UNIT-IV

➢ Digital Signatures and authentication Protocols:

The digital signature must have the following properties:

■ It must verify the author and the date and time of the signature.

■ It must authenticate the contents at the time of the signature.

■ It must be verifiable by third parties, to resolve disputes. Thus, the digital signature function includes the authentication function.

Digital Signature Requirements

On the basis of the properties and attacks just discussed, we can formulate the following requirements for a digital signature.

■ The signature must be a bit pattern that depends on the message being signed.

■ The signature must use some information only known to the sender to prevent both forgery and denial.

■ It must be relatively easy to produce the digital signature.

■ It must be relatively easy to recognize and verify the digital signature.

■ It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.

■ It must be practical to retain a copy of the digital signature in storage.

Direct Digital Signature

The term direct digital signature refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.

Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption). Note that it is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

The validity of the scheme just described depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature. Administrative controls relating to the security of private key scan be employed to thwart or at least weaken this ploy, but the threat is still there, at least to some degree. One example is to require every signed message to include a timestamp (date and time) and to require prompt reporting of compromised keys to a central authority.

Another threat is that a private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

ELGAMAL DIGITAL SIGNATURE SCHEME

Before examining the NIST Digital Signature Algorithm, it will be helpful to under- stand the Elgamal and Schnorr signature schemes. Recall from Chapter 10, that the Elgamal encryption scheme is designed to enable encryption by a user's public key with decryption by the user's private key. The Elgamal signature scheme involves the use of the private key for digital signature generation and the public key for digital signature verification [ELGA84, ELGA85].

Before proceeding, we need a result from number theory. Recall from Chapter 2 that for a prime number q, if a is a primitive root of q, then

$$a, a^2, \subset, a^{q-1}$$

are distinct (mod q). It can be shown that, if a is a primitive root of q, then

1. For any integer m, a^m K 1 (mod q) if and only if m K 0 (mod q - 1).
2. For any integers, i, j, a^i K a^j (mod q) if and only if i K j (mod q - 1).

As with Elgamal encryption, the global elements of **Elgamal digital signature** are a prime number q and a, which is a primitive root of q. User A generates a

private/public key pair as follows.

1. Generate a random integer X_A, such that $1 \leq X_A \leq q - 1$.
2. Compute $Y_A = a^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, a, Y_A\}$.

To sign a message M, user A first computes the hash $m = H(M)$, such that m is an integer in the range $0 \ldots m \ldots q - 1$. A then forms a digital signature as follows.

1. Choose a random integer K such that $1 \ldots K \ldots q - 1$ and $\gcd(K, q - 1) = 1$. That is, K is relatively prime to $q - 1$.
2. Compute $S_1 = a^K \bmod q$. Note that this is the same as the computation of C_1 for Elgamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair (S_1, S_2).

SCHNORR DIGITAL SIGNATURE SCHEME

As with the Elgamal digital signature scheme, the Schnorr signature scheme is based on discrete logarithms [SCHN89, SCHN91]. The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature. The main work for signature generation does not depend on the message and can be done during the idle time of the processor. The message-dependent part of the signature generation requires multiplying a $2n$-bit integer with an n-bit integer.

The scheme is based on using a prime modulus p, with $p - 1$ having a prime factor q of appropriate size; that is, $p - 1 \equiv 0 \pmod q$. Typically, we use $p \approx 2^{1024}$ and $q \approx 2^{160}$. Thus, p is a 1024-bit number, and q is a 160-bit number, which is also
the length of the SHA-1 hash value.

The first part of this scheme is the generation of a private/public key pair, which consists of the following steps.

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $a^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 \leq s \leq q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

NIST DIGITAL SIGNATURE ALGORITHM

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Algorithm (DSA). The DSA makes use of the Secure Hash Algorithm (SHA). The DSA was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2, subsequently updated to FIPS 186-3 in 2009, and FIPS 186-4 in 2013. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

➢ Authentication Protocols

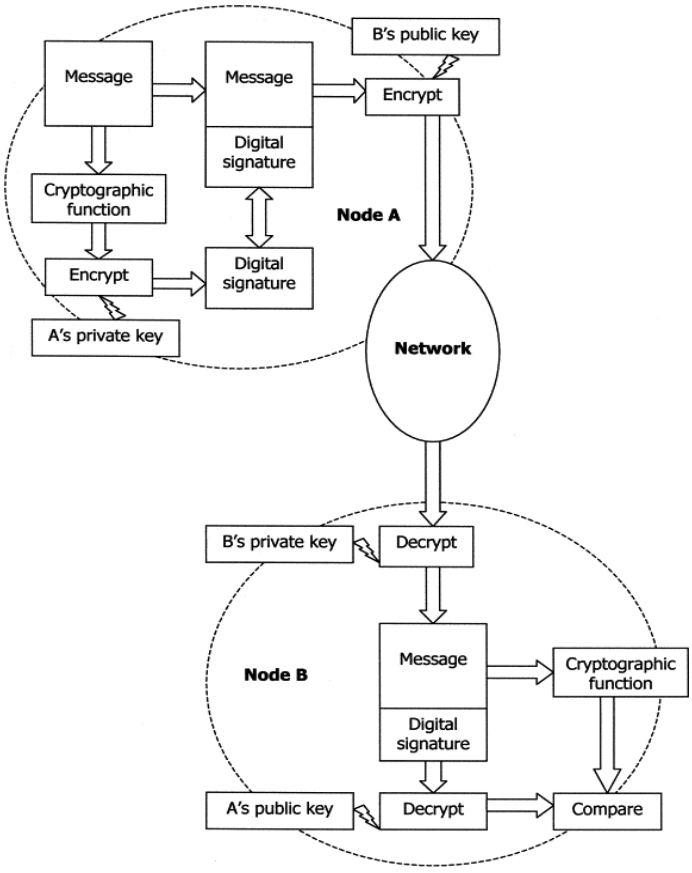
Authentication protocols can be based on shared secret key, public key, key distribution center, or the Kerberos protocol. The protocol based on shared secret key requires users A and B to share a secret key in order to use the protocol. The protocol consists of five message exchanges. A first sends a communication initiation message to B. B does not know whether this message is from A or from an intruder, so B sends a very large random number to A. To prove its identity, A then will encrypt the number with the shared secret key and return it to B. B will decrypt the message to obtain the original number back. Because only A and B know the secret key, B now knows that the message is coming from

A. Next, A sends a challenge (large random number) to B, B encrypts the number with the secret key and sends it back to A, and A decrypts the number to find out whether the message actually came from B. After this, the real communication can start. A problem with the secret key authentication is the secure distribution of the secret key.

The public key authentication protocol uses two keys per node, a public key for encryption and a private key for decryption. Everybody has access to the public key of a node, while the private key is secret. During authentication, random numbers are generated and exchanged, similar to the shared secret key protocol. The only difference is that the public key of the receiving node is used by the sending node to encrypt the random number, while the secret key of the receiving node is used to decrypt the received number. A disadvantage of this protocol is the non-trivial distribution of the public keys.

Another authentication method uses trusted key distribution centers (KDC). Each user has only one key that is shared with the distribution center. Whenever A wants to communicate with B, it generates a session key, encrypts the key with its own secret key, and sends it to the distribution center. The center knows A's secret key and is able to decrypt the session key. It then encrypts the session key with B's secret key and sends it to B, which is able to decrypt the session key again. The session key is then used for secure communication between A and B. To avoid replay attacks where intruders copy messages and resend them at a later time, time stamps or unique numbers are included in the messages to detect the message resending.

The Kerberos authentication protocol consists of a set of two additional servers, the authentication server (AS) and the ticket-granting server (TGS). The AS is similar to a key distribution center in that it shares a secret key with each user. To start a communication between users A and B, A contacts the AS. The server will send back a session key KS and a TGS ticket to A, both encrypted with A's secret key. The TGS ticket can later be used as proof that the sender is really A whenever A requests another ticket from the TGS server. A sends the request to communicate with B to the TGS. This request contains KS and the TGS ticket that A received from the AS. This ticket was encrypted by the AS using a secret TGS key. By decrypting the ticket, the TGS is therefore able to validate that it is communicating with user A. The TGS then creates a session key KAB for the A/B communication and sends two versions back to A: one version encrypted with KS and one version encrypted with B's secret key. A decrypts the first version by using its session key to obtain KAB. It then sends the second version to B, which is also able to decrypt KAB. Now A and B both have the same secret session key KAB to start a secure communication.



Authentication protocols authenticate users only. In many applications, such as financial transactions or e-commerce, messages themselves have to be authenticated as well. Digital signatures were therefore introduced. Digital signatures are used to verify the identity of the sender, to protect against

repudiation of the message by the sender later on, and to detect if a receiver has concocted a message himself. In general, any public key authentication algorithm can be used to produce digital signatures. For example, if user A wants to send a message with a signature to user B, A first generates the signature (by applying a cryptographic function such as a hash function to the message) and encrypts it with his private key and then with B's public key as shown in Fig. 5. After receiving the message and the signature, user B will decrypt the signature first with his private key and then with A's public key. After decrypting the actual message, B can generate a message signature and can compare it with the decrypted signature to verify the message and its sender.

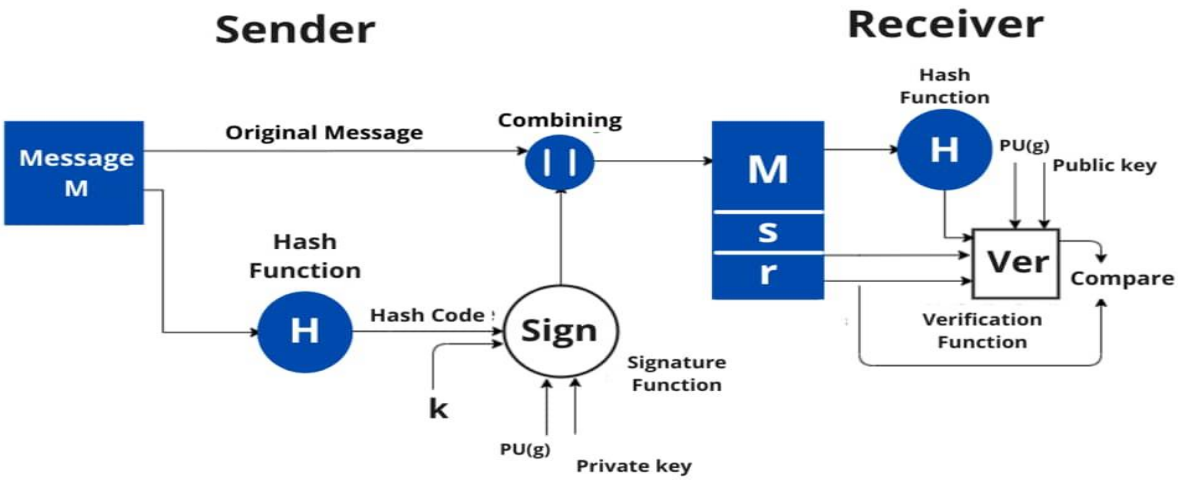
➢ DSS or Digital Signature Standard

It was introduced by the **National Institute of Standards and Technology** (NIST) in 1994. It has become the United States government standard for electronic document authentication. The Federal Information Processing Standard (FIPS) 186 specifies the DSS. It was first proposed in 1991 and revised in 1993as a result of public concerns about the scheme's security.

DSS employs SHA (**Secure Hash Algorithm)** to create digital signatures and offers a new digital signature mechanism known as the Digital Signature Algorithm.

What is the DSS approach?

The DSS is different from the fact that the RSA algorithm uses the public key, private key and hash function whereas the DSS uses the public key, private key, hash function, a random number k, and global public key. Therefore, DSS provides more security than RSA algorithms.



A hash code is generated from the message and given as an input to the signature function on the sender side. The other inputs to a signature function include a unique random number k for the signature, the private key of sender PR(a), and global public key i.e., PU(g).

The output of the signature function consists of two components: s & r, which is concatenated with the input message and then sent to the receiver.

Signature = {s, r}.

At the receiver side, the hash code for the message sent is generated by the receiver by applying a hash function. The verification function is used for verifying the message and signature sent by the sender. The verification function takes hash code generated, signature components s and r, the public key of the sender (PU(a)), and global public key

The signature function is compared with the output of the verification function and if both the values match, the signature is valid because A valid signature can only be generated by the sender using its private key.

Let's see what are the steps involved in DSS.
What are the steps involved in DSS?

- Generation of keys i.e., public and private keys for the source.
- Creation of digital signature by the source for a message.
- The receiver verifies the digital signature.

The DSA (Digital Signature Algorithm)
1. Generation of a global public key component
   - Find a prime number p such that 2^L-1 < p < 2^L, where L is an integer between 512 and 1024 i.e., 512 <= L <= 1024.
   - Find another number q which is a prime divisor of (p-1).
   - Compute: g = h^(p-1)/q mod p, where h is an integer between 1 < h < p-1 and g should greater than 1 or h^(p-1)/q mod p>1.
2. Finding the user private and public keys
   - The private key, x is any random number such that 0 < x< q.
   - The public key, y = g^x mod p
3. Generating the Signature
   - Finding the components of signature s and r.
   - r = (gk mod p) mod q
   - s = [k^ -1 {H(M) + x*r}] mod q where k is an integer such that 0 < k <q.
   - Signature= (r, s)
4. Verifying the signature

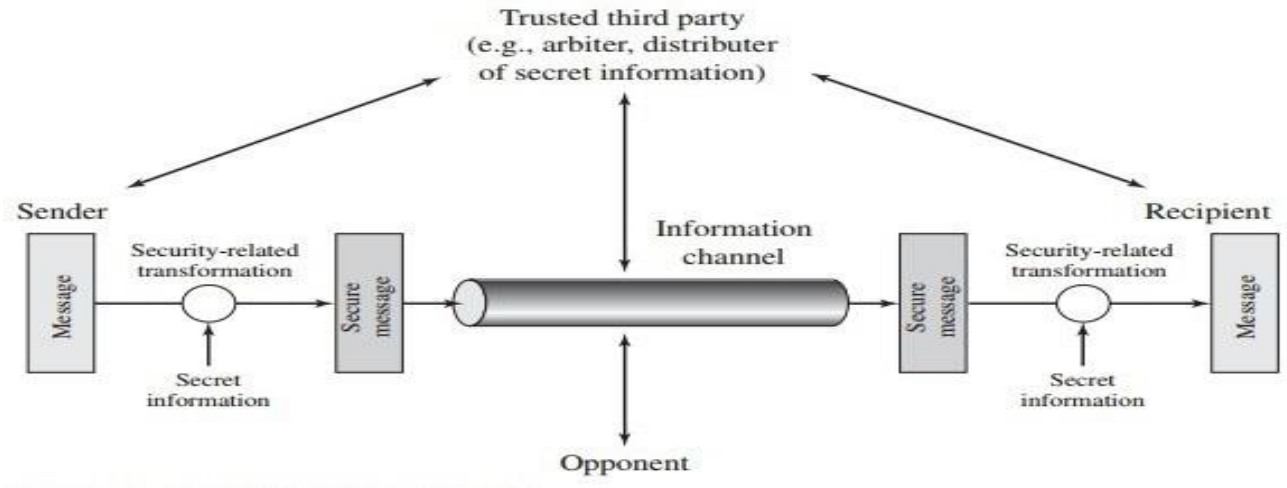
Let M', r', and s' be the message and signature components respectively received corresponding to M, r, and s.
To verify the signature is valid or not, first the following condition needs to be checked:
0 < r' < q and 0 < s' < q
If any of the above conditions is not met, the signature is considered invalid and is therefore discarded; otherwise, if both conditions are met, the following parameters are computed:
v = [(g^u1 y^u2) mod p] mod q
u1 = [H(M)w] mod q
u2 = (r') w mod q
w = (s')^ -1 mod q
Test: v = r'
If v = r', the signature is correct; otherwise, the data or message has been altered.
I hope this module will be useful and you now have a good knowledge of what is **Digital Signature Standard** (DSS) is in Cryptography. Stay tuned for more instructive and engaging modules like this.

➢ Network Security Practice

- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.

- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the



authenticity of a message transmission.

This general model shows that there are four basic tasks in designing a particular securityservice:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

2. Generate the secret information to be used with the algorithm.

3. Develop methods for the distribution and sharing of the secret information.

4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

Parts One through Five of this book concentrate on the types of security mechanisms and services that fit into the model shown in Figure. However, there are other security-related situations of interest that do not neatly fit this model but are considered in this book. A general model of these other situations is illustrated by Figure 1.5, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to

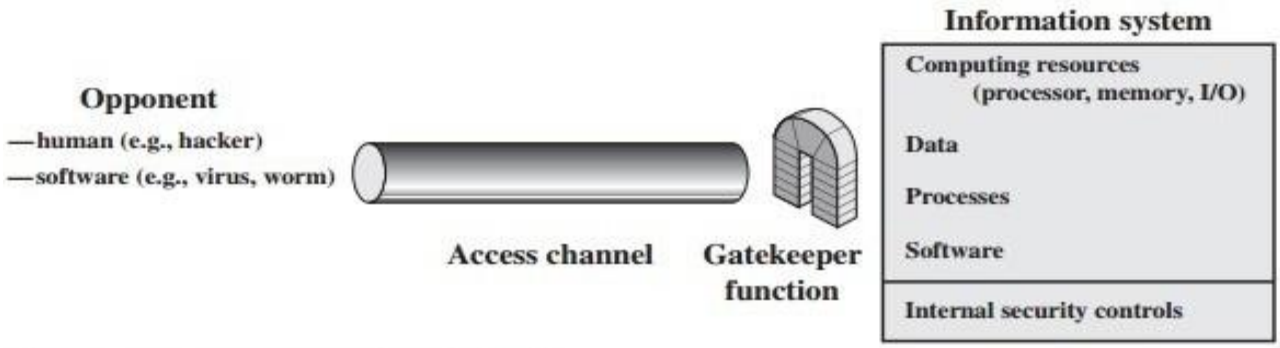


Figure 1.5 Network Access Security Model

penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. The intruder can be a disgruntled employee who wishes to do damage or a criminal who seeks to exploit computer assets for financial gain

Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application pro-grams as well as utility programs,such as editors and compilers. Programs can present two kinds of threats:

- **Information access threats:** Intercept or modify data on behalf of users who shouldnot have access to that data.
- **Service threats:** Exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.

SECURITY MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques.
Encryption or encryption-like transformations of information are the most common means ofproviding security. Some of the mechanisms are

1 **Encipherment**
2 **Digital Signature**
3 **Access Control**

SECURITY SERVIICES

The classification of security services are as follows:

Confidentiality: Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties.

E.g. Printing, displaying and other forms of disclosure.

Authentication: Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

Integrity: Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

Non repudiation: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

Access control: Requires that access to information resources may be controlled by or the targetsystem.

Availability: Requires that computer system assets be available to authorized parties when needed.

➢ Authentication Protocol

An **authentication protocol** is a type of computer communications protocol or cryptographic protocol specifically designed for transfer of authentication data between two entities. It allows the receiving entity to authenticate the connecting entity (e.g. Client connecting to a Server) as well as authenticate itself to the connecting entity (Server to a client) by declaring the type of information needed for authentication as well as syntax. It is the most important layer of protection needed for secure communication within computer networks.

Purpose

With the increasing amount of trustworthy information being accessible over the network, the need for keeping unauthorized persons from access to this data emerged. Stealing someone's identity is easy in the computing world - special verification methods had to be invented to find out whether the person/computer requesting data is really who he says he is. The task of the authentication protocol is to specify the exact series of steps needed for execution of the authentication. It has to comply with the main protocol principles:

1. A Protocol has to involve two or more parties and everyone involved in the protocol must know the protocol in advance.
2. All the included parties have to follow the protocol.
3. A protocol has to be unambiguous - each step must be defined precisely.
4. A protocol must be complete - must include a specified action for every possible situation.

An illustration of password-based authentication using simple authentication protocol:
Alice (an entity wishing to be verified) and Bob (an entity verifying Alice's identity) are both aware of the protocol they agreed on using. Bob has Alice's password stored in a database for comparison.

1. Alice sends Bob her password in a packet complying with the protocol rules.
2. Bob checks the received password against the one stored in his database. Then he sends a packet saying "Authentication successful" or "Authentication failed" based on the result.

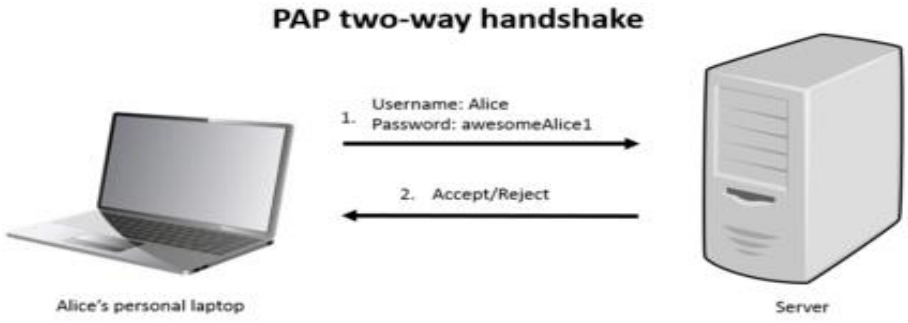
This is an example of a very basic authentication protocol vulnerable to many threats such as eavesdropping, replay attack, man-in-the-middle attacks, dictionary attacks or brute-force attacks. Most authentication protocols are more complicated in order to be resilient against these attacks.
Types

Authentication protocols developed for PPP Point-to-Point Protocol

Protocols are used mainly by Point-to-Point Protocol (PPP) servers to validate the identity of remote clients before granting them access to server data. Most of them use a password as the cornerstone of the authentication. In most cases, the password has to be shared between the communicating entities in advance.

PAP - Password Authentication Protocol

Password Authentication Protocol is one of the oldest authentication protocols. Authentication is initialized by the client sending a packet with credentials (username and password) at the beginning of the connection, with the client repeating the authentication request until acknowledgement is received. It is highly insecure because credentials are sent "in the clear" and repeatedly, making it vulnerable even to the most simple attacks like eavesdropping and man-in-the-middle based attacks. Although widely supported, it is specified that if an implementation offers a stronger authentication method, that method must be offered before PAP. Mixed authentication (e.g. the same client alternately using both PAP and CHAP) is also not expected, as the CHAP authentication would be compromised by PAP sending the password in plain-text.



CHAP - Challenge-handshake authentication protocol

The authentication process in this protocol is always initialized by the server/host and can be performed anytime during the session, even repeatedly. Server sends a random string (usually 128B long). The client uses password and the string received as parameters for MD5 hash function and then sends the result together with username in plain text. Server uses the username to apply the same function and compares the calculated and received hash. An authentication is successful or unsuccessful.

EAP - Extensible Authentication Protocol

EAP was originally developed for PPP(Point-to-Point Protocol) but today is widely used in IEEE 802.3, IEEE 802.11(WiFi) or IEEE 802.16 as a part of IEEE 802.1x authentication framework. The latest version is standardized in RFC 5247. The advantage of EAP is that it is only a general authentication framework for client-server authentication - the specific way of authentication is defined in its many versions called EAP-methods. More than 40 EAP-methods exist, the most common are:

- EAP-MD5
- EAP-TLS
- EAP-TTLS
- EAP-FAST
- EAP-PEAP

AAA architecture protocols (Authentication, Authorization, Accounting)

Complex protocols used in larger networks for verifying the user (Authentication), controlling access to server data (Authorization) and monitoring network resources and information needed for billing of services (Accounting).

TACACS, XTACACS and TACACS+

The oldest AAA protocol using IP based authentication without any encryption (usernames and passwords were transported as plain text). Later version XTACACS (Extended TACACS) added authorization and accounting. Both of these protocols were later replaced by TACACS+. TACACS+ separates the AAA components thus they can be segregated and handled on separate servers (It can even use another protocol for e.g. Authorization). It uses TCP (Transmission Control Protocol) for transport and encrypts the whole packet. TACACS+ is Cisco proprietary.

RADIUS

Remote Authentication Dial-In User Service (RADIUS) is a full AAA protocol commonly used by ISP. Credentials are mostly username-password combination based, it uses NAS and UDP protocol for transport.

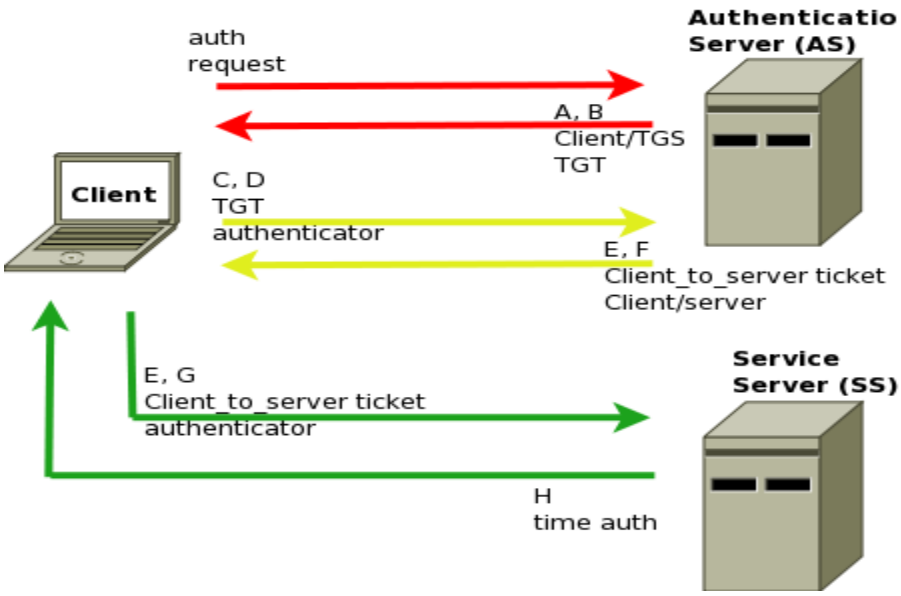
DIAMETER

Diameter (protocol) evolved from RADIUS and involves many improvements such as usage of more reliable TCP or SCTP transport protocol and higher security thanks to TLS.

Other

Kerberos (protocol)

Kerberos is a centralized network authentication system developed at MIT and available as a free implementation from MIT but also in many commercial products. It is the default authentication method in Windows 2000 and later. The authentication process itself is much more complicated than in the previous protocols - Kerberos uses symmetric key cryptography, requires a trusted third party and can use public-key cryptography during certain phases of authentication if need be.



➢ Basic overview of Electronic Mail Security:

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. Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and S/MIME. Both are examined in this chapter. The chapter closes with a discussion of Domain Keys Identified Mail.

➢ PRETTY GOOD PRIVACY

PGP is a remarkable phenomenon. Largely the effort of a single person, 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 (Via crypt, now Network Associates) to provide a fully compatible, low-cost commercial version of PGP. PGP has grown explosively and is now widely used.

The PGP documentation often uses the term *secret key* to refer to a key paired with a public key in a public-key encryption scheme. As was mentioned earlier, this practice risks confusion with a secret key used for symmetric encryption. Hence, we use the term *private key* instead.

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

AUTHENTICATION Figure illustrates the digital signature service provided by PGP.

.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 thehash 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.

CONFIDENTIALITY 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. As always, one must address the problem of key distribution. In PGP, each symmetric key is used only once. That is, a new key is generated as a random 128- bit number for each message. Thus, although this is referred to in the documentation as a session key, it is in reality a one-time key. Because it is to be used only once, the session key is bound to the message and transmitted with it.To protect the key, it is encrypted with the receiver's public key. Figure illustrates the sequence, which canbe described as follows.

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 sessionkey.
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.

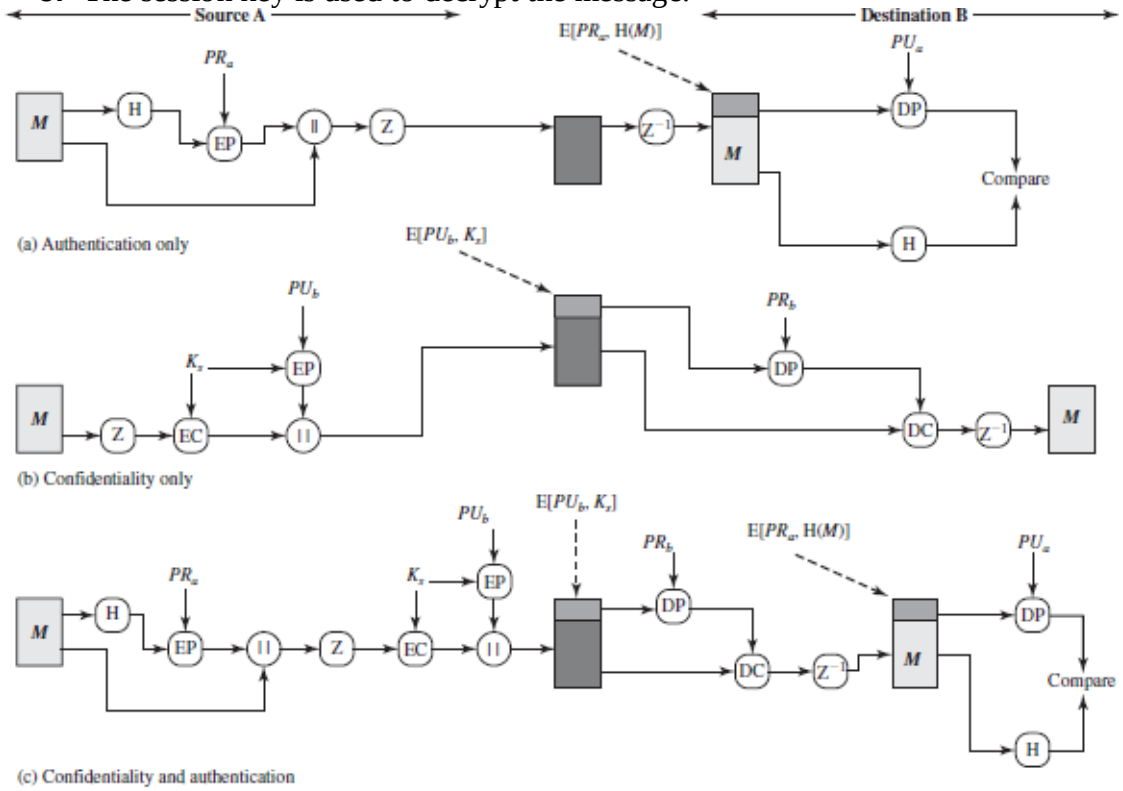
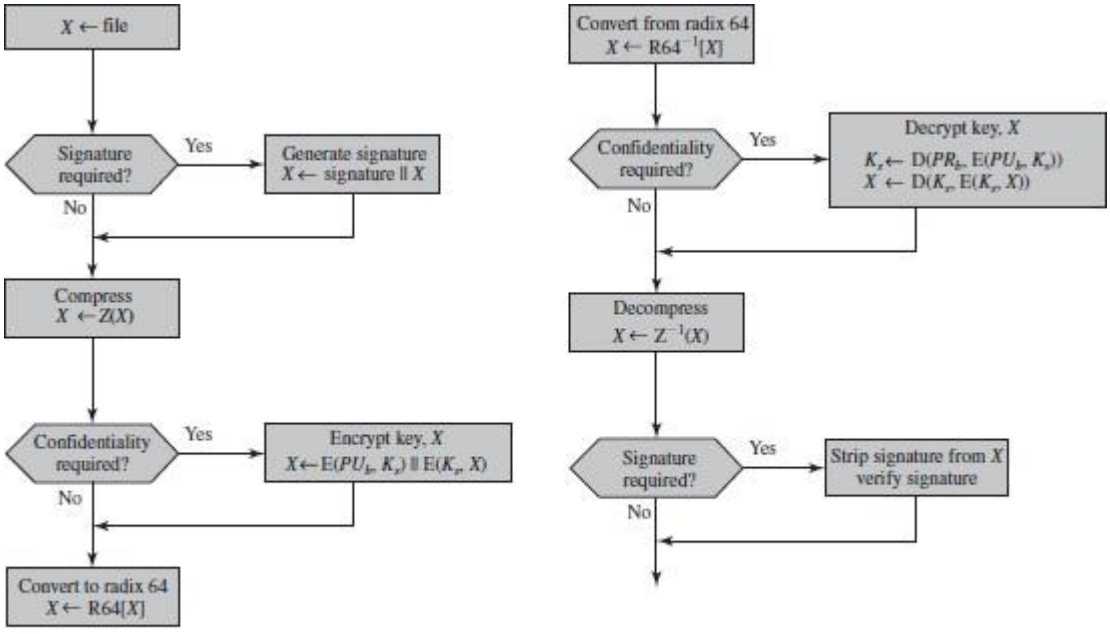


Figure PGP Cryptographic Functions



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Figure Transmission and Reception of PGP Messages

The **message component** includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation. The **signaturecomponent** includes the following.

- **Timestamp:** The time at which the signature was made.
- **Message digest:** The 160-bit SHA-1 digest encrypted with the sender's private signature key. The digest is calculated over the signature timestamp concatenated with the data portion of the message component. The inclusion of the signature timestamp in the digest insures against replay types of attacks. The exclusion of the filename and timestamp portions of the message component ensures that detached signatures are exactly the same as attached signatures prefixed to the message. Detached signatures are calculated on a separate file that has none of the message component header fields.
- **Leading two octets of message digest:** Enables the recipient to determine if the correct public key was used to decrypt the message digest for authentication by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message.
- **Key ID of sender's public key:** Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest.

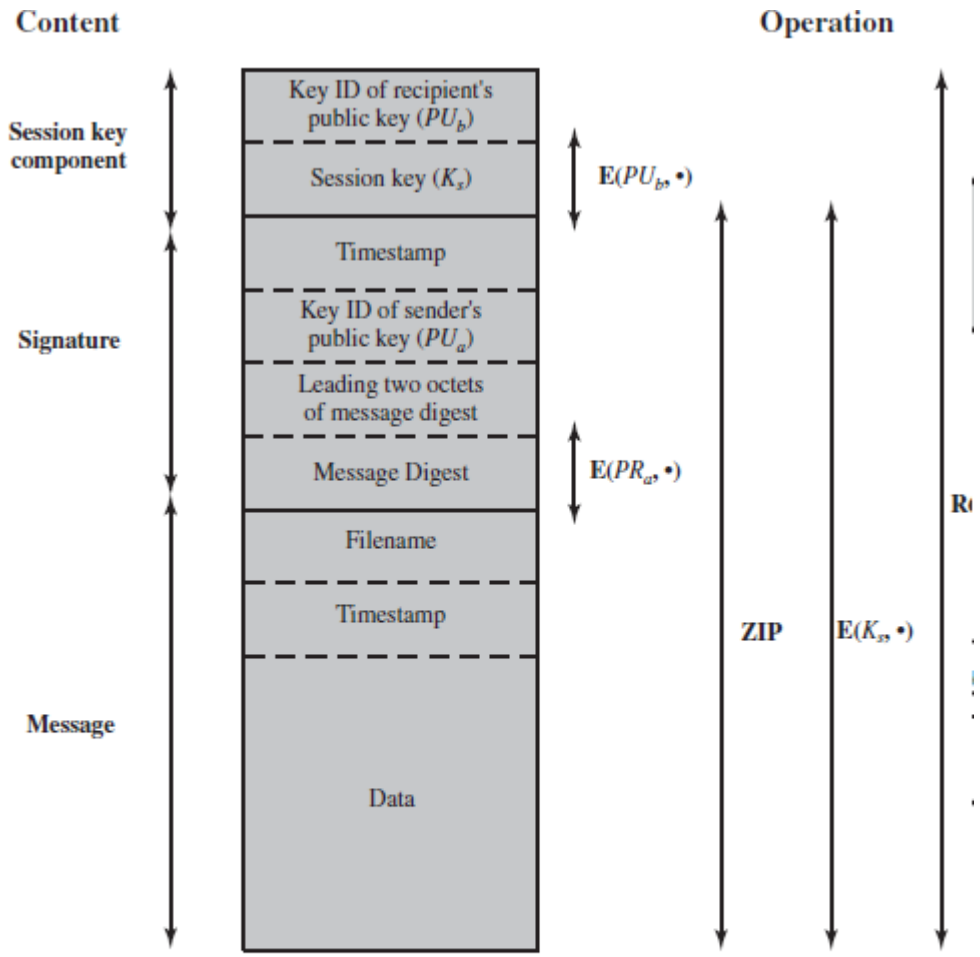


Figure General Format PGP Message (from A to B)

➢ S/MIME

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. Although both PGP and S/MIME are on an IETF standards track, it appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users.

Multipurpose Internet Mail Extensions

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP), defined in RFC 821, or some other mail transfer protocol and RFC 5322 for electronic mail. [PARZ06] lists the following limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/Uudecode scheme.However, none of these is a standard or even a *de facto* standard.

2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.

3. SMTP servers may reject mail message over a certain size.

4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.

5. SMTP gateways to X.400 electronic mail networks cannot handle non textual data included in X.400 messages.

6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:

- Deletion, addition, or reordering of carriage return and linefeed
- Truncating or wrapping lines longer than 76 characters
- Removal of trailing white space (tab and space characters)
- Padding of lines in a message to the same length
- Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations. The specification is provided in RFCs 2045 through 2049.

OVERVIEW The MIME specification includes the following elements.

1. Five new message header fields are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.

2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.

3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

In this subsection, we introduce the five message header fields. The next two subsections deal with content formats and transfer encodings.

The five header fields defined in MIME are

- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

S/MIME Messages

S/MIME makes use of a number of new MIME content types. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort.

We examine each of these in turn after first looking at the general procedures for S/MIME message preparation.

SECURING A MIME ENTITY S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 5322 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. The MIME entity is prepared according to the normal rules for MIME message preparation. Then the MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce

what is known as a PKCS object. A PKCS object is then treated as message content and wrapped in MIME (provided with appropriate MIME headers). This process should become clear as we look at specific objects and provide examples.

In all cases, the message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart.

The use of transfer encoding requires special attention. For most cases, the result of applying the security algorithm will be to produce an object that is partially or totally represented in arbitrary binary data. This will then be wrapped in an outer MIME message, and transfer encoding can be applied at that point, typically base64. However, in the case of a multipart signed message (described in more detail later), the message content in one of the subparts is unchanged by the security process. Unless that content is 7bit, it should be transfer encoded using base 64 or quoted- printable so that there is no danger of altering the content to which the signature wasapplied.

➢ IP SECURITY

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.

Figure 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.

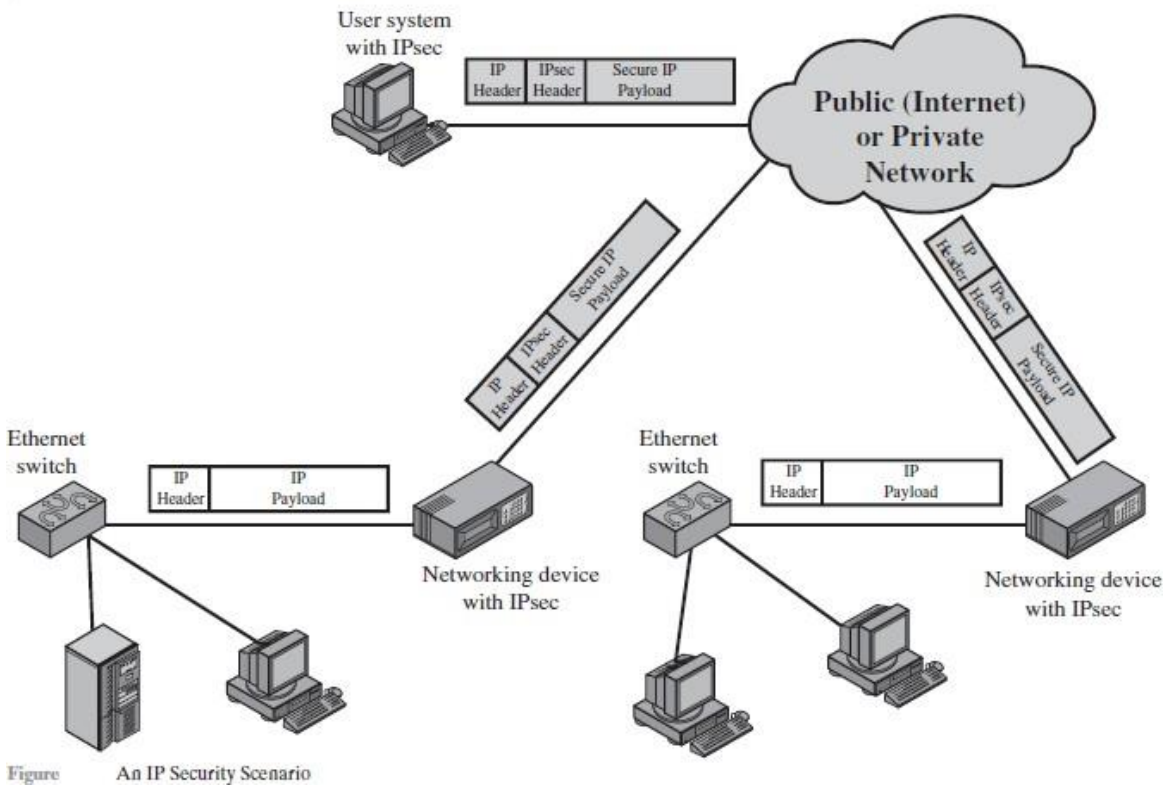


Figure An IP Security Scenario

Benefits of IPsec

Some of the benefits of IPsec are:

• When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.

• IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.

• IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPsec is implemented in the firewall or router. Even if IPsec is implemented in end systems, upper-layer software, including applications, is not affected.

• IPsec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.

• IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual sub network within an organization for sensitive applications.

Routing Applications

In addition to supporting end users and protecting premises systems and networks, IPsec can play a vital role in the routing architecture required for internetworking. The following are the examples of the use of IPsec. IPsec can assure that

• A router advertisement (a new router advertises its presence) comes from an authorized router.

• A neighbor advertisement (a router seeks to establish or maintain a neighbour relationship with a router in another routing domain) comes from an authorized router.

• A redirect message comes from the router to which the initial IP packet was sent.

• A routing update is not forged.

Without such security measures, an opponent can disrupt communications or divert some traffic. Routing protocols such as Open Shortest Path First (OSPF) should be run on top of security associations between

routers that are defined by IPsec.

IPsec Documents

IPsec encompasses three functional areas: authentication, confidentiality, and key management. The totality of the IPsec specification is scattered across dozens of RFCs and draft IETF documents, making this the most complex and difficult to grasp of all IETF specifications.

The documents can be categorized into the following groups.

- **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPsec technology. The current specification is RFC 4301, Security Architecture for the Internet Protocol.
- **Authentication Header (AH):** AH is an extension header to provide message authentication. The current specification is RFC 4302, IP Authentication Header. Because message authentication is provided by ESP, the use of AH is deprecated. It is included in IPsecv3 for backward compatibility but should not be used in new applications.
- **Encapsulating Security Payload (ESP):** ESP consists of an encapsulating header and trailer used to provide encryption or combined encryption/authentication. The current specification is RFC 4303, IP Encapsulating Security Payload (ESP).
- **Internet Key Exchange (IKE):** This is a collection of documents describing the key management schemes for use with IPsec. The main specification is RFC 4306, Internet Key Exchange (IKEv2) Protocol, but there are a number of related RFCs.
- **Cryptographic algorithms:** This category encompasses a large set of documents that define and describe cryptographic algorithms for encryption, message authentication, pseudorandom functions (PRFs), and cryptographic key exchange.
- **Other:** There are a variety of other IPsec-related RFCs, including those dealing with security policy and management information base (MIB) content.

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

Transport and Tunnel Modes

Both AH and ESP support two modes of use: transport and tunnel mode. The operation of these two modes is best understood in the context of a description of ESP, which is covered in Section 19.3. Here we provide a brief overview.

TRANSPORT MODE Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet.1 Examples include a TCP or UDP segment or an ICMP packet, all of which operate directly above IP in a host protocol stack. Typically, transport mode is used for end-to-end communication between two hosts (e.g., a client and a server, or two workstations).When a host runs AH or ESP over IPv4, the payload is the data that normally follow the IP header. For IPv6, the payload is the data that normally follow both the IP header and any IPv6

extensions headers that are present, with the possible exception of the destination options header, which may be included in the protection.

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

TUNNEL MODE Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new outer IP packet with a new outer IP header. The entire original, inner, packet travels through a tunnel from one point of an IP network to another; no routers along the way are able to

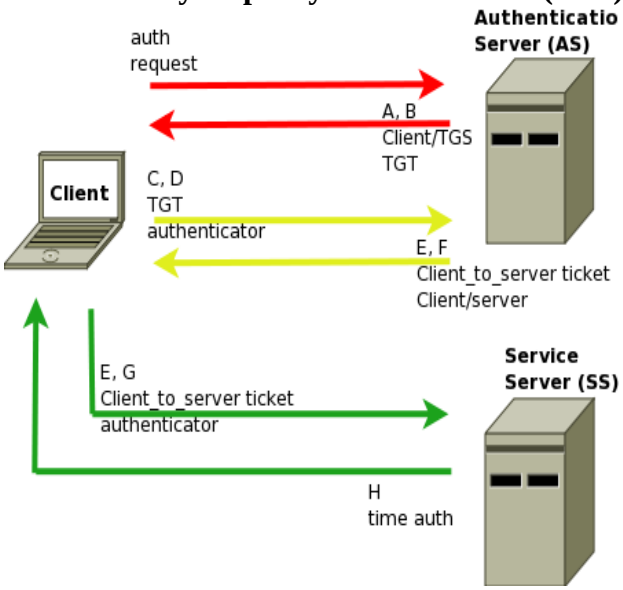
examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding

to the security.Tunnel mode is used when one or both ends of a security association (SA) are a security gateway, such as a firewall or router that implements IPsec.

With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPsec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the IPsec software in the firewall or secure router at the boundary ofthe local network.

Fundamental to the operation of IPsec is the concept of a security policy applied to each IP packet that transits from a source to a destination. IPsec policy is

determined primarily by the interaction of two databases, the **security association database (SAD)** and the **security policy database (SPD)**. This section provides an overview o



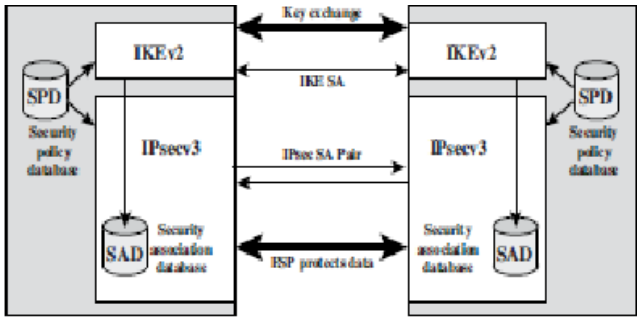


Figure IPsec Architecture

f these two databases and then summarizes their use during IPsec operation. Figure 19.2 illustrates the relevant relationships.

➢ WEB SECURITY

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 Figure. 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

exchange.

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 withone 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

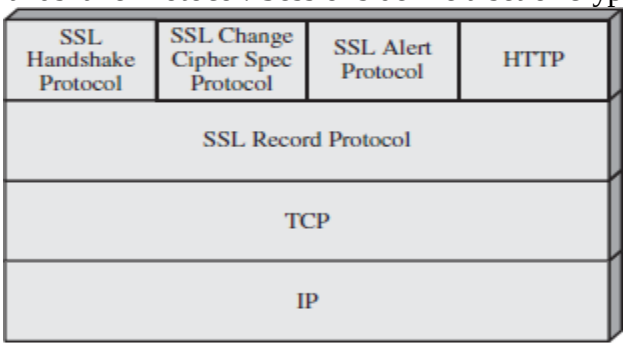


Figure SSL Protocol Stack

security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection. Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice. There are a number of states associated with each session. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.

A session state is defined by the following parameters.

• **Session identifier:** An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

• **Peer certificate:** An X509.v3 certificate of the peer. This element of the state may be null.

• **Compression method:** The algorithm used to compress data prior to encryption.

• **Cipher spec:** Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.

• **Master secret:** 48-byte secret shared between the client and server.

• **Is resumable:** A flag indicating whether the session can be used to initiate new connections. A connection state is defined by the following parameters.

• **Server and client random:** Byte sequences that are chosen by the server and client for each connection.

• **Server write MAC secret:** The secret key used in MAC operations on data sent bythe server.

• **Client write MAC secret:** The secret key used in MAC operations on data sent bythe client.

• **Server write key:** The secret encryption key for data encrypted by the server and decrypted by the client.

• **Client write key:** The symmetric encryption key for data encrypted by the client and decrypted by the server.

• **Initialization vectors:** When a block cipher in CBC mode is used, an initialization vector (IV) is

maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter, the final ciphertext block from each record is preserved for use as the IV with the following record.

• **Sequence numbers:** Each party maintains separate sequence numbers for transmitted and received messages for each connection.When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 264 – 1.

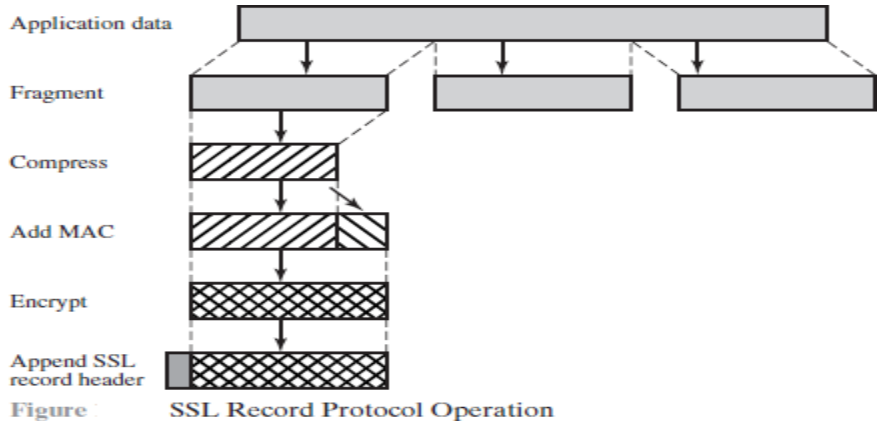


Figure SSL Record Protocol Operation

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 16.3 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.

The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null. The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as hash(MAC_write_secret || pad_2|| hash(MAC_write_secret || pad_1||seq_num || SSLCompressed.type || SSLCompressed.length || SSLCompressed.fragment))

where

|| = concatenation

MAC_write_secret = shared secret key hash = cryptographic hash algorithm; eitherMD5 or SHA-1

pad_1 = the byte 0x36 (0011 0110) repeated48 times (384 bits) for MD5 and 40 times (320 bits) for SHA-1

pad_2 = the byte 0x5C (0101 1100) repeated 48times for MD5 and 40 times for SHA-1

seq_num = the sequence number for this message SSLCompressed.type = the higher-level protocol used to processthis fragment

SSLCompressed.length = the length of the compressed fragment SSLCompressed.fragment = the compressed fragment (if compressionis not used, this is the plaintext fragment)

➢ INTRUDERS

One of the two most publicized threats to security is the intruder (the other is viruses), often referred to as a hacker or cracker. In an important early study of intrusion, Anderson identified three classes of intruders:

- **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
- **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
- **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. Following are the examples of intrusion:

- Performing a remote root compromise of an e-mail server
- Defacing a Web server
- Guessing and cracking passwords
- Copying a database containing credit card numbers
- Viewing sensitive data, including payroll records and medical information,without authorization
- Running a packet sniffer on a workstation to capture usernames and passwords
- Using a permission error on an anonymous FTP server to distribute piratedsoftware and music files
- Dialing into an unsecured modem and gaining internal network access
- Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password
- Using an unattended, logged-in workstation without permission

Some Examples of Intruder Patterns of Behavior

(a) Hacker

1. Select the target using IP lookup tools such as NSLookup, Dig, and others.
2. Map network for accessible services using tools such as NMAP.
3. Identify potentially vulnerable services (in this case, pcAnywhere).
4. Brute force (guess) pcAnywhere password.
5. Install remote administration tool called DameWare.Wait for administrator to log on and capture his password.
6. Use that password to access remainder of network.

(b) Criminal Enterprise

1. Act quickly and precisely to make their activities harder to detect.
2. Exploit perimeter through vulnerable ports.
3. Use Trojan horses (hidden software) to leave back doors for reentry.
4. Use sniffers to capture passwords.
5. Do not stick around until noticed.
6. Make few or no mistakes.

(c) Internal Threat

1. Create network accounts for themselves and their friends.
2. Access accounts and applications they wouldn't normally use for their daily jobs.
3. E-mail former and prospective employers.

4. Conduct furtive instant-messaging chats.
5. Visit Web sites that cater to disgruntled employees, such as f'dcompany.com.
6. Perform large downloads and file copying.
7. Access the network during off hours.

Intrusion Techniques

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Most initial attacks use system or software vulnerabilities that allow a user to execute code that opens a back door into the system. Alternatively, the intruder attempts to acquire information that should have been protected. In some cases, this information is in the form of a user password. With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords. The password file can be protected in one of two ways:

- **One-way function:**

The system stores only the value of a function based on the user's password.When the user presents a password, the system transforms that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the one-way function and in which a fixed-length output is produced.

- **Access control:**

Access to the password file is limited to one or a very few accounts

Few techniques for learning passwords:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
2. Exhaustively try all short passwords (those of one to three characters).
3. Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.
4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.
5. Try users' phone numbers, Social Security numbers, and room numbers.
6. Try all legitimate license plate numbers for this state.
7. Use a Trojan horse to bypass restrictions on access.
8. Tap the line between a remote user and the host system.

Intrusion Detection System:

An intrusion detection system (IDS) is a device, typically another separate computer that monitors activity to identify malicious or suspicious events. An IDS is a sensor, like a smoke detector, that raises an alarm if specific things occur. A model of an IDS is shown in below figure. The components in the figure are the four basic elements of an intrusion detection system, based on the Common Intrusion Detection Framework of [STA96]. An IDS receives raw inputs from sensors. It saves those inputs, analyzesthem, and takes some controlling action.

Types of IDSs

The two general types of intrusion detection systems are signature based and heuristic. Signature-based intrusion detection systems perform simple pattern- matching and report situations that match a pattern corresponding to a known attack type. Heuristic intrusion detection systems, also known as anomaly based, build a model of acceptable behavior and flag exceptions to that model; for the future, the administrator can mark a flagged behavior as acceptable so that the heuristic IDS will now treat that previously unclassified behavior as acceptable.

Intrusion detection devices can be network based or host based. A network-based IDS is a stand-alone

device attached to the network to monitor traffic throughout that network; a host-based IDS runs on a single workstation or client or host, to protect that one host.

Signature-Based Intrusion Detection:

A simple signature for a known attack type might describe a series of TCP SYN packets sent to many different ports in succession and at times close to one another, as would be the case for a port scan. An intrusion detection system would probably find nothing unusual in the first SYN, say, to port 80, and then another (from the same source address) to port 25. But as more and more ports receive SYN packets, especially ports that are not open, this pattern reflects a possible port scan. Similarly, some implementations of the protocol stack fail if they receive an ICMP packet with a data length of 65535 bytes, so such a packet would be a pattern for which to watch. *Heuristic Intrusion* **Detection:**

Because signatures are limited to specific, known attack patterns, another form of intrusion detection becomes useful. Instead of looking for matches, heuristic intrusion detection looks for behavior that is out of the ordinary. The original work in this area focused on the individual, trying to find characteristics of that person that might be helpful in understanding normal and abnormal behavior. For example, one user might always start the day by reading e-mail, write many documents using a word processor, and occasionally back up files. These actions would be normal. This user does not seem to use many administrator utilities. If that person tried to access sensitive system management utilities, this new behavior might be a clue that someone else was acting under the user's identity.

Inference engines work in two ways. Some, called state-based intrusion detection systems, see the system going through changes of overall state or configuration. They try to detect when the system has veered into unsafe modes. Others try to map current activity onto a model of unacceptable activity and raise an alarm when the activity resembles the model.

These are called model-based intrusion detection systems. This approach has been extended to networks in [MUK94]. Later work sought to build a dynamic model of behavior, to accommodate variation and evolution in a person's actions over time. The technique compares real activity with a known representation of normality.

Alternatively, intrusion detection can work from a model of known bad activity. For example, except for a few utilities (login, change password, create user), any other attempt to access a password file is suspect. This form of intrusion detection is known as misuse intrusion detection. In this work, the real activity is compared against a known suspicious area.

Stealth Mode:

An IDS is a network device (or, in the case of a host-based IDS, a program running on a network device). Any network device is potentially vulnerable to network attacks. How useful would an IDS be if it itself were deluged with a denial-of-service attack? If an attacker succeeded in logging in to a system within the protected network, wouldn't trying to disable the IDS be the next step?

To counter those problems, most IDSs run in stealth mode, whereby an IDS has two network interfaces: one for the network (or network segment) being monitored and the other to generate alerts and perhaps other administrative needs. The IDS uses the monitored interface as input only; it *never* sends packets out through that interface. Often, the interface is configured so that the device has no published address through the monitored interface; that is, a router cannot route anything to that address directly, because the router does not know such a device exists. It is the perfect passive wiretap. If the IDS needs to generate an alert, it uses only the alarm interface on a completely separate control network.

Goals for Intrusion Detection Systems:

1. Responding to alarms:

Whatever the type, an intrusion detection system raises an alarm when it finds a match. The alarm can range from something modest, such as writing a note in an audit log, to something significant, such as

paging the system security administrator. Particular implementations allow the user to determine what action the system should take on what events. In general, responses fall into three major categories (any or all of which can beused in a single response):

Monitor, collect data, perhaps increase amount of data collectedProtect, act to reduce exposure

Call a human

2. False Results:

Intrusion detection systems are not perfect, and mistakes are their biggest problem. Although an IDS might detect an intruder correctly most of the time, it may stumble in two different ways: by raising an alarm for something that is not really an attack (called a false positive, or type I error in the statistical community) or not raising an alarm for a real attack (a false negative, or type II error). Too many false positives means the administrator will be less confident of the IDS's warnings, perhaps leading to a real alarm's being ignored. But false negatives mean that real attacks are passing the IDS without action. We say that the degree of false positives and false negatives represents the sensitivity of the system. Most IDS implementations allow the administrator to tune the system's sensitivity, to strike an acceptable balance between false positives and negatives.

IDS strength and limitations:

On the upside, IDSs detect an ever-growing number of serious problems. And as we learn more about problems, we can add their signatures to the IDS model. Thus, over time, IDSs continue to improve. At the same time, they are becoming cheaper and easier to administer. On the downside, avoiding an IDS is a first priority for successful attackers. An IDS that is not well defended is useless. Fortunately, stealth mode IDSs are difficult even to find on an internal network, let alone to compromise. IDSs look for known weaknesses, whether through patterns of known attacks or models of normal behavior. Similar IDSs may have identical vulnerabilities, and their selection criteria may miss similar attacks. Knowing how to evade a particular model of IDS is an important piece of intelligence passed within the attacker community. Of course, once manufacturers become aware of a shortcoming in their products, they try to fix it. Fortunately, commercial IDSs are pretty good at identifying attacks. Another IDS limitation is its sensitivity, which is difficult to measure and adjust. IDSs will never be perfect, so finding the proper balance is critical.

In general, IDSs are excellent additions to a network's security. Firewalls block traffic to particular ports or addresses; they also constrain certain protocols to limit their impact. But by definition, firewalls have to allow some traffic to enter a protected area. Watching what that traffic actually does inside the protected area is an IDS's job, which it does quite well.

Trojan Horses

A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function. Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changes the invoking user’s file permissions so that the files are readable by any user. The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility program or application. An example is a program that ostensibly produces a listing of the user’s files in a desirable format. After another user has run the program, the author of the program can then access the information in the user’s files. An example of a Trojan horse program that would be difficult to detect is a compiler that has been modified to insert additional code into certain programs as they are compiled, such as a system login program. The code creates a backdoor in the login program that permits the author to log on to the system using a special password. This Trojan horse can never be discovered by reading the source code of the login program.

Trojan horses fit into one of three models:

- Continuing to perform the function of the original program and additionally performing a separate malicious activity
- Continuing to perform the function of the original program but modifying the function to perform malicious activity (e.g., a Trojan horse version of a login program that collects passwords) or to disguise other malicious activity (e.g., a Trojan horse version of a process listing program that does not display certain processes that are malicious)
- Performing a malicious function that completely replaces the function of the originalprogram

➢ Viruses

A computer virus is a piece of software that can "infect" other programs by modifying them; the modification includes injecting the original program with a routine to make copies of the virus program, which can then go on to infect other programs. A computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network. In a network environment, the ability to access applications and system services on other computers provides aperfect culture for the spread of a virus.

A virus can do anything that other programs do.The difference is that a virus attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs that is allowed by the privileges of the current user.

A computer virus has three parts :

- **Infection mechanism:** The means by which a virus spreads, enabling it toreplicate.

The mechanism is also referred to as the **infection vector**.

- **Trigger:** The event or condition that determines when the payload is activatedor delivered.
- **Payload:** What the virus does, besides spreading. The

➢ FIREWALLS

Firewalls were officially invented in the early 1990s, but the concept really reflects the reference monitor from two decades earlier.

What is a Firewall?

A firewall is a device that filters all traffic between a protected or "inside" network and a less trustworthy or "outside" network. Usually a firewall runs on a dedicated device; because it is a single point through which traffic is channeled, performance is important, which means non-firewall functions should not be done on the same machine. Because a firewall is executable code, an attacker could compromise that code and execute from the firewall's device. Thus, the fewer pieces of code on the device, the fewer tools the attacker would have by compromising the firewall. Firewall code usually runs on a proprietary or carefully minimized operating system. The purpose of a firewall is to keep "bad" things outside a protected environment. To accomplish that, firewalls implement a security policy that is specifically designed to address what bad things might happen. For example, the policy might be to prevent any access from outside (while still allowing traffic to pass *from* the inside *to* the outside). Alternatively, the policy might permit accesses only from certain places, from certain users, or for certain activities. Part of the challenge of protecting a network with a firewall is determining which security policy meets the needs of the installation.

Design of Firewalls:

A reference monitor must beAlways invoked Tamperproof

Small and simple enough for rigorous analysis

A firewall is a special form of reference monitor. By carefully positioning a firewall within a network, we

can ensure that all network accesses that we want to control must pass through it. This restriction meets the "always invoked" condition. A firewall is typically well isolated, making it highly immune to modification. Usually a firewall is implemented on a separate computer, with direct connections only to the outside and inside networks. This isolation is expected to meet the "tamperproof" requirement. And firewall designers strongly recommend keeping the functionality of the firewall simple.

Types of Firewalls:

Firewalls have a wide range of capabilities. Types of firewalls include

- □ Packet filtering gateways or screening routers
- □ Stateful inspection firewalls
- □ Application proxy
- □ Guards
- □ Personal firewalls

Packet Filtering Gateway:

A packet filtering gateway or screening router is the simplest, and in some situations, the most effective type of firewall. A packet filtering gateway controls access to packets on the basis of packet address (source or destination) or specific transport protocol type (such as HTTP web traffic). As described earlier in this chapter, putting ACLs on routers may severely impede their performance. But a separate firewall behind (on the local side) of the router can screen traffic before it gets to the protected network. Figure 7-34 shows a packet filter that blocks access from (or to) addresses in one network; the filter allows HTTP traffic but blocks traffic using the Telnet protocol.

Stateful Inspection Firewall:

Filtering firewalls work on packets one at a time, accepting or rejecting each packet and moving on to the next. They have no concept of "state" or "context" from one packet to the next. A stateful inspection firewall maintains state information from one packet to another in the input stream.

One classic approach used by attackers is to break an attack into multiple packets by forcing some packets to have very short lengths so that a firewall cannot detect the signature of an attack split across two or more packets. (Remember that with the TCP protocols, packets can arrive in any order, and the protocol suite is responsible for reassembling the packet stream in proper order before passing it along to the application.) A stateful inspection firewall would track the sequence of packets and conditions from one packet to another to thwart such an attack.

Application Proxy

Packet filters look only at the headers of packets, not at the data inside the packets. Therefore, a packet filter would pass anything to port 25, assuming its screening rules allow inbound connections to that port. But applications are complex and sometimes contain errors. Worse, applications (such as the e-mail delivery agent) often act on behalf of all users, so they require privileges of all users (for example, to store incoming mail messages so that inside users can read them). A flawed application, running with all users' privileges, can cause much damage. An application proxy gateway, also called a bastion host, is a firewall that simulates the (proper) effects of an application so that the application receives only requests to act properly. A proxy gateway is a two-headed device: It looks to the inside as if it is the outside (destination) connection, while to the outside it responds just as the insider would.

An application proxy runs pseudo-applications. For instance, when electronic mail is transferred to a location, a sending process at one site and a receiving process at the destination communicate by a protocol that establishes the legitimacy of a mail transfer and then actually transfers the mail message. The protocol between sender and destination is carefully defined. A proxy gateway essentially intrudes in the middle of this protocol exchange, seeming like a destination in communication with the sender that is outside the firewall, and seeming like the sender in communication with the real destination on the inside. The proxy in the middle has the opportunity to screen the mail transfer, ensuring that only acceptable

e-mail protocol commandsare sent to the destination.

Guard:

A guard is a sophisticated firewall. Like a proxy firewall, it receives protocol data units, interprets them, and passes through the same or different protocol data units that achieve either the same result or a modified result. The guard decides what services to perform on the user's behalf in accordance with its available knowledge, such as whatever it can reliably know of the (outside) user's identity, previous interactions, and so forth. The degree of control a guard can provide is limited only by what is computable. But guards and proxy firewalls are similar enough that the distinction between them is sometimes fuzzy. That is, we can add functionality to a proxy firewall until it starts to look a lot like a guard.

Personal Firewalls:

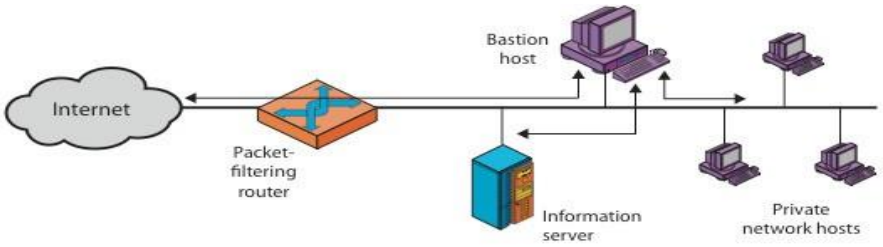
A personal firewall is an application program that runs on a workstation to block unwanted traffic, usually from the network. A personal firewall can complement the work of a conventional firewall by screening the kind of data a single host will accept, or it can compensate for the lack of a regular firewall, as in a private DSL or cable modem connection.

The personal firewall is configured to enforce some policy. For example, the user may decide that certain sites, such as computers on the company network, are highly trustworthy, but most other sites are not. The user defines a policy permitting download of code, unrestricted data sharing, and management access from the corporate segment, but not from other sites. Personal firewalls can also generate logs of accesses, which can be useful to examine in case something harmful does slip through the firewall.

A personal firewall runs on the very computer it is trying to protect. Thus, a clever attacker is likely to attempt an undetected attack that would disable or reconfigure the firewall for the future. Still, especially for cable modem, DSL, and other "always on" connections, the static workstation is a visible and vulnerable target for an ever- present attack community. A personal firewall can provide reasonable protection to clients that are not behind a network firewall.

Firewall Configurations

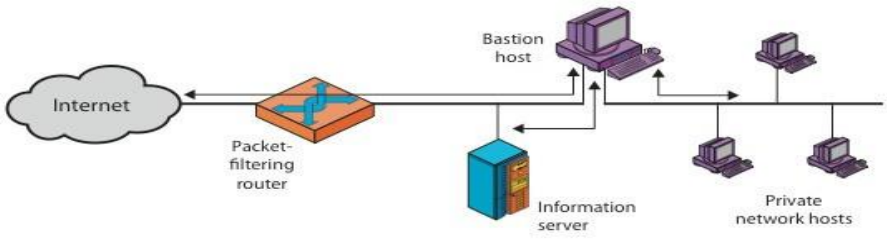
- In addition to the use of a simple configuration consisting of a single system, more complex configurations are possible and indeed more common.
- Figure illustrates three common firewall configurations.
- Figure (a) shows the "screened host firewall, single-homed bastion configuration", where the firewall consists of two systems:
- a packet-filtering router - allows Internet packets to/from bastion only
- a bastion host - performs authentication and proxy functions
- This configuration has greater security, as it implements both packet-level & application-level filtering, forces an intruder to generally penetrate two separate systems to compromise internal security, & also affords flexibility in providing direct Internet access to specific internal servers (eg web) if desired.



(a) Screened host firewall system (single-homed bastion host)

Figure (b) illustrates the "screened host firewall, dual-homed bastion configuration" which physically

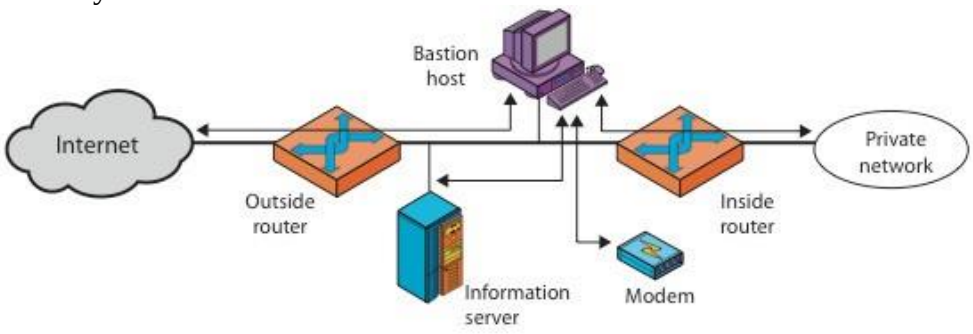
separates the external and internal networks, ensuring two systems must be compromised to breach security. The advantages of dual layers of security are also present here. Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy, but are now separated from the internal network.
re(c) shows the "screened subnet firewall configuration", being the most secure shown. It has two packet-



(b) Screened host firewall system (dual-homed bastion host)

filtering routers, one between the bastion host and the Internet and the other between the bastion host and the internal network, creating an isolated sub network. This may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked.
This configuration offers several advantages:

- There are now three levels of defense to thwart intruders
- The outside router advertises only the existence of the screened subnet to the Internet; therefore the internal network is invisible to the Internet
- Similarly, the inside router advertises only the existence of the screened subnet to the internal network; hence systems on the inside network cannot construct direct routes to the Internet



(c) Screened-subnet firewall system

➢ WORMS

A computer worm is a type of malware that spreads copies of itself from computer to computer. A worm can replicate itself without any human interaction, and it does not need to attach itself to a software program in order to cause damage.
How do computer worms work?

Worms can be transmitted via software vulnerabilities. Or computer worms could arrive as attachments in spam emails or instant messages (IMs). Once opened, these files could provide a link to a malicious website or automatically download the computer worm. Once it's installed, the worm silently goes to work and infects the machine without the user's knowledge.

Worms can modify and delete files, and they can even inject additional malicious software onto a computer. Sometimes a computer worm's purpose is only to make copies of itself over and over —

depleting system resources, such as hard drive space or bandwidth, by overloading a shared network. In addition to wreaking havoc on a computer's resources, worms can also steal data, install a backdoor, and allow a hacker to gain control over a computer and its system settings.

How to tell if your computer has a worm

If you suspect your devices are infected with a computer worm, run a virus scan immediately. Even if the scan comes up negative, continue to be proactive by following these steps.

1. **Keep an eye on your hard drive space**. When worms repeatedly replicate themselves, they start to use up the free space on your computer.
2. **Monitor speed and performance**. Has your computer seemed a little sluggish lately? Are some of your programs crashing or not running properly? That could be a red flag that a worm is eating up your processing power.
3. **Be on the lookout for missing or new files**. One function of a computer worm is to delete and replace files on a computer.

How to help protect against computer worms

Computer worms are just one example of malicious software. To help protect your computer from worms and other online threats, take these steps.

1. Since software vulnerabilities are major infection vectors for computer worms, be sure your computer's operating system and applications are up to date with the latest versions. Install these updates as soon as they're available because updates often include patches for security flaws.
2. Phishing is another popular way for hackers to spread worms (and other types of malware). Always be extra cautious when opening unsolicited emails, especially those from unknown senders that contain attachments or dubious links.
3. Be sure to invest in a strong internet security software solution that can help block these threats. A good product should have anti-phishing technology as well as defenses against viruses, spyware, ransomware, and other online threats.