against internal threats such as terminated or ex-employee or an employee who co-operates with an external attacker.
2. The firewall cannot protect against the transfer of virus-infected programs or files.

Types of firewall-

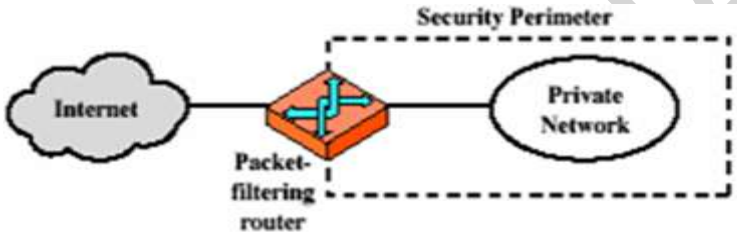
- Packet Filters.
- Application Level Filtering.
- Circuit Level Gateways.

- **Packet filter firewall –**

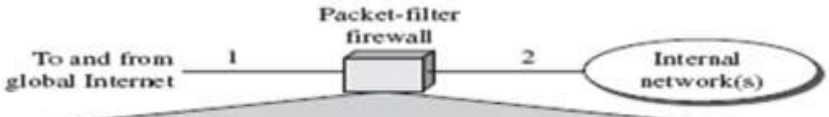
Packet filtering is a feature implemented on routers and firewalls that uses rules specified by administrator to determine whether a packet should be permitted to pass through the firewall. The router is typically configured to filter packets going in both direction.

Filtering rules are based on information contained in network packet:

1. Source IP address – The IP address of the system that originated the IP packet.
2. Destination IP address – The IP address of the system the IP packet is trying to reach.
3. Source and Destination transport-level address(port number) – the transport level port number, which defines application such as SNMP or TELNET.
4. IP Protocol field – defines the transport protocol.
5. Interface - for a router with three or more parts, which interface of the router the packet came from or which interface of the router the packet is destined as shown in fig.



(a) Packet-filtering router



Interface	Source IP	Source port	Destination IP	Destination port
1	131.34.0.0	*	*	*
1	*	*	*	23
1	*	*	194.78.20.8	*
2	*	80	*	*

Advantages of packet filter firewall –

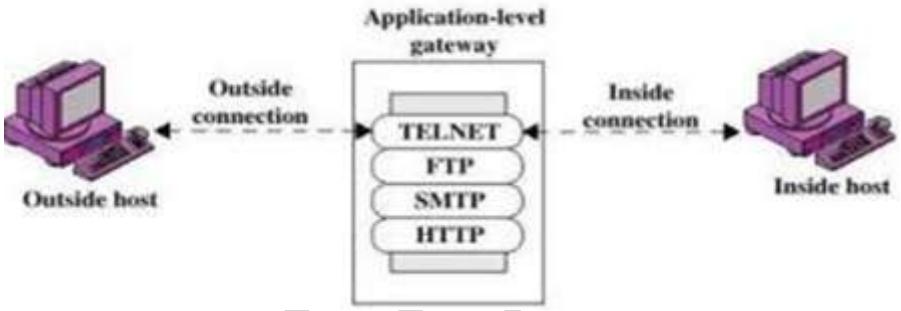
1. Packet – filtering router is simple.
2. Low impact on network performance.
3. Packet filters are normally transparent to user.
4. Packet filters are very fast.
5. Relatively inexpensive in price.

Disadvantages of packet filter firewall –

1. Packet filter firewalls do not examine upper- layer data, they cannot prevent attacks that employ application-specific vulnerabilities or function.
2. They cannot hide the network topology.
3. Most packet filter firewalls do not support advanced user authentication schemes.
4. Packet filtering firewall cannot support all internet applications.
5. Packet filtering firewall cannot have very limited auditing capabilities.

- **Application Level Gateway –**

An application level firewall also called a proxy server. It acts as a relay of application level traffic as shown in fig.



1. The user contacts the gateway using a TCP/IP application such as Telnet or FTP and an the gateway asks the user for the name of the remote host to be accessed.
2. When user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segment containing the application data between the two end points.
3. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.

Advantages of Application level firewall –

1. Application level gateways is more secure than packet filters.
2. Easy to configure.
3. They can hide private network topology.
4. It support user level authentication.
5. Capability to examine the all traffic in detail.

Disadvantages of Application level firewall –

1. High impact on nerwork performance.
2. Slower in operation.

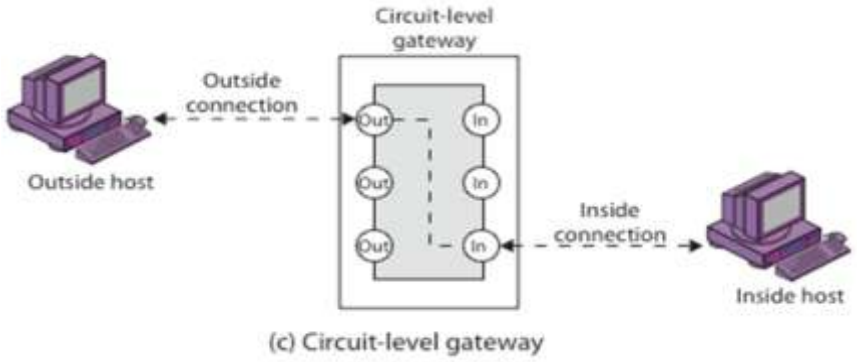
3. Additional processing overhead on each connection.
4. Not transparent to users.
5. Needs an individual server for every application and required modification to client program.

Comparison between Packet filter and Application level firewall –

Sr.No.	Packet Filter	Proxy (Application Level Firewall)
1	Works at network layer of OSI and IP layer of TCP.	Works at application layer of OSI, and TCP layer of TCP.
2	Low impact on network performance.	High impact on network performance.
3	Low level of security as compare to proxy.	High level of security.
4	Packet filtering is not effective with the FTP protocol.	FTP and Telnet are allowed into the protected subnet.
5	Simple level of security and faster than proxy firewall.	Capability to examine the traffic in detail, so slower than packet filtering.
6	Normally transparent to the user.	Not transparent to the user.
7	Difficult to configure as compare to proxy.	Easier to configure than packet filtering.
8	They cannot hide the private network topology.	They can hide the private network topology.

Circuit-Level Gateway –

1. It works at the session layer of the OSI model or the TCP layer of TCP/IP. This can be a stand-alone system or it can be specialized function performed by an Application level Gateway for certain applications.
2. A circuit level gateway does not permit an end-to-end TCP connection; rather, the gateway set up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside itself.
3. Once the two connections are established, the gateway typically relays TCP segment from one connection to the other without examining the contents.
4. The security function consists of determining which connections will be allowed.



(c) Circuit-level gateway

Advantages of circuit level gateway:

1. It is faster than proxy because they perform fewer applications.
2. It protect an entire network by prohibiting connection between specific internet source and internal computers.
3. It protect internal IP address from external users.

Disadvantages of circuit level gateway:

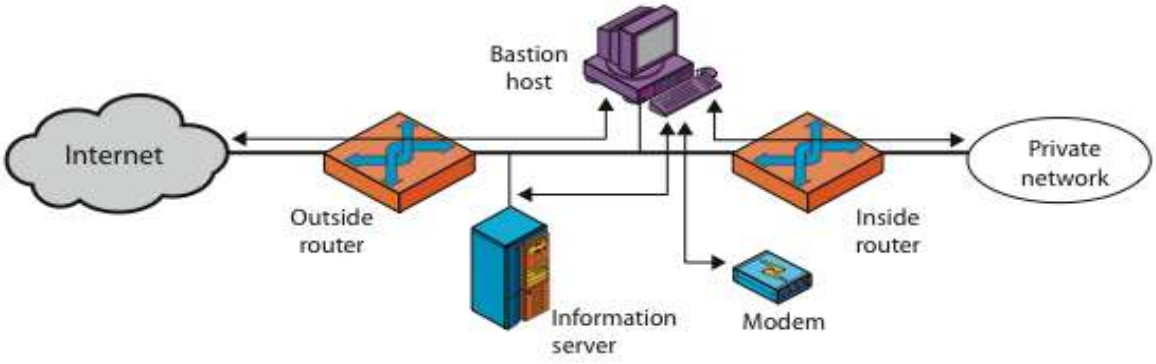
1. Cannot protect higher level protocol.
2. Difficult to test "allow" and "deny" rules.

Bastion Host:

A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Typically, the bastion host serves as platform for an application-level or circuit-level gateway.

The following are common characteristics of a bastion host:

1. The bastion host hardware platform executes a secure version of its operating system, making it a trusted system.
2. Only the services that the network administrator considers essential are installed on the bastion host. These include proxy applications such as Telnet, DNS, FTP, SMTP and user authentication.
3. The bastion host may require additional authentication before a user is allowed access to the proxy services. In addition, each proxy service may require its own authentication before granting user access.
4. Each proxy is configured to support only a subset of the standard application's command set.
5. Each proxy is configured to allow access only to specific host systems. This means that the limited command/feature set may be applied only to a subset of systems on the protected network.
6. A proxy generally performs no disk access other than to read its initial configuration file. This makes it difficult for an intruder to install trojan horse sniffers or other dangerous files on the bastion host.



(c) Screened-subnet firewall system

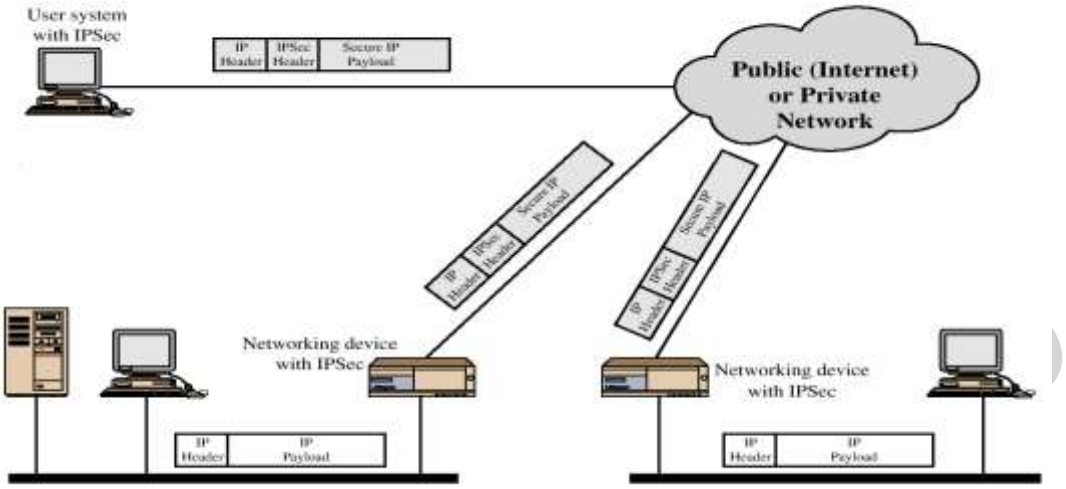
- **IP addresses is not secure. WHY?**
- The most serious types of attacks included IP spoofing, in which intruders create packets with false IP addresses and crash applications that use authentication based on IP.
- **IPSec** is a set of services and protocols that provide a complete security solution for an IP networks.

IP Security(IPSec) Protocols:

- The technology that brings secure communications to the internet protocol is called IP Security, commonly abbreviated IPSec.
- It is a collection of protocols designed by the Internet Engineering Task Force(IETF) to provide an end-to-end security scheme operating in the internet layer.
- It can be used in protecting data flows between a pair of host(host-to-host), between a pair of security gateways(network-to-network), or between a security and a host(network-to-host).
- IPSec provides security in three situations:
  - Host-to-host, host-to-gateway and gateway-to-gateway.
- IPSec operates in two modes:
  -*Transport mode* (for end-to-end).

  -*Tunnel mode* (for VPN).
- IPSec provides two protocols
  - Authentication Header(AH) protocol.

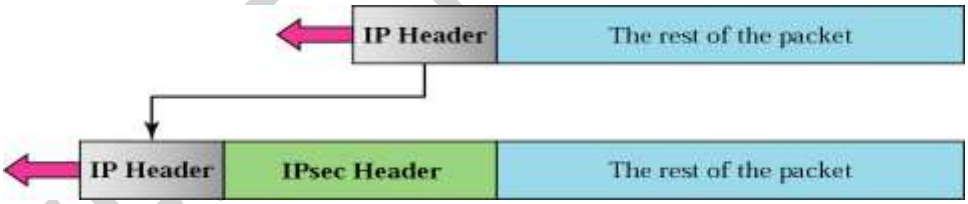
  - Encapsulating Security Payload(ESP) protocol.



Modes of operation:

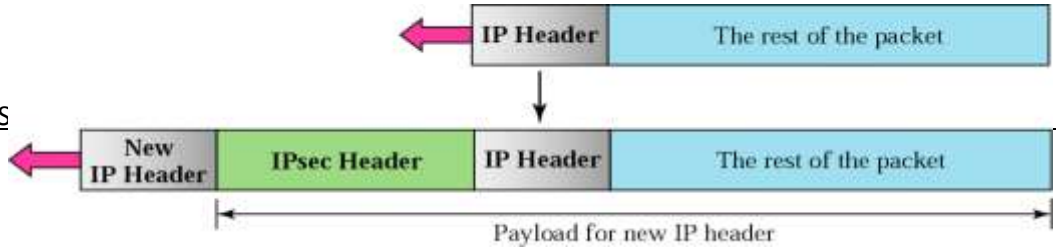
Transport Mode:

- IPSec can be implemented in a host-to-host transport mode, and in a network tunnel mode.
- The transport mode is used when there is a need of host-to-host protection of data.
- The transport mode adds the IPSec header and trailer to the information coming from the transport layer.
- IPSec header is added between the IP header and the rest of the packet.



Tunnel Mode:

- In this mode, IPSec protects the entire IP packet.
- It takes an IP packet, including the header, applies IPSec security methods to the entire packet, and then adds a new IPheader.
- The newly created IP header, has different information than the original IP header.
- The tunnel mode is normally used between two routers, like between a host and a router or between a router and a host.



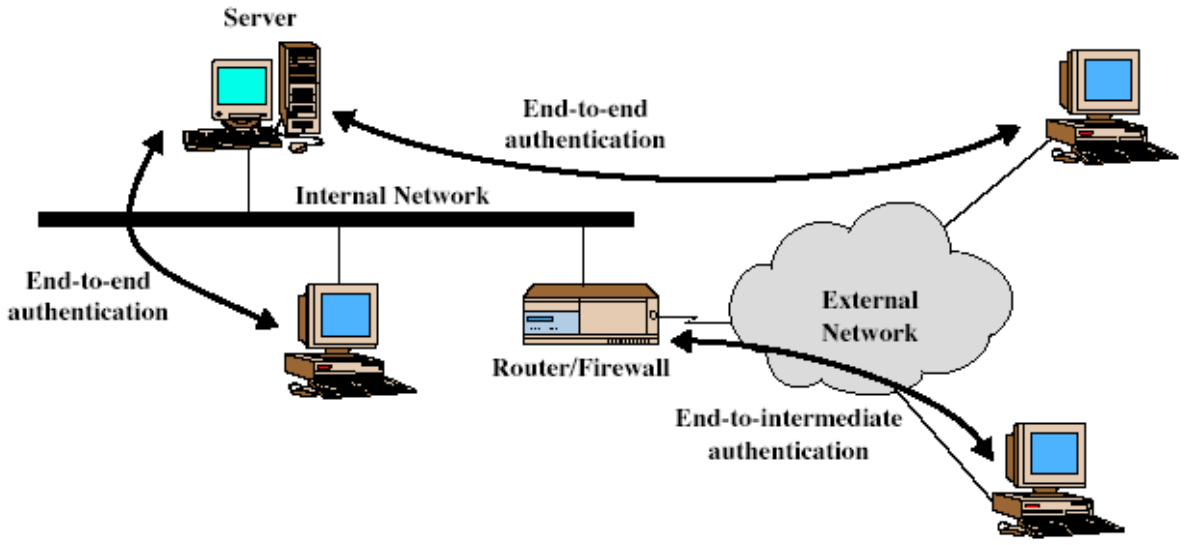


Figure – Transport and Tunnel Mode on Network

IPSec Architecture:

The IPSec architecture comprises of different protocols like

1.Authentication Header(AH) protocol.

2.Encapsulating Security Payload(ESP).

3.Internet Key Exchange(IKE).

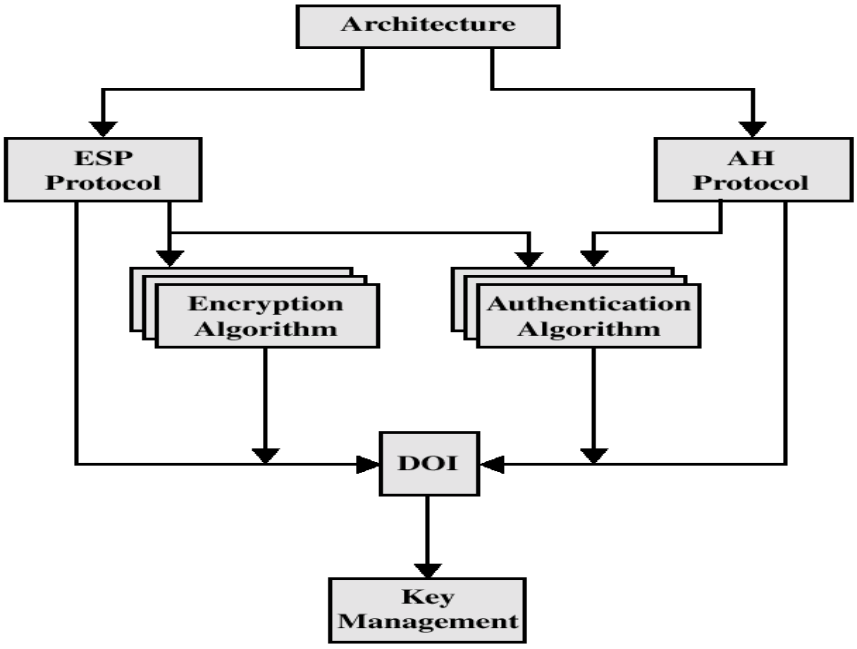


Figure – IPSec Document Overview

A brief description each field:

- Architecture : covers the general concepts, security requirements, definitions and mechanisms defining IPSec technology.
- Encapsulating Security Payload(ESP): covers the packet format and general issues related to the use of the ESP for packet encryption and optionally, authentication.
- Authentication Header(AH): covers the packet format and general issues related to the use of AH for packet authentication.
- Encryption Algorithm: a set of documents that describe how various encryption algorithm are used for ESP.
- Authentication Algorithm: a set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP.
- Key management: documents that describes key management schemes.
- Domain of Interpretation(DOI): contains values needed for the other documents relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as keys.

Authentication Header (AH) Protocol RFC 2402:

- The Authentication Header protocol provides authentication to the source host so as to ensure the integrity(the contents of the message should remain same when the receiver receives it) of the payload carried in the IP packet.

- This protocol uses a hash function and a symmetric key to create a message digest; the digest is inserted in the authentication header.
- The AH is then placed in the appropriate location based on the transport or tunnel mode.

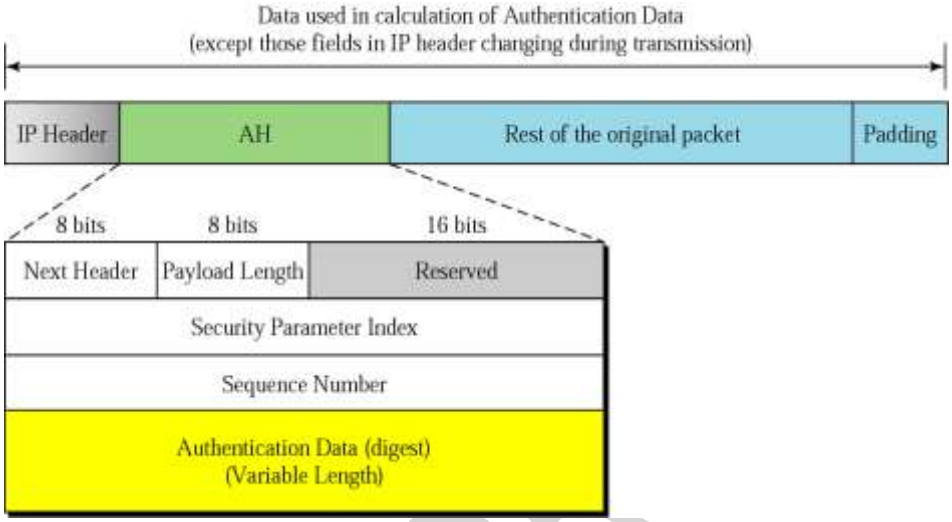
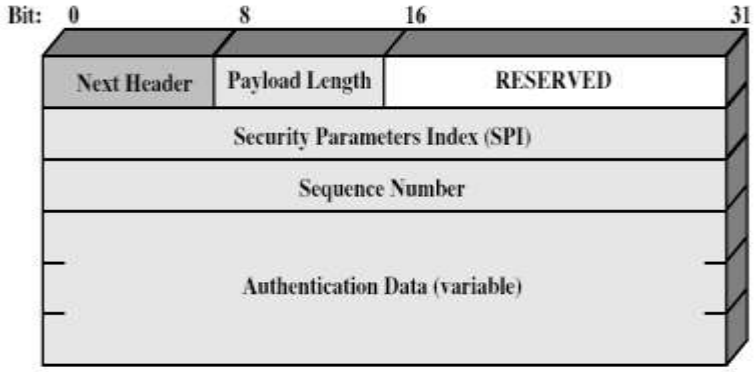


Figure - AH PROTOCOL in transport mode

Steps for Authentication Header:

- AH is added to the payload with the authentication data field set to zero.
- Padding may be added to make the total length even for a particular hashing algorithm
- Hashing is based on total packet. For message digest, only those fields of IP header that don't change during transmission are considered.
- Authentication data are included in the authentication header
- IP header is added after changing the value of protocol field to 51.

Authentication Header Format:



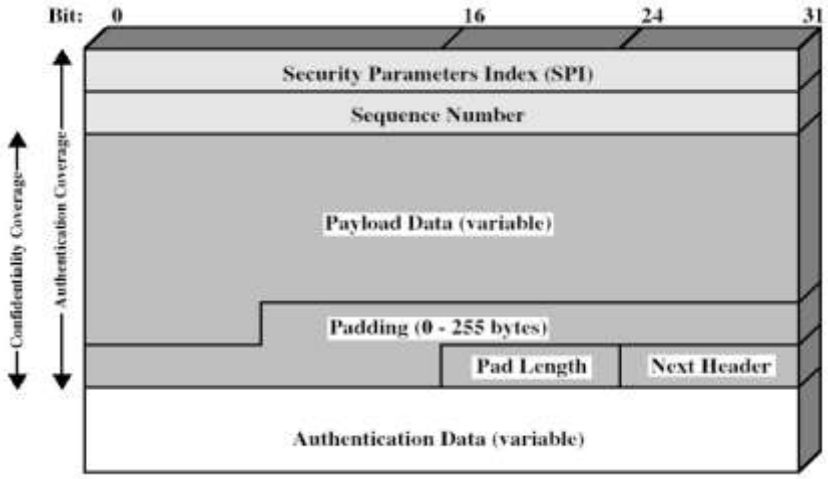
A brief description each field:

- Next header : The 8-bit next-header field defines the type of payload carried by the IP datagram(such as TCP, UDP, ICMP).
- Payload length : this is 8 bit field is misleading. It defines the length of the authentication header.
- Security parameter index : this is 32 bit security parameter index(SPI) field plays the role of a virtual-circuit identifier and is the same for all packets sent during a connection called a security association.
- Sequence Number : a 32-bit sequence number provides ordering information for a sequence of datagram's. The sequence number is not repeated even if a packet is retransmitted.
- Authentication data – finally, the authentication data field is the result of applying a hash function to the entire IP datagram except for the fields that are changed during transmit.

ESP protocol:

- Due to the limitations of the authentication header IPSec defined an alternative protocol that provides source authentication and integrity and privacy called Encapsulating Security Payload(ESP).

ESP protocol header:



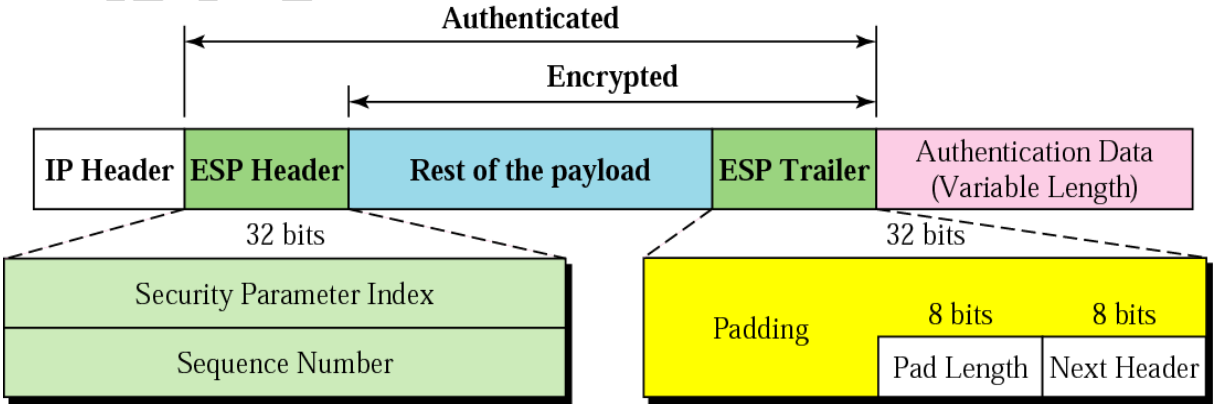
ESP fields:

- Security Parameter Index (SPI)
- Sequence number
- Padding & Pad length – it is reserved bits for next header.
- Next header
- Authentication Data

Steps for ESP:

- ESP trailer is added to the payload
- Payload and trailer or encrypted
- ESP header is added
- ESP header, payload and ESP trailer are used to create authenticated data.
- Authenticated data are added at the end of ESP trailer.
- IP header is added after changing the protocol value to 50.

ESP in transport mode:

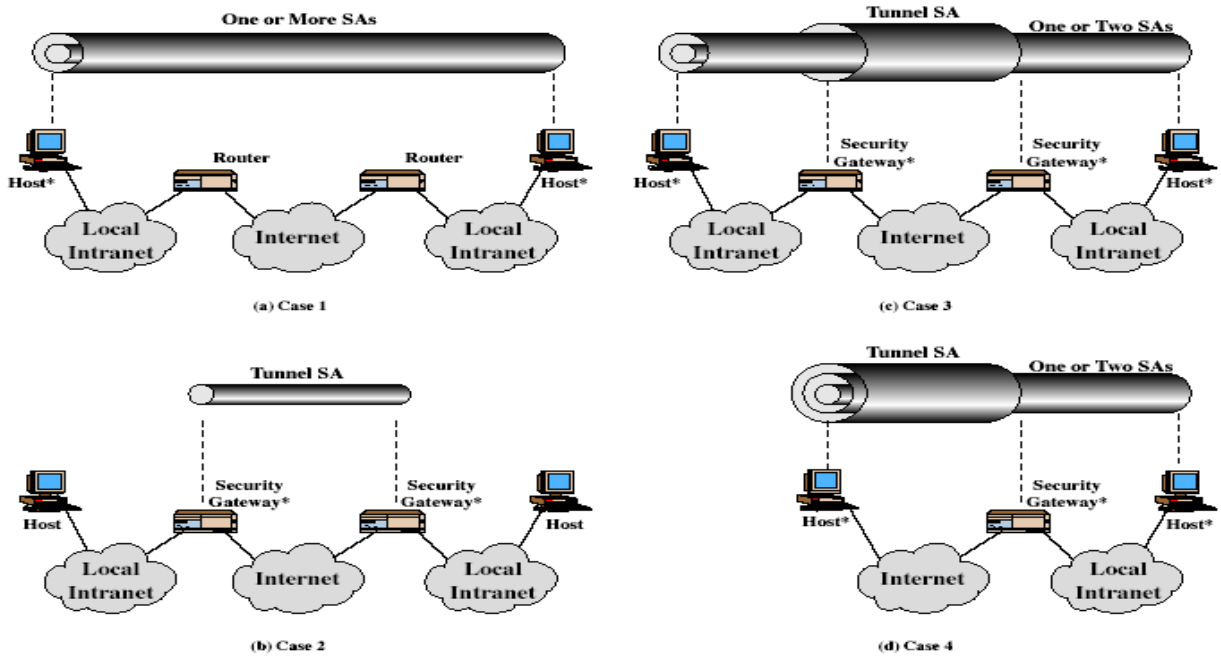


Combining Security Associations:

- SAs can be implemented by using AH or ESP. To implement both we need to combine SAs to form a security association.
- Security Association(SA) is a collection of parameters that specify how data is to be protected when communicating with a peer.

Security Association Components:

- Destination IP address.
- Security Parameter Index(SPI).
- Protocol – ESP or AH. AH is no longer supported.
- Encryption Algorithm – defines how the data is protected – for example, DES algorithm.
- Authentication Algorithm – keyed hash function(digital signatures, hashed message authentication code).
- Mode – the mode IPSec is working as: tunnel or transport.
- Lifetime - the number of seconds or kilobytes that a key should be used; when this lifetime is exceeded, a new key is created.



a. AH in transport mode.

b.ESP in transport mode .

c. AH followed by ESP in transport mode(ESP SA inside an AH SA).

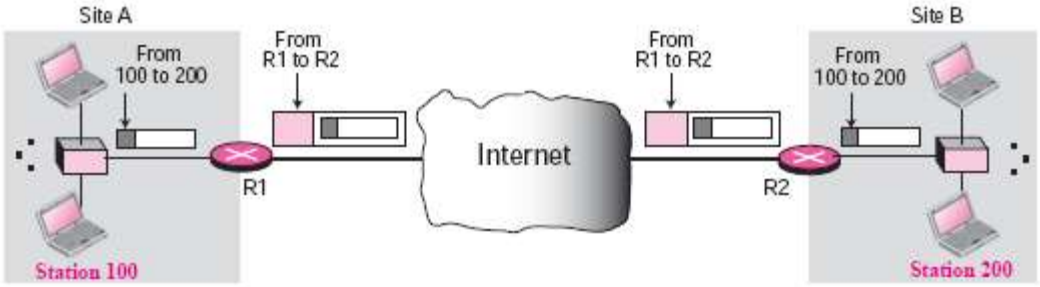
d. any one a, b, c inside an AH or ESP in tunnel mode.

virtual private network

A virtual private network (VPN) extends a private network across a public network, such as the Internet. It enables a computer to send and receive data across shared or public networks as if it is directly connected to the private network, while benefiting from the functionality, security and management policies of the private network. A VPN is created by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryptions.

VPNs have several characteristics:

- Traffic is encrypted so as to prevent eavesdropping.
- The remote site is authenticated.
- Multiple protocols are supported over the VPN.
- The connection is point to point.



Intrusion

Intruders

- Some one who intrudes on the privacy or property of another without permission.
- Intruders may be from outside the network or legitimate users of the network.

Intrusion

- Entrance by force or without permission or welcome.
- Intrusion : Attempting to break into or misuse your system.
- Intrusions are activities that violate the security policy of system.

Intrusion detection

- Intrusion detection is the art and science of sensing when a system or network is being used inappropriately or without authorization.
- Intrusion detection is the process used to identify intrusions.
- An intrusion detection system(IDS) monitors system and network resources and activities and, using information gathered from these sources, notifies the authorities when it identifies a possible intrusions.

Classification an IDS:

1. Anomaly Detection IDS
2. Signature Based misuse IDS
3. Host based IDS
4. Stack based IDS
5. Network based IDS

1. **Anomaly Detection IDS** – This IDS models use on the network as a noise characterization. Anything distinct from the noise is assumed to be an intrusion activity. For example, flooding a host with lots of packet.
   Drawbacks :
   a. Assumes that intrusions will be accompanied by manifestations that are sufficiently unusual so as to permit detection.
   b. These generate many false alarms and hence compromise the effectiveness of the IDS.

2. **Signature based IDS** – These IDS possess an attacked description that can be matched to sensed attack manifestations. IDS system is programmed to interpret a certain series of packets, or certain piece of data contained in those packets, as an attack.
   Drawbacks :
   a. They are unable to detect novel attacks.
   b. Suffer from false alarms.
   c. Have to program again for every new pattern to be detected.

3. **Host/Application based IDS** : These host operating system or the application logs in the audit information. These audit information includes events like the use of identification and authentication mechanisms(logins etc.), file opens and programs executions, admin activities etc. these audit is then analyzed to detect trails of intrusion.

4. **Stack based IDS –** They are integrated closely with the TCP/IP stack, allowing packets to be watched as they traverse their way up the OSI layers. This allows the IDS to pull the packets from the stack before the OS or the application has a chance to process the packets.

5. **Network based IDS** : This IDS looks for attack signatures in network traffic. A filter is usually applied to determine which traffic will be discarded or passed on to an attack recognition module. This helps to filter out known un-malicious traffic.

Internet Security Protocols:

Web security:

- Most of businesses, government agencies and many individuals now have web sites.
- The number of individuals and companies with internet access is expanding rapidly, and all of these have graphical web browsers.
- The general requirements for web security and then focus on two standardized schemes that are becoming increasingly important as part of web commerce :
- SSL/TLS AND SET.

Web Security Consideration:

- ✓ The world wide web is fundamentally a client/server application running over the internet and TCP/IP intranets.
- ✓ Security on the internet: initially, the internet was designed to be open and accessible by anonymous users.
- ✓ Cookies: Cookies are a small file stored on the client computer.
- ✓ Logins and passwords: some websites ask visitors to create an account with a profile. These account settings are typically controlled by what is called a 'session id' and the account information is stored in a database.
- ✓ Hacking: the term hacker was used to refer to a person who worked or used a computer on a daily basis. In contrast, a 'cracker' is the person who commits the illegal, immoral, or unethical actions mentioned.

A short list of the most general 'cracker':

1. XSS – cross site scripting, or the insertion of tags or scripted code into another site's web page.
2. Key logging – a practice that typically uses software that is planted on an unsuspecting person's computer. The software has ability to log the keystrokes used.
3. Phishing: mostly in email system, to get a person to enter their account numbers, passwords etc into a website or form. Their account identity is compromised(identity theft) for illegal activity.

- ✓ Weak passwords: computer programs that can generate passwords are often used against account logins.

The general purpose solution is to implement security above TCP. The foremost example of this approach is the Secure Sockets Layer(SSL) and follow-on Internet standard of SSL known as Transport Layer Security(TLS).

SSL

SSL Architecture:

- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- SSL is not a single protocol but rather two layers of protocols.
- The SSL Record Protocol provides basic security services to various higher-layer protocols.
- The HTTP which provides the transfer service for Web client/server interaction, can operate on top of SSL.

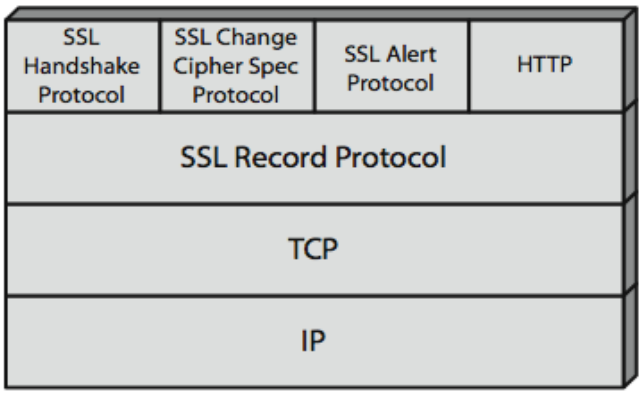


Figure – SSL Architecture/ SSL Header Format

- SSL consists of three higher-layer protocols – the Handshake Protocol, the Change CipherSpec Protocol, and Alert Protocol.
- Two important SSL concepts are the SSL session and the SSL connection.

1. Connection

2. Session

1. Connection – a connection is a transport that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connection are transient. Every connection associated with one session.
2. Session – it manages an association between a client and a server. sessions are created by using the handshake protocol. session define a set of cryptographic security parameters that can be shared among multiple connections. sessions are useful in avoiding the expensive negotiation of new security parameters for each connection.

SSL Record Protocol:

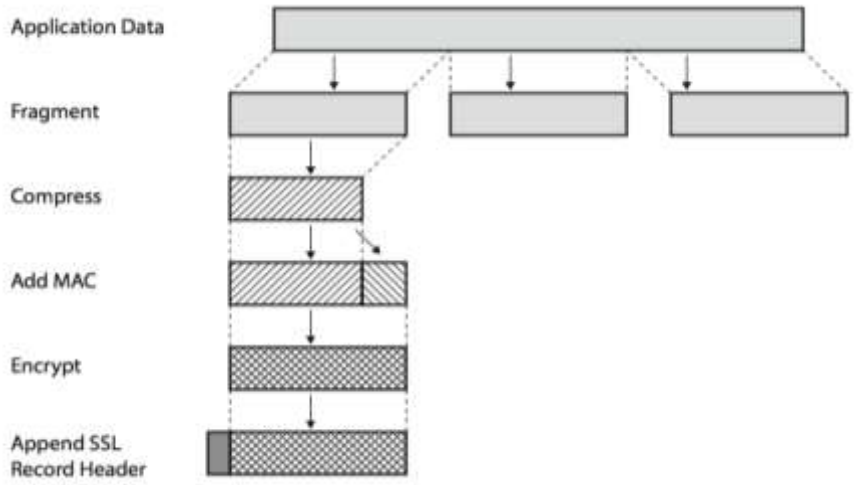
The SSL Record Protocol provides two services for SSL connection:

1. Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.

2. Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a MAC(message authentication code).

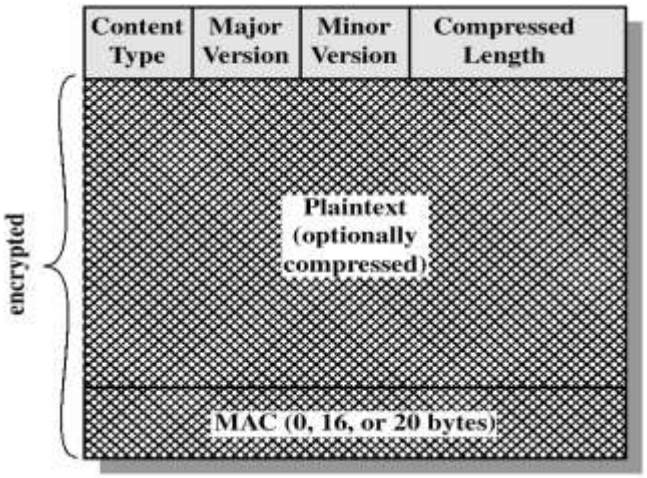
SSL Record Protocol Operation:

- Fragmentation: each upper layer message is fragmented into block of 2^{14} bytes(16384bytes) or less.
- Compression must be lossless and may not increase the content length by more than 1024 bytes.
- Message Authentication Code- it is compute a code over the compressed data. for this purpose a shared secret key is used.
- Next, the compressed message plus the MAC are encrypted using symmetric encryption.



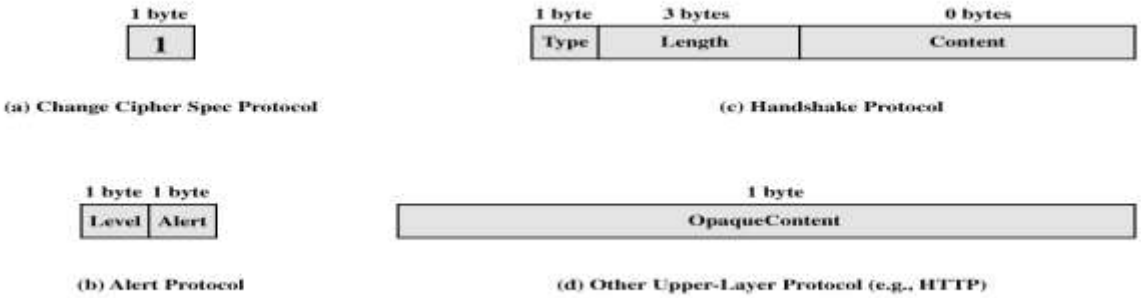
Change Cipher Spec protocol:

- This protocol consists of a single message, which consists of a single byte with the value 1.
- The sole purpose of this message is to cause the pending state to be copied to into the current state, which updates the cipher suite to be used on this connection.



Alert Protocol:

- The alert protocol is used to convey SSL-related alerts to the peer entity.
- Each message in this protocol consists of two bytes.
- The first byte takes the value warning(1) or fatal(2) to convey the severity of the message.

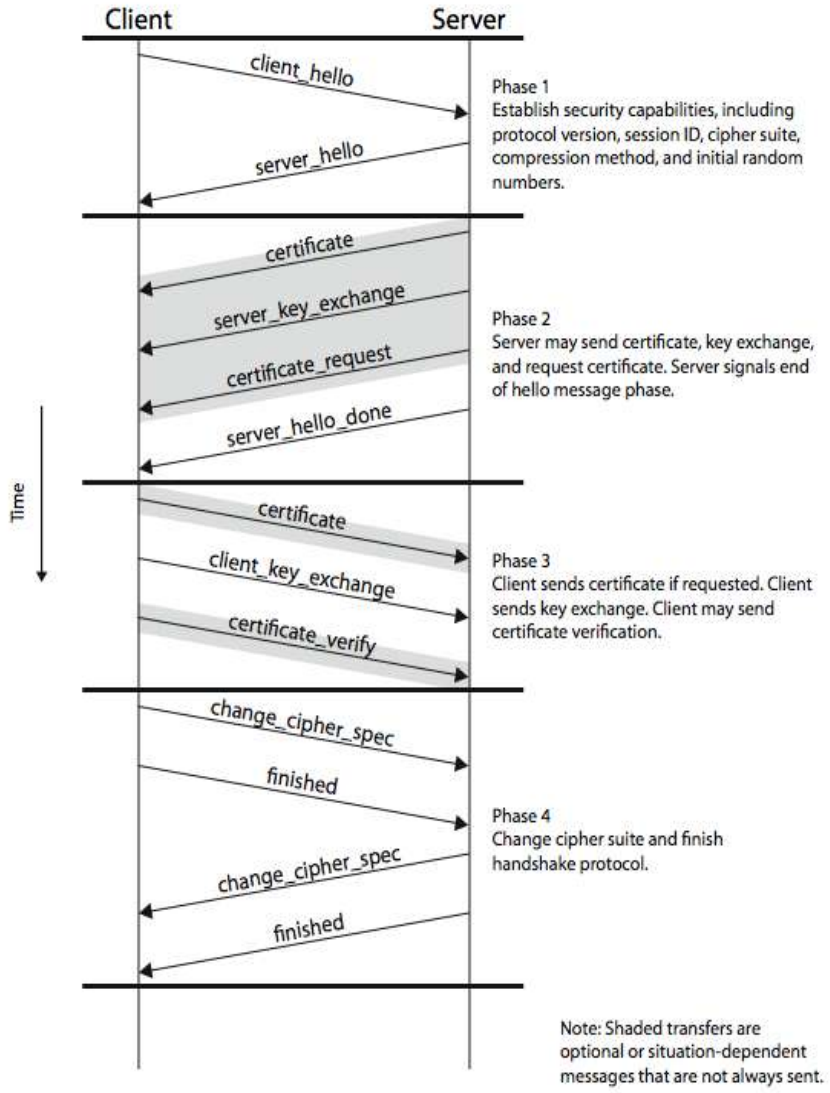


Lists of alert:

- Unexpected_message –an inappropriate message was received.
- Bad_record_mac – an incorrect MAC was received.
- Decompression-failure – the decompression function received improper input.
- Handshake_failure – sender was unable to negotiate an acceptable set of security parameters given the options available.

Handshake Protocol:

- This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record.
- The handshake protocol is used before any application data is transmitted.
- The handshake protocol consists of a series of messages exchanged by client and server.



TLS

Transport Layer Security(TLS):

- The TLS protocol allows client-server applications to communicate across a network in a way designed to prevent eavesdropping(secretly listening to the private conversation of others without their consent) and tampering (a device or process that makes unauthorized access to the protected object easily)
- Transport Layer Security(TLS) is a protocol that ensure privacy between communicating applications and their users on the internet.
- When a server and client communicate, TLS ensures that no third party may eavesdrop or tamper with any message.
- Two main ways of achieving TLS:
  1. Use a different port number for TLS connections.(for eg., port 443 for HTTP's)

  2. Use the regular port number and have the client request that the server switch the connection to TLS using a protocol.

SECURE HYPER TEXT TRANSFER PROTOCOL (SHTTP)

The Secure Hyper Text Transfer Protocol (SHTTP) is a set of security mechanisms defined for protecting the Internet traffic. This includes the data-entry forms and Internet-based transactions.

The services offered by SHTTP are quite similar to those of SSL. However, SSL has become highly successful—SHTTP has not. SHTTP works at the application layer, and is therefore, tightly coupled with HTTP, unlike SSL.

SHTTP supports both authentication and encryption of HTTP traffic between the client and the server.

The key difference between SSL and SHTTP is that SHTTP works at the level of individual messages. It can encrypt and sign individual messages. On the other hand, SSL does not differentiate between different messages. Instead, it aims at making the connection between a client and the server, regardless of the messages that they are exchanging. Also, SSL cannot perform digital signatures.

SET

Secure Electronic Transaction:

- SET is an open encryption and security specification designed to protect credit card transactions on the internet.
- It is developed by VISA and MasterCard for securing credit card transactions over insecure networks, specifically, the internet.
- SET was not itself a payment system, but rather a set of security protocols and formats that enable users to employ the existing credit card payment infrastructure on an open network in secure fashion.

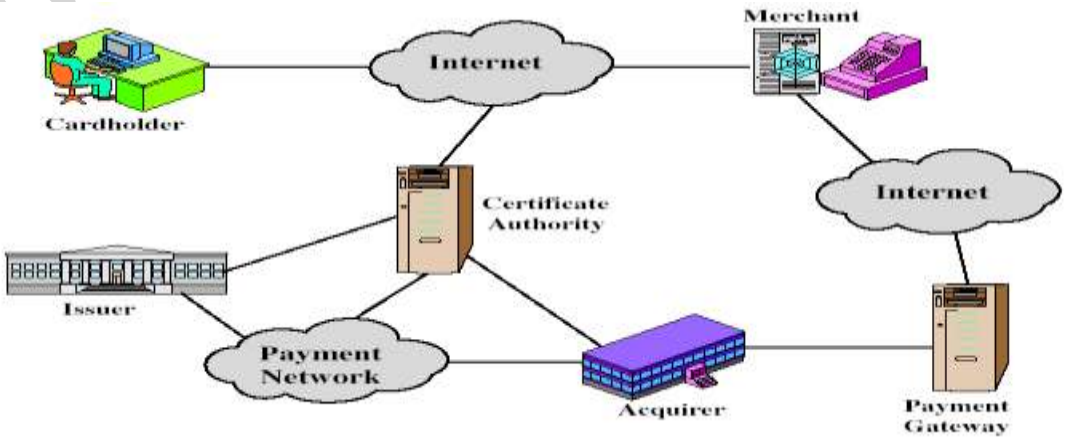
SET specification:

- Uses public key cryptography and digital certificates for validating both consumers and merchants.
- It provides the four security requirements – confidentiality, data integrity, user and merchant authentication, and consumer non-repudiation.

Architecture OR participants of SET:

The SET protocol coordinates the activities of:

1. Card Holder (Consumer) – he is the buyer who is the registered holder of the credit card.
2. Card Issuer(Consumer's Bank) – bank that issues the credit card to card holder.
3. Merchant – refers to the seller who is connected to an acquirer.
4. Acquirer (Merchant's Bank) – bank that serves as an agent to link a merchant to multiple issuers(customer's banks).
5. Payment Gateway – this is connected to acquirer. It is situated between the SET system and the financial network of the credit card system for processing the credit card payment.
6. Certification Authority (CA) – Issues digital signatures to concerned parties.



SET payment transactions:

- Step1 – consumer browses the merchant's website and initiates to purchase.
- Step2 - he sends payment and encrypted financial information along with his digital certificate to the merchant.
- Step3 – Merchant's website transfers the payment information to the acquirer while a CA certifies that digital certificate belongs to the card holder.
- Step4 - Acquirer checks with issuer for payment authorization.
- Step5 – issuer authorizes the payment.
- Step6 – Merchant receives this approval.
- Step7 – Merchant completes the order by capturing the transaction and thus, consumer's credit card is charged.
- Step8 – Issuer sends credit card bill to consumer(card holder).

At last, merchant ships the goods to consumer.

Dual Signature:

The purpose of the SET dual signature is to link two messages that are intended for two different recipients, the order information (OI) for the merchant and the payment information (PI) for the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order, however the two items must be linked in a way that can be used to resolve disputes if necessary. The customer takes the hash (using SHA-1) of the PI and the hash of the OI, concatenates them, and hashes the result. Finally,the customer encrypts the final hash with his or her private signature key, creating the dual signature. This can be summarized as: DS=E(PRc, [H(H(PI)||H(OI))])

SET Purchase Request:

The purchase request exchange consists of four messages: Initiate Request, Initiate Response, Purchase Request, and Purchase Response. In order to send SET messages to the merchant, the cardholder must have a copy of the certificates of the merchant and the payment gateway. The customer requests the certificates in the Initiate Request message, sent to the merchant. The merchant generates a response and signs it with its private signature key. The cardholder verifies the merchant and gateway certificates by means of their respective CA signatures and then creates the OI and PI. Next, the cardholder prepares the Purchase Request message with Purchase-related information & Order-related information. The Purchase Response message

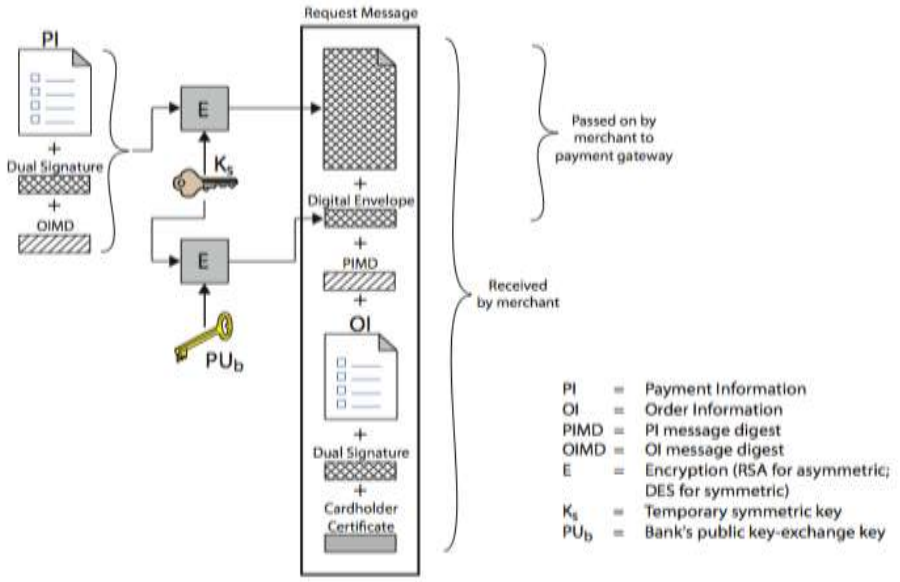
includes a response block that acknowledges the order and references the corresponding transaction number.

Purchase Request – Customer:

Figure. shows the details of the contents of the Purchase Request message generated b y the customer.

The message includes the following:

1. Purchase-related information, which will be forwarded to the payment gateway by the merchant and consists of: PI, dual signature, & OI message digest (OIMD).
2. Order-related information, needed by the merchant and consists of: OI, dual signature, PI message digest (PIMD).
3. Cardholder certificate. This contains the cardholder's public signature key.

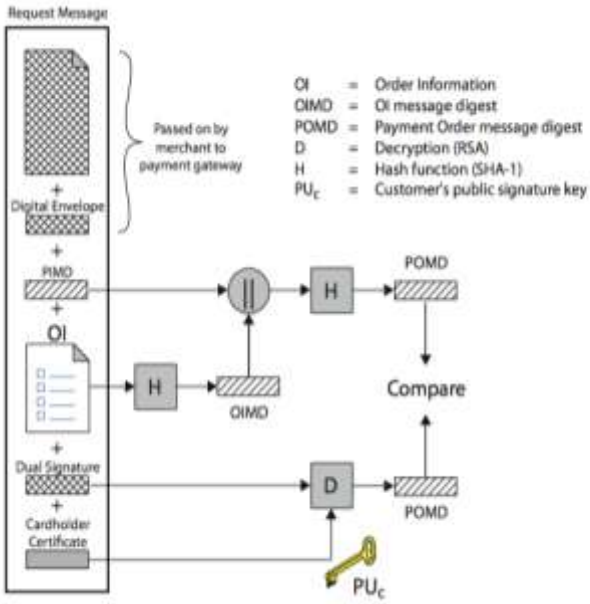


Purchase Request – Merchant

When the merchant receives the Purchase Request message, the actions listed are performed.

The Purchase Response message includes a response block that acknowledges the order and references the corresponding transaction number. This block is signed by the merchant using its private signature key.The block and its signature are sent to the customer, along with the

merchant's signature certificate. Figure. illustrates the crypto processes used by the merchant to verify the customer's purchase request order.



Payment Authorization:

During the processing of an order from a cardholder, the merchant authorizes the transaction with the payment gateway.The payment authorization ensures that the transaction was approved by the issuer, guarantees the merchant will receive payment, so merchant can provide services or goods to customer.

The payment authorization exchange consists of two messages: Authorization Request and Authorization response.The payment gateway performs the tasks shown on receiving the Authorization Request message.

Payment Gateway Authorization:

During the processing of an order from a cardholder, the merchant authorizes the transaction with the payment gateway. The payment authorization ensures that the transaction was approved by the issuer, guarantees the merchant will receive payment, so merchant can provide services or goods to customer.

The payment authorization exchange consists of two messages: Authorization Request and Authorization response.The payment gateway performs the tasks shown on receiving the Authorization Request message.

Payment Capture:

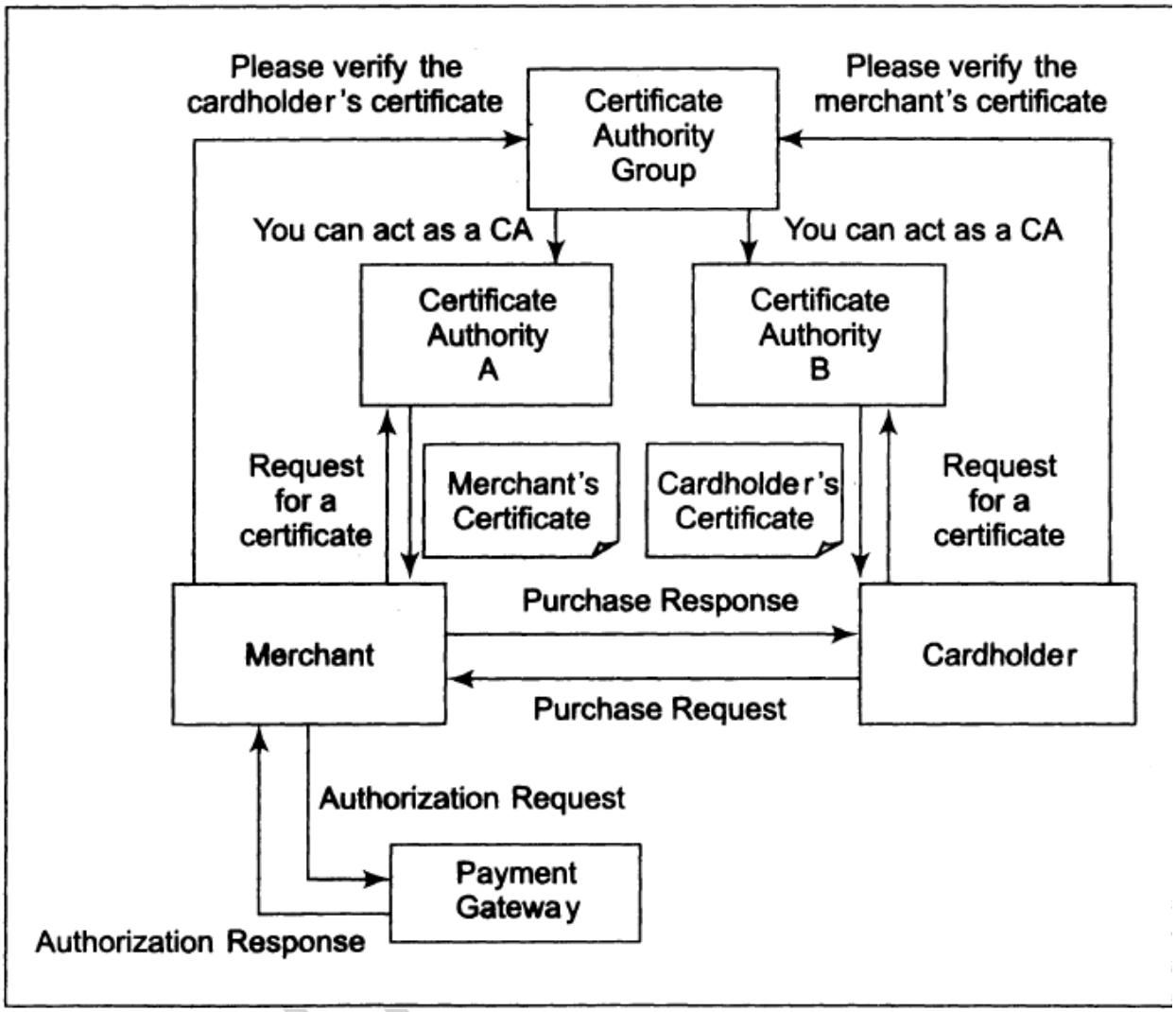
To obtain payment, the merchant sends a capture request message to the payment gateway, for which the merchant generates, signs, and encrypts.a capture request block, including payment amount and transaction ID.

The payment gateway receives the capture request message, decrypts and verifies the capture request block and decrypts and verifies the capture token block. It then checks for consistency between the capture request and capture token.

It then creates a clearing request sent to the issuer over the private payment network, which causes funds to be transferred to the merchant's account.

The gateway then notifies the merchant of payment in a Capture Response message, which includes a capture response block that the gateway signs and encrypts, plus the gateway's signature key certificate. The merchant software stores the capture response to be used for reconciliation with payment received from the acquirer.

SET Model



SSL VERSUS SET

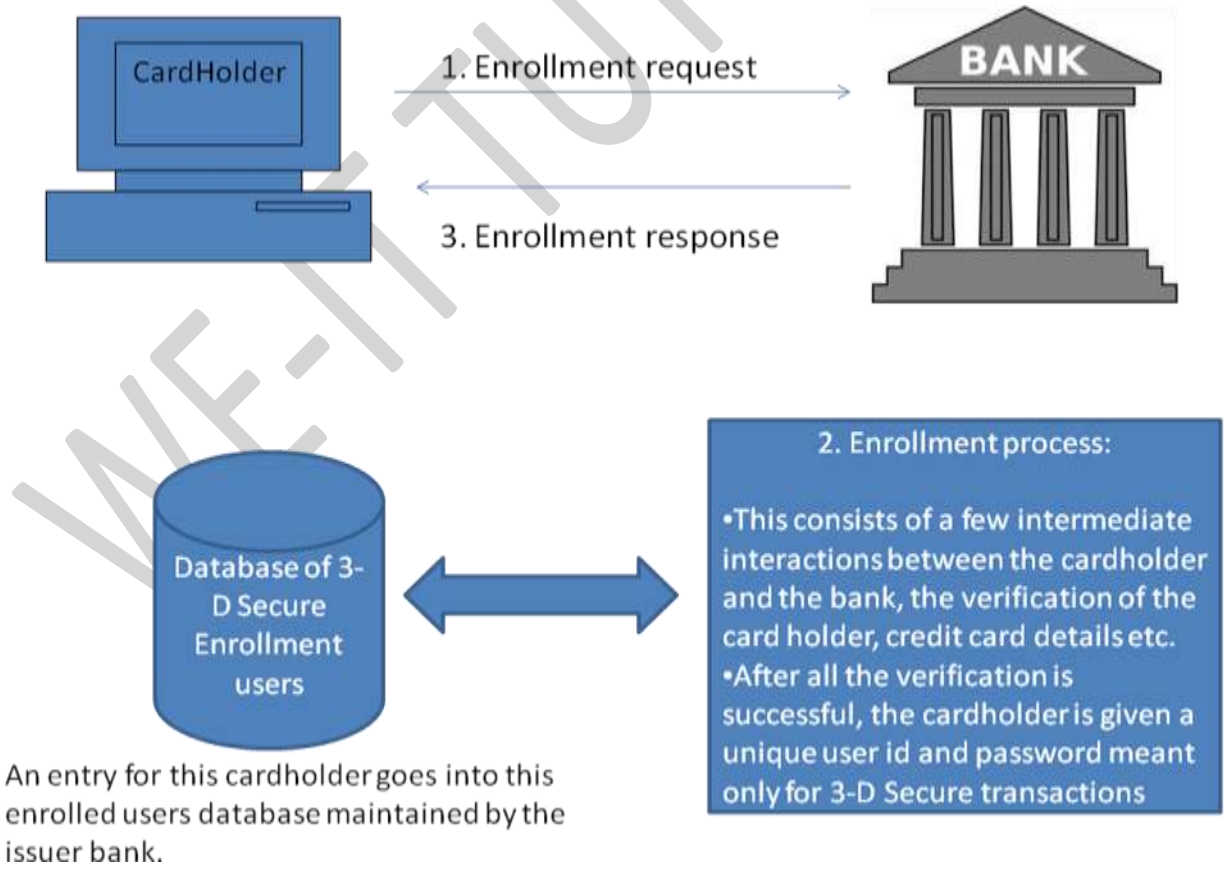
Issue	SSL	SET
Main aim	Exchange of data in an encrypted form	E-commerce related payment mechanism
Certification	Two parties exchange certificates	All the involved parties must be certified by a trusted third party
Authentication	Mechanism s in place, but not very strong	Strong mechanisms for authenticating all the parties involved
Risk of merchant	Possible, since customer gives financial	Unlikely, since customer gives financial

fraud	data to merchant	data to payment gateway
Risk of customer fraud	Possible, no mechanisms exist if a customer refuses to pay later	Customer has to digitally sign payment instructions
Action in case of customer fraud	Merchant is liable	Payment gateway is liable
Practical usage	High	Low at the moment, expected to grow

3-D SECURE PROTOCOL

SET has one limitation: it does not prevent a user from providing someone else's credit-card number. The credit-card number is protected from the merchant. However, how can one prevent a customer from using another person's credit-card number? That is not achieved in SET. Consequently, a new protocol developed by Visa has emerged, called 3-D Secure.

The main difference between SET and 3-D Secure is that any cardholder who wishes to participate in a payment transaction involving the usage of the **3-D** Secure protocol has to enroll on the issuer bank's *Enrollment Server.*



Protocol Overview

Step 1

The user shops using the shopping cart on the merchant site, and decides to pay the amount. The user enters the credit-card details for this purpose, and clicks on the OK button.

Step 2

When the user clicks on the *OK* button, the user will be redirected to the issuer bank's site. The bank site will produce a pop-up screen, prompting the user to enter the password provided by the issuer bank.

internal operations of 3-D Secure. The process uses SSL for confidentiality and server authentication.

1. The customer finalizes on the payment on the merchant site (the merchant has all the data of this customer).
2. A program called *merchant plug-in*, which resides at the merchant Web server, sends the User information to the Visa/MasterCard directory (which is LDAP-based).
3. The Visa/MasterCard directory queries the access control server running at the issuer bank (i.e. the customer's bank), to check the authentication status of the customer.
4. The access-control server forms the response for the Visa directory and sends it back to the Visa/MasterCard directory.
5. The Visa/MasterCard directory sends the payer's authentication status to the merchant plug-in.
6. After getting the response, if the user is currently not authenticated, the plug-in redirects the user to the bank site, requesting the bank or the issuer site to perform the authentication process.
7. The access-control server (running on the bank's site) receives the request for authentication of the user.
8. The authentication server performs authentication of the user based on the mechanism of authentication chosen by the user (e.g. password, dynamic password, mobile etc.) The access control server sends the information to the repository where the history of the user authentication is kept for legal purpose.
9. The plug-in receives the response of the access-control server through the user's browser. This contains the digital signature of the access-control server.
10. The plug-in validates the digital signature of the response and the response from the access control server.
11. If the authentication was successful and the digital signature of the access-control server is validated, the merchant sends the authorization information to its bank (i.e. the acquire bank).

Electronic Money

Electronic money (e-money) is broadly defined as an electronic store of monetary value on a technical device that may be widely used for making payments to entities other than the e-money issuer. The device acts as a prepaid bearer instrument which does not necessarily involve bank accounts in transactions.

E-money products can be hardware-based or software-based, depending on the technology used to store the monetary value.

Hardware-based products

In the case of hardware-based products, the purchasing power resides in a personal physical device, such as a chip card, with hardware-based security features. Monetary values are typically transferred by means of device readers that do not need real-time network connectivity to a remote server.

Software-based products

Software-based products employ specialised software that functions on common personal devices such as personal computers or tablets. To enable the transfer of monetary values, the personal device typically needs to establish an online connection with a remote server that controls the use of the purchasing power. Schemes mixing both hardware and software-based features also exist.

EMAIL SECURITY

The basic phases of an email communication consists of the following steps

1. At the sender's end, an SMTP server takes the message sent by a user's computer.
2. The SMTP server at the sender's end then transfers the message to the SMTP server of the receiver.
3. The receiver's computer then pulls the email message from the SMTP server at the receiver's end, using other email protocols such as Post Office Protocol (POP) or Internet Mail Access Protocol (IMAP)

There are three main email security protocols: **Privacy Enhanced Mail (PEM), Pretty Good Privacy (PGP) and Secure MIME (S/MIME).**

Privacy Enhanced Mail (PEM)

- The Privacy Enhanced Mail (PEM) is an email security standard adopted by the Internet Architecture Board (LAB) to provide secure electronic mail communication over the Internet.
- PEM was initially developed by the Internet Research Task Force (IRTF) and Privacy Security Research Group (PSRG).
- PEM supports the three main cryptographic functions of encryption, non-repudiation, and message integrity

The Working of PEM

Step 1: Canonical Conversion

- PEM transforms each email message into an abstract, canonical representation. This means that regardless of the architecture and the operating system of the sending and the receiving computers, the email message always travels in a uniform, independent format.

Step 2: Digital Signature

- It starts by creating a message digest of the email message using an algorithm such as MD2 or MD5

- The message digest thus created is then encrypted with the sender's private key to form the sender's digital signature.

Step 3: Encryption

- In this step, the original email and the digital signature are encrypted together with a symmetric key. For this, the DES or DES-3 algorithm in CBC mode is used.

Step 4: Base-64 Encoding

- The Base-64 encoding (also called Radix- 64 encoding or ASCII armour) process transforms arbitrary binary input into printable character output.

Pretty Good Privacy (PGP)

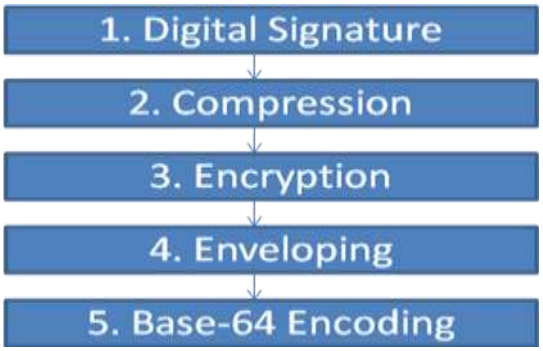
- Phil Zimmerman is the father of the Pretty Good Privacy (PGP) protocol.
- The most significant aspects of PGP are that it supports the basic requirements of cryptography, is quite simple to use, and is completely free, including its source code and documentation.
- a low-cost commercial version of PGP is available from an organization called Viacrypt (now Network Associates).
- PGP has become extremely popular and is far more widely used, as compared to PEM.

Step 1: Digital Signature

In PGP, it consists of the creation of a message digest of the email message using the SHA-1 algorithm. The resulting message digest is then encrypted with the sender's private key.

Step 2: Compression

the input message as well as the digital signature are compressed together to reduce the size of the final message that will be transmitted. For this, the famous ZIP program is used. ZIP is based on the Lempel-Ziv algorithm.



Step 3: Encryption

In this step, the compressed output of step 2 (i.e. the compressed form of the original email and the digital signature together) are encrypted with a symmetric key. For this, generally the IDEA algorithm in CFB mode is used.

Step 4: Digital Enveloping

In this case, the symmetric key used for encryption in step 3 is now encrypted with the receiver's public key.

The output of step 3 and step 4 together form a digital envelope

Step 5: Base-64 encoding
The output of step 4 is Base-64 is encoded now

Secure Multipurpose Internet Mail Extensions (S/MIME)

Multipurpose Internet Mail Extensions (MIME) system extends the basic email system by permitting users to send binary files using the basic email system.

When we enhance the basic MIME system to provide for security features, it is called Secure Multipurpose Internet Mail Extensions (S/MIME).

MIME Overview

The MIME specification adds five new headers to the email system.

These headers are described below.

- **MIME-Version** This contains the MIME version number. Currently, it has a value of 1.1. This field is reserved for future use, when newer versions of MIME are expected to emerge.
- **Content-Type** This describes the data contained in the body of the message. The details provided are sufficient so that the receiver email system can deal with the received email message in an appropriate manner.

- **Content-Transfer-Encoding** Specifies the type of transformation that has been used to represent the body of the message. In other words, the method used to encode the messages into zeroes and ones is defined here.
- **Content-ID Identifies** the MIME entities uniquely with reference to multiple contexts.
- **Content-Description** Used when the body is not readable (e.g. video).

S/MIME Functionality

S/MIME is quite similar to PGP. Like PGP, S/MIME provides for digital signatures and encryption of email messages.

- **Enveloped data**: This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.
- **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
- **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

Cryptographic Algorithms used in S/MIME

S/MIME prefers the usage of the following cryptographic algorithms:

- Digital Signature Standard (DSS) for digital signatures
- Diffie-Hellman for encrypting the symmetric session keys
- RSA for either digital signatures or for encrypting the symmetric session keys
- DES-3 for symmetric key encryption

S/MIME Messages

S/MIME makes use of a number of new MIME content types

Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs 7-mime	signedData	A signed S/MIME entity.
	pkcs 7-mime	envelopedData	An encrypted S/MIME entity.

	pkcs 7-mime	degenerate signedData	An entity containing only public- key certificates.
	pkcs 7-mime	Compressed Data	A compressed S/MIME entity
	pkcs 7-signature	signedData	The content type of the signature subpart of a multipart/signed message.

WIRELESS APPLICATION PROTOCOL (WAP) SECURITY

The Wireless Application Protocol (WAP) had arrived. In simple terms, WAP is a communication protocol that enables wireless mobile devices to have an access to the Internet.

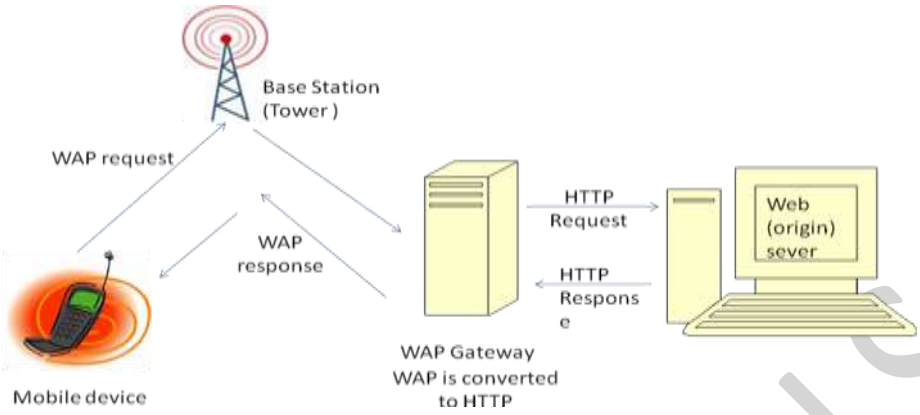
The WAP architecture was developed keeping in mind the basic Internet architecture, and yet realizing that WAP had to deal with the limitations of the mobile devices. Thus, the idea was to model WAP on the Internet architecture, and at the same time, make sure that the differences between the wired and the wireless world would also be taken care of.

Thus, the basic principle was to give the end user a look and feel that is similar to what the Internet gives, and also keep the communication between the content providers and the end users to a basic minimum.

These Internet protocols are too complex for the small and less powerful mobile devices and the mobile channels of communication.

The WAP gateway. Simplistically, the job of the WAP gateway is to translate client requests to the server from WAP to HTTP, and on the way back from the server to the client, from HTTP to WAP.

The WAP requests first originate from the mobile device (usually a mobile phone), which travel to the network carrier's base station (shown as a tower), and from there, they are relayed on to the WAP gateway where the conversion from WAP to HTTP takes place. The WAP gateway then interacts with the Web server (also called origin server) as if it is a Web browser, i.e. it uses HTTP protocol for interacting with the Web server. On return, the Web server sends an HTTP response to the WAP gateway, where it is converted into a WAP response, which first goes to the base station, and from there on, to the mobile device.

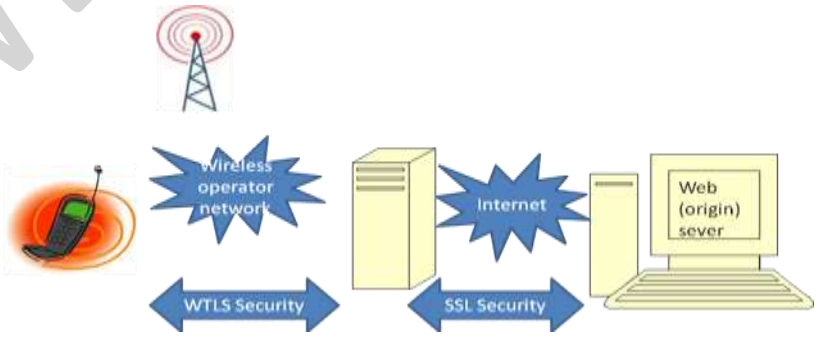


Security Layer

The security layer in the WAP stack is also called Wireless Transport Layer Security (WTLS) protocol. It is an optional layer, that when present, provides features such as authentication, privacy and secure connections.

The Security Layer— Wireless Transport Layer Security (WTLS)

- The wireless world is more vulnerable to security issues as compared to the wired world, as the number of parties involved is more, and the chances of people not taking proper security measures when on the move are significantly higher.
- As a result, the WAP protocol stack includes the Wireless Transport Layer Security (WTLS) as an additional layer, which is not found in other similar protocol stacks.
- WTLS is optional. It is based on the Transport Layer Security (TLS) protocol, which, in turn, is based on the Secure Socket Layer (SSL) protocol. When present, WTLS runs on top of the transport layer of WAP (WDP).
- SSL allows two parties involved in a transaction to make it totally secure and reliable.
- WLTS makes similar attempts in the wireless world. WTLS ensures four things:
    - privacy,
    - server authentication,
    - client authentication, and
    - data integrity.



Problem arises when WTLS-SSL and SSL-WTLS conversion is made.

Because the conversion is in plain text and not encrypted.

SECURITY IN GSM

- In earlier days of mobile telephony, analog technologies such as AMPS(Advanced Mobile Phone System) were used.
- But there was little or no security in such technologies.
- D-AMPS was an improvement to AMPS.
- Another similar technology called GSM (Global System for Mobile Communications) is used.
- An alternative to WAP standard have emerged i.e. GPRS (General Packet Radio Service) which offers an analog modems service wherever GSM wireless service is available.
- GSM is for voice and GPRS is for data.

Elements for GSM:

- The SIM*(Subscriber Identity Module)* contains:
- IMSI *(International Mobile Subscriber Identity)*
- Ki *(subscriber authentication key)*
- Ciphering key generation algorithm (A8)
- The authentication algorithm (A3)
- PIN (Personal Identification Number)
- GSM handset contains:
- Ciphering algorithm (A5)
- AUC (Authentication Center) which is a part of GSM network contains:
- Encryption algorithm (A3, A5 and A8)
- As well as database of identification & authentication information about the subscribers.

There are three key aspects to GSM security:

Authentication:

1. The GSM network authenticates subscriber by a challenge-response mechanism
2. The network sends 128 bit random number to subscriber.
3. 32- bit signed response using A3 and Ki is prepared by the handset and send to network.
4. The network retrieves its value of Ki from database
5. Performs the same operation using A3 algorithm
6. If the result matches with the one received from handset
7. Then the user authentication is successful.

Signaling and data confidentiality:

1. A8 algorithm is used to create a 64 bit ciphering key(Kc).
2. The value of Kc is obtained by applying the same random number as above.
3. Kc is later used for secure communications between the subscriber and the mobile telephony base station.

Voice and data security:

1. A5 algorithm is used to encrypt the voice and data traffic between handset and network.
2. For this, the handset sends a ciphering mode request to the network.

3. In response, the network starts encryption and decryption using A5 and Kc.

Security in 3G

- GPRS has naturally evolved into Universal Mobile Telephone System (UMTS).
- UMTS is called the 3G of wireless /mobile telephony.
- UMTS is used to deliver high tech applications such as video on demand, video/audio streaming, high speed multimedia, videoconferencing, multi-player gaming and improved mobile Internet access.
- The user authentication process involves 3 parties: user's mobile handset, home location and current location.

The UMTS authentication process involves three parties: User's mobile handset, Home location and the Current location (note that the user of a mobile phone can move around, and therefore the current location of the user may be different from his/her home location). The user authentication process consists of four steps, as follows.

1. The user's mobile handset sends its International Mobile Subscriber Id (IMSI) to the Home Location. The IMSI is unique for a handset, and is optionally encrypted before it is sent to the Home Location.

2. The Home Location performs the following steps now:
   (a) It generates a random number (RAND).
   (b) Next, it retrieves the secret key it shares with this user's handset from its database.
   (c) It uses the random number and the key to generate the following items:
      (i) Response (RES)
      (ii) Confidentiality Key (CK)
      (iii) Integrity Key (IK)
      (iv) Authentication Key (AK)
   (d) The Home Location and the user's handset share a secret, called Sequence Number (SEQ). The Home Location calculates a MAC on the combination of the random number (RAND) and the sequence number (SEQ), i.e. it does MAC (RAND, SEQ).
   (e) It now does an XOR operation on the sequence number (SEQ) and Authentication Key (AK), i.e. it does (SEQ XOR AK).
   (f) Finally, it sends the following items to the Current Location of the user: RAND, RES, CK, IK, MAC, (SEQ XOR AK)
3. The Current Location receives these values from the Home Location, and sends the following to the user's handset:
   (a) Random number (RAND)
   (b) XOR of sequence number (SEQ) and Authentication Key (AK), i.e. (SEQ XOR AK)
   (c) MAC
4: The user's handset receives these values, and performs the following tasks:
   (a) It calculates the response (RES), using the random number (RAND) received from the Current Location, and the secret key shared with its Home Location.
   (b) It also generates the various keys such as CK, IK and AK in the same fashion, as was done by the Home Location in step 1.

(c) It now performs an XOR operation using AK over the value (SEQ XOR AK). That is, it performs (SEQ XOR AK) XOR AK. Clearly, this would yield back the sequence number (SEQ).
(d) Using the random number (RAND) and the sequence number (SEQ), it performs a MAC operation, as MAC (RAND, SEQ).
(e) It compares the MAC calculated in step 4 with the MAC received from the Current Location (see step 3).
(f) If all these sub steps happen successfully, the user's handset sends the response (RES) to the Current Location. If this response matches with the response (RES) possessed by the Current Location (recall that it had received this from the Home Location in step 1), the Current Location considers this user as authentic, and makes the appropriate entries in its database.

UNIT 6

User Authentication and Kerberos

AUTHENTICATION BASICS

- Authentication can be defined as determining an identity to the required level of assurance.
- Authentication is the first step in any cryptographic solution. We say this because unless we know who is communicating, there is no point in encrypting what is being communicated.
- As we know, the whole purpose of encryption is to secure communication between two or more parties.
- there is no use of encryption without authentication.

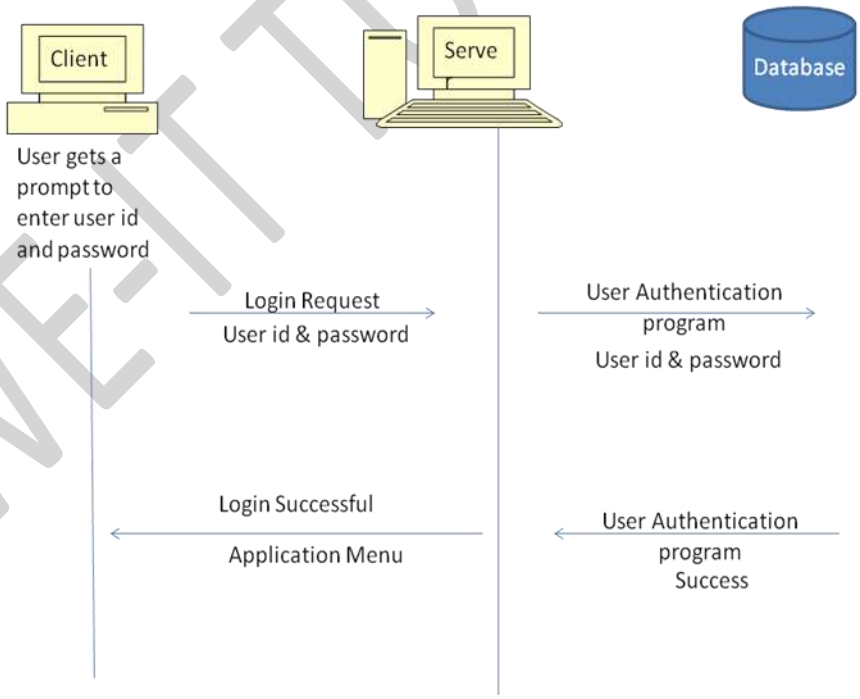
PASSWORDS

Passwords are the most common form of authentication. A password is a string of alphabets, numbers and special characters, which is supposed to be known only to the entity (usually a person) that is being authenticated.

Clear-Text Password

This is the simplest password-based authentication mechanism. Usually, every user in the system is assigned a user id and an initial password. The user changes the password periodically for security reasons. The password is stored in clear text in the user database against the user id on the server.

working:



Problems with the Scheme

Problem 1— Database Contains Passwords in Clear Text Firstly, the user database contains user ids and passwords in clear text. Therefore, if an attacker succeeds in obtaining an access to the database, the whole list of user ids and passwords is available to the attacker.

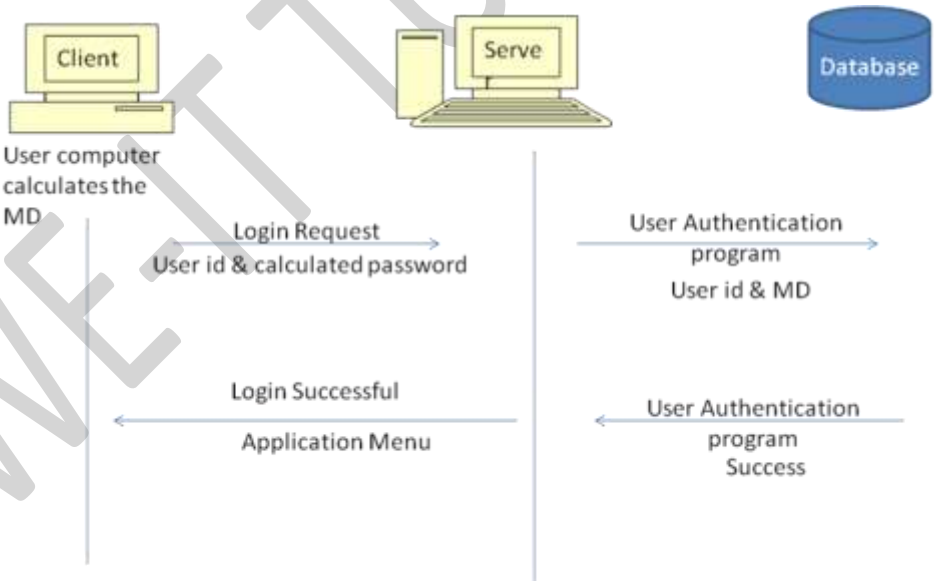
Problem 2— Password Travels in Clear Text from the User's Computer to the Server Even if we store encrypted passwords in the database, the password would travel in clear text from the user to the server. Therefore, if an attacker breaks into the communication link between the user's computer and the server, the attacker can easily obtain the clear-text password.

Something Derived from Passwords

The variation from the basic password-based authentication is not to use the password itself, but to use something that is *derived from* the password. That is, instead of storing the password as it is, or in an encrypted format, we can run some algorithm on the password and store the output of this algorithm as the (derived) password in the database.

When the user wants to be authenticated, the user enters the password, the user's computer performs the same algorithm locally, and sends the derived password to the server, where it is verified.

Step:1 Create a message digest of passwords
Step:2 Store the user id and message digest of the passwords in the user database.



Adding Randomness

To improve the security of the solution, we need to add a bit of unpredictability or randomness to the earlier scheme. This is to ensure that although the message digest of the password is always the same, the

exchange of information between the user's computer and the server is never the same. This will ensure that a *replay attack* is foiled.

Step 1: Storing Message Digests as Derived Passwords in the User Database
Step 2: User Sends a Login Request

Step 3: Server Creates a Random Challenge : If the user id is valid, the server now creates a random challenge (a random number, generated using a pseudorandom number generation technique), and sends it back to the user.

Step 4: User Signs the Random Challenge with the Message Digest of the Password : At this stage, the application displays the password-entry screen to the user. In response, the user enters the password on the screen. The application executes the appropriate message-digest algorithm on the user's computer to create a message digest of the password entered by the user. This message digest of the password is now used to encrypt the random challenge received from the server. This encryption is, of course, of a symmetric-key encryption form.

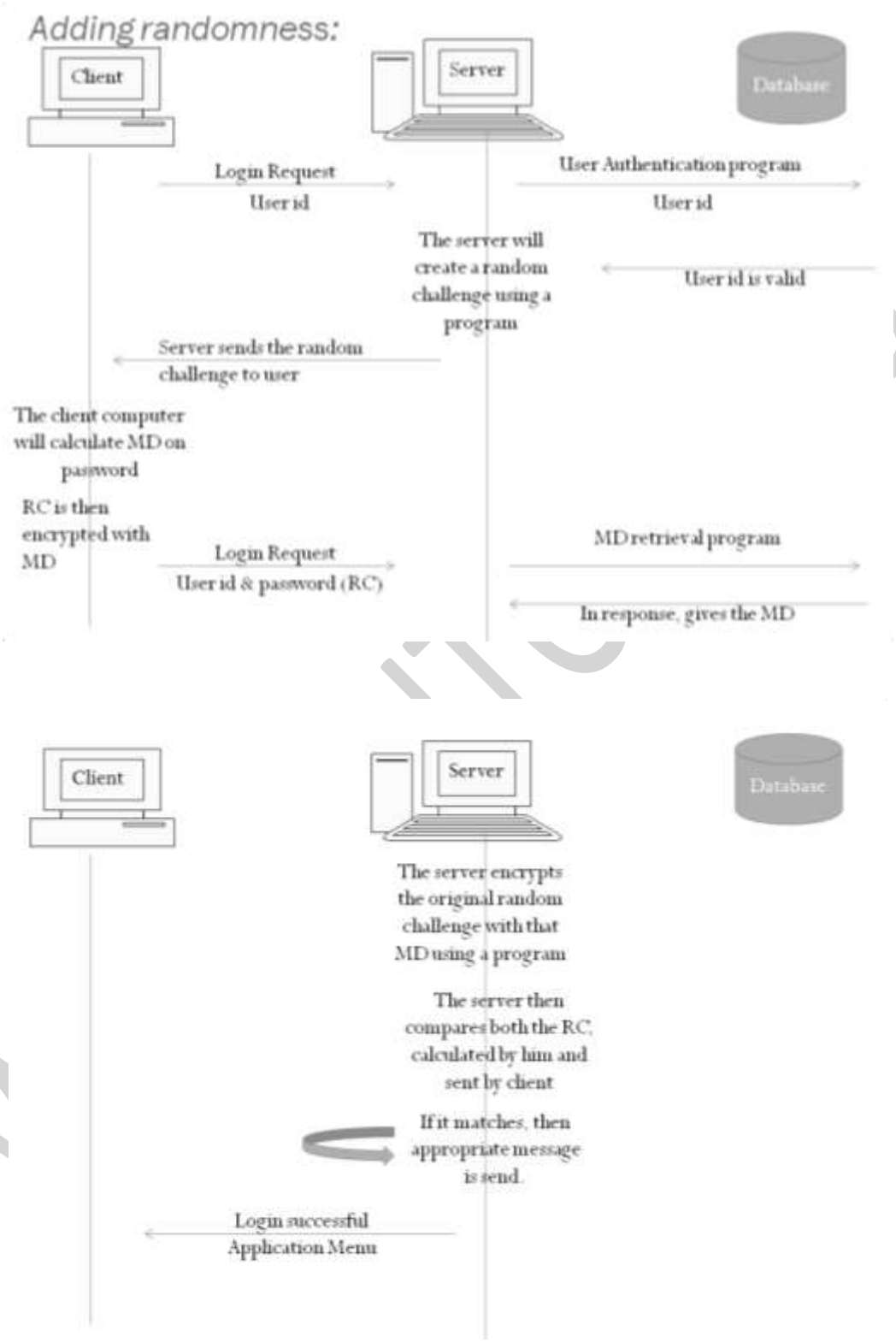
Step 5: Server Verifies the Encrypted Random Challenge Received from the User The server receives the random challenge, which was encrypted by the password of the user's message digest. In order to verify that the random challenge was indeed encrypted by the message digest of the user's password, the server must perform an identical operation.

This can be done in two ways:

- The server can decrypt the encrypted random challenge received from the user with the message digest of the user's password. As we know, the message digest of the user's password is available to the server via the user database. If this decryption matches the original random challenge available on the server, the server can be assured that the random challenge was indeed encrypted by the message digest of the user's password.
- Alternatively, the server can simply encrypt its own version of the random challenge (i.e. the one which was sent earlier to the user) with the message digest of the user's password. If this encryption produces an encrypted random challenge, which matches with the encrypted random challenge received from the user, the server can be assured that the random challenge was indeed encrypted by the message digest of the user's password.

Step 6: Server Returns an Appropriate Message back to the User Finally, the server sends an appropriate message back to the user, depending on whether the previous operations yielded success or failure.

Note that the random challenge is different every time. Therefore, the random challenge encrypted with the message digest of the password would also be different every time.


Adding randomness:
Client
Server
Database
Login Request
User id
User Authentication program
User id
The server will create a random challenge using a program
User id is valid
Server sends the random challenge to user
The client computer will calculate MD on password
RC is then encrypted with MD
Login Request
User id & password (RC)
MD retrieval program
In response, gives the MD
Client
Server
Database
The server encrypts the original random challenge with that MD using a program
The server then compares both the RC, calculated by him and sent by client
If it matches, then appropriate message is send.
Login successful
Application Menu

AUTHENTICATION TOKENS

An **authentication token** is an extremely useful alternative to a password. An authentication token is a small device that generates a new random value every time it is used. This random value becomes the basis for authentication.

- Processor
- Liquid Crystal Display (LCD) for displaying outputs
- Battery
- (Optionally) a small keypad for entering information
- (Optionally) a real-time clock

Each authentication token (i.e. each device) is pre-programmed with a unique number, called a **random seed,** or just **seed.**

Step 1: Creation of a Token

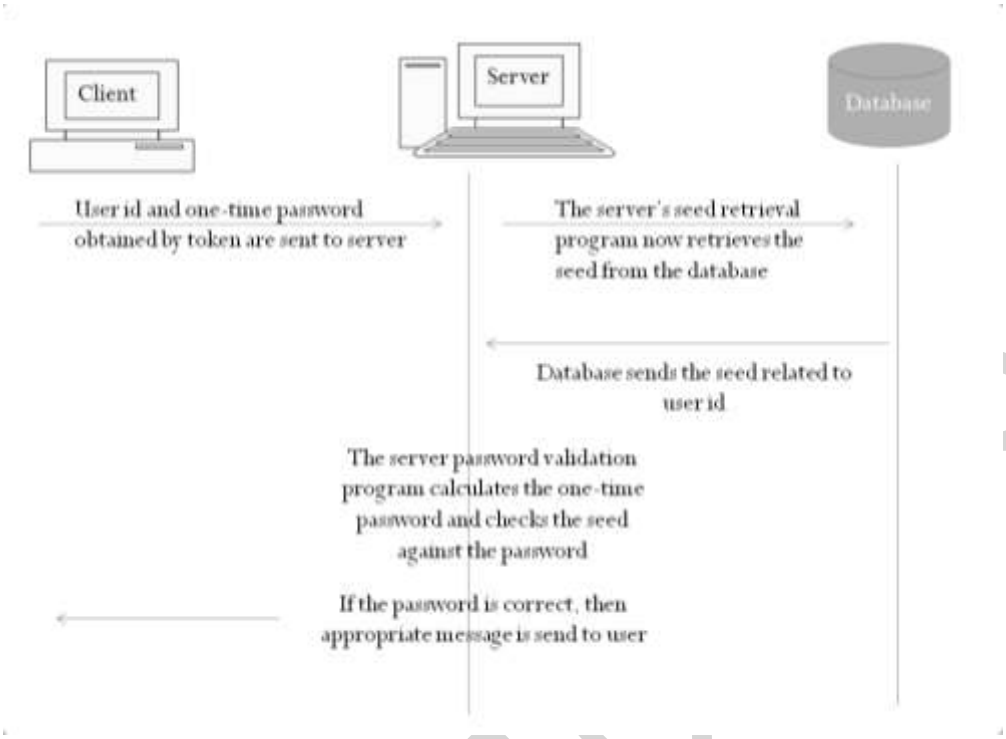
- Whenever an authentication token is created, the corresponding random seed is generated for the token by the **authentication server** (a special server that is configured to work with authentication tokens).
- This seed is stored or pre-programmed inside the token, as well as its entry is made against that user's record in the user database.

Step 2: Use of Token

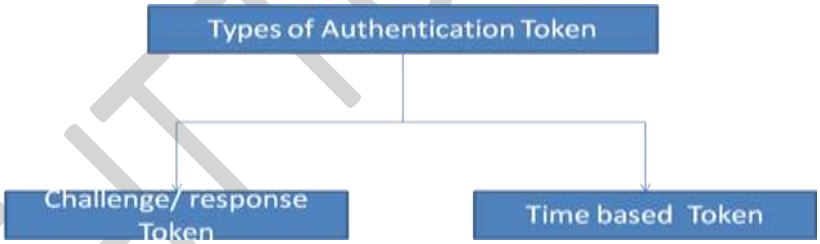
- An authentication token automatically generates pseudorandom numbers, called **one-time passwords** or **one-time passcodes.** One-time passwords are generated randomly by an authentication token, based on the seed value that they are pre-programmed with.
- They are *one-time* because they are generated, used once, and discarded for ever. When a user wants to be authenticated, the user will get a screen to enter the user id and the latest one-time password.

Step 3: Server Returns an Appropriate Message back to the User

- Finally, the server sends an appropriate message back to the user, depending on whether the previous operations yielded success or failure.



Authentication Token Types



1. ***Challenge/Response Tokens***

Step 1: User Sends a Login Request In this technique, the user sends the login request only with his/her user id (and not the one-time password).

Step 2: Server Creates a Random Challenge

If the user id is valid, the server now creates a random challenge (a random number, generated using a pseudo-random number generation technique), and sends it back to the user.

Step 3: User Signs the Random Challenge with the Message Digest of the Password
The user gets a screen, which displays the user id, the random challenge received from the server, and a data entry field, with the label *Password.*

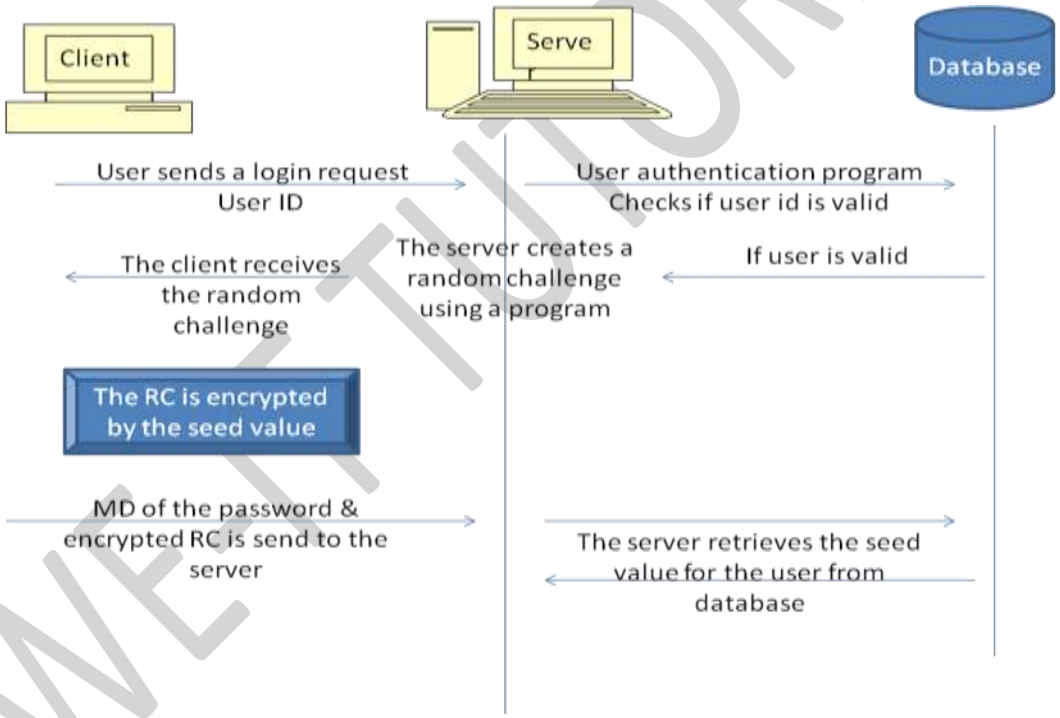
At this stage, the user reads the random challenge displayed on the screen. It first opens the token using her PIN and then keys in the random challenge received from the server inside the token. In order to do this, the token has a small keypad. The token accepts the random challenge, and encrypts it with the seed value, which is known only to itself. The result is the random challenge encrypted with the seed. This result is displayed on the display (LCD) of the token. The user reads this value and types it in the Password field on the screen. This request is then sent to the server as the login request.

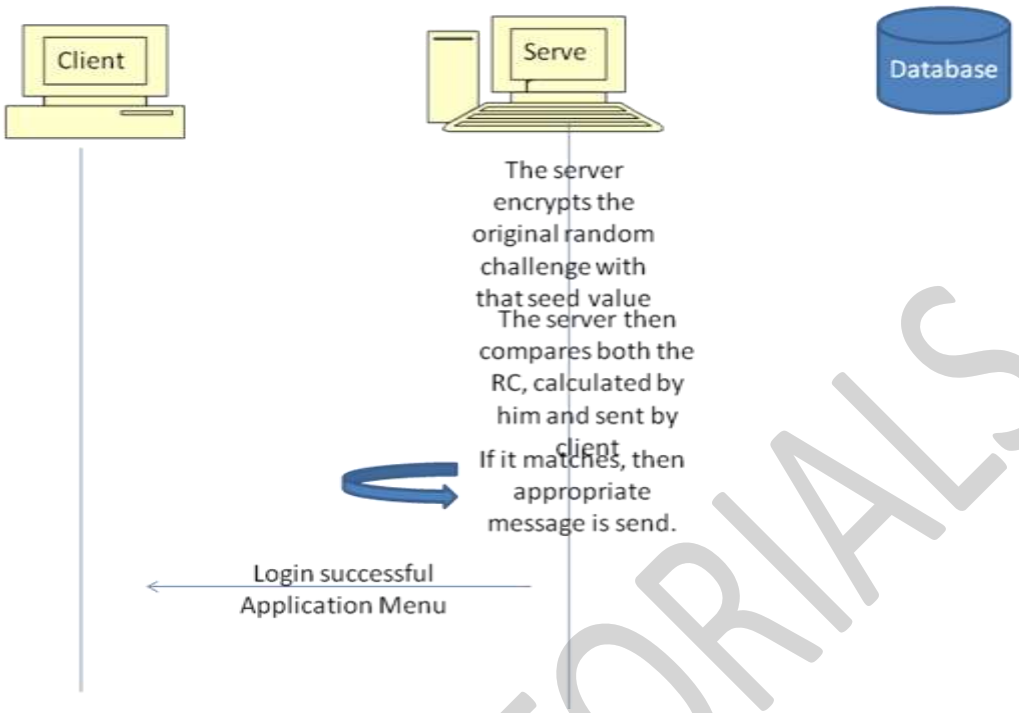
Step 4: Server Verifies the Encrypted Random Challenge Received from the User

The server receives the random challenge, which was encrypted with the seed by the user's authentication token. In order to verify that the random challenge was indeed encrypted by the correct seed, the server must perform an identical operation.

Step 5: Server Returns an Appropriate Message Back to the User

Finally, the server sends an appropriate message back to the user, depending on whether the previous operations yielded success or failure.





2. ***Time-based Tokens***

In spite of the improvement to the Challenge/Response mechanism discussed earlier (using message digest instead of encryption), there is still a practical problem.

Note that the user has to make three entries:

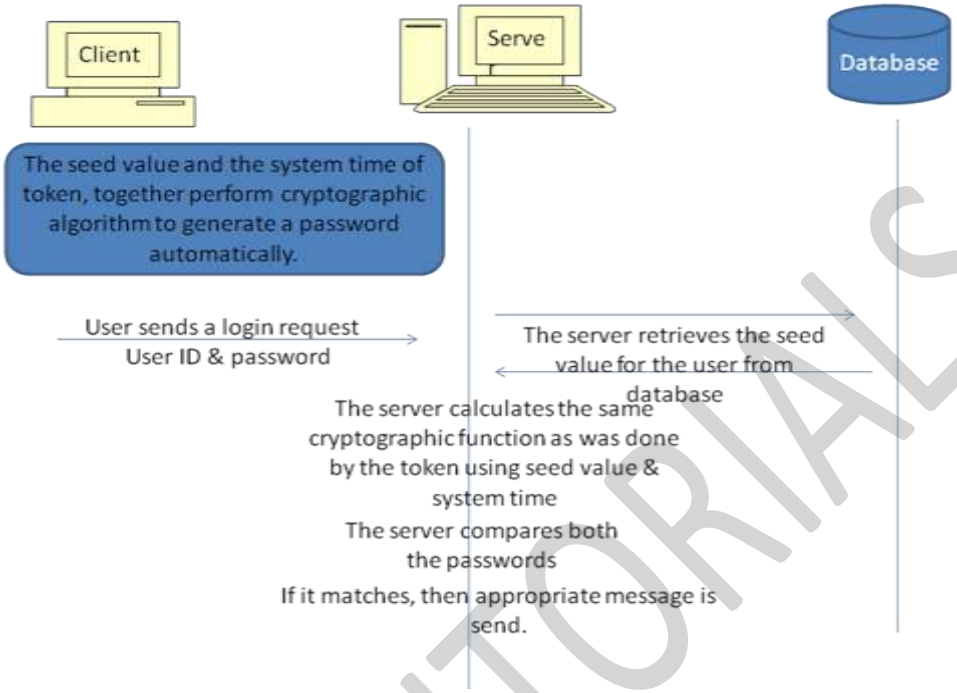
- firstly, the user has to enter the PIN to access the token;
- secondly, the user has to read the random challenge from the screen and key in the random number challenge into the token, and
- thirdly, the user has to read the encrypted random challenge from the LCD of the token and enter it into the *Password* field.

firstly, the user has to enter the PIN to access the token; secondly, the user has to read the random challenge from the screen and key in the random number challenge into the token, and thirdly, the user has to read the encrypted random challenge from the LCD of the token and enter it into the *Password* field.

Step 1: **Password Generation and Login Request** The token is pre-programmed with a seed, as usual. The copy of the seed is also available to the authentication server. As we know, a challenge/ response token performs an operation such as encryption or message-digest creation only based on the user's inputs. However, in the case of time-based tokens, this is handled differently.

Step 2: **Server-side Verification** The server receives the password. It also performs an independent cryptographic function on the user's seed value and the current system time to generate its version of the password. If the two values match, it considers the user as a valid one.

Step 3: **Server Returns an Appropriate Message Back to the User** Finally, the server sends an appropriate message back to the user, depending on whether the previous operations yielded success or failure.



CERTIFICATE-BASED AUTHENTICATION

- Another emerging authentication mechanism is that of **certificate-based authentication.** This is based on the digital certificate of a user. FIPS-196 is a standard that specifies the operation of this mechanism.
- As we know, in PKI, the server and (optionally) the client are required to possess digital certificate in order to perform digital transactions. The digital certificates can then be reused for user authentication as well. In fact, if we use SSL, the server *must* have a digital certificate, whereas the clients (users) *may* have digital certificates.
- This is because the client authentication is optional in SSL, but not the server authentication.
- Certificate-based authentication is a stronger authentication mechanism as compared to a password based authentication mechanism, because here the user is expected to *have* something (certificate) and not *know* something (password).
- At the time of login, the user is requested to send his/her certificate to the server, over the network as a part of the login request. A copy of the certificate exists on the server, which can be used to verify that the certificate is indeed a valid one.

The Working of Certificate-based Authentication

Step 1: Creation, Storage and Distribution of Digital Certificates

- The first step in certificate-based authentication is actually a pre-requisite.
- Here, the digital certificates are created by the CA for each user and the certificates are sent to the respective users.

- Moreover, a copy of the certificate is stored by the server in its database (usually in a binary format), in order to verify the certificate during the user's certificate-based authentication.

Step 2: Login Request

- During a login request, the user sends only his/her user id to the server

Step 3: Server Creates a Random Challenge

- When the server receives the user's login request containing the user id alone, it first checks to see if the user id is a valid one (note that only the user id is checked).
- If it is not, it sends an appropriate error message back to the user.
- If the user id is valid, the server now creates a random challenge (a random number, generated using a pseudo-random number generation technique), and sends it back to the user.
- The random challenge can travel as plain text from the server to the user's computer

Step 4: User Signs the Random Challenge

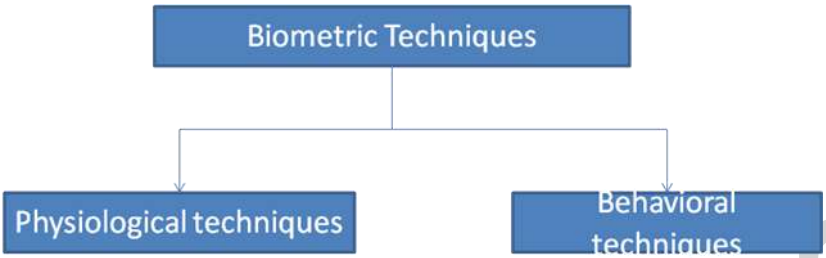
- The user has to now sign the random challenge with her private key.
- As we know, this private key corresponds to the user's public key, with the latter being mentioned in the user's certificate. For this purpose, the user needs to access his/her private key, which is stored as a disk file on her computer.
- However, the private keys are not available directly to anybody. In order to protect them, passwords are used.
- Only a correct password can open a private-key file. Therefore, the user must enter the secret password for opening up the private key file.

Step 5: Server Returns an Appropriate Message Back to the User

- Finally, the server sends an appropriate message back to the user, depending on whether the previous operations yielded success or failure.

BIOMETRIC AUTHENTICATION

- A biometric device works on the basis of human characteristics.
- Every time, the sample produced in biometric varies slightly.
- That is why many biometric samples are combined and their average is stored in the database.
- Thus an approximate match is acceptable.
- In biometric authentication system, 2 configurable parameters are defined:
- FAR (False Accept Ratio) = is a measurement of the chance that a user who should be rejected is actually accepted by the system as good enough.
- FRR (False Reject Ratio) = is a measurement of the chance that a user who should be accepted as valid is actually rejected by the system as not good enough.
- If the two samples match to the expected degree on the basis of values of FAR and FRR, the user is authenticated otherwise not.



Physiological techniques

- Face – check and measure the distance between the various facial features such as eyes, nose & mouth.
- Voice – characteristics of sound waves, pitch and tone of voice.
- Fingerprint – authentication uses 2 approaches:
  - Minutiae based: graph of individual ridge positions is drawn
  - Image based: image of fingerprint is stored.
- Iris – unique pattern of inside the iris
- Retina – the vessels carrying blood supply at the back of human eye are examined

Behavioral techniques

- Keystroke – speed of typing, strength of keystrokes, time between 2 keystrokes, error percentage and frequency.
- Signature – physically signed by the authorizer is compared by the computerized scanned copy

What is Kerberos?

- It is a Network authentication protocol.
- It provides for strong authentication for client-server applications.
- Kerberos is an authentication service developed as part of Project Athena at MIT in the mid 1980s.
- It is available as open source or in supported commercial software.
- It uses secret – key cryptography to provide strong authentication.
- It provide strong security on physically insecure network.
- A centralized authentication server which authenticates users to servers and servers to users.

Motivation/ Requirement of Kerberos:

Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.

- Security – A network eavesdropper should not be able to obtain the necessary information to impersonate a user i.e. secure against eavesdropping. Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- Reliable – For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.
- Transparent – Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password i.e. almost invisible to user.
- Scalable – The system should be capable of supporting large numbers of users and servers.

To support these requirements, the overall scheme of Kerberos is that of a trusted third-party authentication service that uses a protocol.

It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. Assuming the Kerberos protocol is well designed, then the authentication service is secure if the Kerberos server itself is secure.

Need of Kerberos:

an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. The servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services. In particular, the following three threats exist:

- A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access.

Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

Design Requirement:

In a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers. In this environment, three approaches to security can be provided:

4. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).
5. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.

6. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

Version of Kerberos:

Kerberos Version 4

- Version 4 of Kerberos makes use of DES, to provide the authentication service.
- An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database.
- In addition, the AS shares a unique secret keywith each server. These keys have been distributed physically or in some other securemanner. Consider the following aspects:
  (1) C AS: $ID_C||P_C||ID_V$

  (2) AS C: *Ticket*

  (3) C V: $ID_C||Ticket$

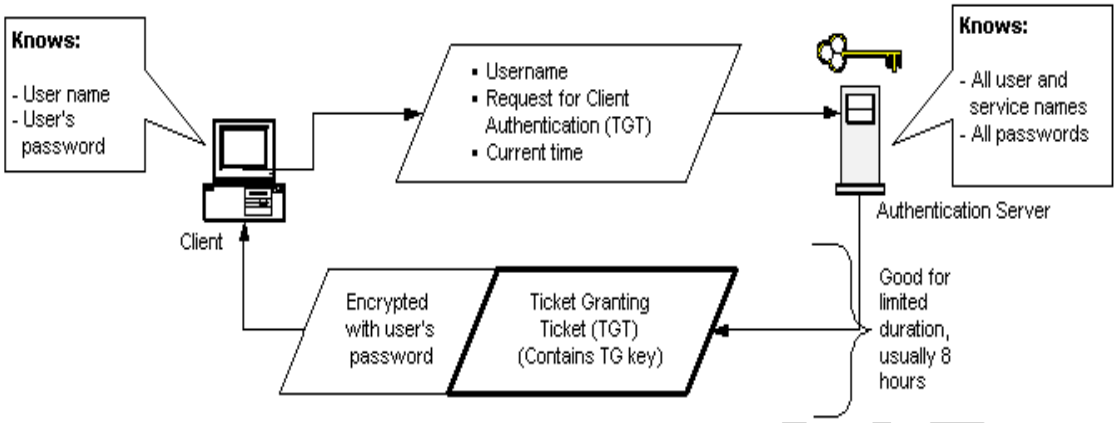
  $Ticket = E(K_v, [ID_C||AD_C||ID_V])$

where

C = client
AS = authentication server
V =server
ID_C = identifier of user on C
ID_V = identifier of V
P_C = password of user on C
AD_C = network address of C
K_v = secret encryption key shared by AS and V

- In this scenario, the user logs on to a workstation and requests access to server V. The client module C in the user's workstation requests the user's password and then sends a message to the AS that includes the user's ID, the server's ID, and the user's password.
- The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V. If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic. To do so, the AS creates a ticket that contains the user's ID and network address and the server's ID.
- This ticket is encrypted using the secret key shared by the AS and this server. This ticket is then sent back to C. Because the ticket is encrypted, it cannot be altered by C or by an opponent.
- With this ticket, C can now apply to V for service. C sends a message to V containing C's ID and the ticket. V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message. If these two match, the server considers the user authenticated and grants the requested service.

Initial Issuance of Ticket Granting Ticket



- Although the foregoing scenario solves some of the problems of authentication in an open network environment, the first problem is like to minimize the number of times that a user has to enter a password.
- Suppose each ticket can be used only once. If user C logs on to a workstation in the morning and wishes to check his or her mail at a mail server, C must supply a password to get a ticket for the mail server. If C wishes to check the mail several times during the day, each attempt requires reentering the password.
- We can solve this problem for a single logon session, the workstation can store the mail server ticket after it is received and use it on behalf of the user for multiple accesses to the mail server.
- The second problem is involved a plaintext transmission of the password. An eavesdropper could capture the password and use any service accessible to the victim.To solve these problems, we introduce a scheme for avoiding plaintext passwords and a new server, known as the ticket-granting server (TGS).

- **Once per user logon session:**
  (1) C AS: $ID_C||ID_{tgs}$
  (2) AS C: $E(K_c, Ticket_{tgs})$

- **Once per type of service:**
  (3) C TGS: $ID_C||ID_V||Ticket_{tgs}$
  (4) TGS C: $Ticket_v$

- **Once per service session:**
  Once per user logon session:

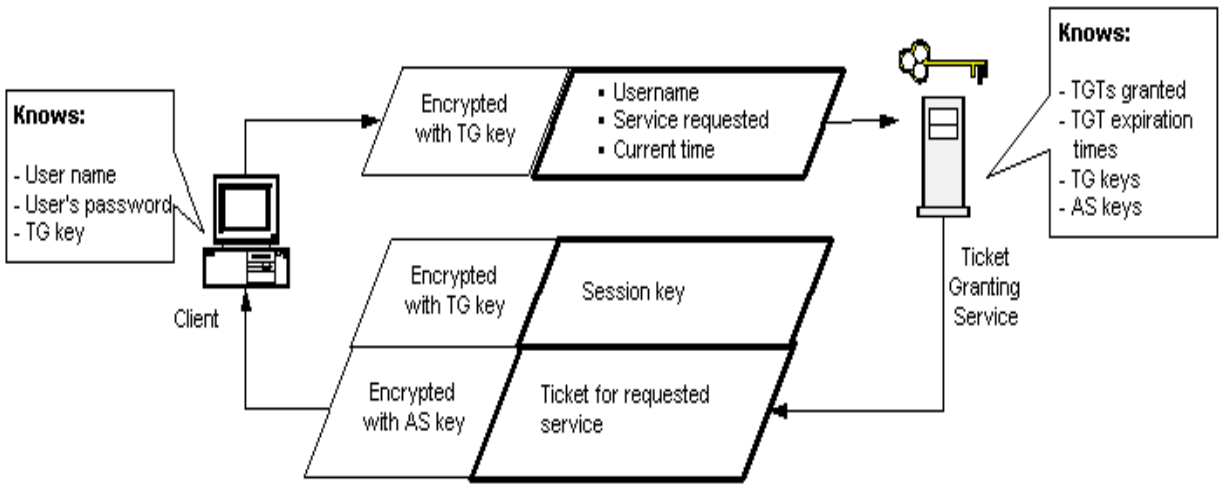
  (5) C V: $ID_C||Ticket_v$

  $Ticket_{tgs} = E(K_{tgs}, [ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1])$

  $Ticket_v = E(K_v, [ID_C||AD_C||ID_V||TS_2||Lifetime_2])$

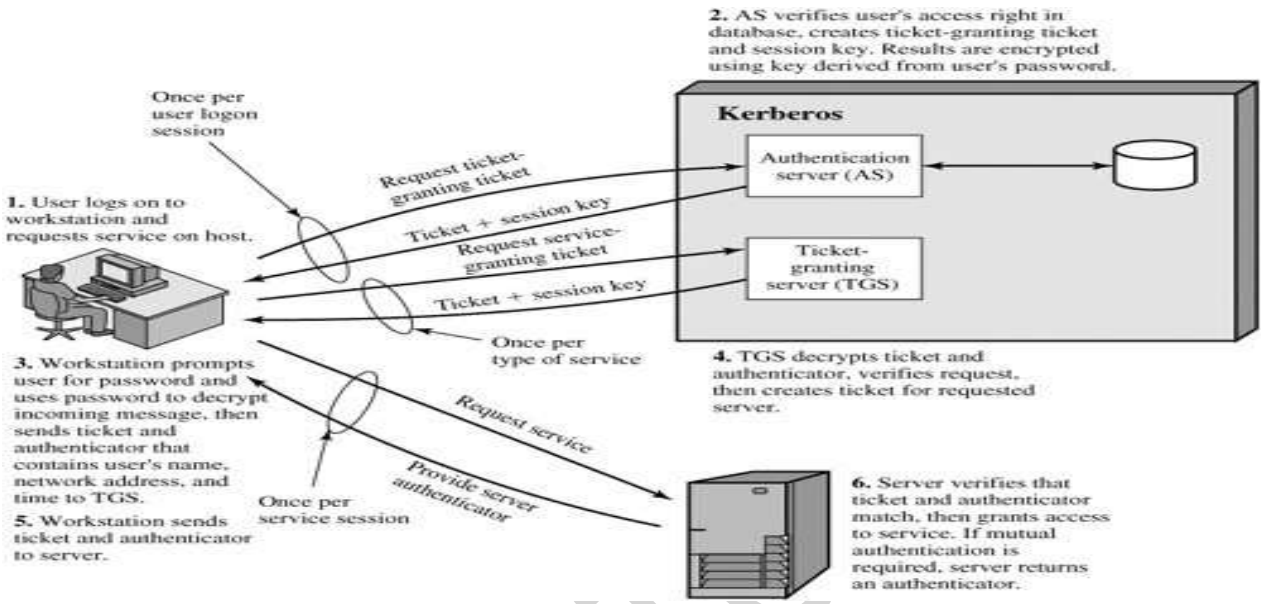
The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS.

Susequent Requests for Services from the Ticket Granting Service



1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID and password to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.
3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.
5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

Figure - Overview of Kerberos



Kerberos Realms and Multiple Kerberi –

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.

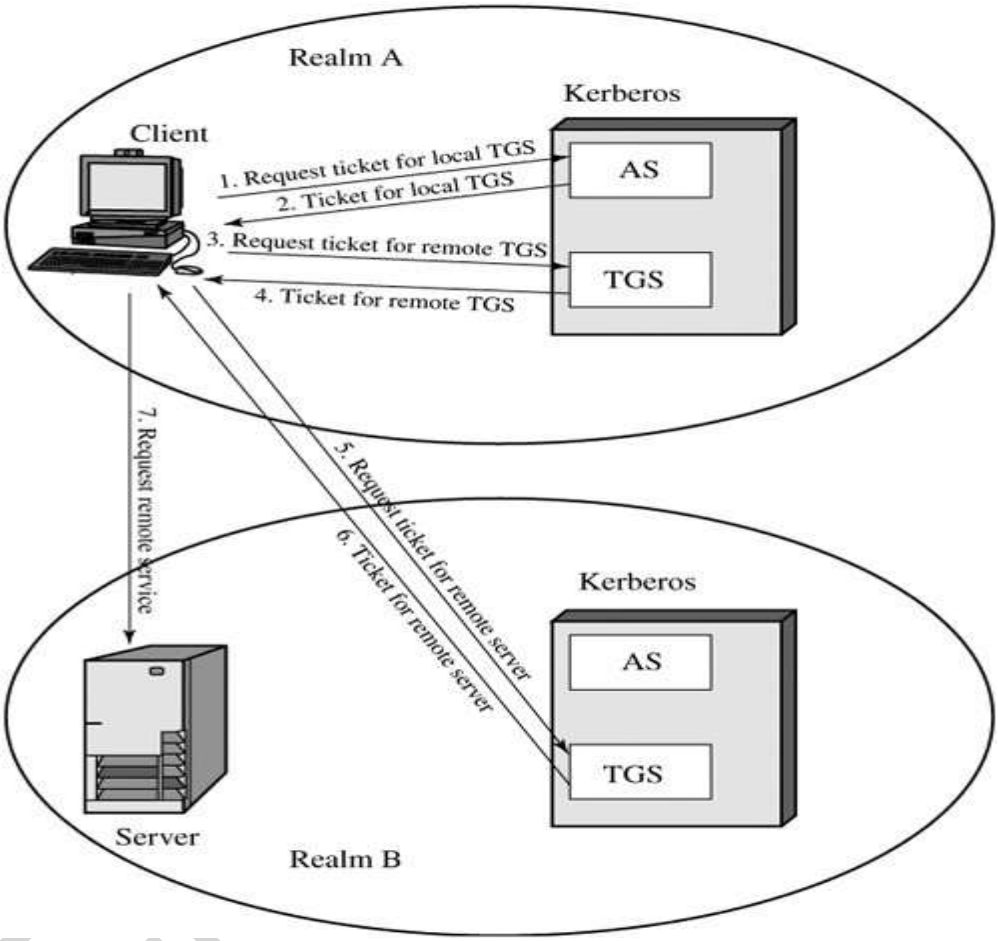
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a **Kerberos realm**.

- A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room.
- A read-only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password.
- Kerberos provides a mechanism for supporting such interrealm authentication. For two realms to support interrealm authentication, a third requirement is:

3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

Figure - Request for Service in Another Realm



Explain differences between Versions 4 and 5

Version 5 is intended to address the limitations of version 4 in two areas: environmental shortcomings and technical deficiencies.

Environmental Limitations / Shortcomings:

1. **Encryption system dependence:** Version 4 requires the use of DES. In version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used.

2. **Internet protocol dependence:** Version 4 requires the use of Internet Protocol (IP) addresses. Other address types, such as the ISO network address, are not accommodated. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

3. **Message byte ordering:** In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address. This techniques works but does not follow established conventions. In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.

4. **Ticket lifetime:** Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is 28 x 5 = 1280 minutes, or a little over 21 hours. This may be inadequate for some applications (e.g., a long-running simulation that requires valid Kerberos credentials throughout execution). In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.

5. **Authentication forwarding:** Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of the client. For example, a client issues a request to a print server that then accesses the client's file from a file server, using the client's credentials for access. Version 5 provides this capability.

6. **Interrealm authentication:** In version 4, interoperability among N realms requires on the order of $N2$ Kerberos-to-Kerberos relationships. Version 5 supports a method that requires fewer relationships.

There are **technical deficiencies** in the version 4 protocol itself and version 5 attempts to address these. The deficiencies are the following:

1. **Double encryption:** If tickets provided to clients are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful.

2. **PCBC encryption:** Encryption in version 4 makes use of a non standard mode of DES known as propagating cipher block chaining (PCBC). It has been demonstrated that this mode is vulnerable to an attack involving the interchange of ciphertext blocks. PCBC was intended to provide an integrity check as part of the encryption operation. Version 5 provides explicit integrity mechanisms, allowing the standard CBC mode to be used for encryption. In particular, a checksum or hash code is attached to the message prior to encryption using CBC.

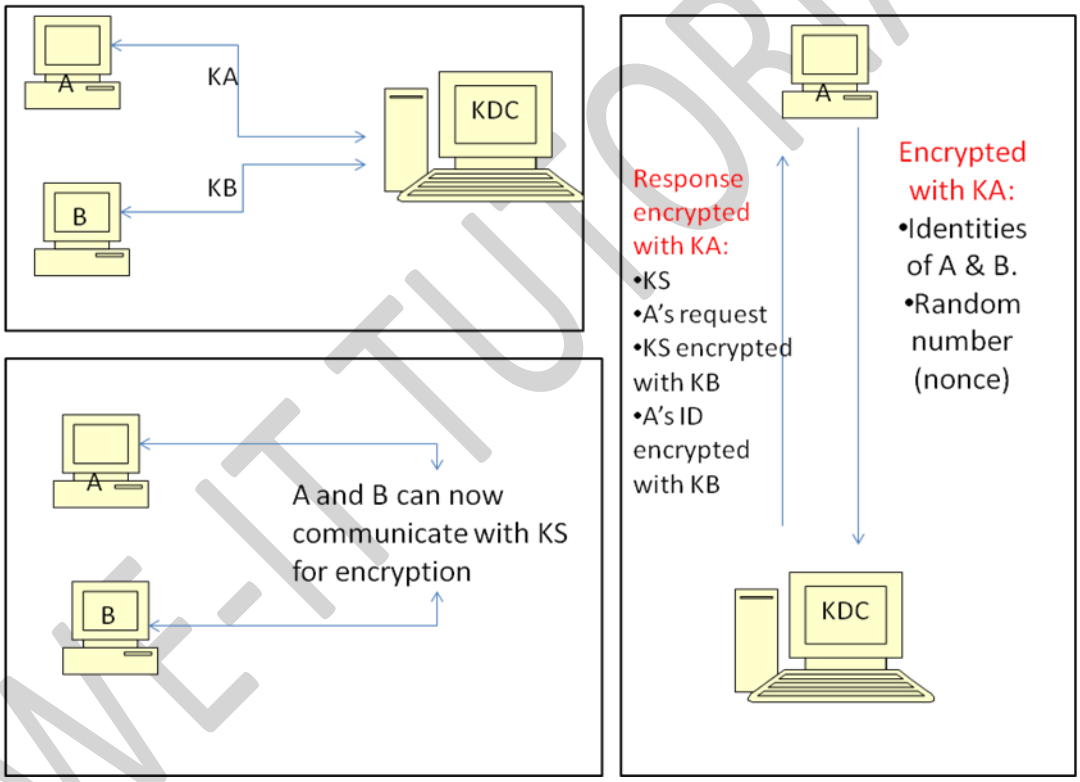
3. **Session keys: T**he same ticket may be used repeatedly to gain service from a particular server, there is the risk that an opponent will replay messages from an old session to the client or the server. In version 5, it is possible for a client and server to negotiate a subsession key, which is to be used only for that one connection. A new access by the client would result in the use of a new subsession key.

4. **Password attacks:** Both versions are vulnerable to a password attack. The message from the AS to the client includes material encrypted with a key based on the client's password. An opponent can capture this message and attempt to decrypt it by trying various passwords. Version 5 does provide a mechanism known as preauthentication, which should make password attacks more difficult, but it does not prevent them.

KEY DISTRIBUTION CENTER (KDC)

Key Distribution Center (KDC) is a central authority dealing with keys for individual computers (nodes) in a computer network. It is similar to the concept of the Authentication Server (AS) and Ticket Granting Server (TGS) in Kerberos. The basic idea is that every node shares a unique secret key with the KDC. Whenever user A wants to communicate securely with user B, the following happens:

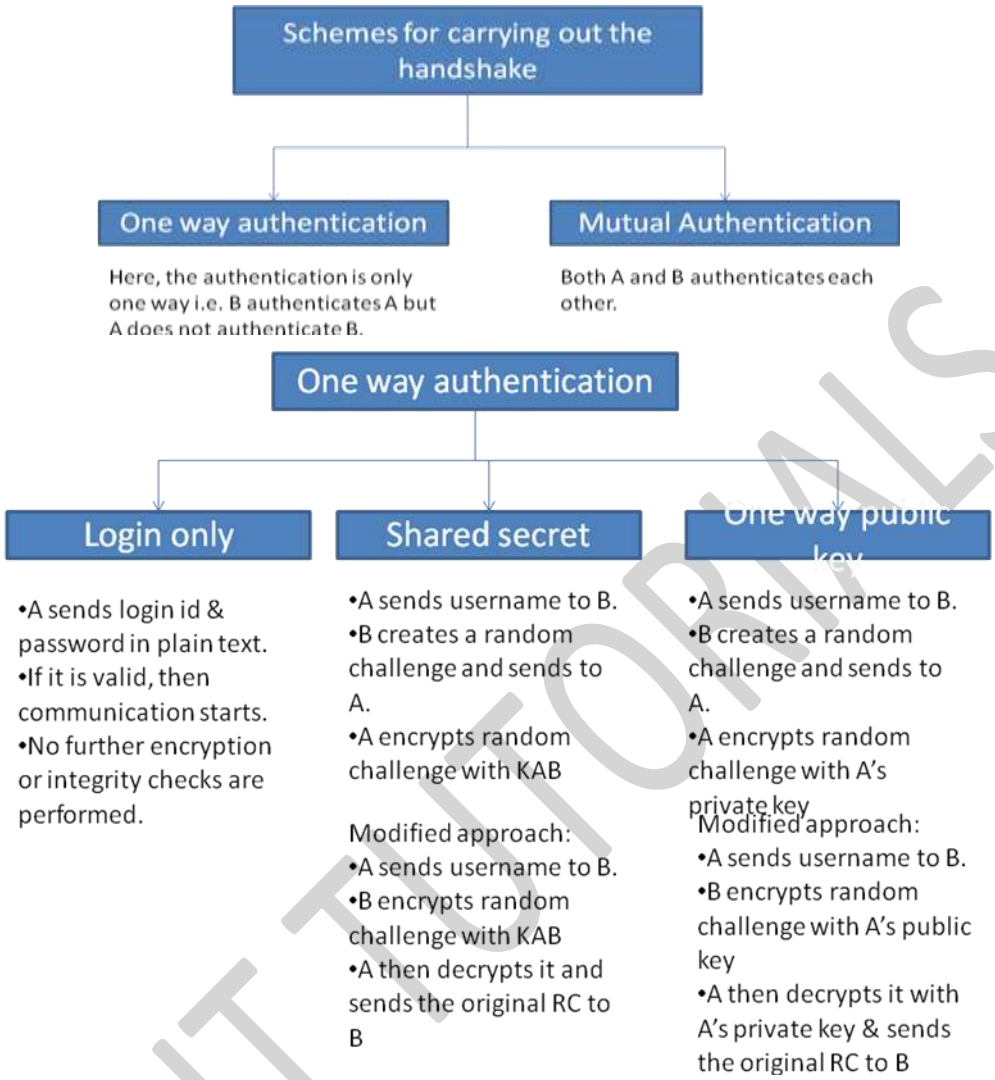
1. The background is that *A* has a shared secret key KA with KDC. Similarly, *B* is assumed to share a secret key KB with the KDC.
2. *A* sends a request to KDC encrypted with KA, which includes
   (a) Identities of *A* and *B*
   (b) A random number *R,* called a **nonce**
3. KDC responds with a message encrypted with KA, containing
   (a) One-time symmetric key KS
   (b) Original request that was sent by *A,* for verification
   (c) Plus, *KS* encrypted with *KB* and ID of *A* encrypted with *KB.*
4. *A* and *B* can now communicate by using *KS* for encryption.

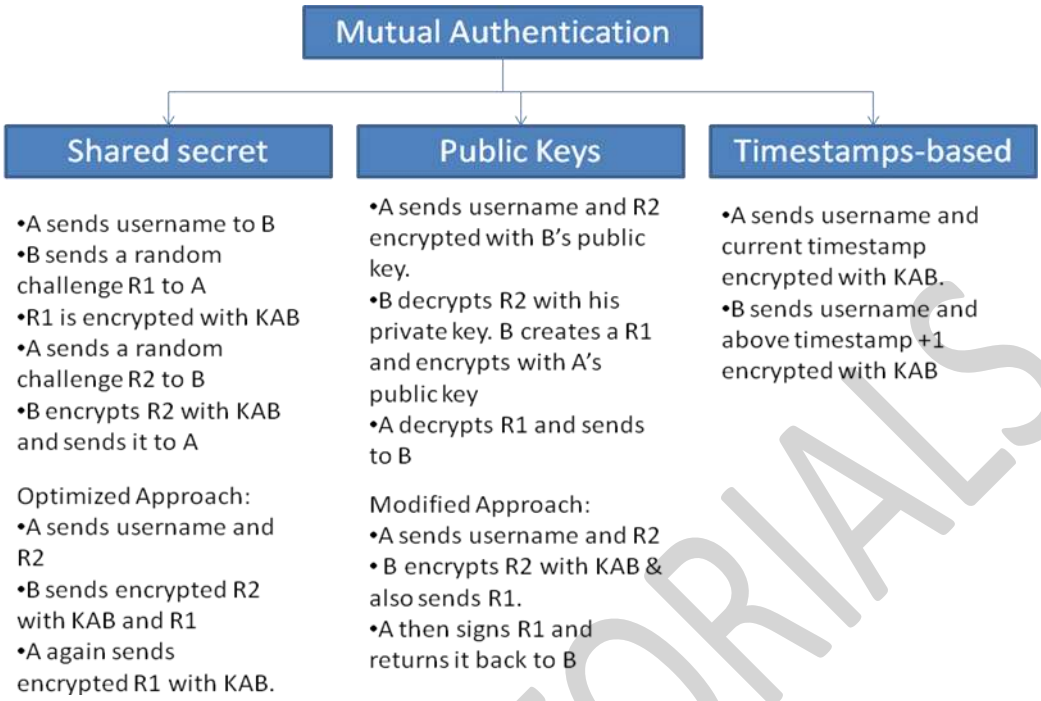


SECURITY HANDSHAKE PITFALLS

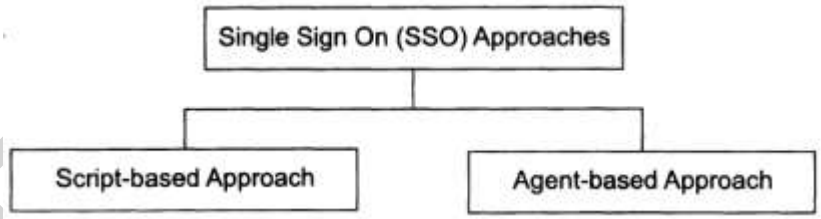
These occur at the stage of the handshake, i.e. when the communicating parties are trying to authenticate each other.

There are two broad-level schemes for carrying out the handshake **one-way authentication** and **mutual authentication.**





SINGLE SIGN ON (SSO) APPROACHES



Scripting

In the scripting technique, the SSO software mimics user actions. It does this by interpreting a program (the script), which simulates the user-depressing keyboard keys, and reacting to individual end-system sign-on prompts. The SSO product itself holds and manages the different sets of authentication information required by the end-systems. It then extracts this information from its database and inserts it in the data stream simulated to be from the user, at the appropriate points. If needed, the script can be programmed to prompt the user to enter information at the appropriate places in the script.

In this approach, batch files and scripts containing authentication information (usually user ids and passwords, along with the login commands, if any) for each application/platform are created. When a user requests an access, a script runs in the background, and performs the same commands/tasks that the user would have to perform. The scripts can contain macros to replay user keystrokes/commands within a

shell. This is easy for users, but quite tough for system administrators, as they have to first play a role in the creation of scripts, have to maintain these scripts securely (as they contain user ids and passwords, etc.) and have to also ensure change coordination when users want to change their passwords.

Agents

In the agent-based approach, every Web server running an application must have a piece of software, called as an agent. Additionally, there is a single SSO server, which interacts with the user database to validate user credentials. Agents interact with the SSO server to achieve single sign on.

Whenever a user wants to access an application/a site participating in SSO, the agent sitting on the particular Web server intercepts the user's HTTP request, and checks for the presence of a cookie. There are two possibilities now.

(a) If the cookie is not present, the agent sends the login page to the user, where the user must enter the SSO user id and password. This login request goes to the SSO server, which validates the user credentials, and if this process is successful, it creates a cookie for the user.

(b) If the cookie is present, the agent opens it, validates its contents, and if they are found OK, allows further processing of the user's request.