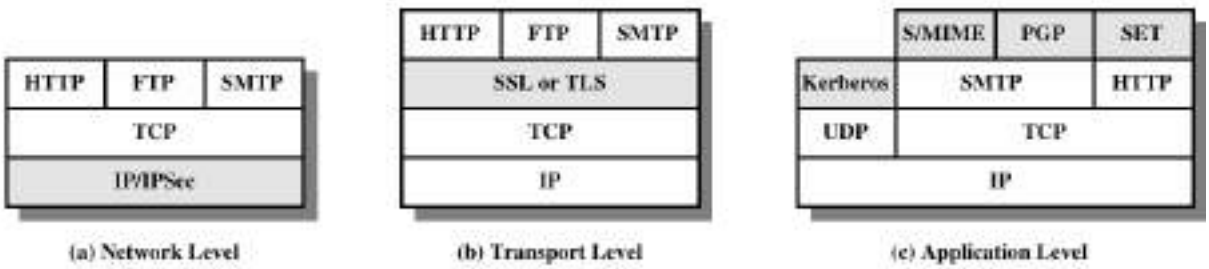
Unit-6: Network Security and Public Key Infrastructure

Overview of Network Security

- Network security is a broad term that covers a multitude of technologies, devices and processes.
- In its simplest term, it is a set of rules and configurations designed to protect the integrity, confidentiality and accessibility of computer networks and data using both software and hardware technologies.
- Network security consists of the policies and practices adopted to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator
- Network Security deals with all aspects related to the protection of the sensitive information assets existing on the network. It covers various mechanisms developed to provide fundamental security services for data communication.

Security at different network layers



Digital Certificates and X.509 certificates

- A digital certificate is a digital form of identification, like a passport. A digital certificate provides information about the identity of an entity.
- A digital certificate is issued by a Certification Authority (CA).
- Examples of trusted CA across the world are Verisign, Entrust, etc. The CA guarantees the validity of the information in the certificate.

❖ For analogy, a certificate can be considered as the ID card issued to the person. People use ID cards such as a driver's license, passport to prove their identity. A digital certificate does the same basic thing in the electronic world, but with one difference.

- Digital Certificates are not only issued to people but they can be issued to computers, software packages or anything else that need to prove the identity in the electronic world.

▪ Digital certificates are based on the ITU standard X.509 which defines a standard certificate format for public key certificates and certification validation. Hence digital certificates are sometimes also referred to as X.509 certificates.

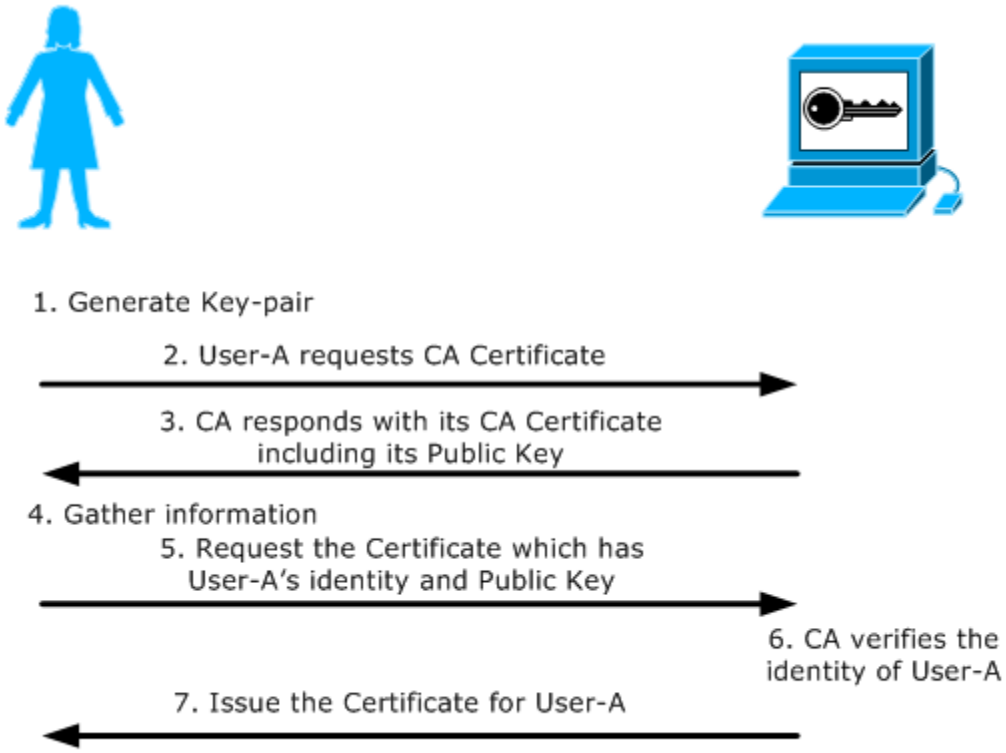
▪ Public key pertaining to the user client is stored in digital certificates by The Certification Authority (CA) along with other relevant information such as client information, expiration date, usage, issuer etc.

❖ CA digitally signs this entire information and includes digital signature in the certificate.

❖ Anyone who needs the assurance about the public key and associated information of client, he carries out the signature validation process using CA's public key. Successful validation assures that the public key given in the certificate belongs to the person whose details are given in the certificate.

The Process of Obtaining a Digital Certificate

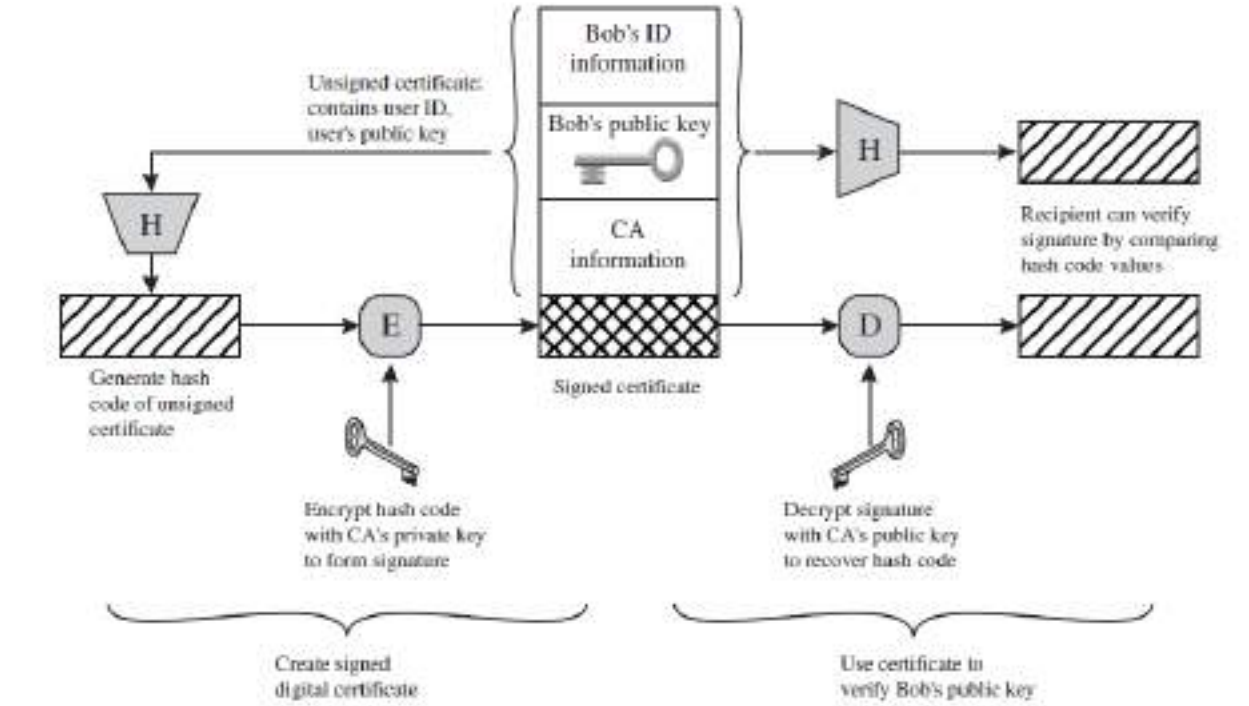
The following diagram shows the process of obtaining a Digital Certificate from a CA.



1. **Generate Key-pair:** User-A generates a Public and Private key-pair or is assigned a key-pair by some authority in their organization.
2. **Request CA Certificate:** User-A first requests the certificate of the CA Server.
3. **CA Certificate Issued:** The CA responds with its Certificate. This includes its Public Key and its Digital Signature signed using its Private Key.
4. **Gather Information:** User-A gathers all information required by the CA Server to obtain its certificate. This information could include User-A email address, fingerprints, etc. that the CA needs to be certain that User-A claims to be who she is.
5. **Send Certificate Request**: User-A sends a certificate request to the CA consisting of her Public Key and additional information. The certificate request is signed by CA's Public Key.
6. **CA verifies User-A:** The CA gets the certificate request, verifies User-A's identity and generates a certificate for User-A, binding her identity and her Public Key. The signature of CA verifies the authenticity of the Certificate.
7. **CA issues the Certificate:** The CA issues the certificate to User-A.

X.509 Certificates

- ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service.
- The directory is, in effect, a server or distributed set of servers that maintains a database of information about users.
- The information includes a mapping from user name to network address, as well as other attributes and information about the users.
- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users.
- The directory may serve as a repository of public-key certificates
- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.
- **X.509 is based on the use of public-key cryptography and digital signatures**. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function. Again, the standard does not dictate a specific hash algorithm. The 1988 recommendation included the description of a recommended hash algorithm; this algorithm has since been shown to be insecure and was dropped from the 1993 recommendation.
- Following figure illustrates the generation of a public-key certificate.



- X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts.

Certificate

- The heart of the X.509 scheme is the public-key certificate associated with each user.
- These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.
- The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.
- Following Figure shows the general format of a certificate, which includes the following elements

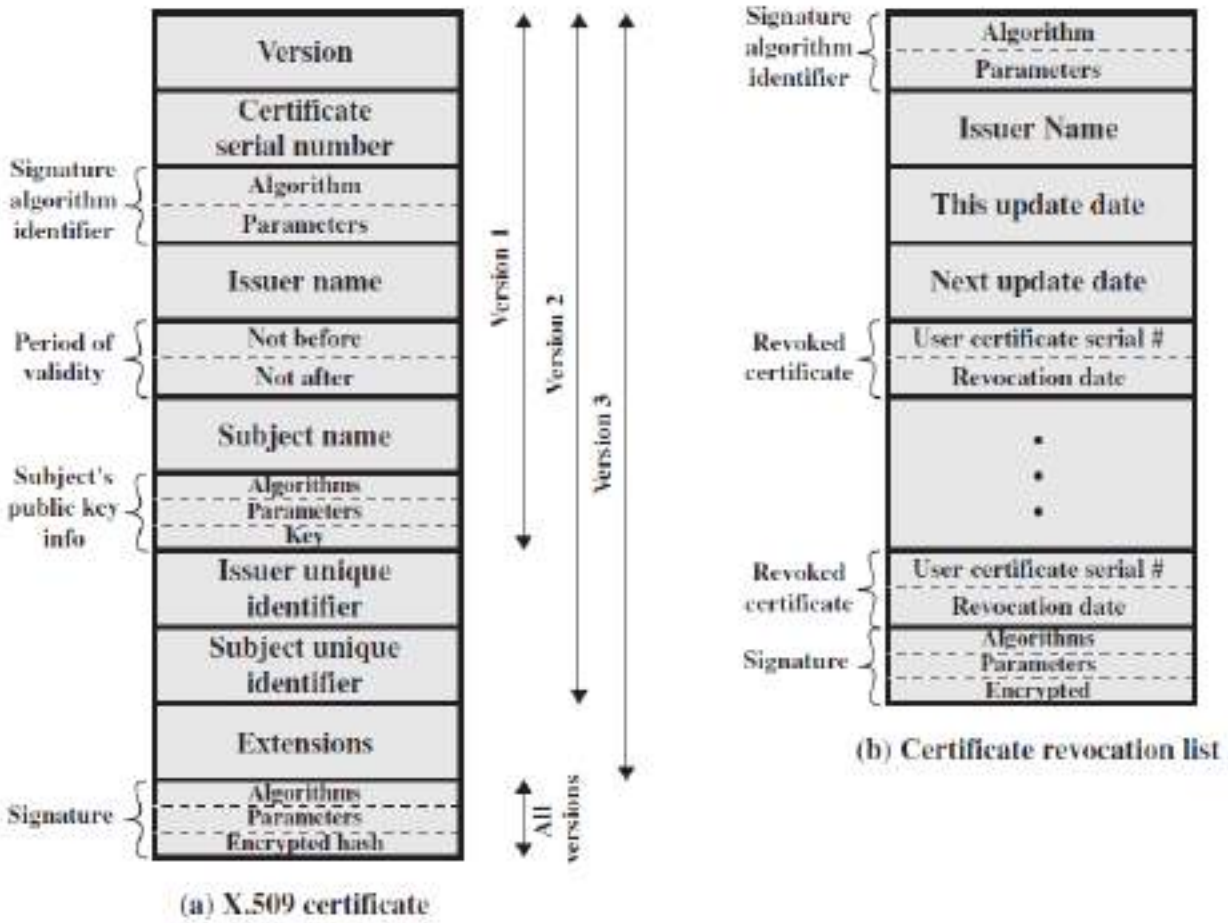


Figure : X.509 Formats

- **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the *issuer unique identifier* or *subject unique identifier* are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
- **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name:** X.500 name of the CA that created and signed this certificate.

- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier:** An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.

User certificates generated by a CA have the following characteristics:

- Any user with access to the public key of the CA can verify the user public key that was certified.
- No party other than the certification authority can modify the certificate without this being detected

Revocation of Certificates

- Each certificate includes a period of validity, much like a credit card. Typically, a new certificate is issued just before the expiration of the old one. In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons.
    1. The user's private key is assumed to be compromised.
    2. The user is no longer certified by this CA. Reasons for this include that the subject's name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA's policies.

3. The CA's certificate is assumed to be compromised

- Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. These lists should also be posted on the directory

Certificate Life Cycle Management

- All digital certificates have a finite lifespan and are no longer recognized as valid upon expiration. Certificates may have varying periods of validity. Minimally, certificates need to be replaced at the end of their life to avoid service disruption and decreased security.
- Organizations without proper certificate lifecycle management can face security and management gaps such as multiple and disjointed authorization mechanisms, and certificates that get lost in the system, expire and cause lost revenue and reputation.
- The **life cycle of a certificate can be broken into following distinct stages, as discussed in the following sections:**

❖ **Certificate Enrollment**

Certificate enrollment is initiated by a user request to the appropriate CA. This is a cooperative process between a user (or a user's PKI software, such as an e-mail or Web browser application) and the CA. The enrollment request contains the public key and enrollment information. Once a user requests a certificate, the CA verifies information based on its established policy rules, creates the certificate, posts the certificate, and then sends an identifying certificate to the user. During the certificate distribution the CA sets policies that affect the use of the certificate.

❖ **Certificate Validation**

When a certificate is used, the certificate status is checked to verify that the certificate is still operationally valid. (Certificate validation is also known as status checking.) During the validation process, the CA checks the status of the certificate and verifies that the certificate is not on a CRL (Certificate Revocation List)

❖ **Certificate Revocation**

A certificate issued by a CA includes an expiration date that defines how long the certificate is valid. If a certificate needs to be revoked before that date, the CA can be instructed to add the certificate to its CRL. Reasons a certificate might need to be revoked include the certificate being lost or compromised, or the person the certificate was issued to leaving the company. When a certificate is revoked, the CA administrator must supply a reason code.

- **Certificate Renewal**

When a certificate reaches its expiration date, and if the certificate policy allows it, it is renewed either automatically, or by user intervention. When renewing a certificate, you must choose whether or not to generate new public and private keys.

- **Certificate Destruction**

When a certificate is no longer in use, the certificate and any backup copies or archived copies of the certificate should be destroyed, along with the private key associated with the certificate. This helps to ensure that the certificate is not compromised and used.

- **Certificate Auditing**

Certificate auditing involves tracking the creation, expiration, and revocation of certificates. In certain instances, it can also track each successful use of a certificate.

PKI

- RFC 4949 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.
- The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys.
- A public key infrastructure (PKI) consists of the components necessary to securely distribute public keys. Ideally, it consists of certificates (Certification Authorities (CAs)), a repository for retrieving certificates, a method of revoking certificates, and a method of evaluating a chain of certificates from public keys that are known and trusted in advance (trust anchors) to the target name.

PKI Trust Models

- Suppose Alice wants to send an encrypted email message to Bob. She needs to securely find out Bob's public key. The PKI trust model defines where Alice gets her trust anchors, and what paths would create a legal chain from a trust anchor to the target name ("Bob" in this example).

1. Monopoly Model

— In this model, the world chooses one organization, universally trusted by all companies, countries, universities, and other organizations to be the single CA for the world.

— The key of that one organization is embedded in all software and hardware as the PKI trust anchor. Everyone must get certificates from it.

— This is a wonderfully simple model, mathematically.

— This is the model favored by organizations hoping to be the monopolist.

— However, there are problems with it:

- There is no one universally trusted organization.
- Given that all software and hardware would come preconfigured with the monopoly organization's key, it would be infeasible to ever change that key in case it were compromised, since that would involve reconfiguration of every piece of equipment and software.
- It would be expensive and insecure to have a remote organization certify your key.
- Once enough software and hardware was deployed so that it would be difficult for the world to switch organizations, the organization would have monopoly control, and could charge whatever it wanted for granting certificates.
- The entire security of the world rests on that one organization never having an incompetent or corrupt employee who might be bribed or tricked into issuing bogus certificates or divulging the CA's private key.

2. Monopoly plus Registration Authorities (RAs)

- This model is just like Monopoly Model except that the single CA chooses other organizations (known as RAs) to securely check identities and obtain and vouch for public keys.
- The RA then securely communicates with the CA, perhaps by sending signed email with the information that would go into the certificate, and the CA can then issue a certificate because it trusts the RA.

- This model's advantage over the Monopoly Model is that it is more convenient and secure to obtain certificates, since there are more places to go to get certified. However, all the other disadvantages of the monopoly model apply.

Note: RAs can be added to any of the models we'll talk about. Some people believe that it is better for their organization to run an RA and leave the operation of the CA to an organization more expert at what it takes to be a CA. However, in practice, the CA just rubber-stamps whatever information is verified by the RAs. It is the RA that has to do the security-sensitive operations of ensuring the proper mapping of name to key. The CA might be better able to provide a tamper-proof audit trail of certificates it has signed.

3. Delegated CAs

- In this model the trust anchor CA can issue certificates to other CAs, vouching for their keys and vouching for their trustworthiness as CAs
- Users can then obtain certificates from one of the delegated CAs instead of having to go to the trust anchor CA.
- The difference between a delegated CA and an RA is whether Alice sees a chain of certificates from a trust anchor to Bob's name, or sees a single certificate. Assuming a monopoly trust anchor, this model has security and operational properties similar to Monopoly plus Registration Authorities (RAs).
- Chains of certificates through delegated CAs can be incorporated into any of the models we'll discuss.

4. Oligarchy

- This is the model commonly used in browsers. In this model, instead of having products preconfigured with a single key, the products come configured with many trust anchors, and a certificate issued by any one of them is accepted.
- Usually in such a model it is possible for the user to examine and edit the list of trust anchors, adding or deleting trust anchors.
- It has the advantage over the monopoly models that the organizations chosen as trust anchors will be in competition with each other, so the world might be spared monopoly pricing.
- However it is likely to be even less secure than the monopoly model:

- — In the monopoly model, if the single organization ever has a corrupt or incompetent employee, the entire security of the world is at risk. In the oligarchy model, though, any of the trust anchor organizations getting compromised will put the security of the world at risk. It is of course far more likely that at least one of n organizations will wind up with a misused key when n is bigger than 1.
- — The trust anchor organizations are trusted by the product vendor, not by the user.
- — Users will not understand the concept of trust anchors.
- — There is no practical way for even a knowledgeable user to be able to examine the set of trust anchors and tell if someone has modified the set.

5. Anarchy Model

- This is the model used by PGP. Each user is responsible for configuring some trust anchors, for instance, public keys of people he has met and who have handed him a business card with a PGP fingerprint (the message digest of the public key), and sent him email containing a public key with that digest.
- Then anyone can sign certificates for anyone else. Some organizations (for instance, MIT does this today) volunteer to keep a certificate database into which anyone can deposit certificates.
- To get the key of someone whose key is not in your set of trust anchors, you can search through the public database to see if you can find a path from one of your trust anchors to the name you want.
- This absolutely eliminates the monopoly pricing, but it is really unworkable on a large scale:
  - — The database would get unworkably large if it were deployed on Internet scale. If every user donated, say, ten certificates, the database would consist of billions of certificates. It would be impractical to search through the database and construct paths.
  - — Assuming somehow Alice could piece together a chain from one of her trust anchors to the name Bob, how would she know whether to trust the chain? So, Carol

(her trust anchor) vouches for Ted's key. Ted vouches for Gail's key. Gail vouches for Ken's key. Ken vouches for Bob's key. Are all these individuals trustworthy?

- As long as this model is used within a small community where all the users are trustworthy, it will work, but on the Internet scale, when there are individuals who will purposely add bogus certificates, it would be impossible to know whether to trust a path.

Note: Some people have suggested that if you can build multiple chains to the name that you can be more assured of the trustworthiness. But once someone decides to add bogus certificates, he can create arbitrary numbers of fictitious identities and arbitrary numbers of certificates signed by those entities. So sheer numbers will not be any assurance of trustworthiness.

PKIX

- The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet
- Figure below shows the interrelationship among the key elements of the PKIX model.

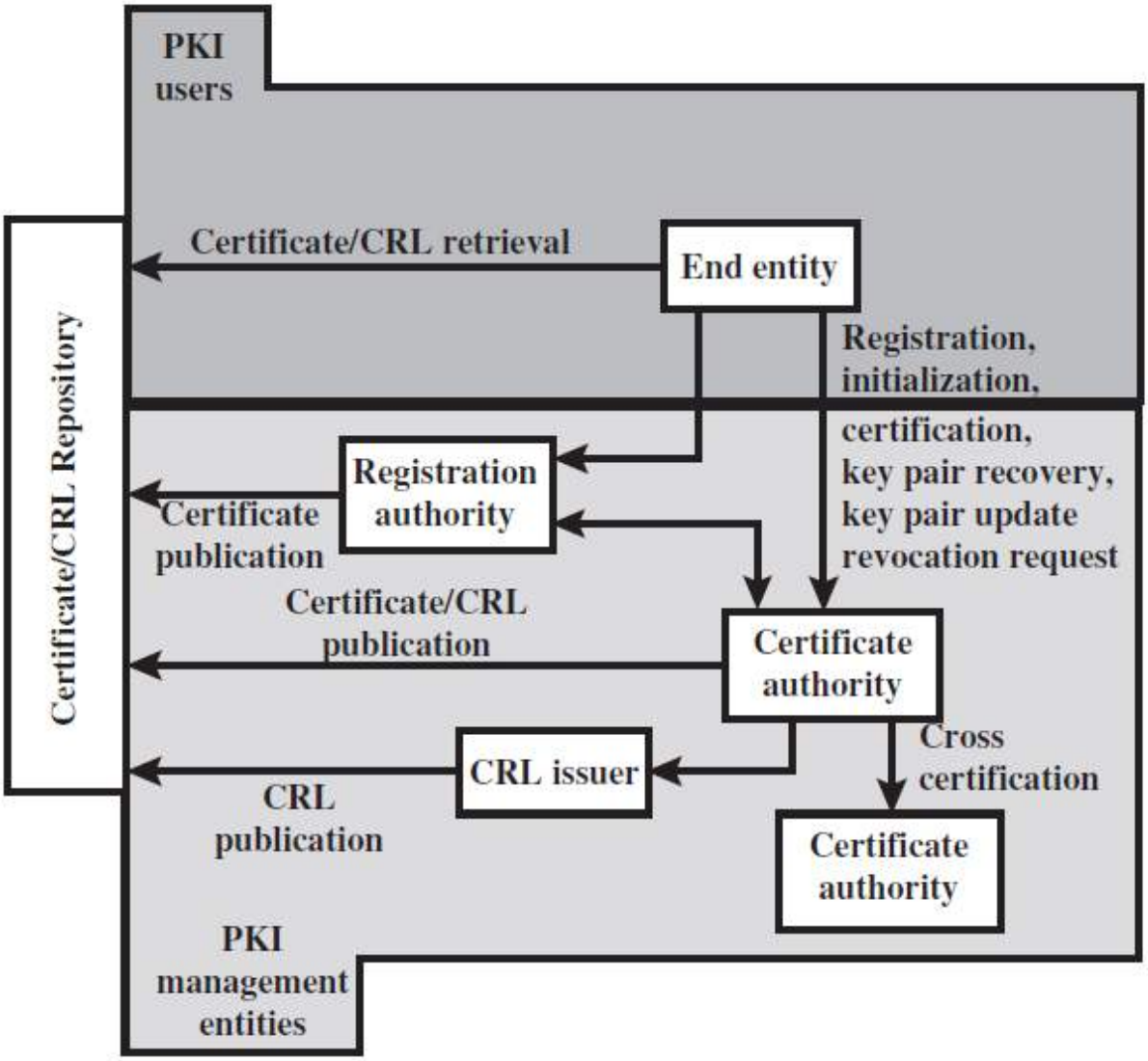


Figure: PKIX Architectural Model

- **End entity:** A generic term used to denote end users, devices (e.g., servers,routers), or any other entity that can be identified in the subject field of a public-key certificate. End entities typically consume and/or support PKI-related services.

- **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.
- **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the end entity registration process but can assist in a number of other areas as well.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

PKIX Management Functions

PKIX identifies a number of management functions that potentially need to be supported by management protocols. These are indicated in above figure and include the following:

- **Registration:** This is the process whereby a user first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate or certificates for that user. Registration begins the process of enrolling in a PKI. Registration usually involves some offline or online procedure for mutual authentication. Typically, the end entity is issued one or more shared secret keys used for subsequent authentication.
- **Initialization:** Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure. For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s), to be used in validating certificate paths.
- **Certification:** This is the process in which a CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository.
- **Key pair recovery:** Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both. When a key pair is used for encryption/decryption, it is important to provide a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible, otherwise it will not be possible to recover the encrypted data. Loss of access to the decryption key

can result from forgotten passwords/ PINs, corrupted disk drives, damage to hardware tokens, and so on. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility (typically, the CA that issued the end entity's certificate).

- **Key pair update**: All key pairs need to be updated regularly (i.e., replaced with a new key pair) and new certificates issued. Update is required when the certificate lifetime expires and as a result of certificate revocation.
- **Revocation request:** An authorized person advises a CA of an abnormal situation requiring certificate revocation. Reasons for revocation include private key compromise, change in affiliation, and name change.
- **Cross certification:** Two CAs exchange information used in establishing a cross-certificate. A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

PKIX Management Protocols

— The PKIX working group has defines two alternative management protocols between PKIX entities that support the management functions listed in the preceding subsection.

— RFC 2510 defines the certificate management protocols (CMP).

  - Within CMP, each of the management functions is explicitly identified by specific protocol exchanges. CMP is designed to be a flexible protocol able to accommodate a variety of technical, operational, and business models.

— RFC 2797 defines certificate management messages over CMS (CMC), where CMS refers to RFC 2630, and cryptographic message syntax. CMC is built on earlier work and is intended to leverage existing implementations. Although all of the PKIX functions are supported, the functions do not all map into specific protocol exchanges.

Email Security

- Email security refers to the collective measures used to secure the access and content of an email account or service. It allows an individual or organization to protect the overall access to one or more email addresses/accounts.
- Email security is a broad term that encompasses multiple techniques used to secure an email service.
- An email service provider implements email security to secure subscriber email accounts and data from hackers - at rest and in transit.
- In virtually all distributed environments, electronic mail is the most heavily used network based application. Users expect to be able to, and do, send e-mail to others who are connected directly or indirectly to the Internet, regardless of host operating system or communications suite.
- With the explosively growing reliance on e-mail, there grows a demand for authentication and confidentiality services.
- Email security uses technology to inspect incoming emails for malicious threats and encrypt--or secure - outbound email traffic to protect mailboxes, data, users, and organizations from cybersecurity attacks and schemes.
- Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and S/MIME (Secure/Multipurpose Internet Mail Extensions).
- **Secure/Multipurpose Internet Mail Extension** (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security.

Pretty Good Privacy (PGP)

- PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
- PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

- In essence, Zimmermann has done the following:
  1. Selected the best available cryptographic algorithms as building blocks.
  2. Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.
  3. Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks such as AOL (America On Line).
  4. Entered into an agreement with a company (Viacrypt, now Network Associates) to provide a fully compatible, low-cost commercial version of PGP.

Why PGP is widely used?

PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

1. It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more. In addition, the commercial version satisfies users who want a product that comes with vendor support.
2. It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.
3. It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet and other networks.
4. It was not developed by, nor is it controlled by, any governmental or standards organization. For those with an instinctive distrust of "the establishment," this makes PGP attractive.
5. PGP is now on an Internet standards track (RFC 3156; MIME Security with OpenPGP). Nevertheless, PGP still has an aura of an antiestablishment endeavor.

Notations:

K_s = session key used in symmetric encryption scheme
PR_a = private key of user A, used in public-key encryption scheme
PU_a = public key of user A, used in public-key encryption scheme
EP = public-key encryption
DP = public-key decryption
EC = symmetric encryption
DC = symmetric decryption
H = hash function
|| = concatenation

Z = compression using ZIP algorithm
R64 = conversion to radix 64 ASCII format[1]

Services provided by PGP

Operational Description

- The actual operation of PGP, as opposed to the management of keys, consists of four services: **authentication, confidentiality, compression, and e-mail compatibility**

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed for storage or transmission using ZIP.
E-mail compatibility	Radix-64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

Authentication (Digital Signature)

- Following figure illustrates the digital signature service provided by PGP.

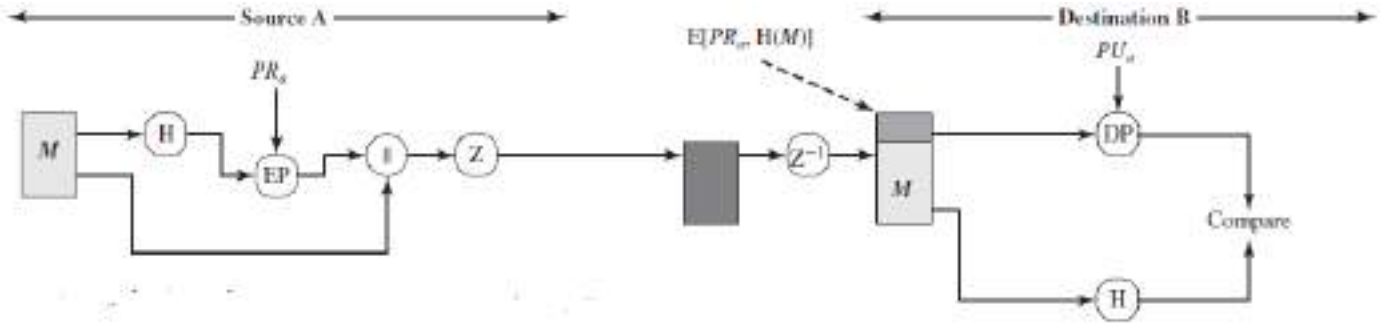


Figure: Authentication Service

- The sequence is as follows.
  1. The sender creates a message.
  2. SHA-1 is used to generate a 160-bit hash code of the message.
  3. The hash code is encrypted with RSA using the sender's private key, and the result is prepended to the message.
  4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
  5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic
- The combination of SHA-1 and RSA provides an effective digital signature scheme.

Confidentiality (Message Encryption)

⇒ Another basic service provided by PGP is confidentiality, which is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the symmetric encryption algorithm CAST-128 may be used.

- ❖ Alternatively, IDEA or 3DES may be used. The 64-bit cipher feedback (CFB) mode is used.

Figure below illustrates the sequence, which can be described as follows.

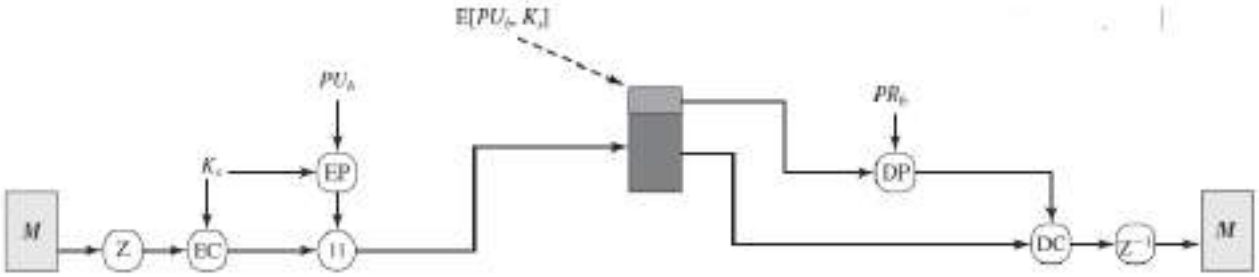


Figure: Confidentiality Service

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

- As an alternative to the use of RSA for key encryption, PGP provides an option referred to as Diffie-Hellman.

Confidentiality and Authentication

As the figure illustrates, both services may be used for the same message.

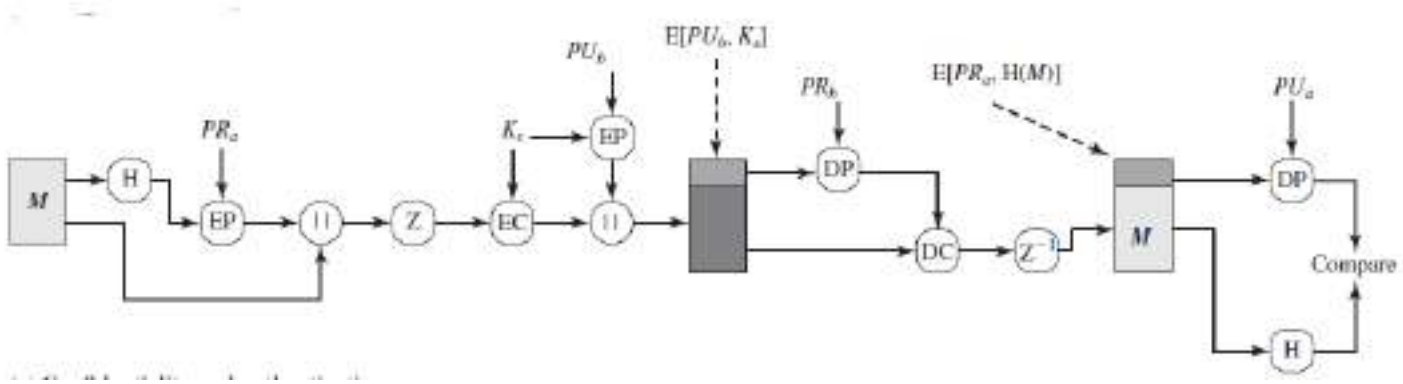


Figure: Confidentiality and Authentication

- First, a signature is generated for the plaintext message and prepended to the message
- Then the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal).
- This sequence is preferable to the opposite: encrypting the message and then generating a signature for the encrypted message.
- It is generally more convenient to store a signature with a plaintext version of a message.
- Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature.
- In summary, when both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and finally encrypts the session key with the recipient's public key.

Compression

- As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.
- The compression algorithm is indicated by Z and Z^{-1} is decompression.
    - The signature is generated before compression
    - Message encryption is applied after compression to strengthen cryptographic security.
    - The compression algorithm used is ZIP

E-mail Compatibility

→ When PGP is used, at least part of the block to be transmitted is encrypted.

→ If only the signature service is used, then the message digest is encrypted (with the sender's private key).

→ If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key).

→ Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets.

→ However, many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters.

→ The scheme used for this purpose is radix-64 conversion. Each group of three octets of binary data is mapped into four ASCII characters. This format also appends a CRC(Cyclic redundancy check) to detect transmission errors.

Transmission and Reception of PGP Messages

Following figure shows **the relationship among the four services** so far discussed.

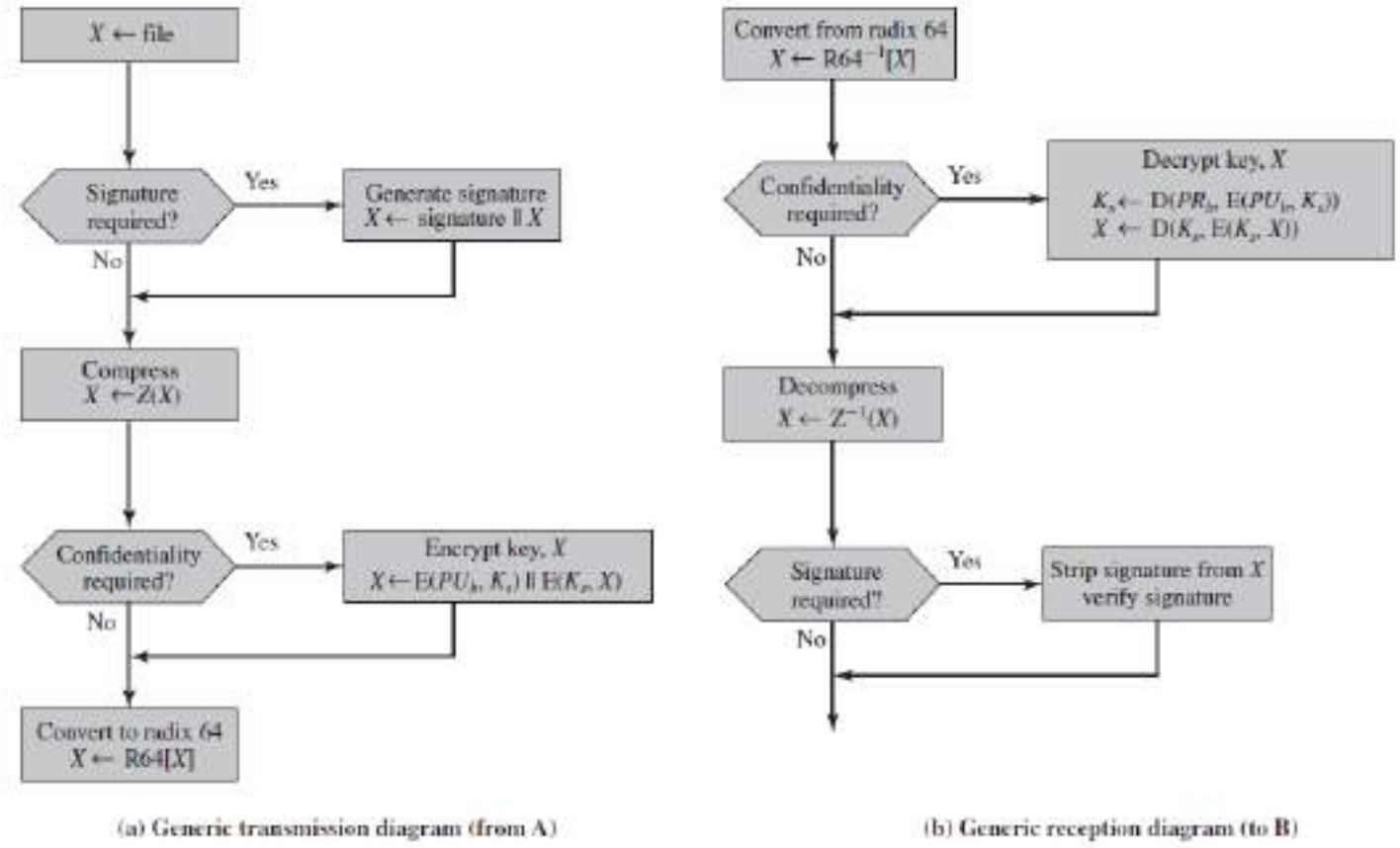


Figure: Transmission and Reception of PGP Messages

→ **On transmission** (if it is required), a signature is generated using a hash code of the uncompressed plaintext.

→ Then the plaintext (plus signature if present) is compressed.

→ Next, if confidentiality is required, the block (compressed plaintext or compressed signature plus plaintext) is encrypted and prepended with the publickey- encrypted symmetric encryption key.

→ Finally, the entire block is converted to radix-64 format.

→ **On reception**, the incoming block is first converted back from radix-64 format to binary.

→ Then, if the message is encrypted, the recipient recovers the session key and decrypts the message.

→ The resulting block is then decompressed. If the message is signed, the recipient recovers the transmitted hash code and compares it to its own calculation of the hash code.

Secure Socket Layer (SSL) Protocol

- One of the most widely used security services is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS).
- SSL is a general-purpose service implemented as a set of protocols that rely on TCP.
- At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications.
- Alternatively, SSL can be embedded in specific packages. For example, most browsers come equipped with SSL, and most Web servers have implemented the protocol.

✓ **Secure Sockets Layer (SSL)** is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser, or a mail server and a mail client (e.g., Outlook).

✓ More specifically, **SSL is a security protocol**. Protocols describe how algorithms should be used. In this case, the SSL protocol determines variables of the encryption for both the link and the data being transmitted.

✓ All browsers have the capability to interact with secured web servers using the SSL protocol. However, the browser and the server need what is called an SSL Certificate to be able to establish a secure connection.

✓ SSL secures millions of peoples' data on the Internet every day, especially during online transactions or when transmitting confidential information. Internet users have come to associate their online security with the lock icon that comes with an SSL-secured website or green address bar that comes with an Extended Validation SSL-secured website. SSL-secured websites also begin with https rather than http.

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, as illustrated in following figure.
  - **The SSL Record Protocol**
  - **Three Higher Layer Protocols** (SSL Handshake Protocol, SSL Change Cipher Spec Protocol, and SSL Alert Protocol)
- **The SSL Record Protocol** provides basic security services to various higher layer protocols.
- In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL.
- **Three higher-layer protocols** are defined as part of SSL: the **Handshake Protocol**, The **Change Cipher Spec Protocol**, and the **Alert Protocol**. These SSL-specific protocols are used in the management of SSL exchanges.

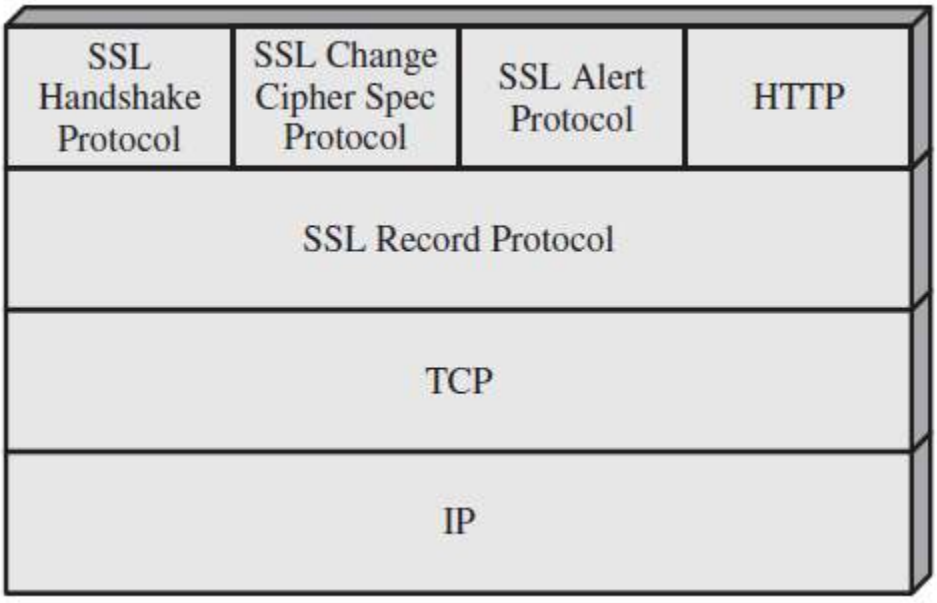


Figure: SSL Protocol Stack

- Two important SSL concepts are the **SSL session** and the **SSL connection**, which are defined in the specification as follows.
    - **Connection:** A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
    - **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

SSL Record Protocol

- The SSL Record Protocol provides two services for SSL connections:
    - **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
    - **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).
- Figure blow indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users

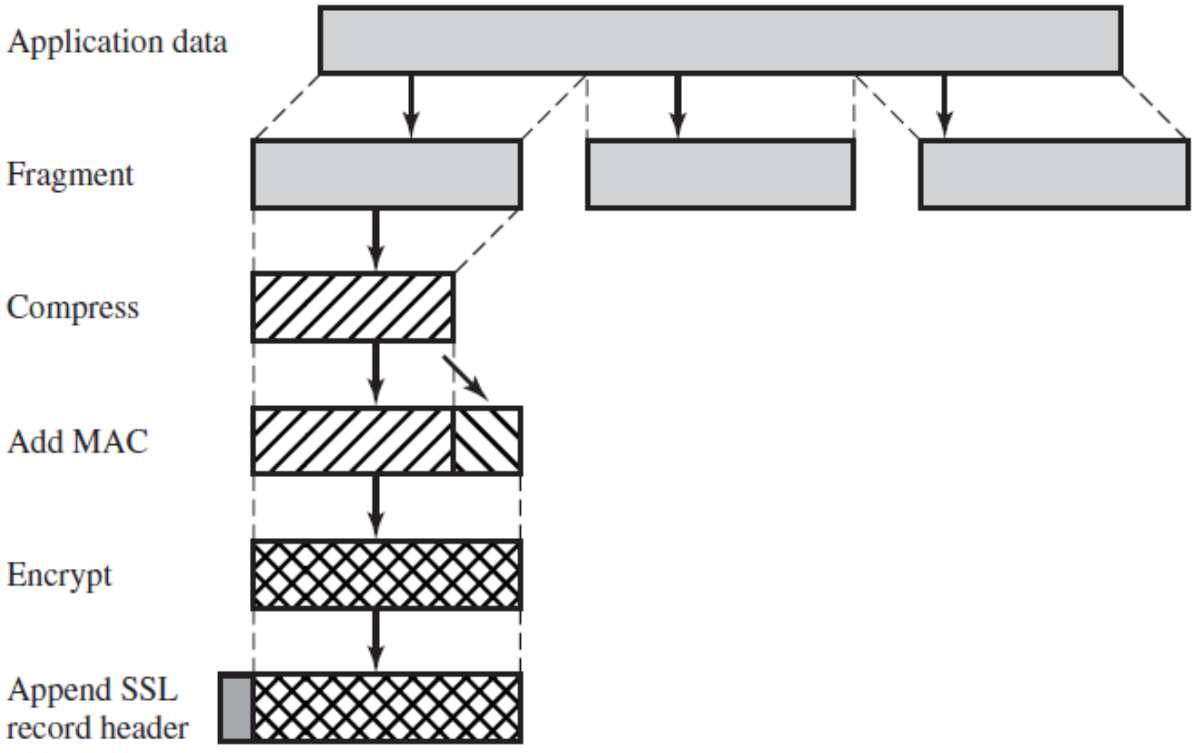


Figure: SSL Record Protocol Operation

(See the text book for detail explanation)

Following figure shows the SSL Record Format

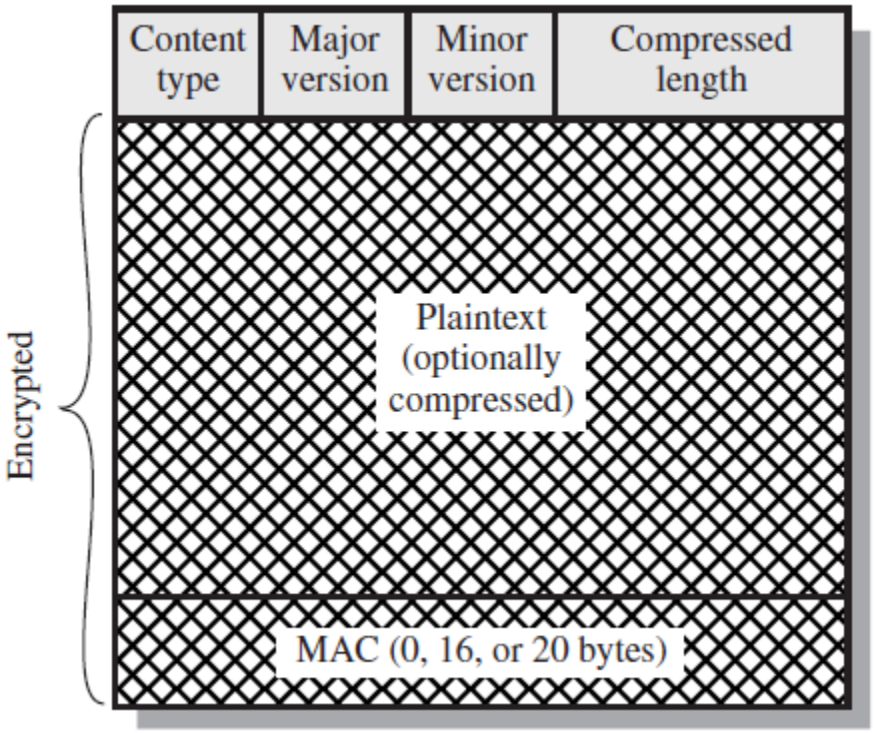


Figure: SSL Record Format

Supported Encryption Algorithms

Block Cipher		Stream Cipher	
Algorithm	**Key Size**	**Algorithm**	**Key Size**
AES	128, 256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Change Cipher Spec Protocol

- The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. 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 into the current state, which updates the cipher suite to be used on this connection.

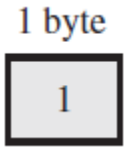


Figure: Change Cipher Spec Protocol Payload

Alert Protocol

- The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.
- Each message in this protocol consists of two bytes (Figure blow). The first byte takes the value warning (1) or fatal (2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert.

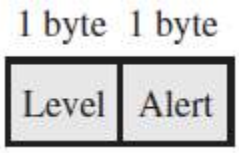


Figure: Alert Protocol Payload

(See the text book for list of fatal and other allerts)

Handshake Protocol

- The most complex part of SSL is the 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. All of these have the format shown in following figure.

Figure: Handshake Protocol Payload

- Each message has three fields:
    - **Type (1 byte):** Indicates one of ten messages.
    - **Length (3 bytes):** The length of the message in bytes.
    - **Content (# 0 bytes):** The parameters associated with this message.

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Following figure shows the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases

Phase 1. Establish Security Capabilities

Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

Phase 2. Server Authentication and Key Exchange

Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3. Client Authentication and Key Exchange

Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4. Finish

Change cipher suite and finish handshake protocol.

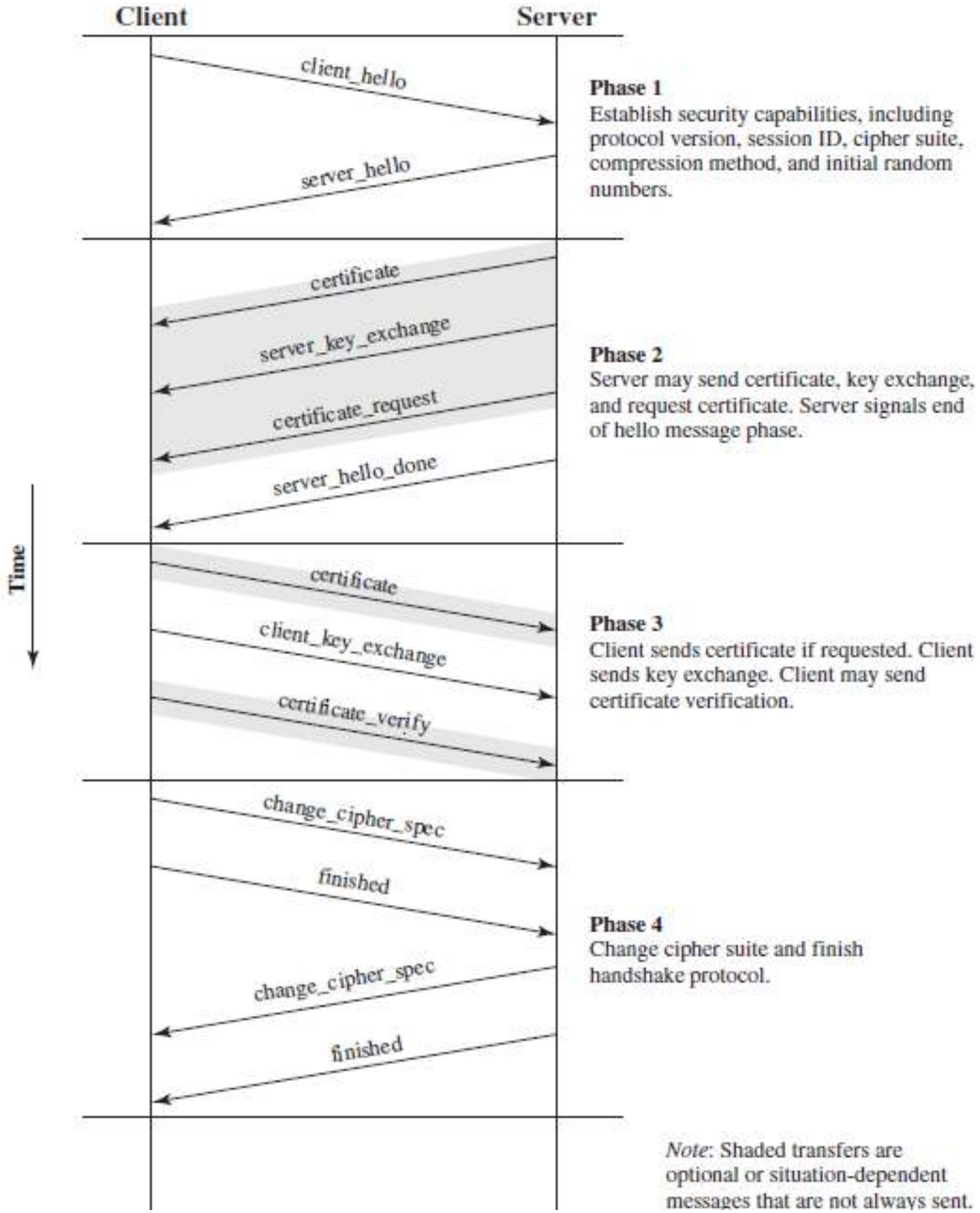


Figure: Handshake Protocol Action

History

- The Secure Sockets Layer (SSL) protocol was first introduced by Netscape in 1994. Version 1.0 of SSL was never released because it had serious security flaws.
- The first official release of SSL, version 2.0, was out in 1995.
- The final version of the SSL protocol, SSL 3.0, was released in November 1996.
- In 2011, the Internet Engineering Task Force (IETF) announced that SSL version 2.0 is deprecated. IETF recommended SSL v2 to be completely abandoned because according to a document that they released (RFC 6176) the protocol has several major deficiencies.. In June 2015, IETF also announced that SSL 3.0 is deprecated.
- As stated in a document released by IETF (RFC 7568), any TLS version is more secure than all versions of SSL.
- The Transport Layer Security (TLS) protocol was first introduced in 1999 as an upgrade to SSL v3.
- The TLS 1.0 RFC document (RFC 2246) document states that the differences between TLS 1.0 and SSL 3.0 are not dramatic, but they are significant enough to preclude interoperability.
- TLS 1.1 (RFC 4346) was a minor update to TLS 1.0 released in April 2006.
- TLS 1.2 (RFC 5246) was released in August 2008. Changes included adding cipher-suite-specified pseudorandom functions (PRFs), adding AES cipher suites, removing IDEA and DES cipher suites, and several other enhancements.
- TLS 1.3, was released in August 2018 (RFC 8446). It took IETF 10 years and 28 drafts to complete.
- All modern browsers support TLS v1.3.

Transport Layer Security (TLS) Protocol

⇒ TLS is a cryptographic protocol that provides end-to-end communications security over networks and is widely used for internet communications and online transactions

⇒ TLS is an IETF(Internet Engineering Task Force) standardization initiative whose goal is to produce an Internet standard version of SSL

⇒ It is an IETF standard intended to prevent eavesdropping, tampering and message forgery. Common applications that employ TLS include Web browsers, instant messaging, e-mail and voice over IP.

o TLS very similar to SSLv3 with following highlighted differences:

Version Number

- **The TLS Record Format** is the same as that of the SSL Record Format (Figure 17.4), and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the major version is 3 and the minor version is 3.

Message Authentication Code

o There are two differences between the SSLv3 and TLS MAC schemes: the actual algorithm and the scope of the MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104.

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

where

H = embedded hash function (for TLS, either MD5 or SHA-1)

M = message input to HMAC

K^+ = secret key padded with zeros on the left so that the result is equal to the block length of the hash code (for MD5 and SHA-1, block length = 512 bits)

ipad = 00110110 (36 in hexadecimal) repeated 64 times (512 bits)

opad = 01011100 (5C in hexadecimal) repeated 64 times (512 bits)

SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. The level of security should be about the same in both cases.

- For TLS, the MAC calculation encompasses the fields indicated in the following expression:

```
MAC(MAC_write_secret,seq_num || TLSCompressed.type ||
    TLSCompressed.version || TLSCompressed.length ||
    TLSCompressed.fragment)
```

The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus the field **TLSCompressed.version**, which is the version of the protocol being employed.

Pseudorandom Function

⇒ TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation.

⇒ The objective is to make use of a relatively small shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs

⇒ The PRF is based on the data expansion function given as

```
P_hash(secret, seed)= HMAC_hash(secret,A(1) || seed) ||
                      HMAC_hash(secret,A(2) || seed) ||
                      HMAC_hash(secret,A(3) || seed) || ...
```

where `A()` is defined as

$$A(0) = \text{seed}$$
$$A(i) = \text{HMAC\_hash}(\text{secret}, A(i-1))$$

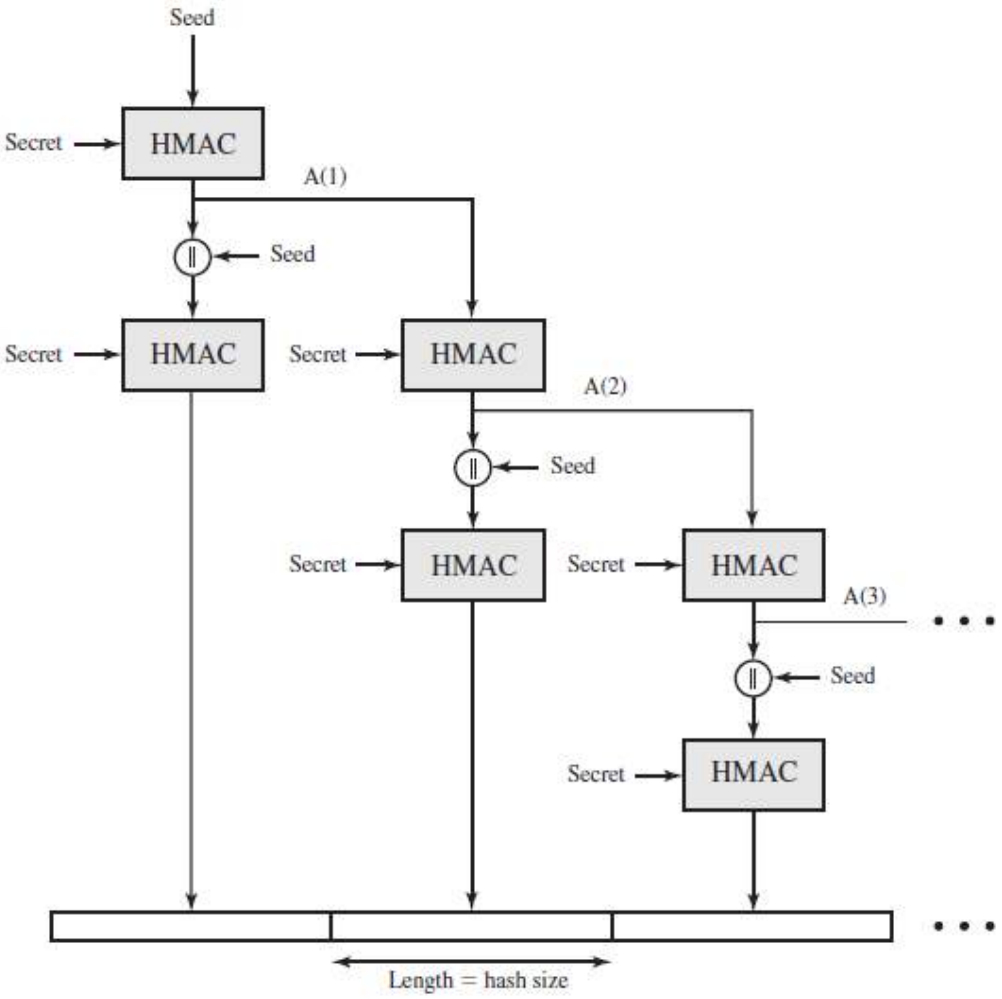


Figure: TLS Function P_hash(secret, seed)

Alert codes

⇒ TLS supports all of the alert codes defined in SSLv3 and supports other more

Cipher Suites

There are several small differences between the cipher suites available under SSLv3 and under TLS:

• **Key Exchange:** TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.

• **Symmetric Encryption Algorithms:** TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza.

Client Certificate Types

⇒ TLS defines the following certificate types to be requested in a *certificate_request* message*: rsa_sign, dss_sign, rsa_fixed_dh*, and *dss_fixed_dh*.

⇒ These are all defined in SSLv3. In addition, SSLv3 includes *rsa_ephemeral_dh, dss_ephemeral_dh*, and *fortezza_kea*.

⇒ Ephemeral Diffie-Hellman involves signing the Diffie-Hellman parameters with either RSA or DSS.

⇒ For TLS, the *rsa_sign* and *dss_sign* types are used for that function; a separate signing type is not needed to sign Diffie-Hellman parameters.

⇒ TLS does not include the Fortezza scheme.

certificate_verify and Finished Messages

⇒ In the TLS *certificate_verify* message, the MD5 and SHA-1 hashes are calculated only over *handshake_messages*. Recall that for SSLv3, the hash calculation also included the master secret and pads. These extra fields were felt to add no additional security.

⇒ As with the finished message in SSLv3, the finished message in TLS is a hash based on the shared *master_secret*, the previous handshake messages, and a label that identifies client or server. The calculation is somewhat different. For TLS, we have

```
PRF(master_secret,finished_label,MD5(handshake_messages)||
    SHA-1(handshake_messages))
```

where *finished_label* is the string “client finished” for the client and “server finished” for the server.

Cryptographic Computations

⇒ The *pre_master_secret* for TLS is calculated in the same way as in SSLv3.

⇒ As in SSLv3, the *master_secret* in TLS is calculated as a hash function of the *pre_master_secret* and the two hello random numbers.

⇒ The form of the TLS calculation is different from that of SSLv3 and is defined as

```
key_block = PRF(master_secret, "key expansion",
            SecurityParameters.server_random ||
            SecurityParameters.client_random)
```

until enough output has been generated.

⇒ As with SSLv3, the *key_block* is a function of the master_secret and the client and server random numbers, but for TLS, the actual algorithm is different.

Padding

⇒ In SSL, the padding added prior to encryption of user data is the minimum amount required so that the total size of the data to be encrypted is a multiple of the cipher's block length.

⇒ In TLS, the padding can be any amount that results in a total that is a multiple of the cipher's block length, up to a maximum of 255 bytes.

⇒ For example, if the plaintext (or compressed text if compression is used) plus MAC plus padding length byte is 79 bytes long, then the padding length (in bytes) can be 1, 9, 17, and so on, up to 249. A variable padding length may be used to frustrate attacks based on an analysis of the lengths of exchanged messages.

IP Security (IPSec) Protocol

⇒ There are application-specific security mechanisms for a number of application areas, including electronic mail (S/MIME, PGP), client/server (Kerberos), Web access (Secure Sockets Layer), and others.

⇒ However, users have security concerns that cut across protocol layers. For example, an enterprise can run a secure, private IP network by disallowing links to untrusted sites, encrypting packets that leave the premises, and authenticating packets that enter the premises. By implementing security at the IP level, an organization can ensure secure networking not only for applications that have security mechanisms but also for the many security-ignorant applications.

⇒ IP-level security encompasses three functional areas: **authentication, confidentiality, and key management.**

- The **authentication** mechanism assures that a received packet was, in fact, transmitted by the party identified as the source in the packet header. In addition, this mechanism assures that the packet has not been altered in transit.
- The **confidentiality** facility enables communicating nodes to encrypt messages to prevent eavesdropping by third parties.
- The **key management** facility is concerned with the secure exchange of keys

The IP security (IPSec) is an Internet Engineering Task Force (IETF) standard suite of protocols between two communication points across the IP network that provide data authentication, integrity, and confidentiality. It also defines the encrypted, decrypted and authenticated packets. The protocols needed for secure key exchange and key management are defined in it.

Applications of IPsec

⇒ IPsec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include:

- **Secure branch office connectivity over the Internet**
  - A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.
- **Secure remote access over the Internet**
  - An end user whose system is equipped with IP security protocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.
- **Establishing extranet and intranet connectivity with partners**
  - IPsec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security**
  - Even though some Web and electronic commerce applications have built-in security protocols, the use of IPsec enhances that security. IPsec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer

⇒ The principal feature of IPsec that enables it to support these varied applications is that it can encrypt and/or authenticate all traffic at the IP level. Thus, all distributed applications (including remote logon, client/server, e-mail, file transfer, Web access, and so on) can be secured.

IP Security Scenario

⇒ Figure below is a typical scenario of IPsec usage.

⇒ An organization maintains LANs at dispersed locations.

⇒ Non secure IP traffic is conducted on each LAN.

⇒ For traffic offsite, through some sort of private or public WAN, IPsec protocols are used.

⇒ These protocols operate in networking devices, such as a router or firewall, that connect each LAN to the outside world.

⇒ The IPsec networking device will typically encrypt and compress all traffic going into the WAN and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN.

⇒ Such user workstations must implement the IPsec protocols to provide security.

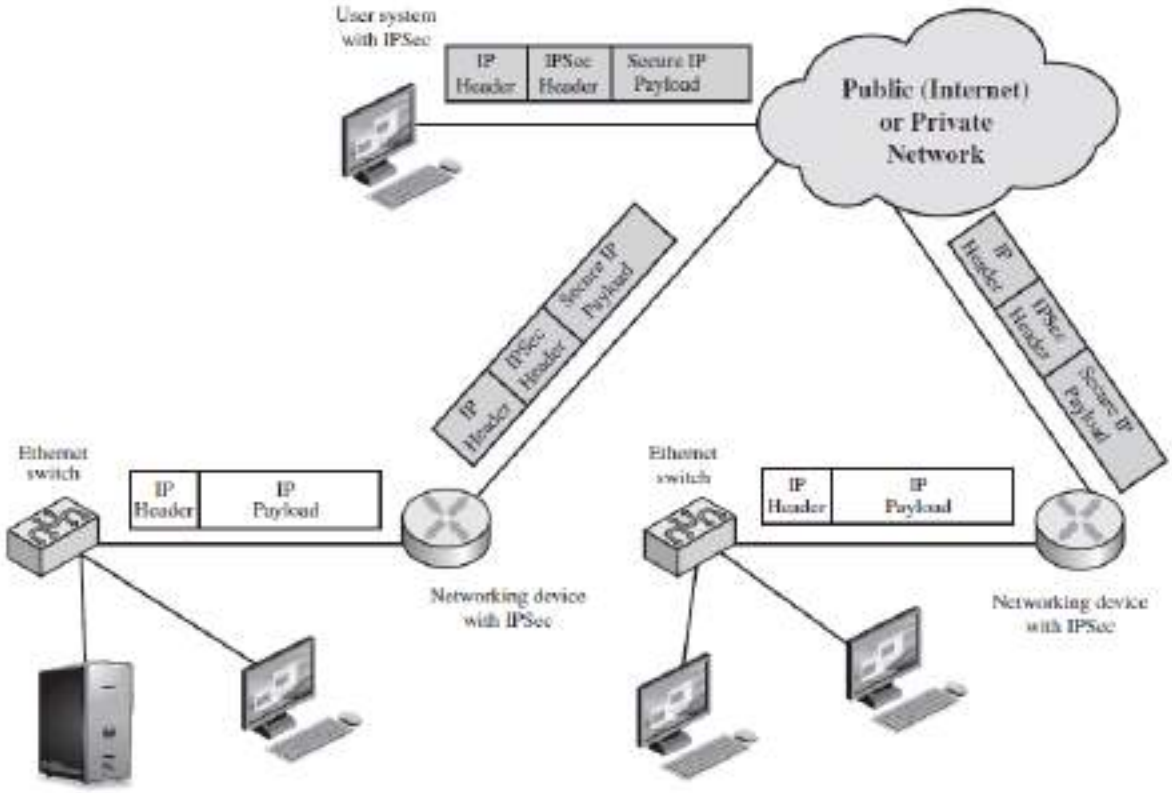


Figure: IP Security Scenario

IPsec Services

⇒ IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services.

⇒ Two protocols are used to provide security:

- an **authentication protocol** designated by the header of the protocol, **Authentication Header (AH);**
- and a combined encryption/ authentication protocol designated by the format of the packet for that protocol, **Encapsulating Security Payload (ESP).**

⇒ RFC 4301 lists the following services:

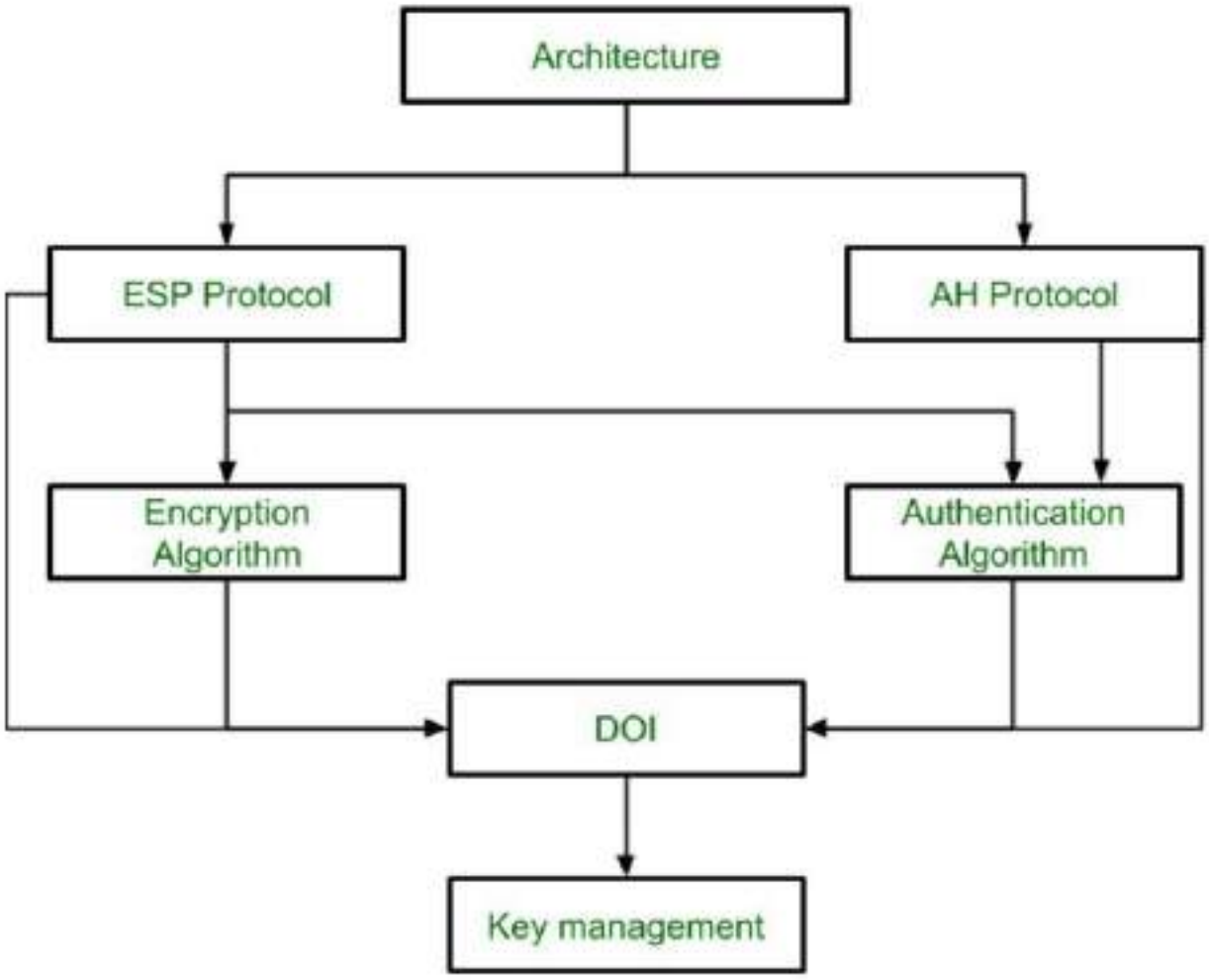
- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

- ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.

IPsec Architecture

— **IPSec** architecture uses two protocols to secure the traffic or data flow. These protocols are **ESP (Encapsulation Security Payload)** and **AH (Authentication Header)**.

— IPSec Architecture include protocols, algorithms, DOI, and Key Management.



Architecture

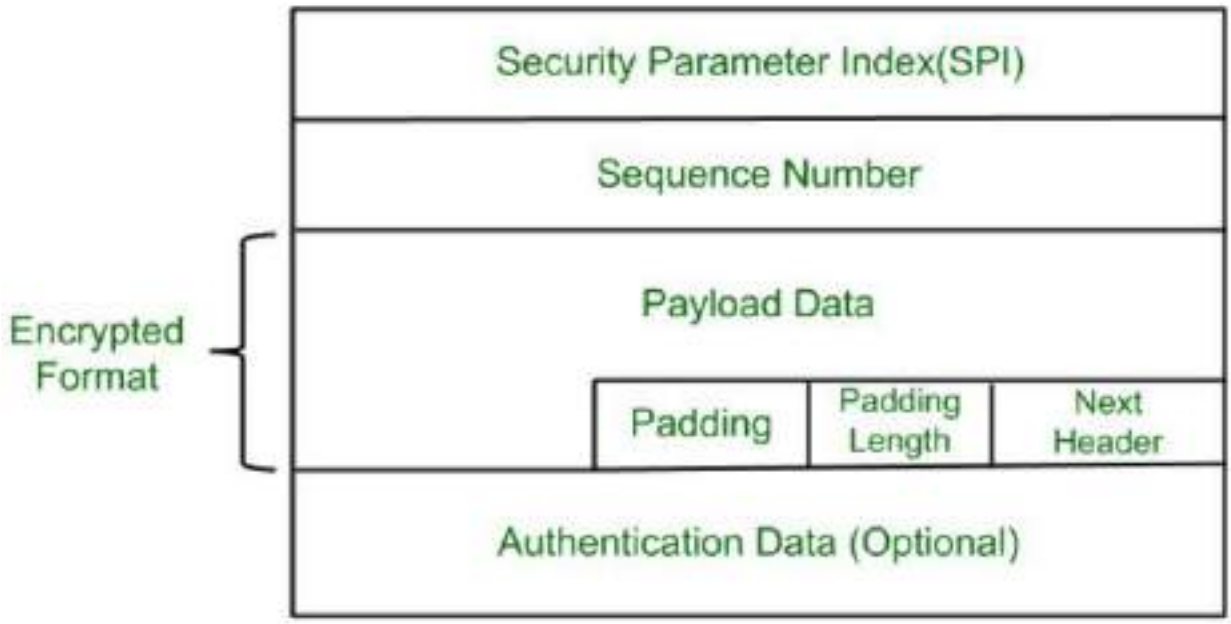
Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms and security requirements of IP Security technology.

ESP Protocol

ESP (Encapsulation Security Payload) provide the confidentiality service. Encapsulation Security Payload is implemented in either two ways:

- ESP with optional Authentication.
- ESP with Authentication.

Packet Format:



- **Security Parameter Index (SPI)**
  This parameter is used in Security Association. It is used to give a unique number to the connection build between Client and Server.
- **Sequence Number**
  Unique Sequence number are allotted to every packet so that at the receiver side packets can be arranged properly.

- **Payload Data**

  Payload data means the actual data or the actual message. The Payload data is in encrypted format to achieve confidentiality.
- **Padding**

  Extra bits or space added to the original message in order to ensure confidentiality. Padding length is the size of the added bits or space in the original message.
- **Next Header**

  Next header means the next payload or next actual data.
- **Authentication Data**

  This field is optional in ESP protocol packet format.

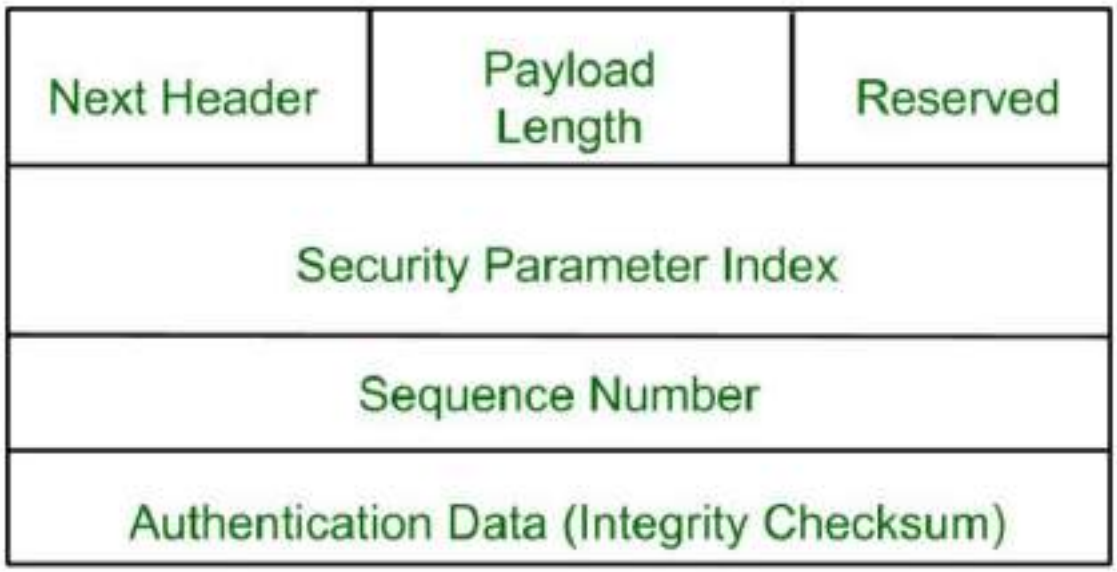
Encryption algorithm

Encryption algorithm is the document that describes various encryption algorithm used for Encapsulation Security Payload.

AH Protocol

AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity

Authentication Header covers the packet format and general issue related to the use of AH for packet authentication and integrity.



Authentication Algorithm

Authentication Algorithm contains the set of the documents that describe authentication algorithm used for AH and for the authentication option of ESP.

DOI (Domain of Interpretation)

DOI is the identifier which support both AH and ESP protocols. It contains values needed for documentation related to each other.

Key Management

Key Management contains the document that describes how the keys are exchanged between sender and receiver.

Firewalls, Firewall Characteristics, Types of Firewalls: Packet filtering firewall, Circuit-level gateway, Stateful inspection firewall, Proxy firewall, Next-generation firewall