

RT 801 - Security in Computing

Module IV

Network & Application Security

Instructor: Dr. SABU M THAMPI, Rajagiri School of
Engineering and Technology, Kochi, India

Module 4

- Network & Application Security: Kerberos – X509 Authentication service – IP security Architecture – Secure socket layer – Electronic mail security – Pretty Good privacy – S/MIME – secure Electronic Transactions – Firewalls – Security mechanisms in JAVA platform – Applet security – Security policy and Security Manager.

- Authentication is the process of proving one's identity to someone else.
- As humans, we authenticate each other in many ways:
 - We recognize each other's faces when we meet, we recognize each other's voices on the telephone.

- Modern computer systems provide service to multiple users and require the ability to accurately identify the user making a request.
- In traditional systems, the user's identity is verified by checking a password typed during login; the system records the identity and uses it to determine what operations may be performed.
- The process of verifying the user's identity is called **authentication**.

- An authentication protocol would run before the two communicating parties in the system run some other protocol.
- The authentication protocol first establishes the identity of the parties to each other's satisfaction
 - only after authentication do the parties get down to the work at hand.
- It is a fundamental building block for a secure networked environment.

- The Internet is an insecure place.
- Tools to "sniff" passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable.
- Some sites attempt to use firewalls to solve their network security problems.
 - Unfortunately, firewalls assume that "the bad guys" are on the outside, which is often a very bad assumption.
 - Most of the really damaging incidents of computer crime are carried out by insiders.

- Password based authentication is not suitable for use on computer networks.
- Passwords sent across the network can be intercepted and subsequently used by eavesdroppers to impersonate the user.

- A *principal* is the party whose identity is verified.
- The *verifier* is the party who demands assurance of the principal's identity.

Kerberos

In Greek myth Cerberus (**Kerberos**) was a horrific dog who stood watch at the gates of Hades, the world of the dead.

Cerberus had three heads and was so vicious that he was feared even by the gods.





The Three A's of Security

- Authentication --the capability of one entity to prove its identity to another entity
- Authorization – the process of discovering whether you have the rights or permissions to do what you have asked to do
- Auditing –the process of checking to see whether sth. has been done the way it is supposed to have been done

Kerberos

- Kerberos is an authentication service developed at MIT (Massachusetts Institute of Technology)
- Uses symmetric key encryption techniques and a key distribution centre
- It is an add-system that can be used with existing network.
- Versions 4 & 5

- Kerberos addresses the following problem:
 - Assume a distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
 - We would like the servers to be able to restrict access to authorized users and to be able to authenticate requests for service.

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

- In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access.
- Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

- Kerberos provides the following requirements:
 - **Secure**: a network eavesdropper should not be able to obtain the necessary information to impersonate a user.
 - **Reliable**: Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.
 - **Transparent**: the user should not be aware that the authentication is taking place, beyond the requirement to enter a password.
 - **Scalable**: the system should be capable of supporting large numbers of clients and servers.

Kerberos version 4

- A simple authentication procedure must involve three steps:
 1. The client C requests the user password and then send a message to the AS of the Kerberos system that includes the user's ID, the server's ID and the user's password.

2. The AS check its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to the server V.

If both tests are passed, the AS accept the user as authentic and must now convince the server that this user is authentic.

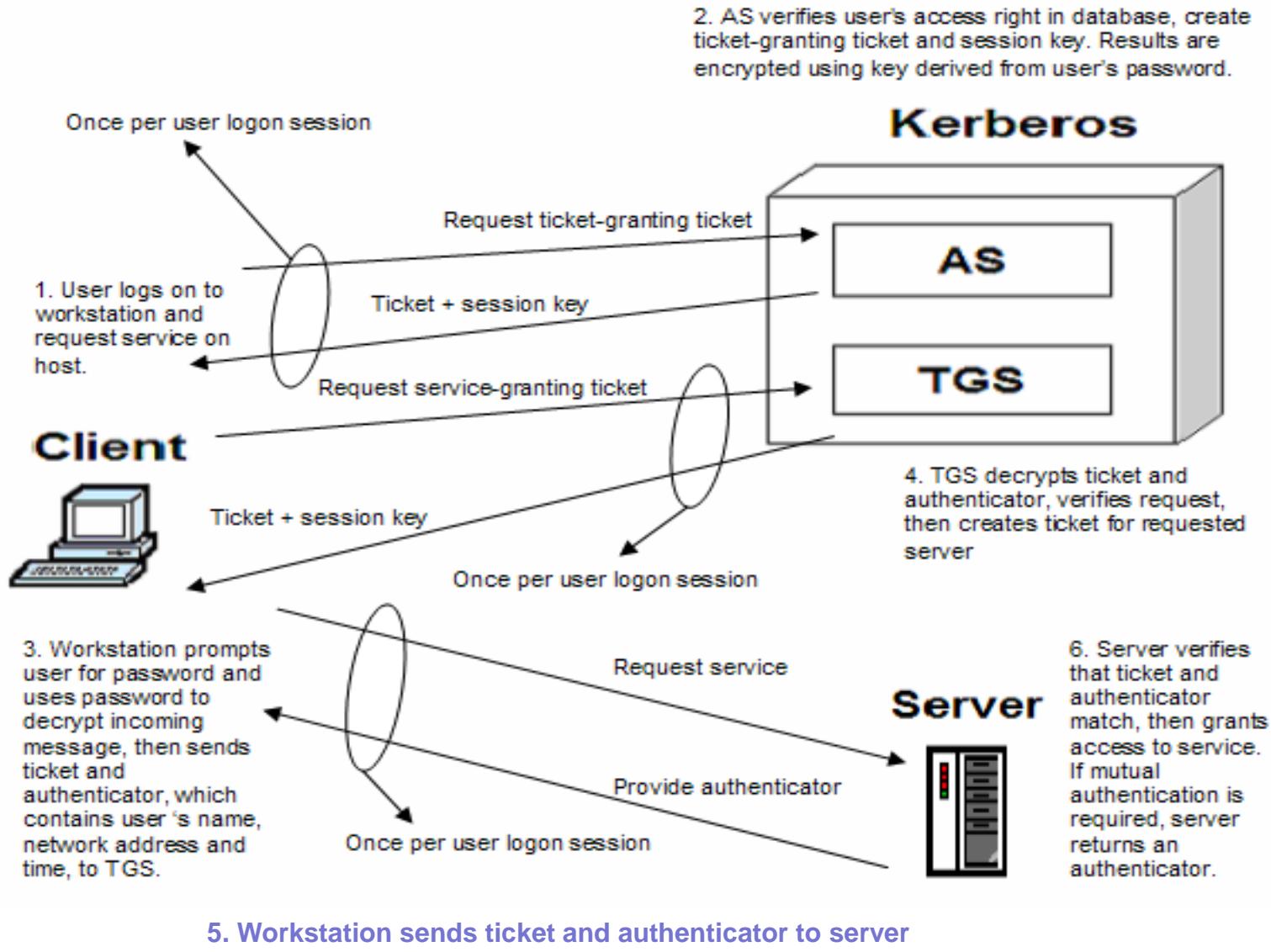
Thus the AS creates and sends back to C **a ticket** that contains the user's ID and network address and the server's ID.

Then it is encrypted with the secret key shared by the AS and the server V.

3. C can now apply to V for the service.
It sends a message to V containing C's ID and the ticket.
V decrypts the ticket and verifies that the user ID in the ticket is the same of the one which came with the ticket.
If these two match, the server grants the requested service to the client.

- This scheme is correct, but this is not enough in a distributed environment:
 - it has some leaks related to security and requires that the user introduce the password every time he needs to access to a service.
- Indeed, the client sends its password unencrypted to the AS; an eavesdropper could capture the password and use any services accessible to the victim.
- Moreover, it is better to minimize the number of times that the user has to enter a password.

- A scheme is introduced for avoiding plaintext passwords
 - a new server, known as the **Ticket-Granting Server** (TGS) is also employed.
- The new service issues tickets to users who have been authenticated to AS.
- Each time the user require access to a new service, the client applies to the TGS using the ticket supplied by the AS to authenticate itself.
- The TGS then grants a ticket to the particular service and the client saves this ticket for future use.



1. The client request a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service
2. The AS responds with a message, encrypted with a key derived from the user's password that **contains the ticket** for the TGS.

The encrypted message also contains a copy of the session key used by C and the TGS. In this way, only the user's client can read it.

The same session key is included in the ticket, which can be read only by the TGS.

Now C and the TGS share a common key.

3. Armed with the ticket and the session key, C is ready to approach the TGS.

C sends the TGS a message that includes the ticket plus the ID of the requested service.

In addition, C transmits an **authenticator**, which includes the ID and address of C's user and a timestamp; it is encrypted with the session key known only by C and TGS.

Unlike the ticket, which is reusable, the authenticator is intended to use only once and has a very short lifetime.

- The TGS can decrypt the ticket with the key that it shares with the AS.
- The TGS decrypt the authenticator with K_c gained from the ticket and check the name and the address from the authenticator with that of the ticket and with the network address of the incoming message.
- If all match, then the TGS is assured that the sender of the ticket is indeed the ticket's real owner.

4. Then the TGS replies to the client, sending a message encrypted with the common key that they share.

It includes a new session key to be used with the server V that must provide the service, the ID of V, **the ticket valid for a specific service** (the ticket is encrypted with a secret key(K_i) known only to the TGS and the server) and the timestamp of the ticket. The ticket itself includes the new session key.

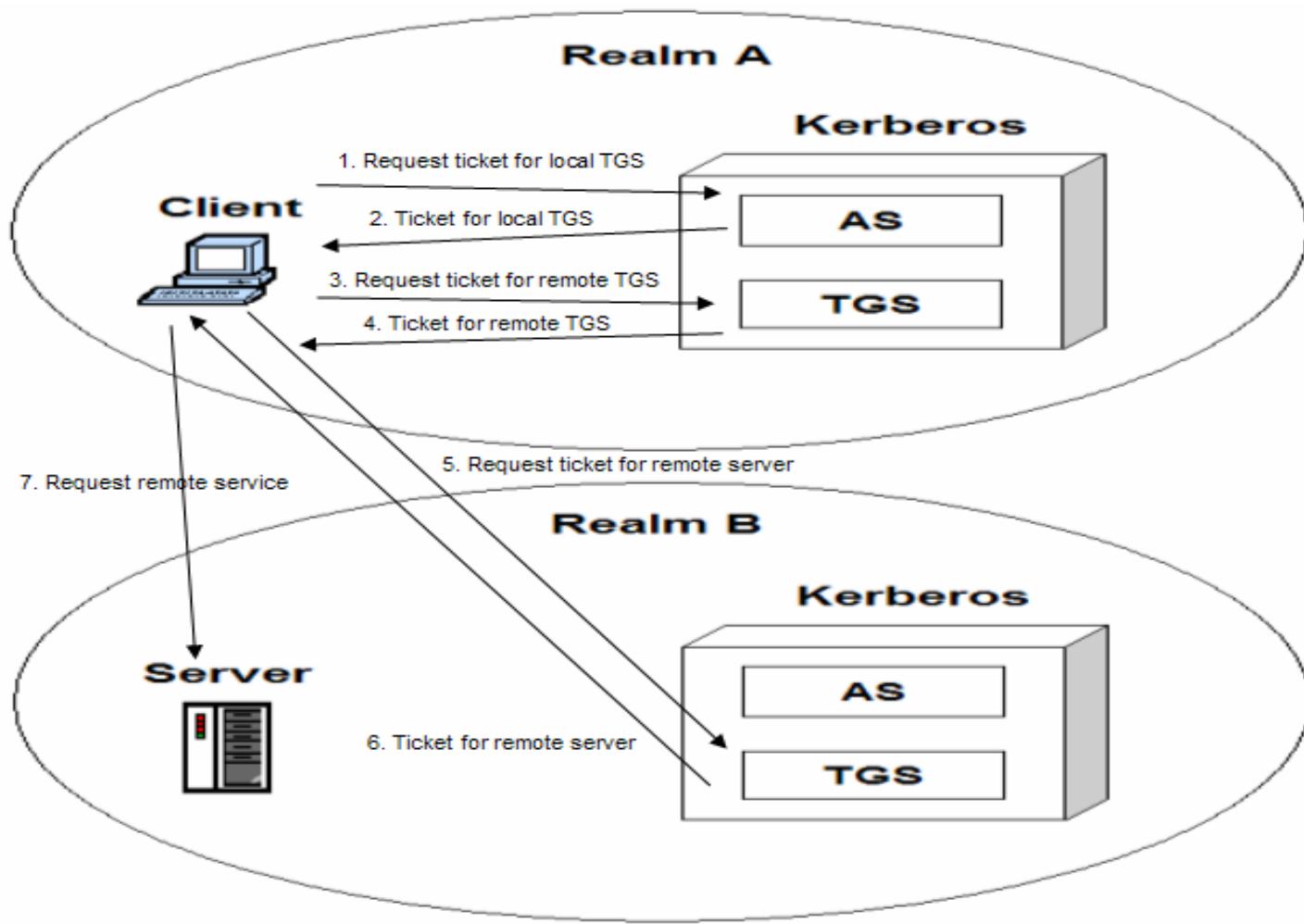
5. C now has a reusable service-granting ticket for V. When C presents this ticket, it also sends an authenticator. The server can decrypt the ticket using K_i , recover the session key and decrypt the authenticator.

- A full-service Kerberos environment, consisting of a Kerberos server, a number of clients and a number of application servers, requires that the Kerberos server must have the user ID (UID) and hashed passwords of all participating users in its database.
- All users are registered with the Kerberos server. Such an environment is referred as a **realm**.
- Moreover, the Kerberos server must share a secret key with each server and every server is registered with the Kerberos server.

- Sometimes, user in one realm may need access to server in other realms and they need to be authenticated before to use services provided by those servers.
- *Kerberos has a mechanism for supporting such inter-realm authentication.*
- The only requirement requested is that the Kerberos server in each interoperating realm **shares a secret key** with the server in the second realm.
 - The two Kerberos server are registered with each other.

- The communication mechanism between a client and a server in two different realms is the following:
 - The client C in the α realm needs a ticket in order to communicate with the server in the β realm. Thus, the user's client follows the usual procedures to gain access to the local TGS and then request a ticket-granting ticket for the remote TGS in realm β .
 - The client can then apply to the remote TGS for a service-granting ticket valid for the desired server in that different realm.
 - The client can now use the ticket obtained from the remote TGS with the server V in the other realm.

2 Inter realm communication



- Unlike UNIX passwords, which are encrypted with a one-way algorithm that cannot be reversed, Kerberos passwords are stored on the server encrypted with a conventional encryption algorithm (typically DES), so that they can be decrypted by the server when needed.
- A user proves his/her identity to the Kerberos server by demonstrating knowledge of the key.

Kerberos limitations

- One of the main problems of Kerberos (which is not relative to security of the protocol) is that any application which wants to use the Kerberos protocol, have to be modified in the code in order to establish a secure communication.
 - This means high costs in terms of time and money, and is not reliable for all applications nor enterprises.

- Another problem relative to security relies on the high dependency on the Kerberos server.
 - If this server goes down, then the whole network goes down as well. This is something very expensive and not desirable in a distributed system.
 - In addition to this, all security relies on the server, so if someone can access the Kerberos server, then all network connection could be hacked.

- Someone could steal from the network the message from the AS to the client, this message, is encrypted by the client key which is derived from the client password.
- Therefore, someone could try to guess this key and then he would be able to identify himself as the original client.
- "Kerberos is not effective against password guessing attacks; if a user chooses a poor password, then an attacker guessing that password can impersonate the user".

X.509 Authentication service

X.500 directory service

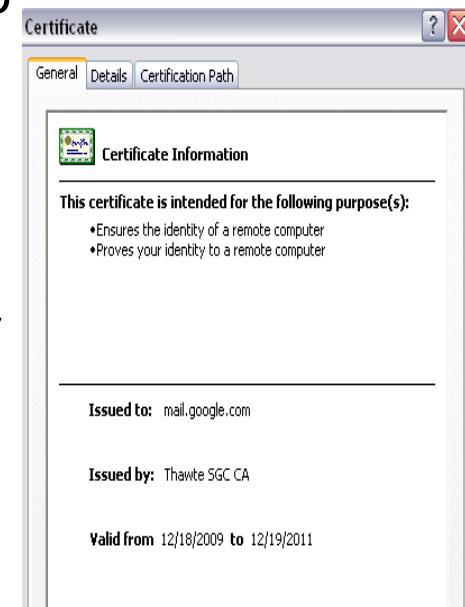
- The X.500 directory service is a global directory service.
- It provides the capability to look up information by name (a white-pages service) and to browse and search for information (a yellow-pages service).

- 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.

- X.509 is based on the use of public-key cryptography and digital signatures.
- The standard doesn't dictate the use of specific algorithm but recommends RSA.
- The digital signature scheme employs hash function
- Distributed set of servers that maintains a database about users
- Each certificate contains the public key of a user and is signed with the private key of a CA.

• Public-key Certificates

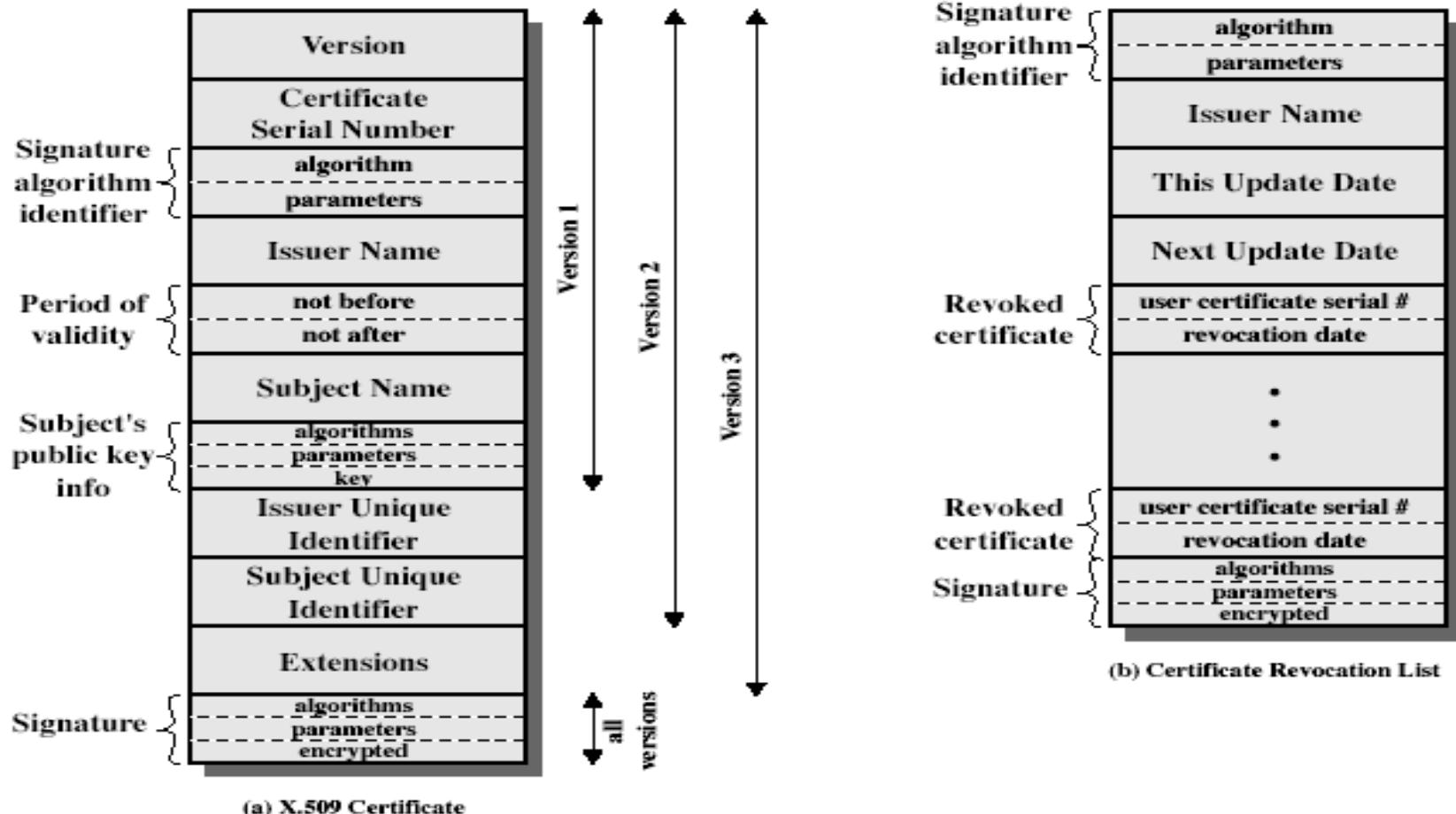
- Use certificates that can be used by participants to exchange keys without contacting a public key authority.
- A certificate consists of a public key plus an identifier of the key owner with the whole block signed by a trusted third party (certificate authority – a Govt. agency, financial institution)
- A user can present his/her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate.
- Anyone needed this users public key can obtain the certificate and verify that it is a valid by way of the attached trusted signature.



X.509 Certificates

- CA signs a certificate with its own secret key
- Anyone can verify a user's certificate, using the CA's public key
- Only CA can place (or modify) a certificate in the directory
- Certificates are unforgeable
- No need to protect certificates in the CA's directory.
- No party other than the CA can modify the certificate without this being detected.

X.509 Certificates



X.509 Certificates

- issued by a Certification Authority (CA), containing:
 - version (1, 2, or 3)
 - serial number (unique within CA) identifying certificate
 - signature algorithm identifier
 - issuer X.500 name (CA)
 - period of validity (from - to dates)
 - subject X.500 name (name of owner)
 - subject public-key info (algorithm, parameters, key)
 - issuer unique identifier (v2+)
 - subject unique identifier (v2+)
 - extension fields (v3)
 - signature (of hash of all fields in certificate, encrypted by the private key of the CA)

- The standard uses the following notation to define a certificate:
- $CA<<A>> = CA(V, S_N, AI, CA, TA, A, AP)$
 - Where $Y<<X>>$ = the certificate of user X issued by certification authority Y
 - V – version of the certificate
 - S_N – serial number of the certificate
 - AI – Identifier of the algorithm used to sign the certificate
 - CA – name of certificate authority
 - TA – period of validity of the certificate
 - A – name of user A
 - AP – Public key of user A

- A and B both ask their own authority for a copy of each others' public key certificate.
- If necessary, the authorities exchange A's and B's certificates.
- The certificates are forwarded to A and B.
- A and B can now securely communicate directly.

Certificate

General Details Certification Path

Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer

* Refer to the certification authority's statement for details.

Issued to: www.irctc.co.in

Issued by: VeriSign Class 3 Extended Validation SSL SGC CA

Valid from 19- 08- 2010 **to** 19- 08- 2012

[Issuer Statement](#)

Learn more about [certificates](#)

OK

Certificate

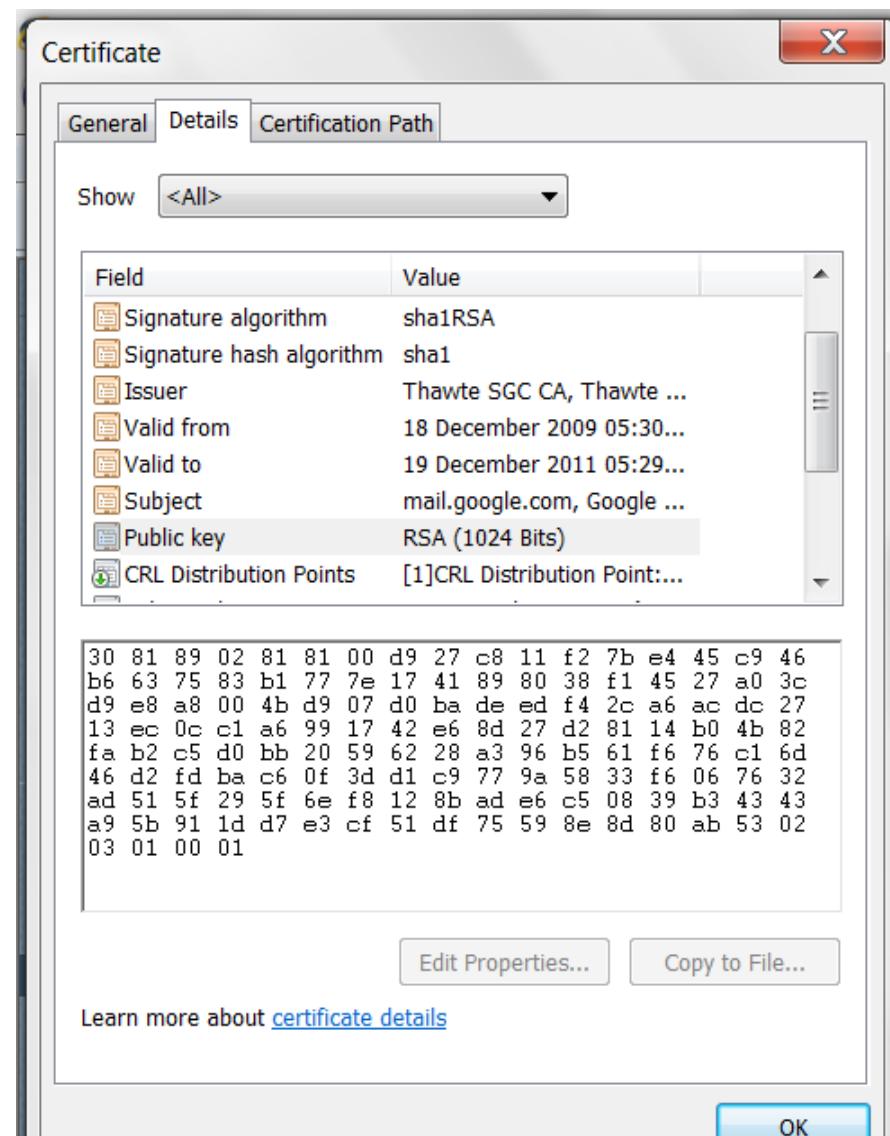
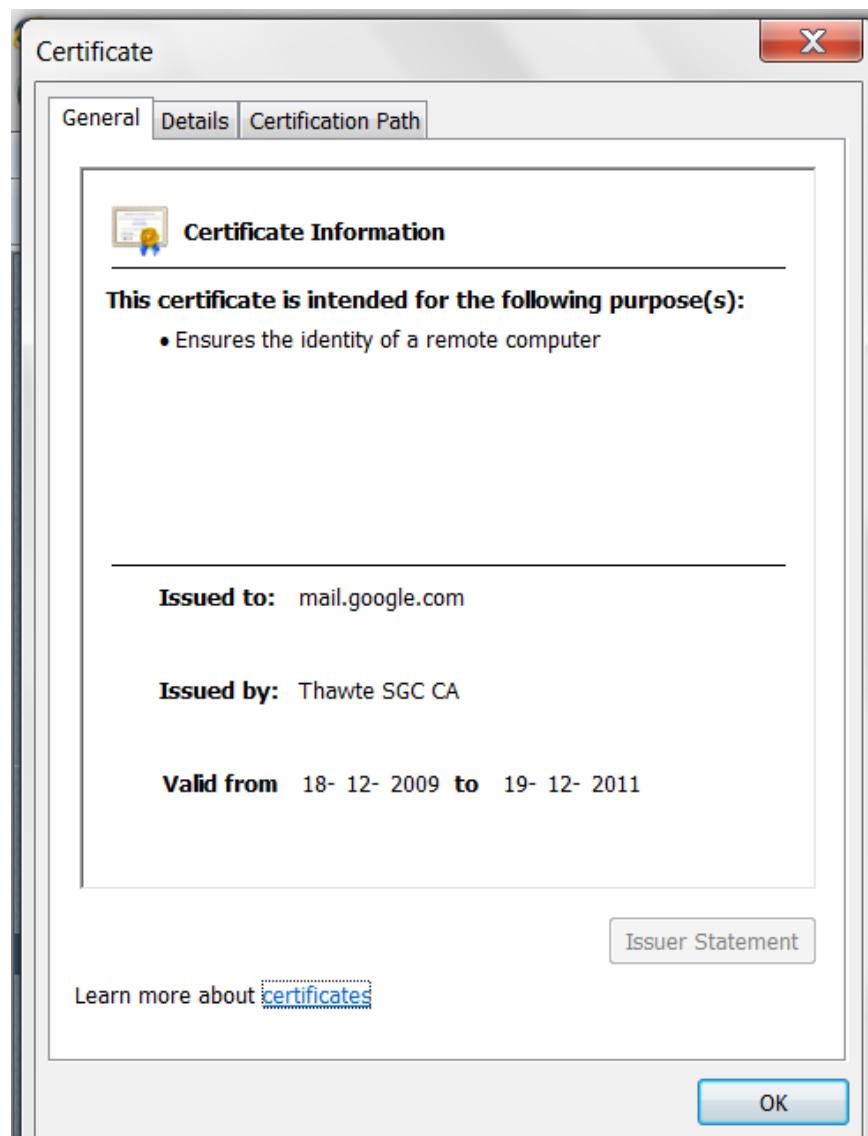
General Details Certification Path

Show Version 1 Fields Only

Field	Value
Serial number	4c e2 9e f3 77 6b 92 eb c...
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	VeriSign Class 3 Extende...
Valid from	19 August 2010 05:30:00
Valid to	19 August 2012 05:29:59
Subject	www.irctc.co.in, IRCTC...
Public key	RSA (2048 Bits)

```
30 82 01 0a 02 82 01 01 00 a9 25 a7 88 ab 10 cf  
3a bb 6e 01 11 23 05 40 5f cb db 10 c8 8f 27 cc  
b3 e7 bd 52 f9 0d 1d 12 c1 7b b7 48 5c 26 c6 e7  
17 86 5d 3a b7 1f 34 b0 00 39 e7 df bd 34 80 17  
61 64 b2 c1 e7 e1 83 eb da 29 e9 1b 1a ab b0 ad  
2c 12 fb d4 8c 58 b9 a6 5a 08 0c 03 40 17 6c c2  
78 5c a3 c5 b1 ae a4 a2 9e de a8 1c 2f 9e 57 de  
71 03 11 0c 1f ed 8c 91 0f dd f9 89 b1 fa ad d9  
c3 2e 9b a2 91 1c 53 01 91 f2 ee a6 02 02 bb f8  
6d 1d e7 63 b0 90 ae 46 09 bf da 60 9e 3f 9e 25  
e8 32 e4 13 8b f1 a2 fb 36 b9 46 6e 17 fb 86 74
```

Edit Properties... Copy to File...



CA's Certificate - Thawte

The image displays three windows from a certificate management application, likely OpenSSL's OpenSSL GUI or a similar tool, illustrating the Thawte SGC CA certificate.

Left Window (Certification Path): Shows the certificate chain. The root is VeriSign Class 3 Public Primary CA, which issued the Thawte SGC CA certificate, which in turn issued the mail.google.com certificate.

Middle Window (Certificate Information): Provides detailed information about the Thawte SGC CA certificate.

- Purpose:** Ensures the identity of a remote computer.
- Issued to:** Thawte SGC CA
- Issued by:** Class 3 Public Primary Certification Authority
- Valid from:** 13-05-2004 to 13-05-2015

Right Window (Raw Hex Dump): Displays the raw hex representation of the certificate's content.

Field	Value
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	Class 3 Public Primary Ce...
Valid from	13 May 2004 05:30:00
Valid to	13 May 2015 05:29:59
Subject	Thawte SGC CA, Thawte ...
Public key	RSA (1024 Bits)
Key Usage	Certificate Signing, Off-lin...

Hex Dump Content:

```
30 81 89 02 81 81 00 d4 d3 67 d0 8d 15 7f ae cd 31  
fe 7d 1d 91 a1 3f 0b 71 3c ac cc c8 64 fb 63 fc 32  
4b 07 94 bd 6f 80 ba 2f e1 04 93 c0 33 fc 09 33 23  
e9 0b 74 2b 71 c4 03 c6 d2 cd e2 2f f5 09 63 cd ff  
48 a5 00 bf e0 e7 f3 88 b7 2d 32 de 98 36 e6 0a ad  
00 7b c4 64 4a 3b 84 75 03 f2 70 92 7d 0e 62 f5 21  
ab 69 36 84 31 75 90 f8 bf c7 6c 88 1b 06 95 7c c9  
e5 a8 de 75 a1 2c 7a 68 df d5 ca 1c 87 58 60 19 02  
03 01 00 01
```

Buttons: View Certificate, OK, Issuer Statement, Edit Properties..., Copy to File..., Learn more about certificates, Learn more about certificate details, OK.

Thawte issues Certificate to Gmail

- The CA signs the certificate with its private key. If the corresponding public key is known to user, then that user can verify that certificate signed by the CA is valid.
Similar to Digital Certificate.
- Because certificates are unforgeable, they can be placed in a directory without need for the directory to make special efforts to protect them.
- Any user with access to CA can get any certificate from it
- Only the CA can modify a certificate

- If all users subscribe to the same CA, then there is a common trust of that CA.
- All user certificates can be placed in the directory for access by all users.
- A user can transmit his or her certificate directly to other users.

- If there is a large community of users, it may not be practical for all users to subscribe to the same CA.
 - Because it is the CA that signs the certificates, each participating user must have a copy of the CA's own public key to verify signatures.

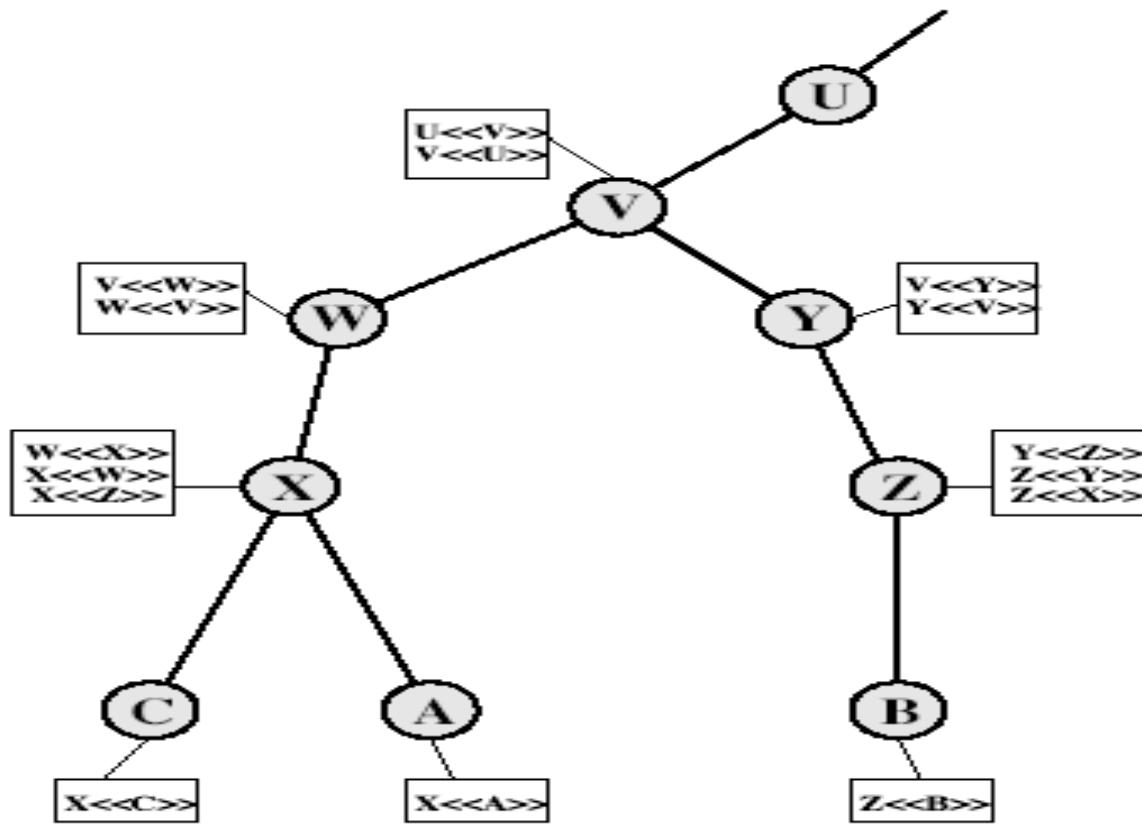
Multiple CAs

- Users in one CA are OK
- What if users from different CAs
 - A from X1
 - B from X2
 - B's certificate is useless to A w/o knowing X2's public key
 - Can work if two CAs exchanged public keys
 - A can use $X1<<X2>>$, $X2<>$
- Chain: $X1<<X2>>$ $X2<<X3>>$... $XN<>$

- Procedure for A to obtain B's public key:
 - A obtains from the directory, the certificate of X2 signed by X1. Because A securely knows X1's public key, A can obtain X2's public key from its certificate and verify it by means of X1's signature on the certificate.
 - A then goes back to the directory and obtains the certificate of B signed by X2. Because A now has a trusted copy of X2's public key, A can verify the signature and securely obtain B's public key.

- In X.509, this chain of certificates is expressed as
 $X_1 << X_2 >> X_2 << B >>$
- B obtains A's public key with the reverse chain
 $X_2 << X_1 >> X_1 << A >>$
- A chain with n elements is
 $X_1 << X_2 >> X_2 << X_3 >> \dots X_n << B >>$
where each pair (X_i, X_{i+1}) of CAs in the chain have created certificates for each other (stored in the directory).

CA Hierarchy Use



User A can acquire the following certificates from the directory to establish a connection path to B:

X << W >> W << V >> V << Y >> Y << Z >> Z << B >>

and then unwrap to obtain a trusted copy of B's public key.

Similarly, B can obtain A's public key from

Z << Y >> Y << V >> V << W >> W << X >> X << A >>

Certificate Revocation

- Certificates have a period of validity
- may need to revoke before expiry, eg:
 1. user's private key is compromised
 2. user is no longer certified by this CA
 3. CA's certificate is compromised
- CA's maintain list of revoked certificates
 - the Certificate Revocation List (CRL)
- users should check with CA's CRL

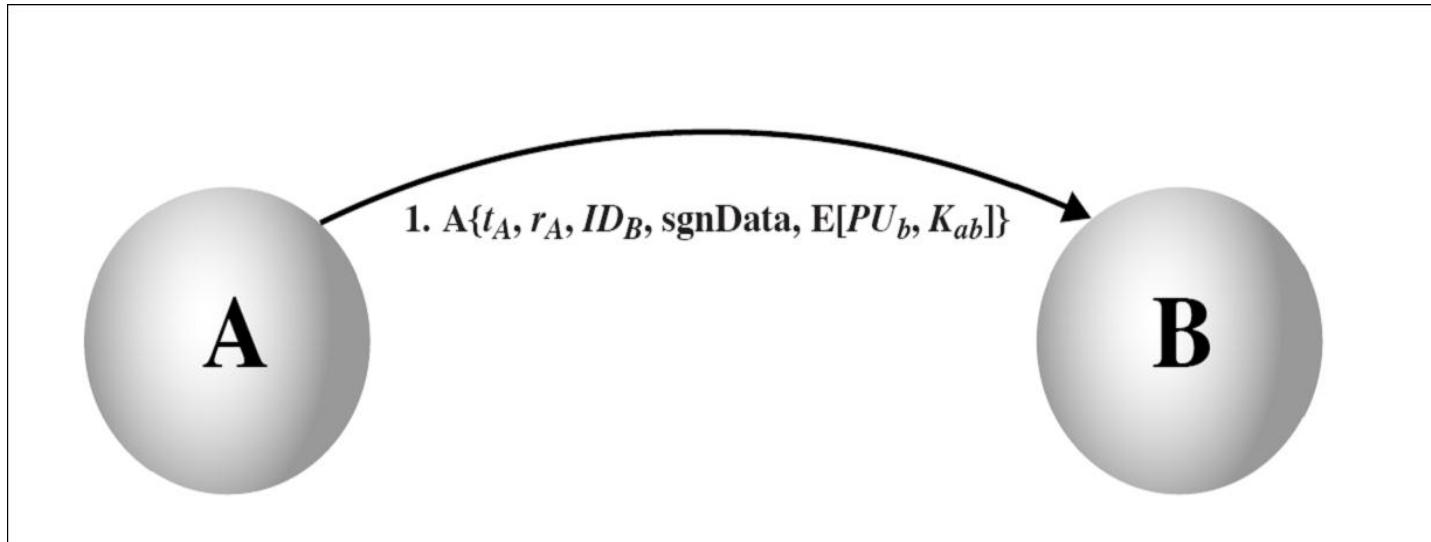
Authentication Procedures

- X.509 includes three alternative authentication procedures:
 - Assumes each already knows the certified public key of the other
 - One-Way Authentication
 - Two-Way Authentication
 - Three-Way Authentication
 - all use public-key signatures

One-Way Authentication

- 1 message (A->B) used to establish
 - the identity of A and that message is from A
 - message was intended for B
 - integrity & originality of message
- message must include timestamp, nonce, B's identity and is signed by A

One-Way Authentication



- Establishes:
 - identity of A and that the message was generated by A.
 - message was intended for B
 - integrity and originality (not replayed) of message
- t_A = timestamp,
- r_A = nonce (unique within expiration time),
- sgnData = information (guaranteed authenticity and integrity),
- K_{ab} = optional session key

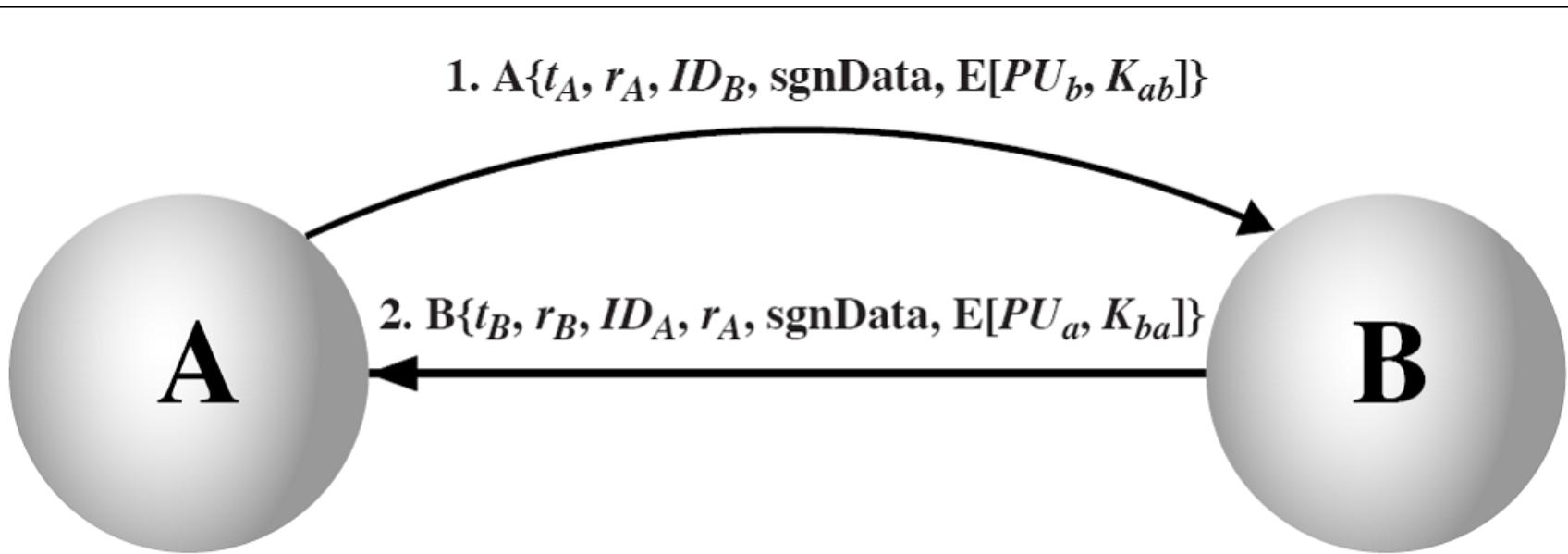
The one-way protocol is:

- (1) Alice generates a random number, R_A .
- (2) Alice constructs a message, $M = \{T_A, R_A, I_B, d\}$, where T_A is Alice's time-stamp, I_B is Bob's identity, and d is an arbitrary piece of data. The data may be encrypted with Bob's public key, E_B , for security.
- (3) Alice sends $\{C_A, D_A(M)\}$ to Bob. (C_A is Alice's certificate; D_A is Alice's private key.)
- (4) Bob verifies C_A and obtains E_A . He makes sure these keys have not expired. (E_A is Alice's public key.)
- (5) Bob uses E_A to decrypt $D_A(M)$. This verifies both Alice's signature and the integrity of the signed information.
- (6) Bob checks the I_B in M for accuracy.
- (7) Bob checks the T_A in M and confirms that the message is current.
- (8) As an option, Bob can check R_A in M against a database of old random numbers to ensure the message is not an old one being replayed.

Two-Way Authentication

- 2 messages ($A \rightarrow B$, $B \rightarrow A$) which also establishes in addition:
 - the identity of B and that reply is from B
 - that reply is intended for A
 - integrity & originality of reply
- reply includes original nonce from A, also timestamp and nonce from B

Two-Way Authentication



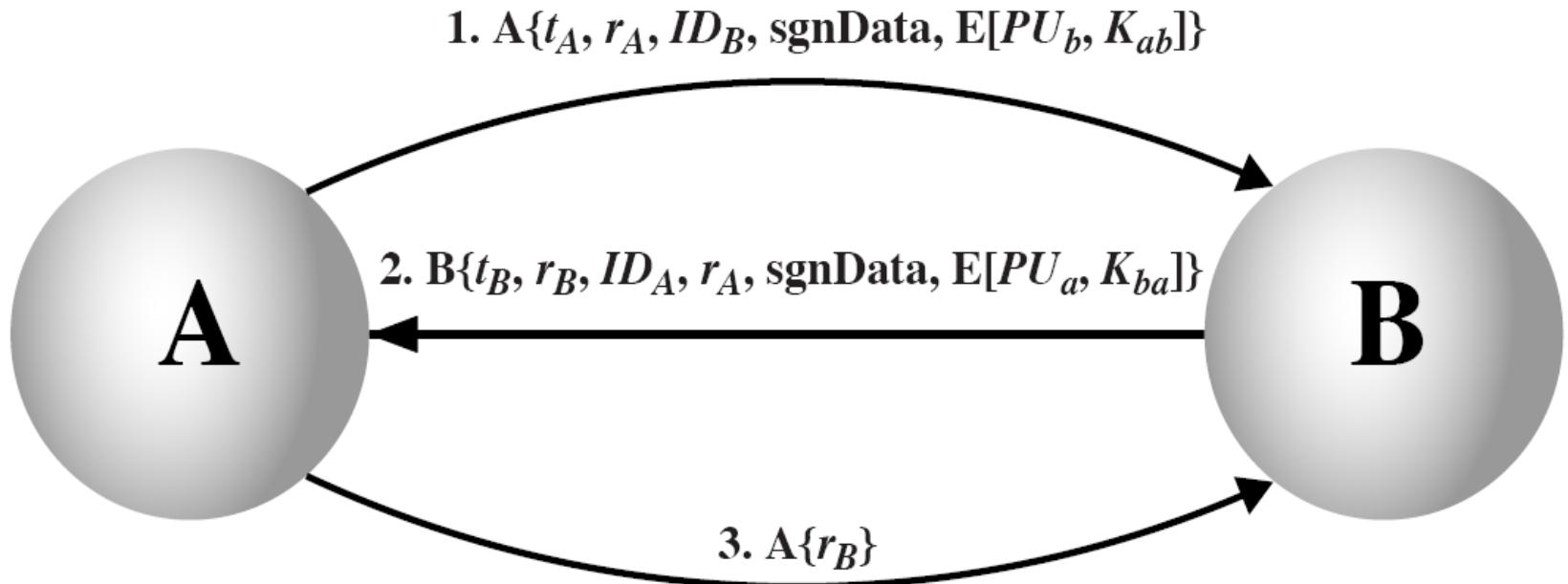
- In addition to one-way authentication, establishes:
 - identity of B and that the reply message was generated by B
 - message was intended for A
 - Integrity and originality of the reply

The two-way protocol consists of the one-way protocol and then a similar one-way protocol from Bob to Alice. After executing steps (1) through (8) of the one-way protocol, the two-way protocol continues with:

- (9) Bob generates another random number, R_B .
- (10) Bob constructs a message $M' = [T_B, R_B, I_A, R_A, d]$, where T_B is Bob's time-stamp, I_A is the identity of Alice and d is arbitrary data. The data may be encrypted with Alice's public key, E_A , for security. R_A is the random number Alice generated in step (1).
- (11) Bob sends $D_B[M']$ to Alice.
- (12) Alice uses E_B to decrypt $D_B[M']$. This verifies both Bob's signature and the integrity of the signed information.
- (13) Alice checks the I_A in M' for accuracy.
- (14) Alice checks the T_B in M' and confirms that the message is current.
- (15) As an option, Alice can check the R_B in M' to ensure the message is not an old one being replayed.

Three-Way Authentication

- Used when synchronized clocks are not available
- 3 messages (A->B, B->A, A->B) which enables the authentication
- has reply from A back to B containing signed copy of nonce from B
 - means doesn't require the transfer of timestamps



- Timestamps need not be checked, signed nonces used

The three-way protocol accomplishes the same thing as the two-way protocol, but without timestamps. Steps (1) through (15) are identical to the two-way protocol, with $T_A = T_B = 0$.

- (16) Alice checks the received version of R_A against the R_A she sent to Bob in step (3).
- (17) Alice sends $D_A(R_B)$ to Bob.
- (18) Bob uses E_A to decrypt $D_A(R_B)$. This verifies both Alice's signature and the integrity of the signed information.
- (19) Bob checks the received version of R_B against the R_B he sent to Alice in step (10).

PGP



Email security



- *Email security is important*
- *If messages are not secure,*
 - *They may be viewed in transit on the network*
 - *Viewed by suitably privileged users on destination systems (e.g. sys admins)*
 - *Easy to masquerade as someone*
 - *There are emailers you can use to send emails claiming you are Bill.Gates@microsoft.com*
 - *Message integrity upon reception is not guaranteed*

Email Security and PGP

- PGP *can be used for email security*
 - It stands for Pretty Good Privacy
 - It is a software application created by Phil Zimmermann
 - Public-key rings is the main feature

From Wikipedia page on PGP:

In 2003, an incident involving seized Psion PDAs belonging to members of the Red Brigade indicated that neither the Italian police nor the FBI were able to decode PGP-encrypted files stored on them.

A more recent incident in December 2006 involving US customs agents and a seized laptop PC which allegedly contained child pornography indicates that US Government agencies find it “nearly impossible” to access PGP-encrypted files. Additionally, a judge ruling on the same case in November 2007 has stated that forcing the suspect to reveal his PGP pass-phrase would violate his Fifth Amendment rights i.e. a suspect’s constitutional right not to incriminate himself.

PGP Features

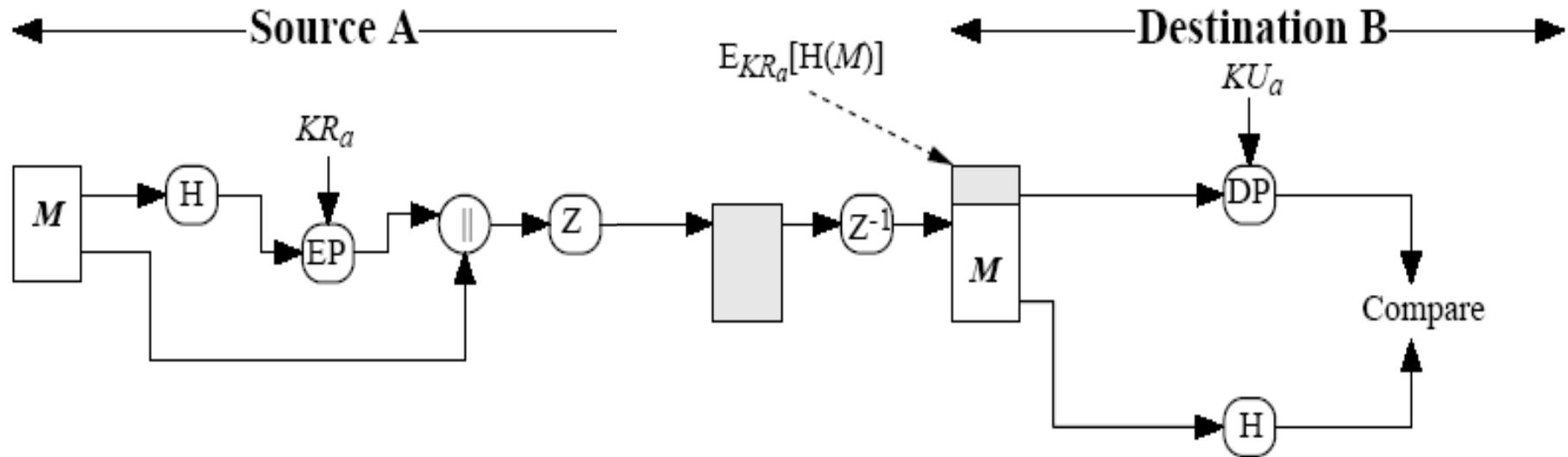
- It is based on the best available crypto algorithms
 - Considered very strong and secure
- Mainly used for email and file storage applications
- Independent of governmental organizations
- Messages are automatically compressed

PGP Components

- There are **five** important services in PGP
 - Authentication (Sign/Verify)
 - Confidentiality (Encryption/Decryption)
 - Confidentiality can be used for storing files locally or transmitting them over insecure channel.
 - Compression
 - Email compatibility
 - Segmentation and Reassembly

PGP: Authentication steps

- **Sender:**
 - Creates a message
 - Hashes it to 160-bits using SHA1
 - Encrypts the hash code using her private key, forming a signature
 - Attaches the signature to message
- **Receiver:**
 - Decrypts attached signature using sender's public key and recovers hash code
 - Recomputes hash code using message and compares with the received hash code
 - If they match, accepts the message



M = original message

H = hash function

\parallel = concatenation (join)

Z = compression

Z^{-1} = decompression

EP = public key encryption

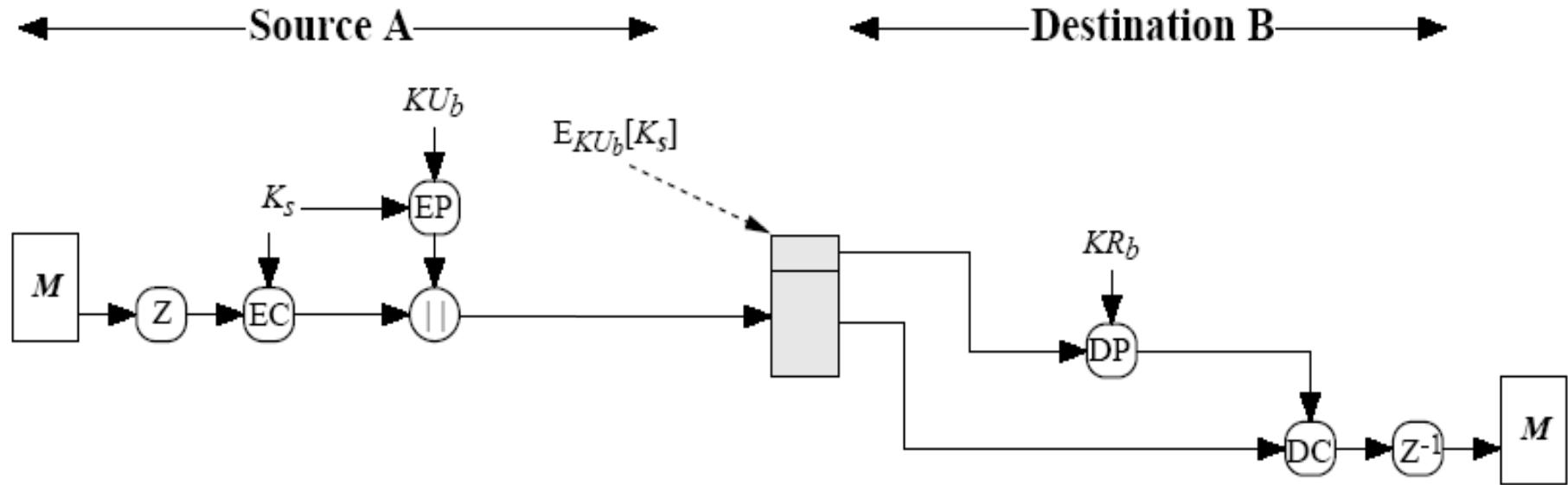
DP = public key decryption

KR_a = A's private key

KU_a = A's public key

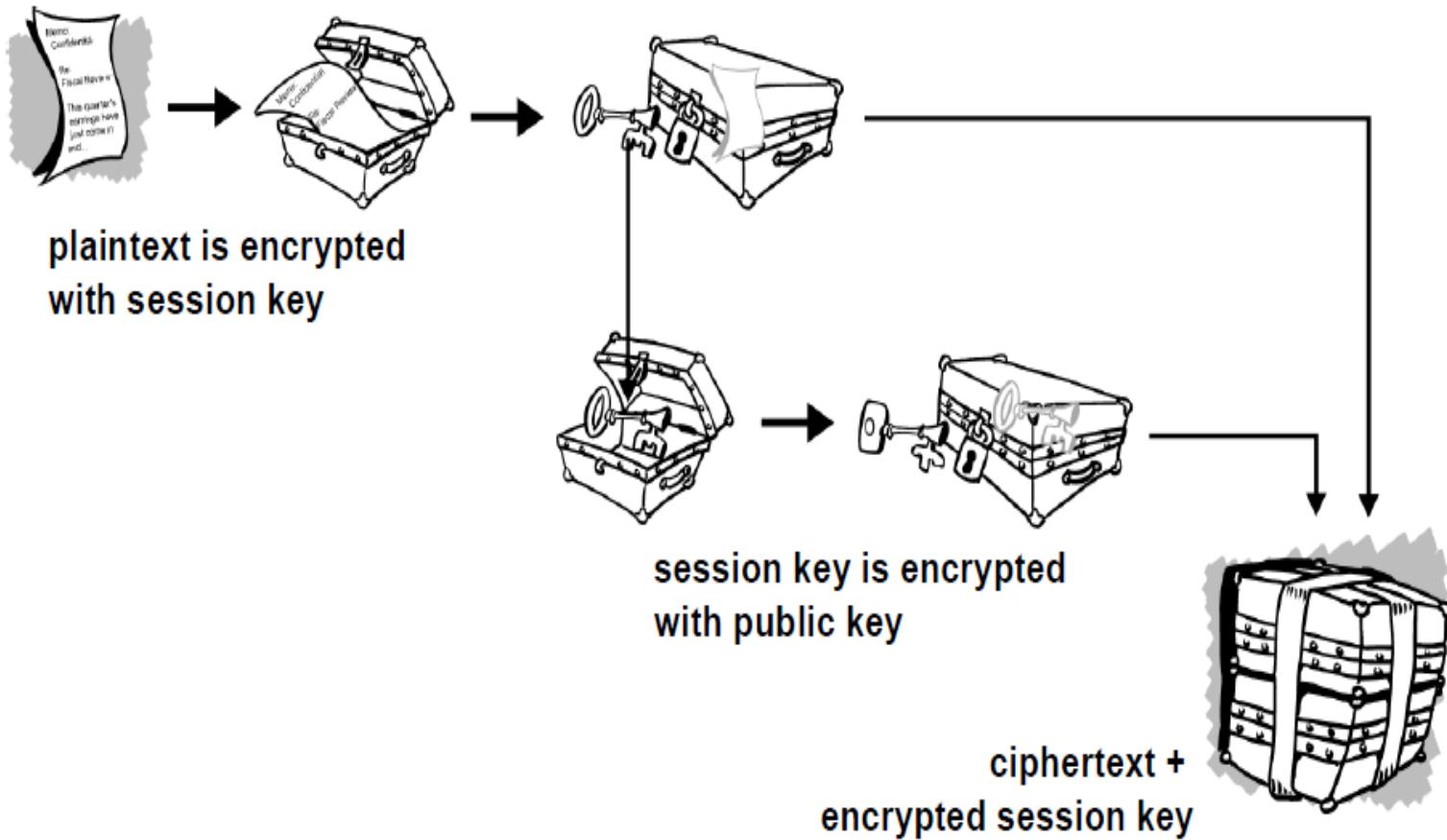
PGP: Confidentiality

- **Sender:**
 - Generates message and a random number (session key) only for this message
 - Encrypts message with the session key using AES, 3DES, etc..
 - Encrypts session key itself with recipient's public key using RSA
 - Attaches it to message
- **Receiver:**
 - Recovers session key by decrypting using his private key
 - Decrypts message using the session key.

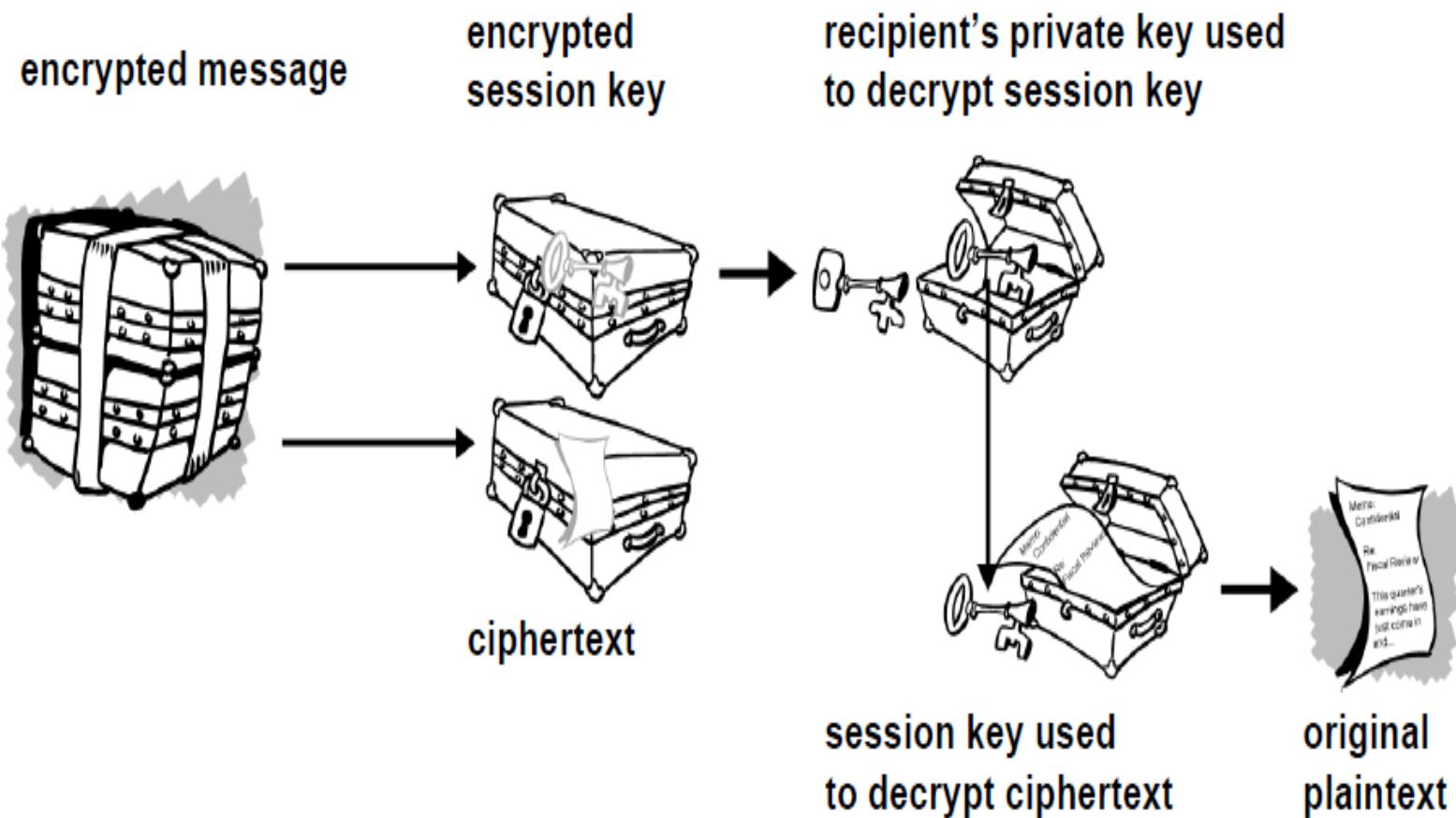


EC = symmetric encryption
DC = symmetric decryption
 K_s = session key

PGP Encryption



PGP Decryption

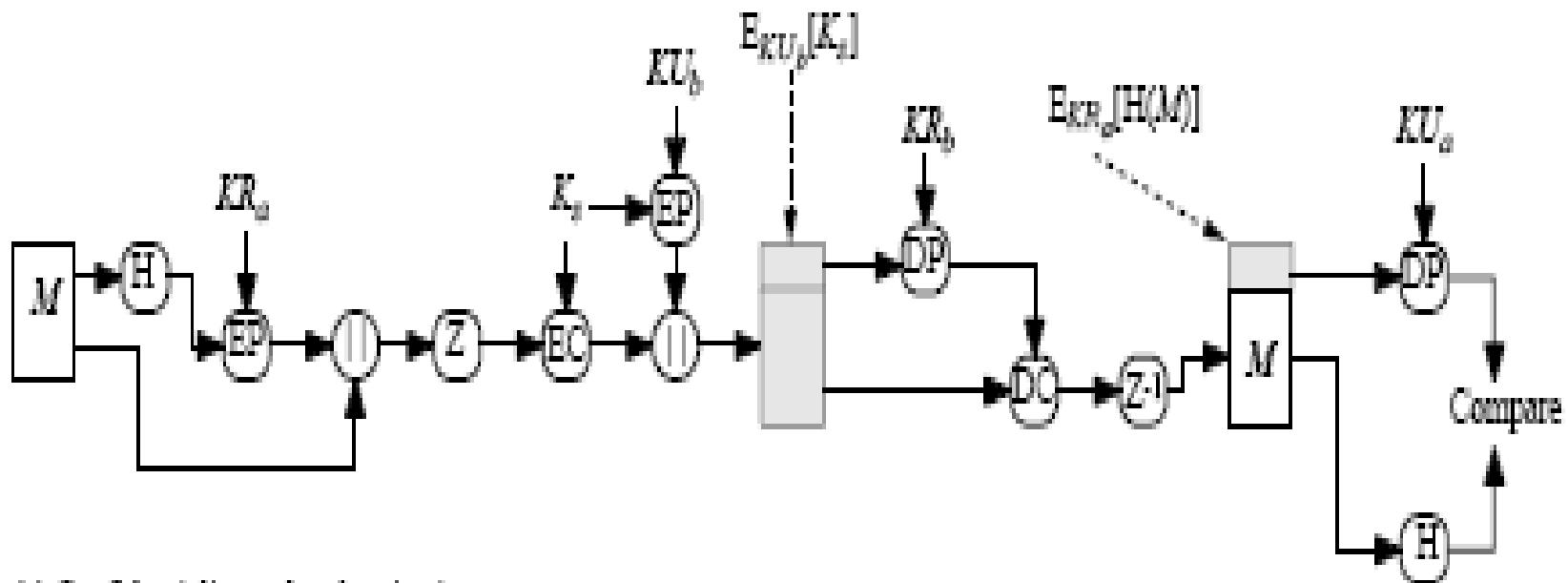


Confidentiality pitfall

- Here, confidentiality service provides no assurance to the receiver as to the identity of sender (i.e. no authentication)
- Only provides confidentiality for sender that only the recipient can read the message (and no one else)

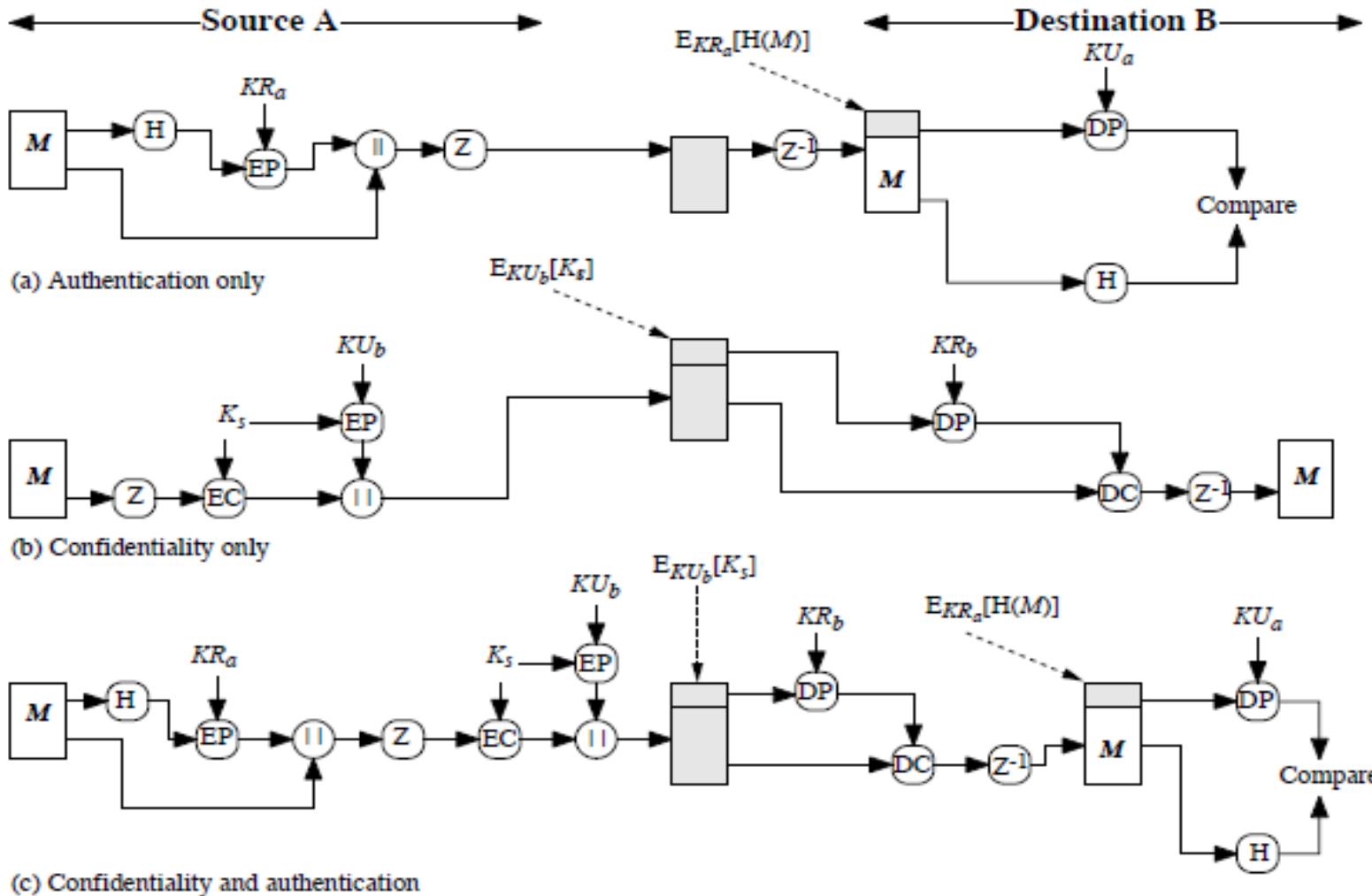
Combining authentication and confidentiality in PGP

- Authentication and confidentiality can be combined
 - A message can be both signed **and** encrypted
- That is called **authenticated confidentiality**
- Encryption/Decryption process is “nested” within the process shown for authentication alone.



(c) Confidentiality and authentication

PGP Modes



PGP Compression

Compression is done before encryption

When a user encrypts plaintext with PGP, PGP first compresses the plaintext.

Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security.

Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher.

Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis. (redundancy reduced)

uses ZIP compression algorithm

PGP Operation – Email Compatibility

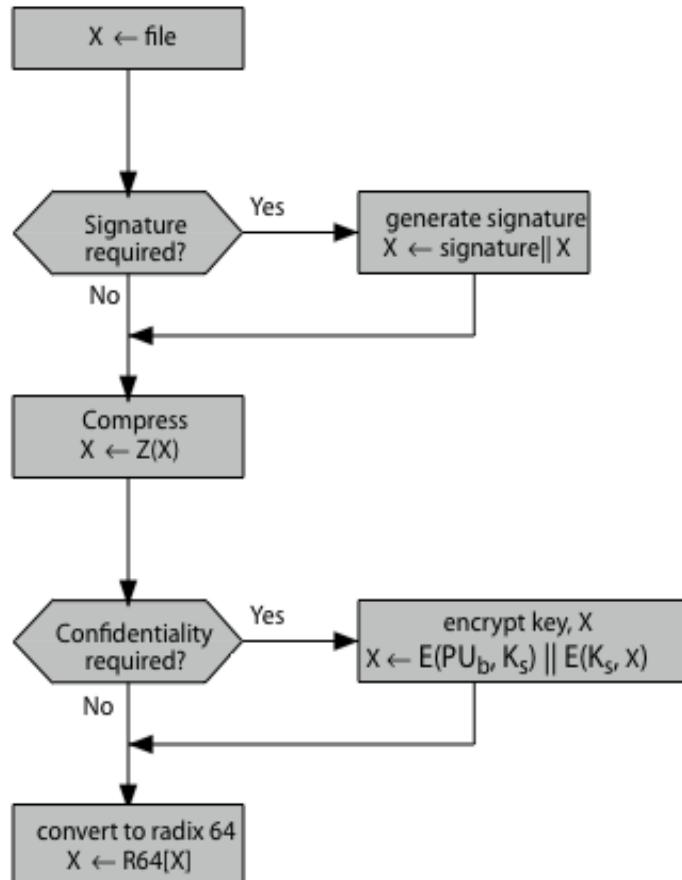
- when using PGP will have binary data to send (encrypted message, etc)
- however email was designed only for text
- hence PGP must encode raw binary data into printable ASCII characters
- uses radix-64 algorithm
 - maps 3 bytes to 4 printable chars
- PGP also segments messages if too big (maximum length 50,000 octets)

PGP Email compatibility

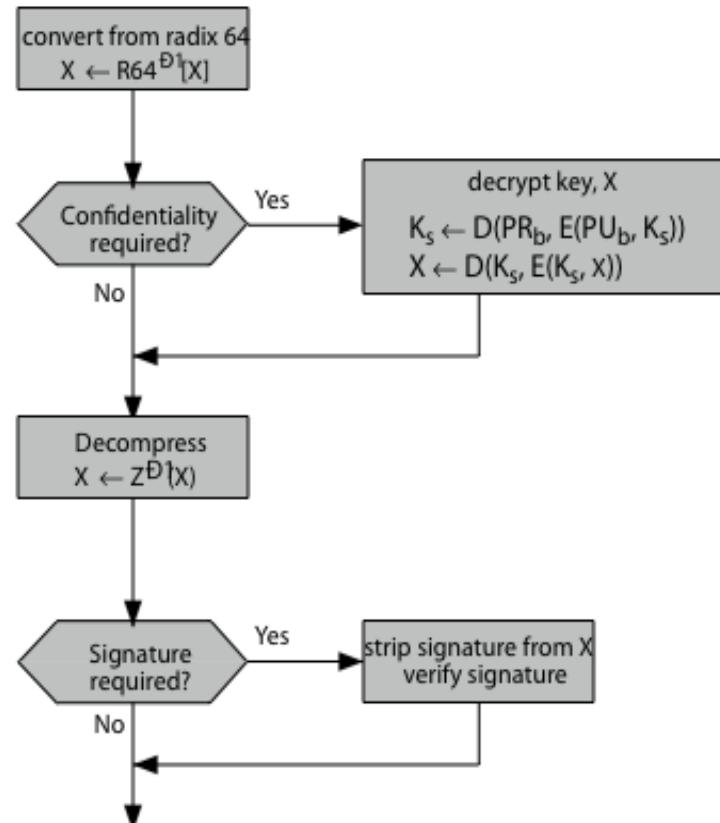
- Output of encryption and compression functions is divided into 6-bit blocks
 - Each block is mapped onto an ASCII Character
 - This is called RADIX-64 encoding
 - Has the side-effect of increasing the size of the data by about 33%

6-bit value	character encoding						
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
					(pad)		=

PGP Operation – Summary



(a) Generic Transmission Diagram (from A)



(b) Generic Reception Diagram (to B)

PGP Segmentation/Reassembly

- Email protocols have a maximum allowed size for messages
 - Like 100 KB
- PGP divides messages that are too large into smaller ones
 - Divide and conquer
- Reassembly at the receiving end is required before verifying signature or decryption

Algorithms for PGP Services

Function	Algorithm Used
Digital Signature	DSS/SHA or RSA/SHA
Message Encryption	AES, triple DES, IDEA or CAST (symmetric) with Diffie-Hellman or RSA (public key)
Compression	ZIP
E-mail Compatibility	Radix-64 conversion
Segmentation	-

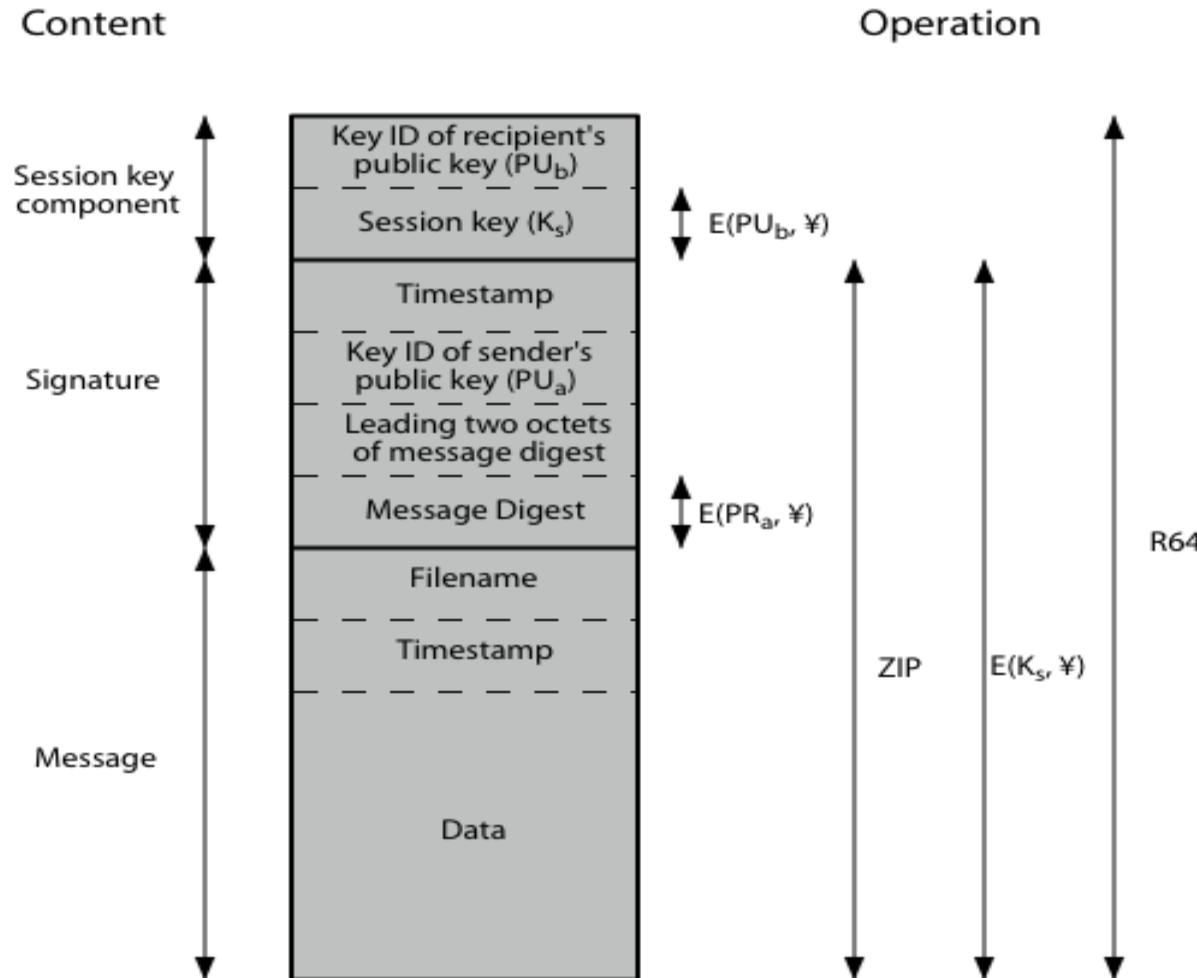
PGP Key Identifiers

- What is a key identifier
- Consider this:
 - A user may have many public/private key pairs at his disposal
 - He wishes to encrypt or sign a message using one of his keys
 - How does he let the other party know which key he has used?
 - Attaching the whole public key every time is inefficient
- Solution?

PGP Key Identifiers

- Generate a key identifier (least significant 64-bits of the key)
 - Key ID of public key K_{Ua} is $(K_{Ua} \bmod 2^{64})$
- This will most likely be unique and can also be used for signatures

PGP Message Format



PGP Key Rings

- PGP uses key rings to identify the key pairs that a user **owns** or **trusts**
- Private-key ring contains public/private key pairs of keys he owns
- Public-key ring contains public keys of others he trusts

PGP Key Rings

- Keys are stored in encrypted form. PGP stores the keys in two files on your hard disk; one for public keys and one for private keys. *These files are called keyrings.*
- As you use PGP, you will typically add the public keys of your recipients to your public keyring.
- Your private keys are stored on your private keyring.
- If you lose your private keyring, you will be unable to decrypt any information encrypted to keys on that ring.

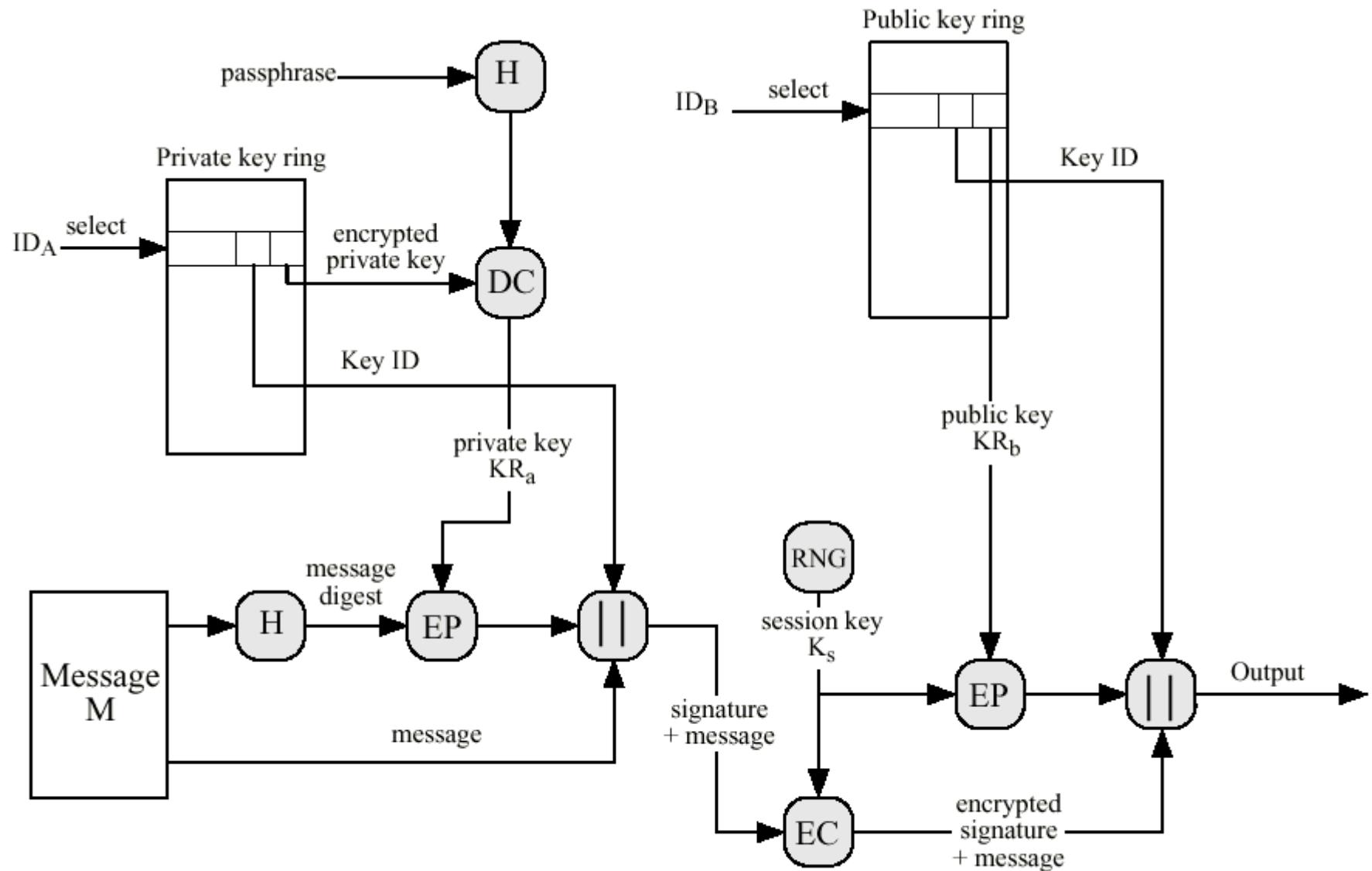


Figure 12.5 PGP Message Generation (from User A to User B; no compression or radix 64 conversion) 94

PGP Message Generation

- Signing the message
 - PGP retrieves the sender's private key from private-key ring using your user-id as an index.
 - PGP prompts the user for the passphrase to recover the unencrypted private key.
 - The signature component of the message is constructed.
- Encrypting the message
 - PGP generates a session key and encrypts the message.
 - PGP retrieves the recipient's public key from the public-key ring using her user-id as an index.
 - The session key component of the message is constructed.

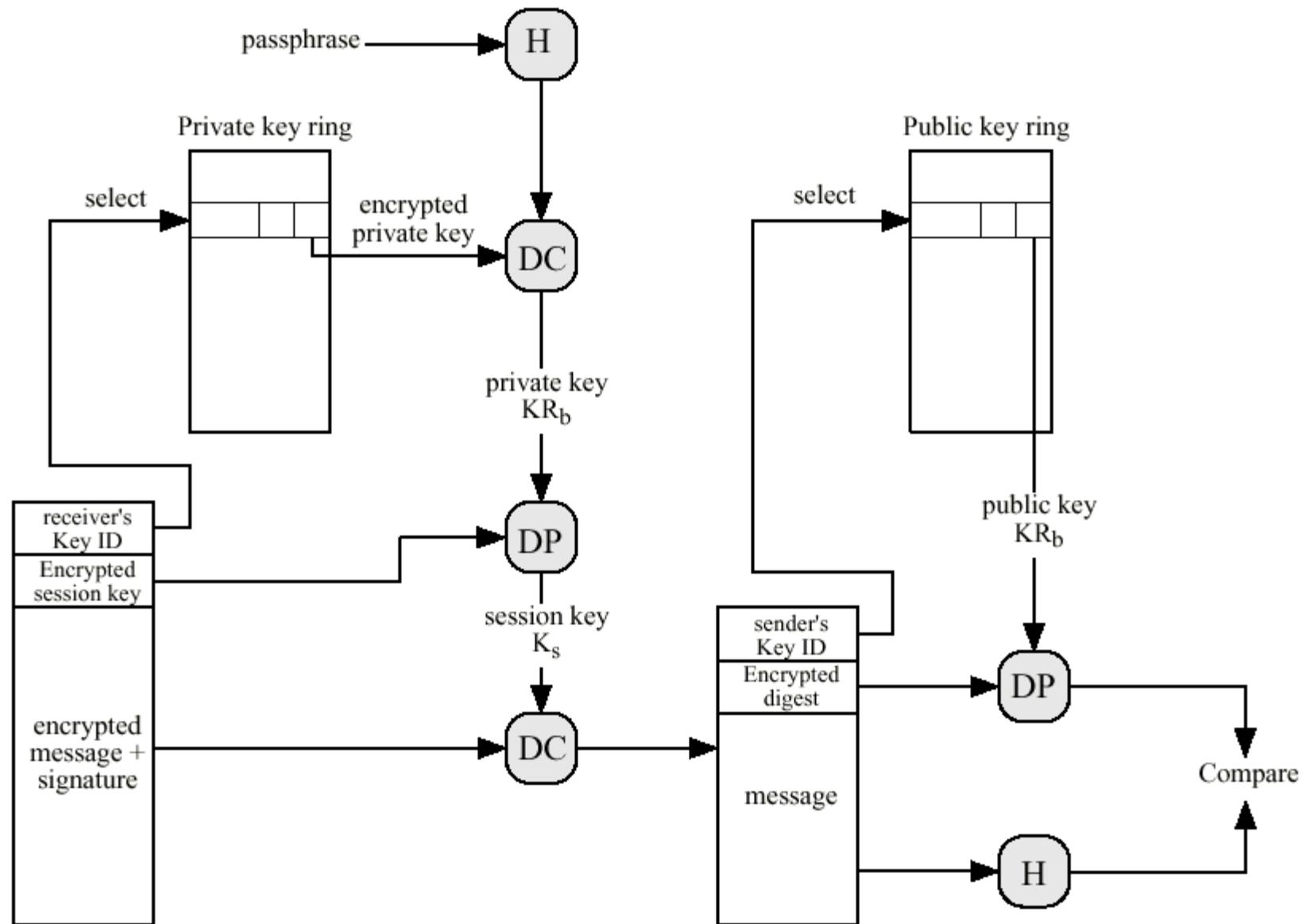


Figure 12.6 PGP Message Reception (from User A to User B; no compression or radix 64 conversion) 96

PGP Message Reception

- Decrypting the message
 - PGP retrieves the receiver's private key from the private-key ring using key ID field in the session key component of the message as an index.
 - PGP prompts the user for the passphrase to recover the unencrypted private key
 - PGP then recovers the session key and decrypts the message.
- Authenticating the message
 - PGP retrieves the sender's public key from the public-key ring, using the key-ID field in the signature key component of the message as an index
 - PGP recovers the transmitted message digest.
 - PGP recomputes the message digest for the received message and compares it to the transmitted message digest to authenticate.

S/MIME

SMTP scheme limitations

1. SMTP cannot transmit executable files or other binary files.
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.

What is S/MIME?

- When email was first developed, people could only send plain text messages
- MIME was developed in early 90s to allow people to send pictures, sound, programs and general attachments -- “Multipurpose Internet Mail Extension”
 - MIME provides a standardized way of dealing with a wide variety of information representations in a multimedia environment.
- MIME has no security features, can be read along its route or forged (easily)
- S/MIME is a secure version of MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard.

S/MIME is *likely to emerge as the industry standard* for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many.

What does S/MIME give us?

- *Secrecy* – Only intended recipient can read the message.
- *Authentication* – Recipient knows the message came from the apparent sender.
- *Integrity* – Recipient knows the message was not changed en route.

S/MIME employs....

- Secret key (symmetric cipher)
- Public key Cryptography – PKC (asymmetric cipher)
 - Secrecy
 - Authentication
 - Secrecy and authentication
- Hashing (message digest)
- Public key certificate (X.509)

- S/MIME puts all these techniques together to create a practical, efficient, reasonably secure email protocol
- Standard (symmetric) cipher – RC2 or TripleDES
- Public key (asymmetric) cipher – RSA
- Hashing – SHA-1 or MD5

S/MIME

Cryptographic algorithms

Function	Requirement
Create a message digest to be used in forming a digital signature.	MUST support SHA-1 and MD5. SHOULD use SHA-1.
Encrypt message digest to form digital signature.	Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.
Encrypt session key for transmission with message.	Sending and receiving agents MUST support Diffie-Hellman. Sending agent SHOULD support RSA encryption with key sizes 512 bits to 1024 bits. Receiving agent SHOULD support RSA decryption.
Encrypt message for transmission with one-time session key.	Sending agents SHOULD support encryption with tripleDES and RC2/40. Receiving agents MUST support decryption using tripleDES and SHOULD support decryption with RC2/40.

S/MIME for secrecy only

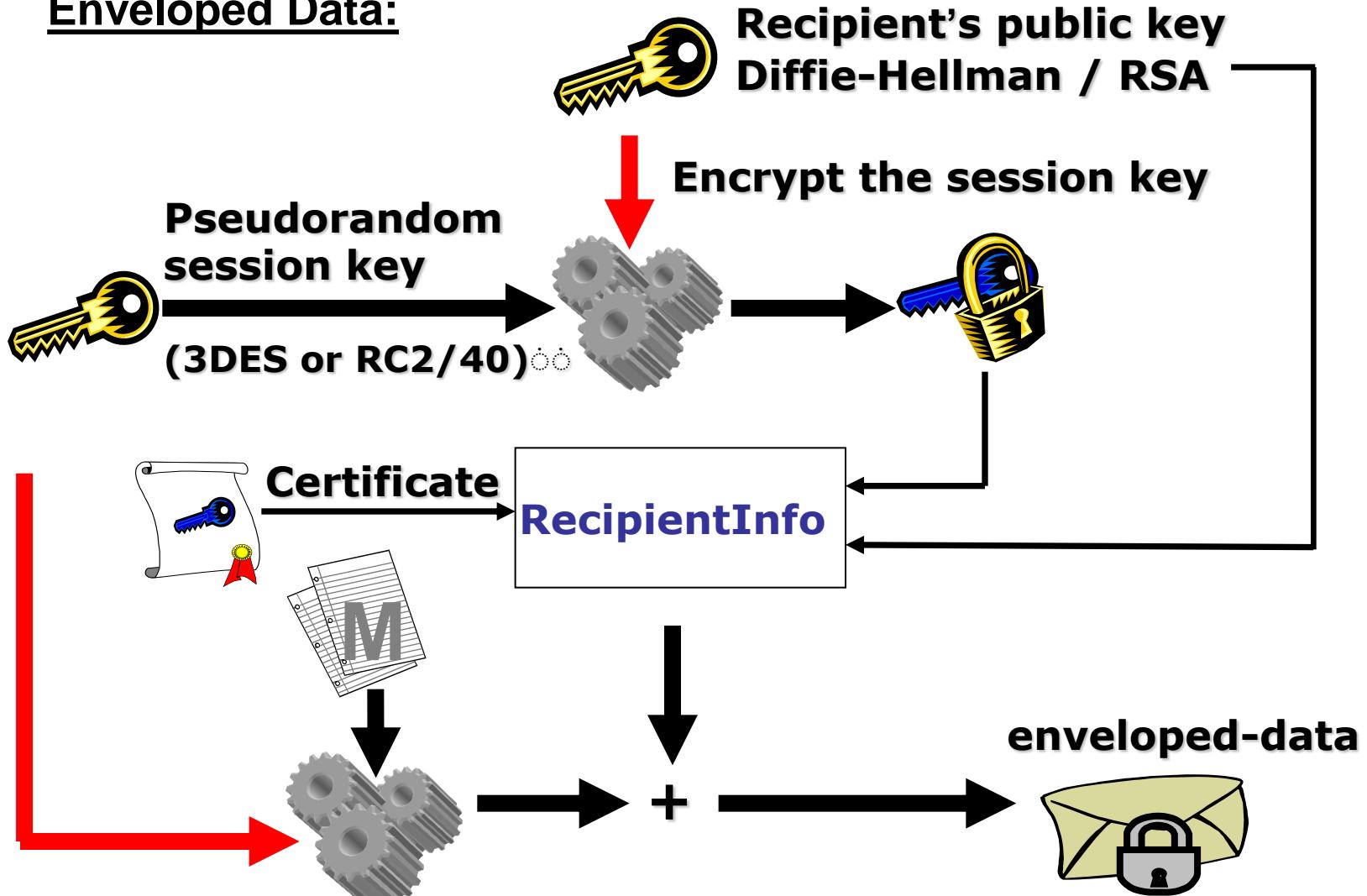
1. Alice's email program creates a random key (*session key*) to be used in a symmetric cipher.
2. Alice's email program encrypts the message with the symmetric cipher and session key.
3. Alice's email program encrypts the session key with PKC and Bob's public key.
4. Alice's email program creates a package of: encrypted message, encrypted session key, his X.509 certificate, names of encryption algorithms.

S/MIME for secrecy....

5. Alice's email program sends package to Bob. This is an S/MIME email message.
6. Bob's email program receives package.
7. Bob's email program uses her private key (and named PKC method) to decrypt the session key.
8. Bob's email program uses session key (and named symmetric cipher) to decrypt the message.

S/MIME - Message

Enveloped Data:



S/MIME for authentication only

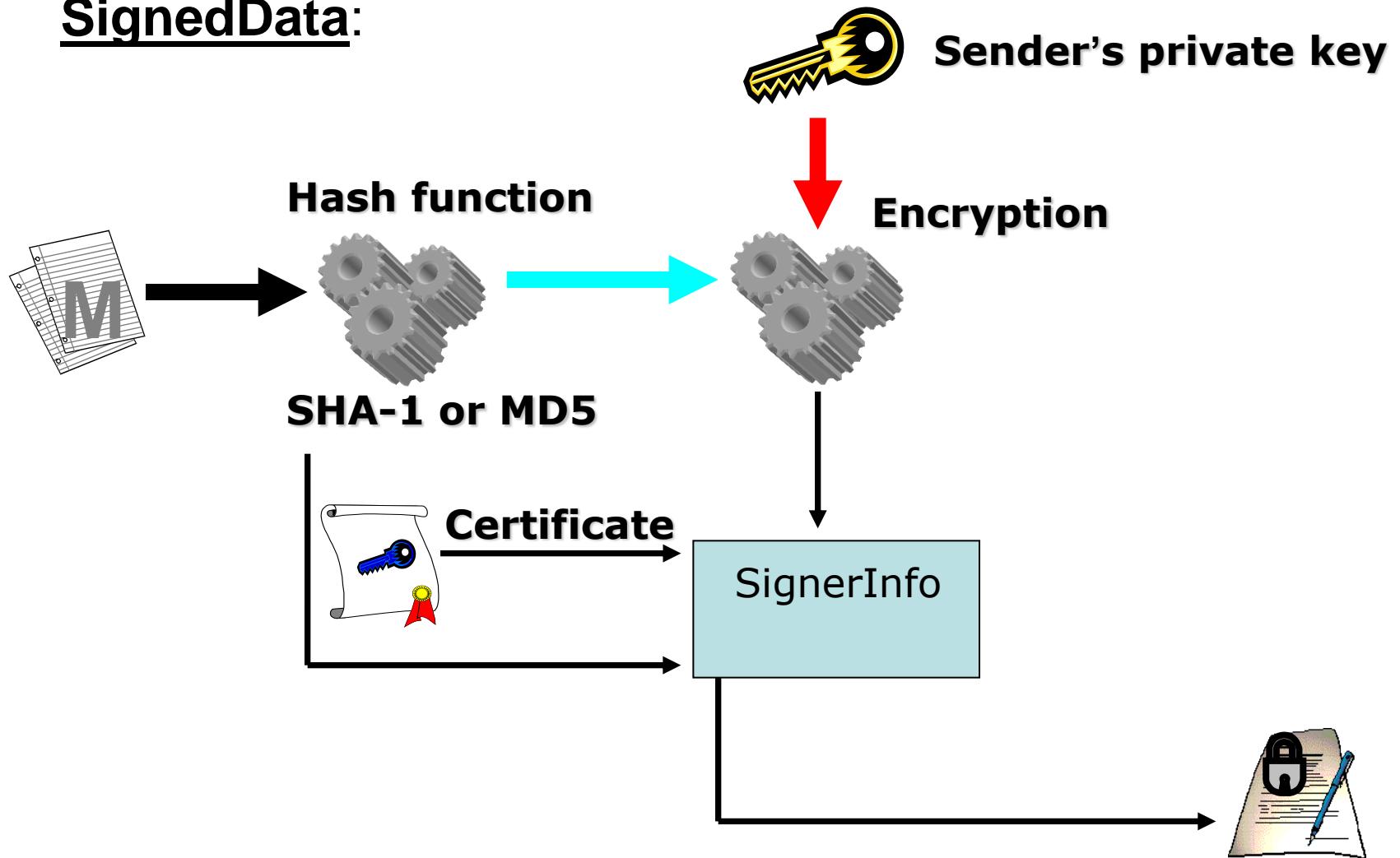
1. Alice's email program uses hash function to create message digest
2. Alice's email program encrypts message digest with PKC and his private key
3. Alice's email program creates a package of: original message, encrypted message digest, his X.509 certificate, names of encryption algorithms
4. Alice's email program sends package to Bob.
5. Bob's email program receives package

S/MIME for authentication....

6. Bob's email program verifies Alice's X.509 certificate by testing signature of CA
7. Bob's email program gets Alice's public key from his certificate
8. Bob's email program uses Alice's public key to decrypt the message digest
9. Bob's email program independently computes the message digest, using the same hash function
10. Bob's email program compares the two message digests to verify sender and message integrity

S/MIME - Message

SignedData:



S/MIME for secrecy and authentication

1. Message is authenticated just as shown above
2. Authenticated package is made secret, just as shown above
3. Secret package is sent to recipient
4. Receiver uses his/her private key to decrypt session key
5. Receiver uses session key to decrypt rest of secret package, yielding authenticated message
6. Receiver authenticates message, just as shown above

S/MIME Security

- Public-key tampering can be detected using trusted CAs
- Most replay attacks can be eliminated using timestamps
- Signing, or the authentication of sender with digital signatures

MIME content types

Type	Subtype	Description
Text	Plain Enriched	Unformatted text (ASCII or ISO 8859). Provides greater format flexibility.
Multipart	Mixed Parallel Alternative Digest	The different parts are independent but are to be transmitted together. Should be presented to the receiver in their original order. Differs from mixed only in that no order is defined. The different parts are alternative versions of the same information. Similar to Mixed but the default type/subtype of each part is message/rfc822.
Message	rfc822 Partial External body	The body is itself an encapsulated message that conforms to RFC822. Used to allow fragmentation in a transparent way to the recipient. Contains a pointer to an object exists else where.

Type	Subtype	Description
Image	Jpeg gif	The image is in JPEG format. The image is in GIF format.
Video	Mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8kHz
Application	Postscript Octet-stream	Adobe Postscript. General binary data consisting of 8-bit bytes.

A MIME attachment with the content type "application/octet-stream" is a binary file. Typically, it will be an application or a document that must be opened in an application, such as a spreadsheet or word processor.

IPSec

Internet Threats

- **Data integrity**

The contents of a packet can be accidentally or deliberately modified.

- **Identity spoofing**

The origin of an IP packet can be forged.

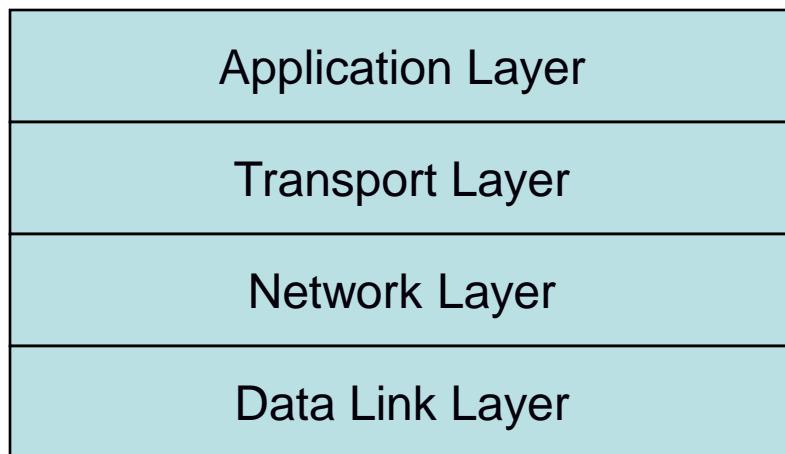
- **Anti-reply attacks**

Unauthorized data can be retransmitted.

- **Loss of privacy**

The contents of a packet can be examined in transit.

Security at What Level?



PGP, Kerberos etc.

Transport Layer Security (TLS)

IP Security

Hardware encryption

Security at Application Layer

- PGP, Kerberos, etc.
- Implemented in end-hosts
- Advantages
 - Extend application without involving operating system.
 - Application can understand the data and can provide the appropriate security.
- Disadvantages
 - Security mechanisms have to be designed independently of each application.

Security at Transport Layer

Transport Layer Security (TLS)

- Implemented in end-hosts
- Advantages
 - Existing applications get security seamlessly
- Disadvantages
 - Protocol specific

Security at Data Link Layer

- (Hardware encryption)
- Need a dedicated link between host/routers.
- Advantages
 - Speed.
- Disadvantages
 - Not scalable.
 - Need dedicated links.

Security at Network Layer

IP Security (IPSec)?

IP Security (IPSec)

- IPSec is a framework of open standards developed by the Internet Engineering Task Force (IETF).

Creates **secure, authenticated, reliable communications over IP networks**

Goals of IPSec

- to verify sources of IP packets
 - *authentication*
- to prevent replaying of old packets
- to protect integrity and/or confidentiality of packets
 - *Data Integrity/Data Encryption*

IPSec Security Services

- **Connectionless integrity**

Assurance that received traffic has not been modified. Integrity includes anti-reply defenses.

- **Data origin authentication**

Assurance that traffic is sent by legitimate party or parties.

- **Confidentiality (encryption)**

Assurance that user's traffic is not examined by non-authorized parties.

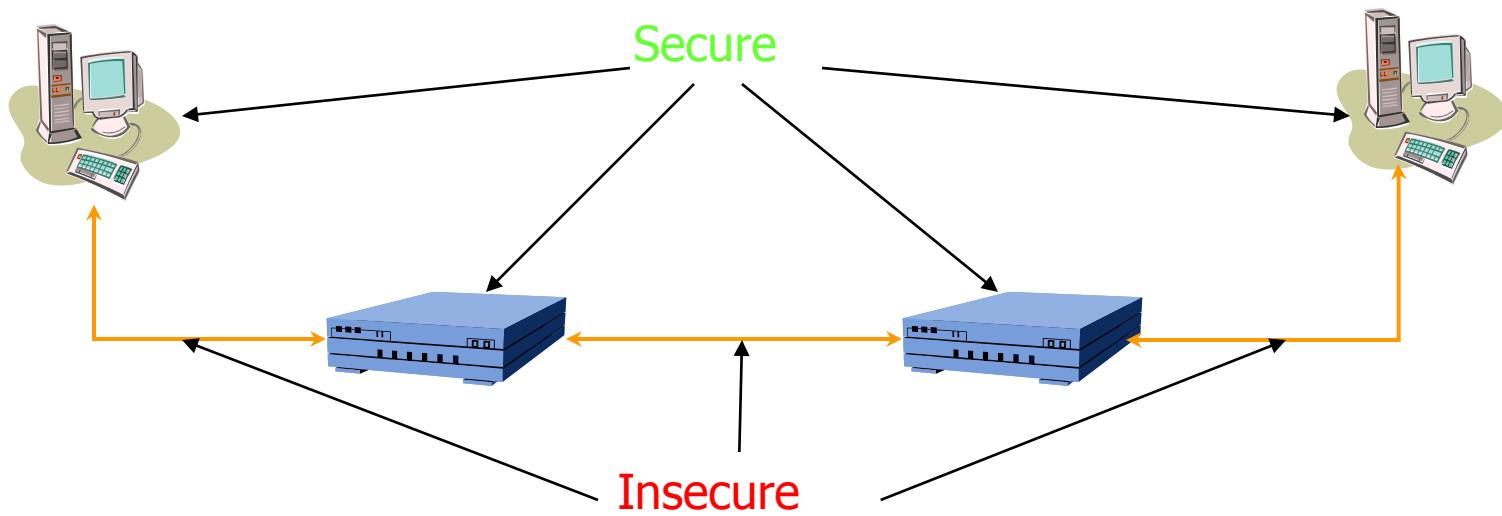
- **Access control**

Prevention of unauthorized use of a resource.

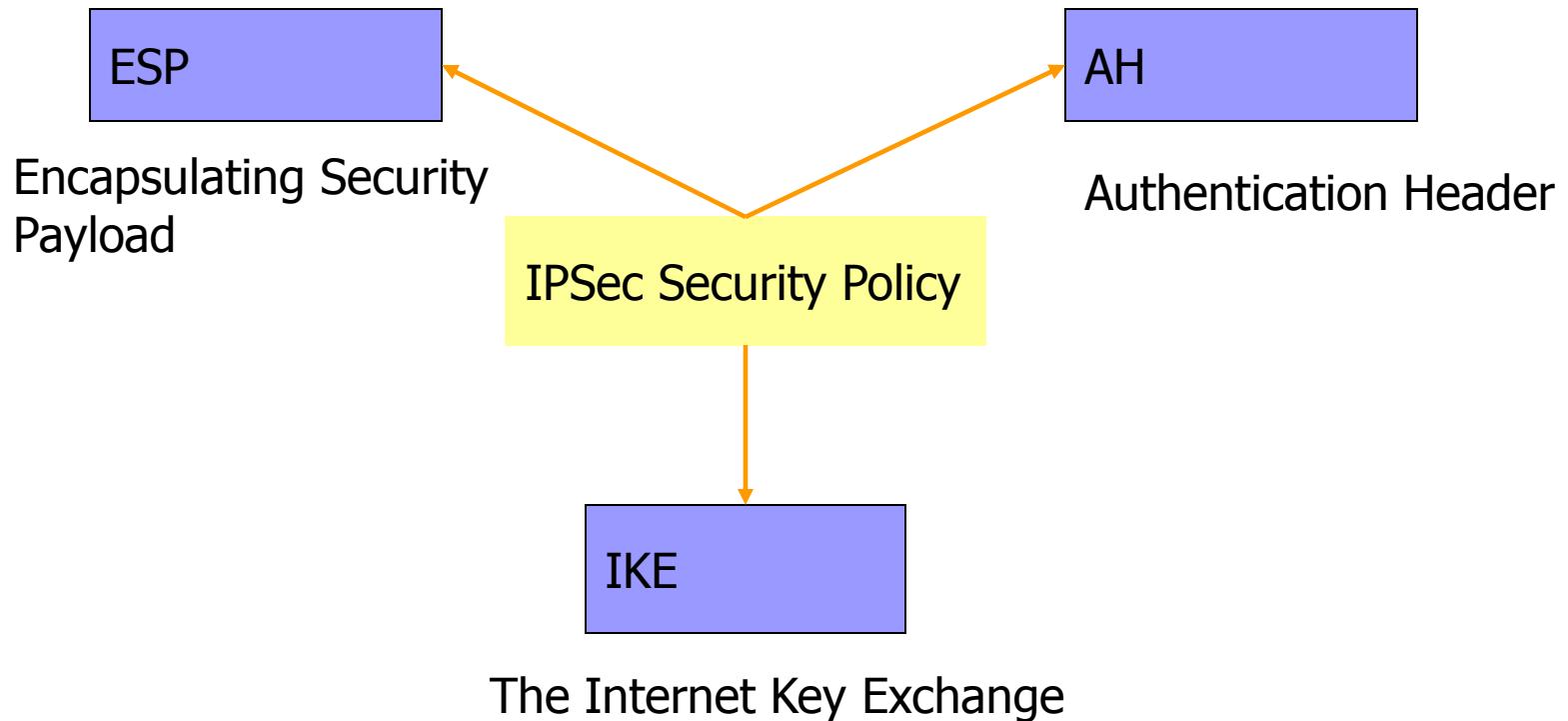
Where can IPSec be used

- These protocols can operate in
 - networking devices,
 - such as a router or firewall
 - or they may operate directly on the workstation or server.

The IPSec Security Model



IPSec Architecture

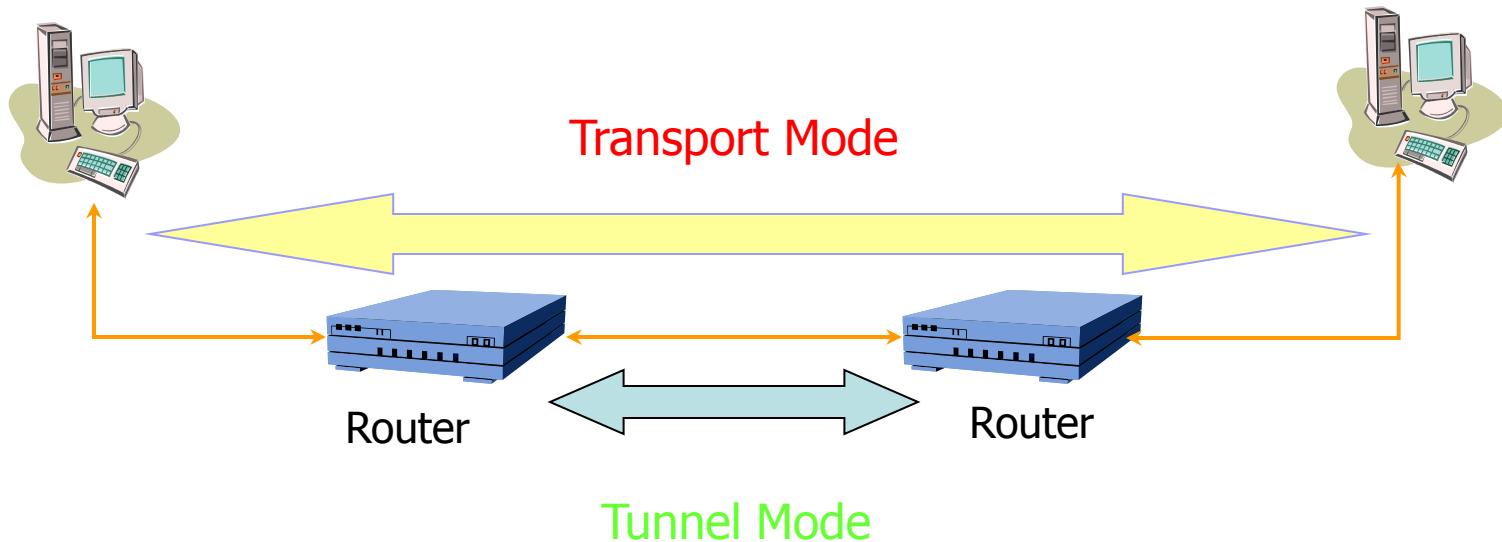


IPSec Architecture

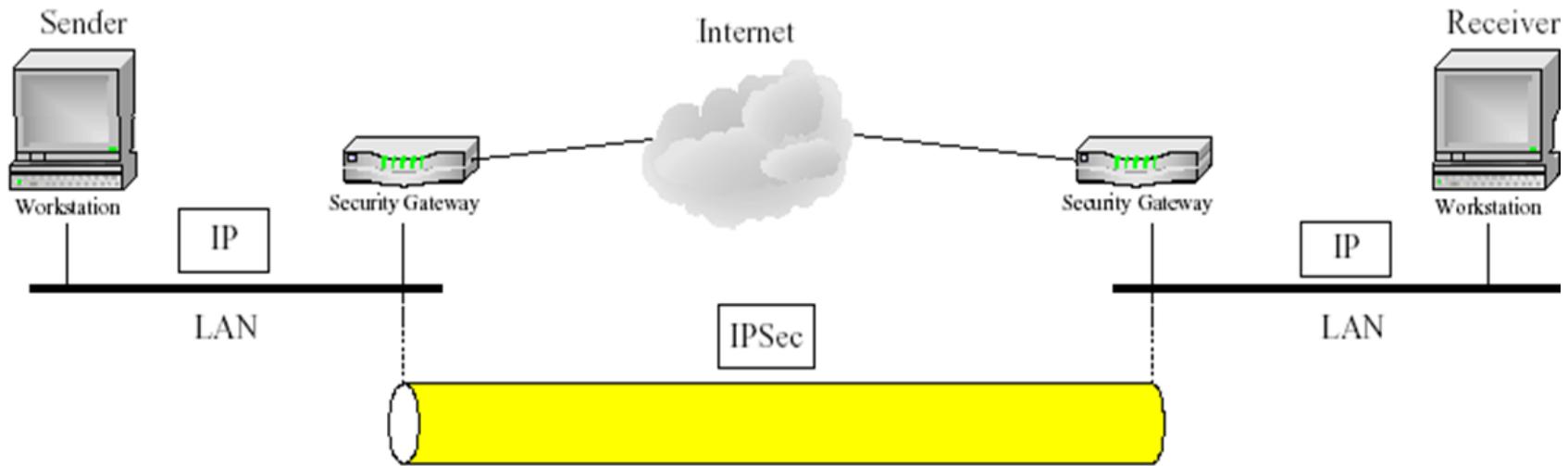
- IPSec provides security in three situations:
 - Host-to-host, host-to-gateway and gateway-to-gateway
 - Example: Router to router, Firewall to router, PC to router, PC to server
- IPSec operates in two modes:
 - *Transport mode* (for end-to-end)
 - *Tunnel mode* (for VPN)

- IPSec has two encryption modes: *tunnel* and *transport*.
 - Tunnel encrypts the header and the payload of each packet while transport only encrypts the payload.
- Only systems that are IPSec compliant can take advantage of this protocol.
- Also, all devices must use a common key and the firewalls of each network must have very similar security policies set up.

IPsec Architecture



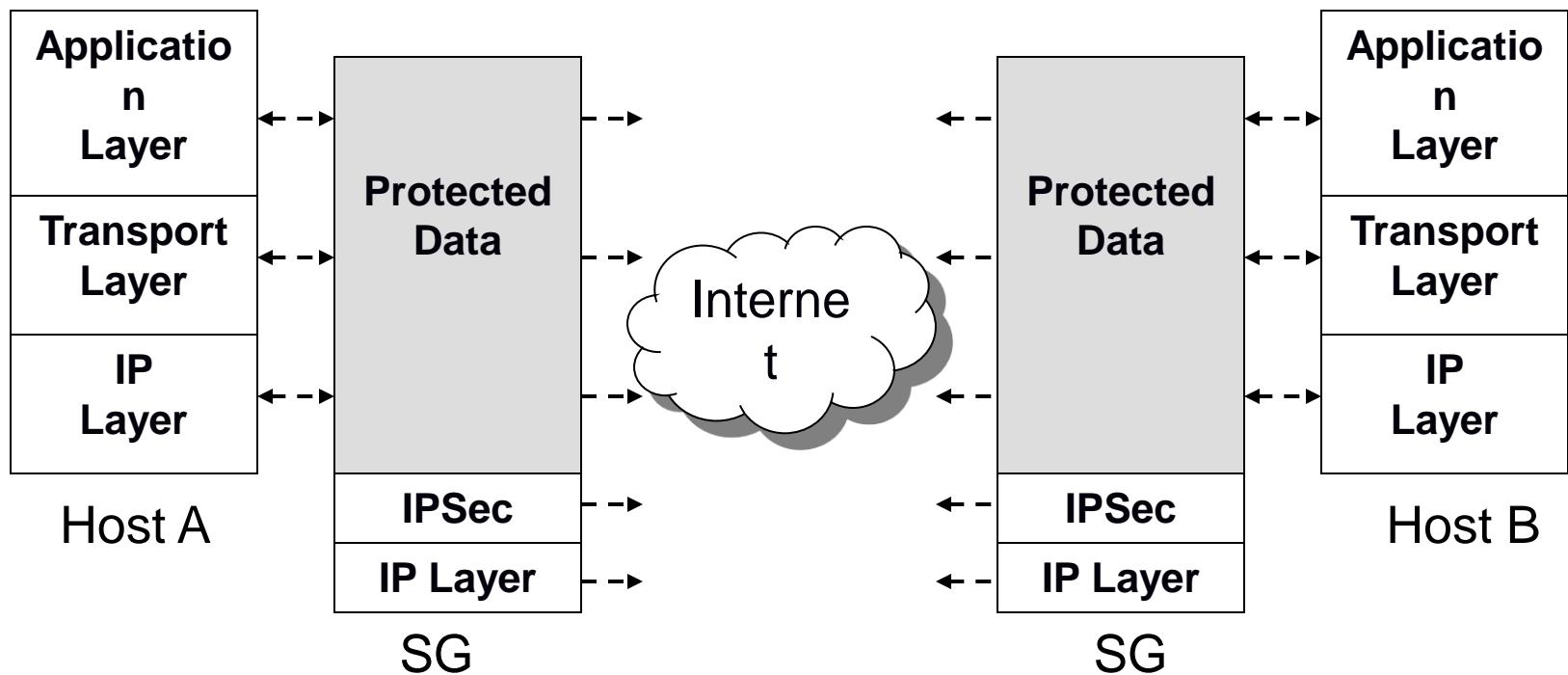
Tunnel Mode



- IPSec implemented between security gateways

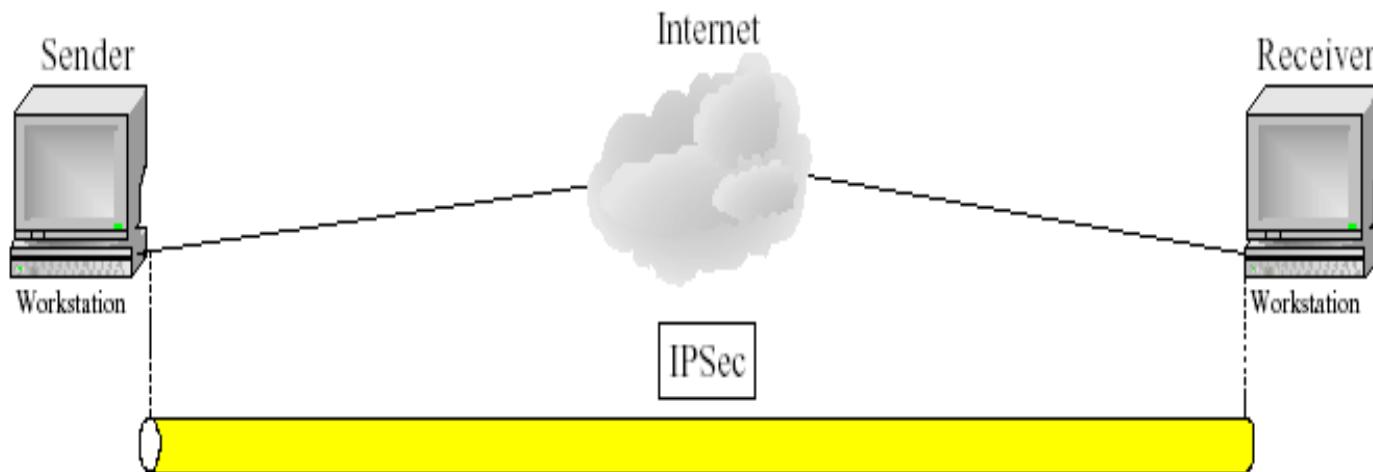
Tunnel Mode

- Host-to-Network, Network-to-Network



SG = Security Gateway

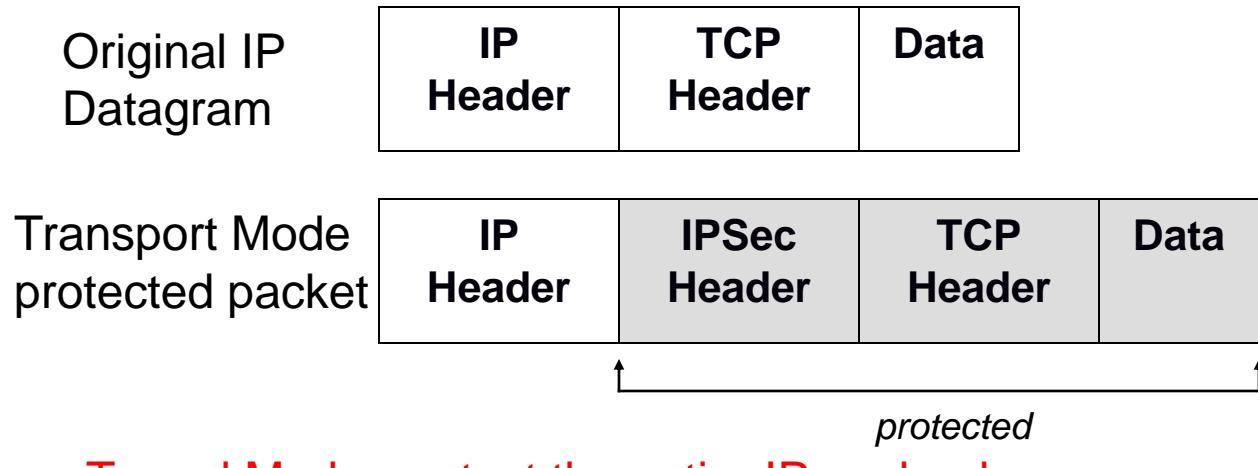
Transport Mode



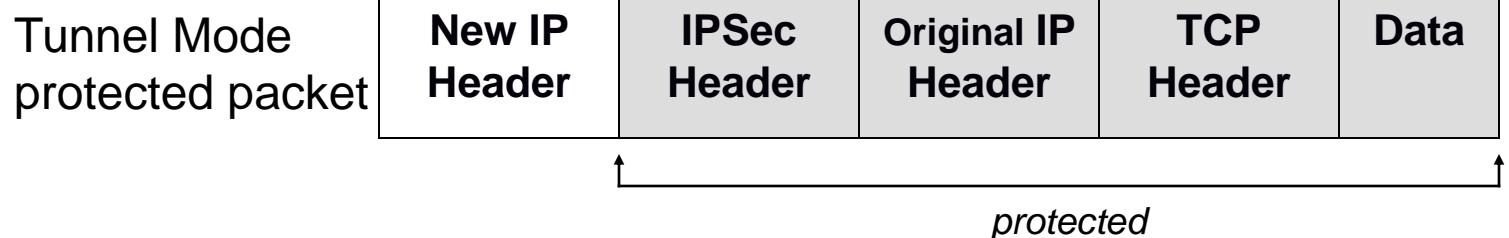
IPSec implementation in end-to-end communication scheme

IPSec Modes of Operation

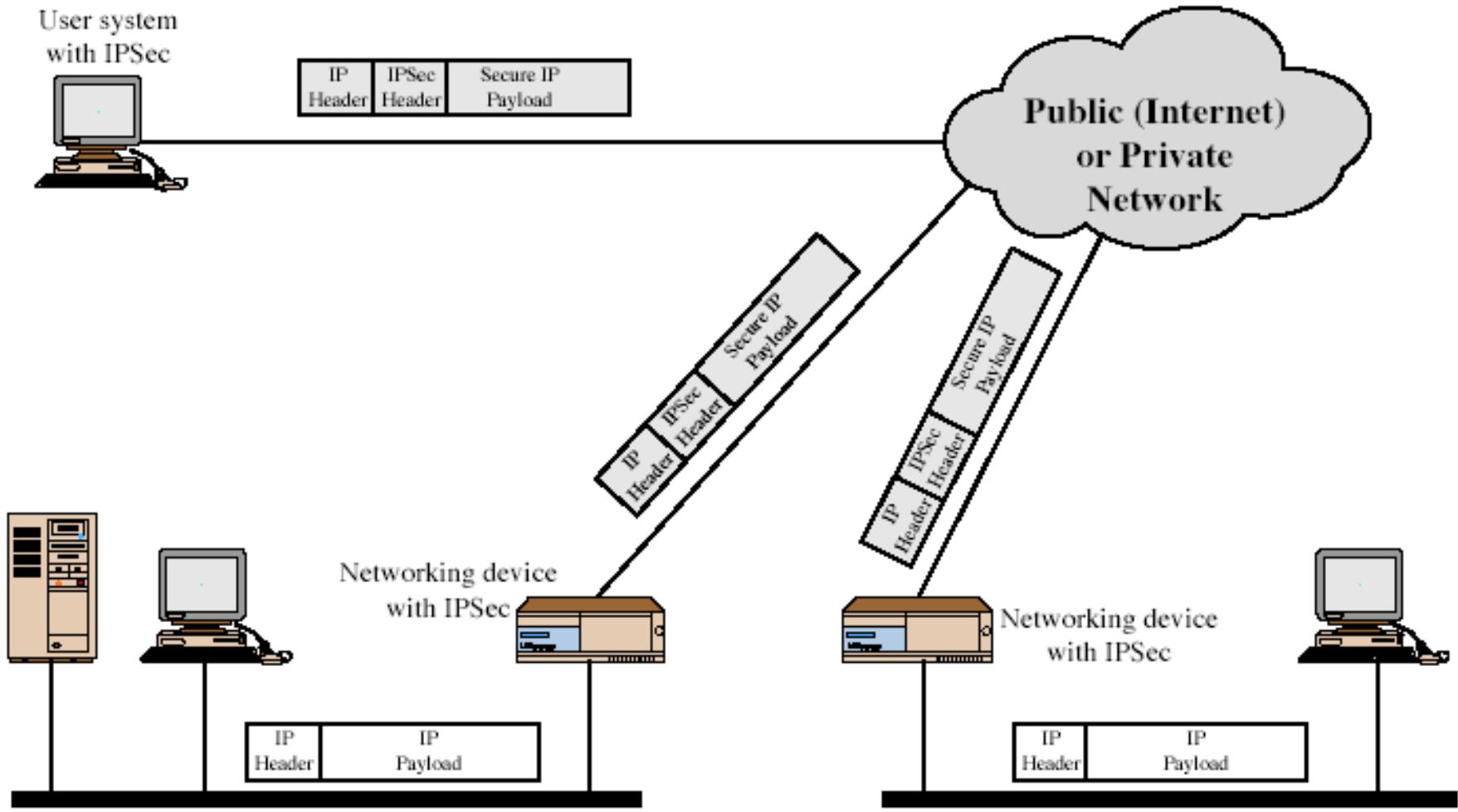
- Transport Mode: protect the upper layer protocols



- ◆ Tunnel Mode: protect the entire IP payload



IPSec Uses



IPSec Security Protocols

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

IPSec Security Protocols

- Authentication Header (AH) provides:
 - Connectionless integrity
 - Data origin authentication
 - Protection against replay attacks
- Encapsulating Security Payload (ESP) provides:
 - Confidentiality (encryption)
 - Connectionless integrity
 - Data origin authentication
 - Protection against reply attacks
- Both protocols may be used alone or applied in combination with each other.

IPSec

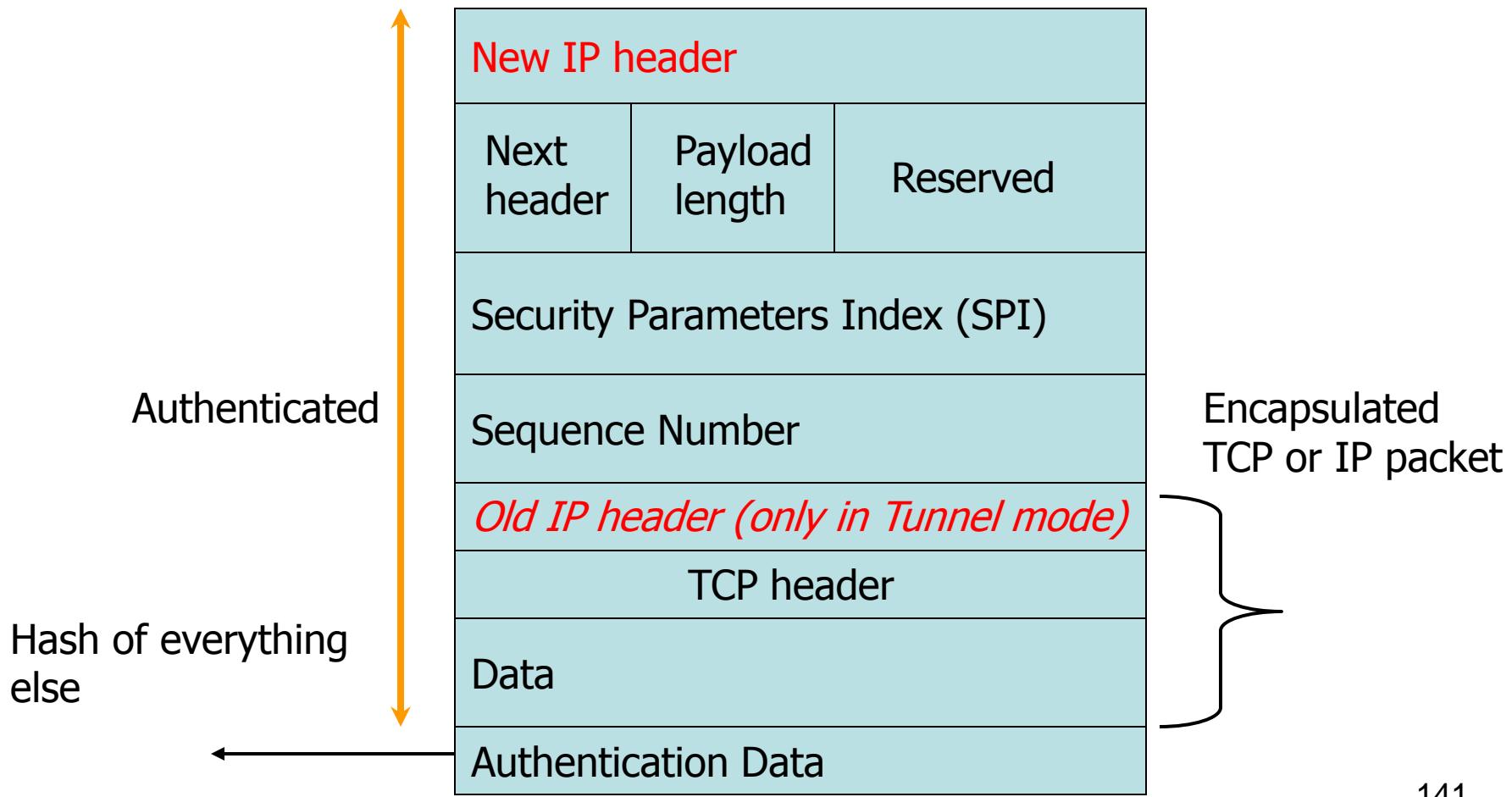
- IPSec architecture requires the host to provide confidentiality using ESP, and data integrity using either AH or ESP

Authentication Header (AH)

- Provides source authentication
 - Protects against source spoofing
- Provides data integrity
 - Use cryptographically strong hash algorithms to protect data integrity (96-bit)
 - Use symmetric key cryptography
- Protects against replay attacks
 - Use 32-bit monotonically increasing sequence numbers (stamp)
 - Protects against denial of service attacks
- **NO support for confidentiality!**

A monotonically increasing sequence is one for which each term is greater than or equal to the term before it

AH Packet Details



Next Header: an 8-bit field that specifies the type of data contained in the payload such as TCP or UDP (protocol ID)

Payload Length: specifies header size

Reserved: Not used; set to zeroes.

Security Parameter Index (SPI): The SPI is a 32-bit random number generated by the sender to identify the SA to the recipient.

Sequence Number: This is a counter field that is initialized to zero when a security association (SA) is formed between two devices, and then incremented for each datagram sent using that SA.

This uniquely identifies each datagram on an SA and is used to provide protection against replay attacks by preventing the retransmission of captured datagrams.

Authentication Data: This field contains the result of the hashing algorithm performed by the AH protocol.

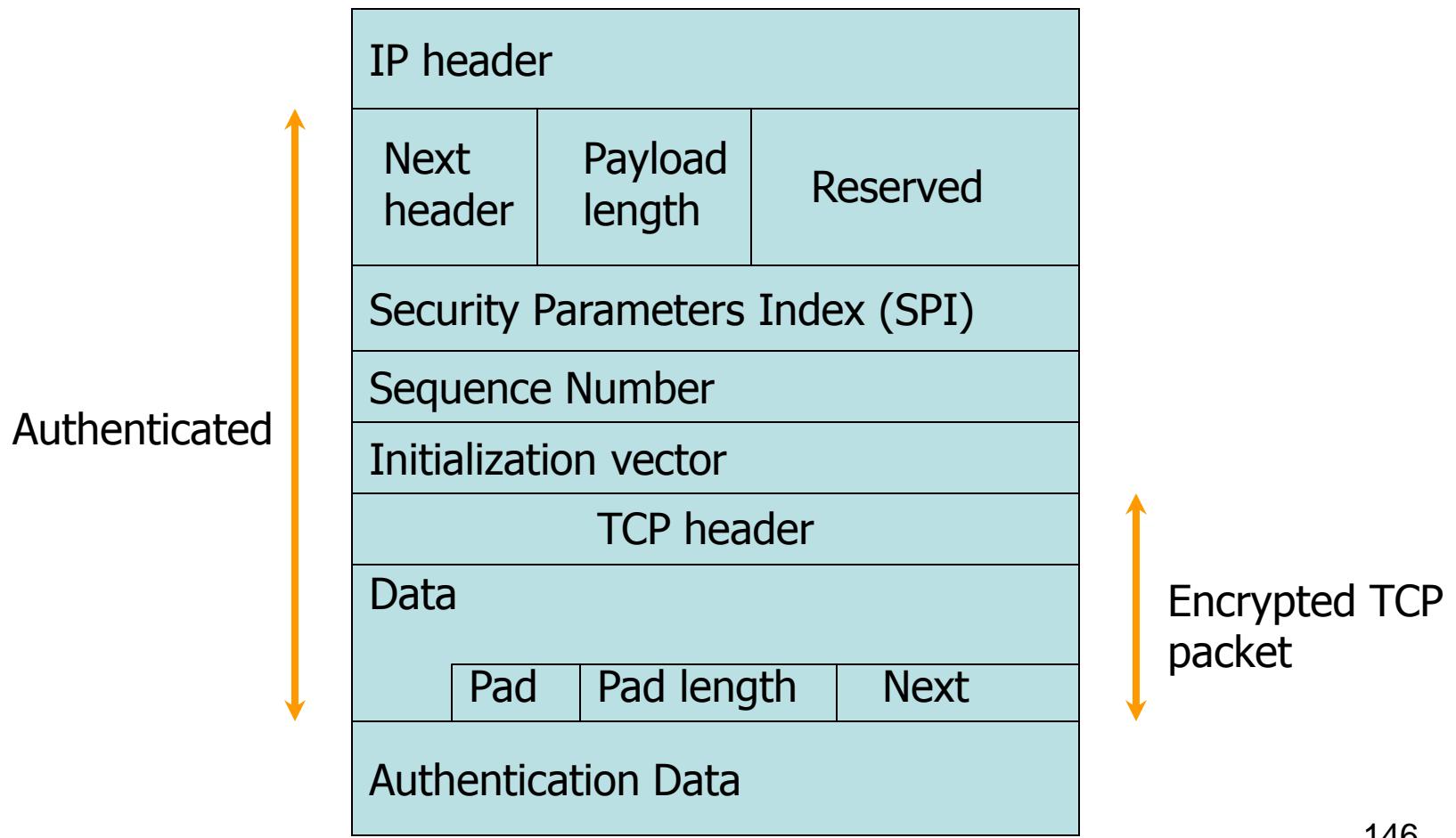
Security Parameter Index (SPI)

- A Security Association (SA) is an agreement between two devices about how to protect information during communication.
 - It also indicates the parameters, such as keys and algorithms.
- SPI provides a mechanism for the destination to identify which SA to use to check the security of the received packet.
- The SPI is a 32-bit random number generated by the sender to identify the SA to the recipient.

Encapsulating Security Payload (ESP)

- Provides all that AH offers, and
- in addition provides **data confidentiality**
 - Uses symmetric key encryption

ESP Packet Details

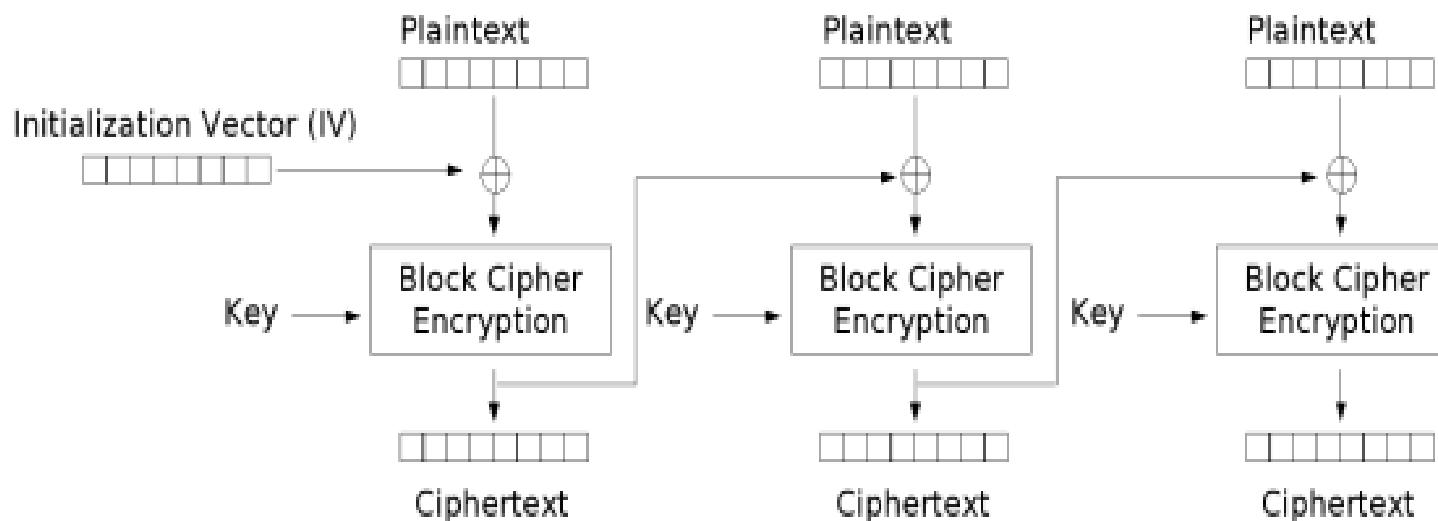


ESP provides confidentiality for IP datagrams by encrypting the payload data to be protected.

ESP uses the Cipher Block Chaining (CBC) mode of the Data Encryption Standard (DES) algorithm

This mode of DES requires an Initialization Vector (IV)

Each datagram contains its own IV. Including the IV in each datagram ensures that decryption of each received datagram can be performed



Cipher Block Chaining (CBC) mode encryption

Benefits of IPsec

- If implemented in a firewall or router, provides strong security to all traffic crossing the perimeter
- Resides below the transport layer, hence transparent to application layer
- Can be transparent to end users

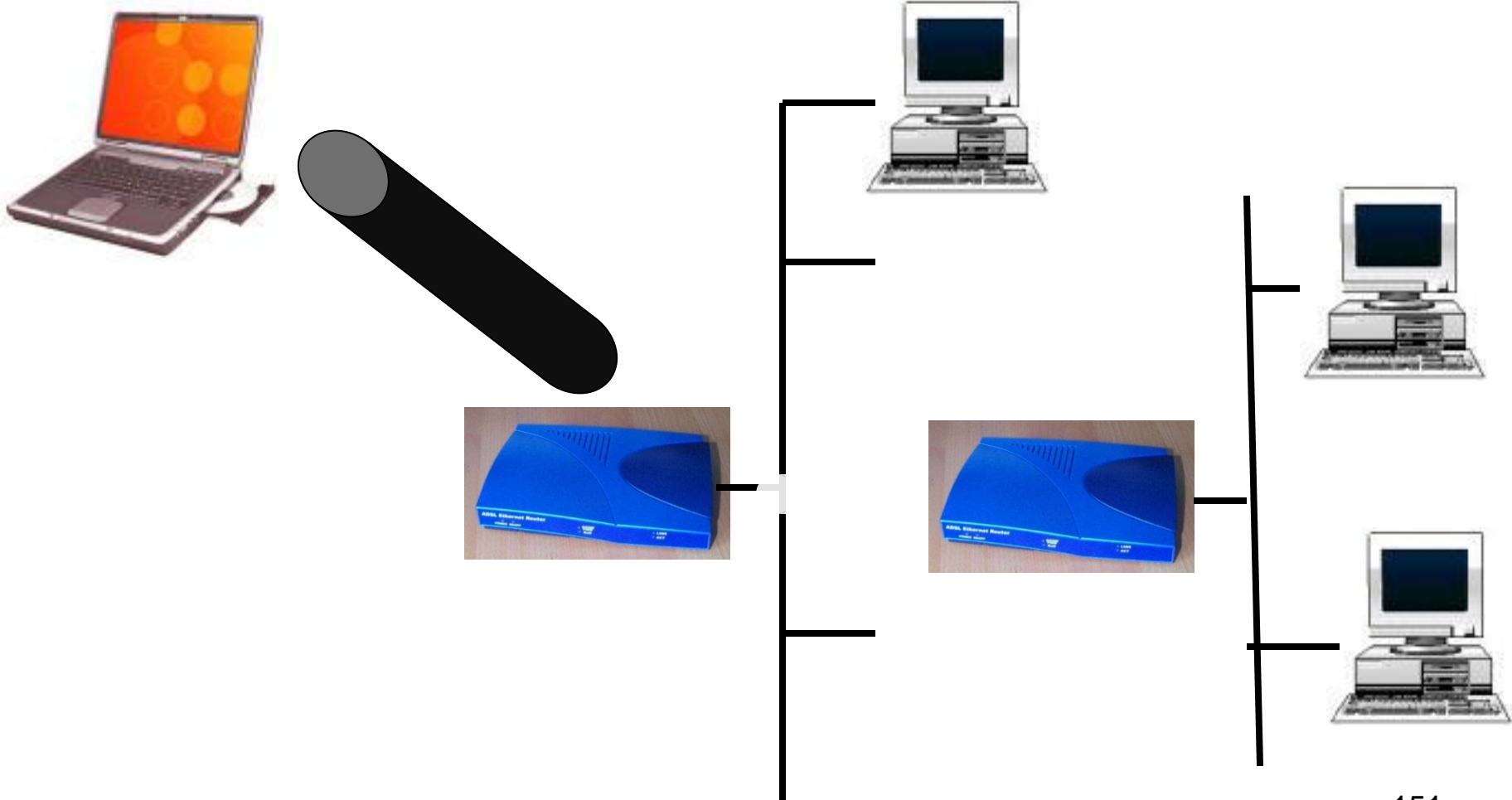
Internet Security Association and Key Management Protocol (ISAKMP)

- It defines packet formats for exchanging key (IKE) generation and authentication data.

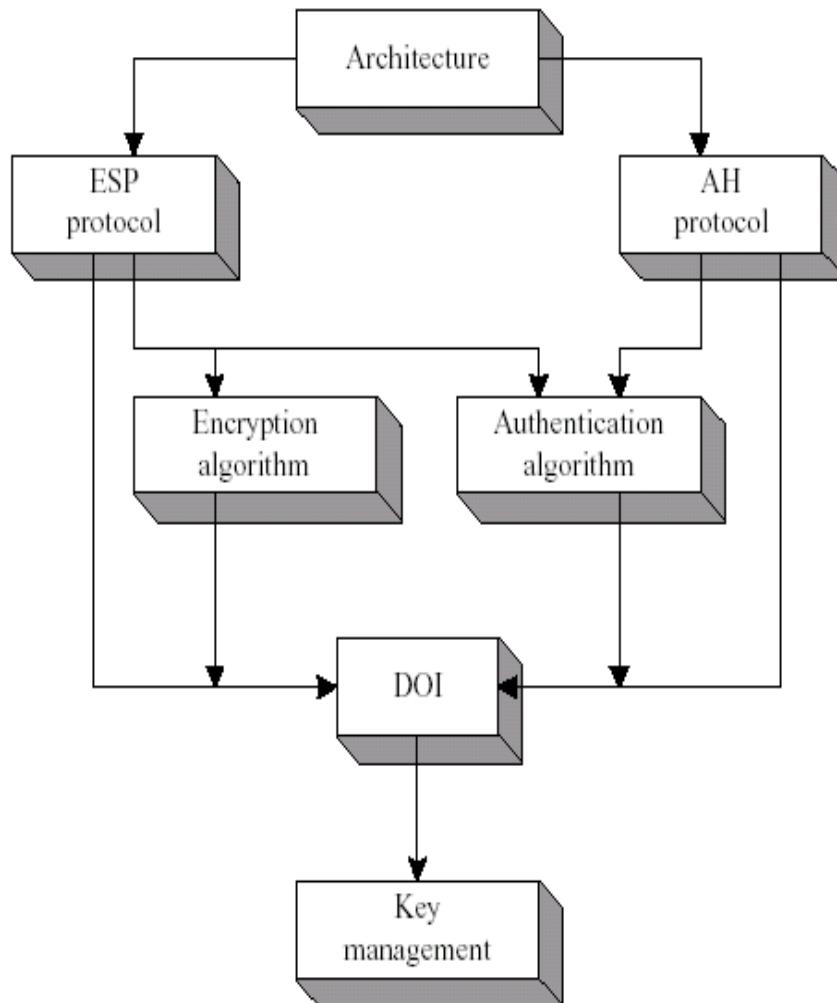
Alice is Traveling

- Alice works for the mergers and acquisitions (M&A) department of takeover.com
- She is at Hicktown taking over a meat-packing plant
- She wants to access the M&A server at her company (confidentially of course)
- VPN

Alice is Traveling



IPSec Overview



- DOI:- Domain of Interpretation: A Domain of Interpretation (DOI) defines payload formats, exchange types, and conventions for naming security-relevant information such as security policies or cryptographic algorithms and modes.

Secure Socket Layer

Why?

- Virtually all businesses, most government agencies, and many individuals now have Web sites.
- The number of individuals and companies with Internet access is expanding rapidly, and all of them have graphical Web browsers.
- As a result, businesses are enthusiastic about setting up facilities on the Web for electronic commerce.
- But the reality is that the Internet and the Web are extremely vulnerable to compromises of various sorts.
- Businesses need to use trusted security mechanisms.

- An increasingly popular general-purpose solution is to implement security as a protocol that sits between the underlying transport protocol (TCP) and the application.
- The foremost example of this approach is the **Secure Socket Layer (SSL)**

Unidirectional versus bi-directional cryptography

- Sometimes, both parties to a conversation need to send secret information
 - bi-directional cryptography is needed
 - each party must send his public key to the other
- Sometimes, only one party to a conversation needs to send secret information
 - only unidirectional cryptography is needed
 - only the recipient of the secret information needs to own a public key
 - which he must send to sender of the secret information

Much e-Commerce involves only unidirectional cryptography

- A customer making an online purchase needs to send secret information, such as a credit card number, to the company
- The company does not need to send any secret information to the customer
- In this case, only unidirectional cryptography is needed
- Thus, the customer need not own any cryptographic key, but the company must
 - the server will have to send its public key to the customer's browser

An example

- Consider buying a ticket from Aer Lingus
- Initially, no secret information is being exchanged
 - Aer Lingus merely lists flight availability
 - the customer selects flights
- Then, however, the customer must provide credit card details
 - before that can happen, Aer Lingus must send its public key to the customer

Customer specifies Cork-Heathrow itinerary http protocol in use; no lock on status bar

aerlingus.com - for the cheapest Aer Lingus fares to and from Ireland, UK and USA. - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Favorites W Mail Print Go Address http://www.aerlingus.com/cgi-bin/obel01im1/bookonline/index.jsp Links Teaching addArticleAhram addArticleSolo addArticleSoloJour addAuthor AltaVista - Babel Fish Translation Amazon Google "mortice lock" Search 2178 blocked AutoFill Options mortice lock

aerlingus.com

BOOK | CHANGE BOOKING | SERVICES & FREQUENT FLYER | NEED HELP? | ABOUT US | LOGIN/JOIN

Search for flights.

return one-way multicity

From Dublin (DUB)
Cork (ORK)
Shannon (SNN)
--- scroll for more ---

To Lanzarote (ACE)
Lisbon (LIS)
Liverpool (LPL)
London (LHR)

Departing 17 December

Returning 18 December

My travel dates are flexible Yes No

Fare Type Lowest

Adults 1 Children 0 Infants 0
2-11 yrs < 2 yrs

Search >>

Sign-up for email offers.
Our best deals sent directly to you.

Quick links

- Extra seats to Malaga this Christmas
- Sky Shopping New Range

New direct route from **Dublin to Dubai** from **€179** click here

Latest low fares:

From Dublin

Bristol	€1
Glasgow	€1
Manchester	€1
Liverpool	€1
London	€1
Amsterdam	€1
Brussels	€1
Paris	€1
Frankfurt	€1
Dusseldorf	€1
Milan/Linate	€24
Venice	€24
Lisbon	€24
Geneva	€24

CarHire **Hertz**
Rentals from €15 a day

Hotels Hotels from €11 pps

Sky Shopping Get your Xmas gifts

Travel Insurance Buy online from €5

Ski Deals

Done Internet

160

Customer selects flights still http protocol; no lock on status bar

The screenshot shows a Microsoft Internet Explorer window with the title "Aer Lingus - Price - Microsoft Internet Explorer". The address bar contains the URL http://www.flyaerlingus.com/cgi-bin/obel01im1/bookonline/flightScheduleLookupDispatchAction.do?BV_SessionID=@@@@0163039272.1134718971@0. The page content is for flight booking, specifically the "Price" step. It features a yellow header with the Aer Lingus logo and navigation links like HOME, BOOKING HELP, UNACCOMPANIED CHILD & INFANT INFORMATION, and START OVER. Below the header, a menu bar lists PLAN --- SELECT --- PRICE --- BOOK --- PURCHASE --- CONFIRMATION. The main content area is titled "Price" and includes sections for "Your Journey" (describing a non-stop flight from Dublin to London on Dec 17, 2005), "Flight Info." (table showing flight details), "Price for this Journey" (table showing costs for 1 adult passenger), and "Restrictions/Endorsements" (checkbox for accepting fare rules). The status bar at the bottom indicates "Done" and "Internet".

Flight Info.

Flight Info.	Departing	Arriving	Fare
AER LINGUS EI152 non-stop	Dublin 06:40 Sat 17 Dec 2005	London 08:05 Sat 17 Dec 2005	Economy
AER LINGUS EI151 non-stop	London 07:50 Sun 18 Dec 2005	Dublin 09:00 Sun 18 Dec 2005	Economy

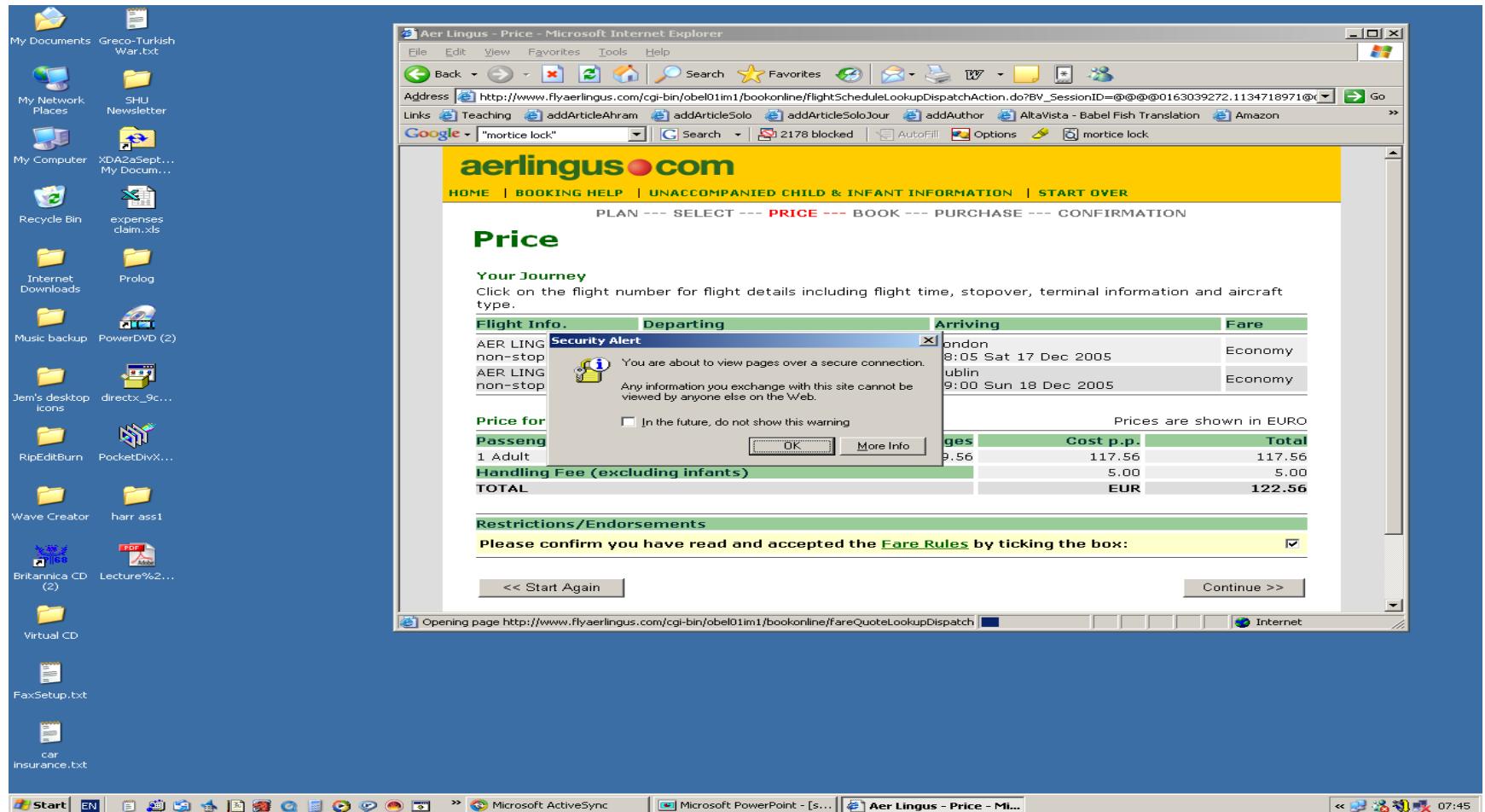
Price for this Journey

Passengers	Fare p.p.	Taxes & Charges	Cost p.p.	Total
1 Adult	68.00	49.56	117.56	117.56
Handling Fee (excluding infants)			5.00	5.00
TOTAL			EUR	122.56

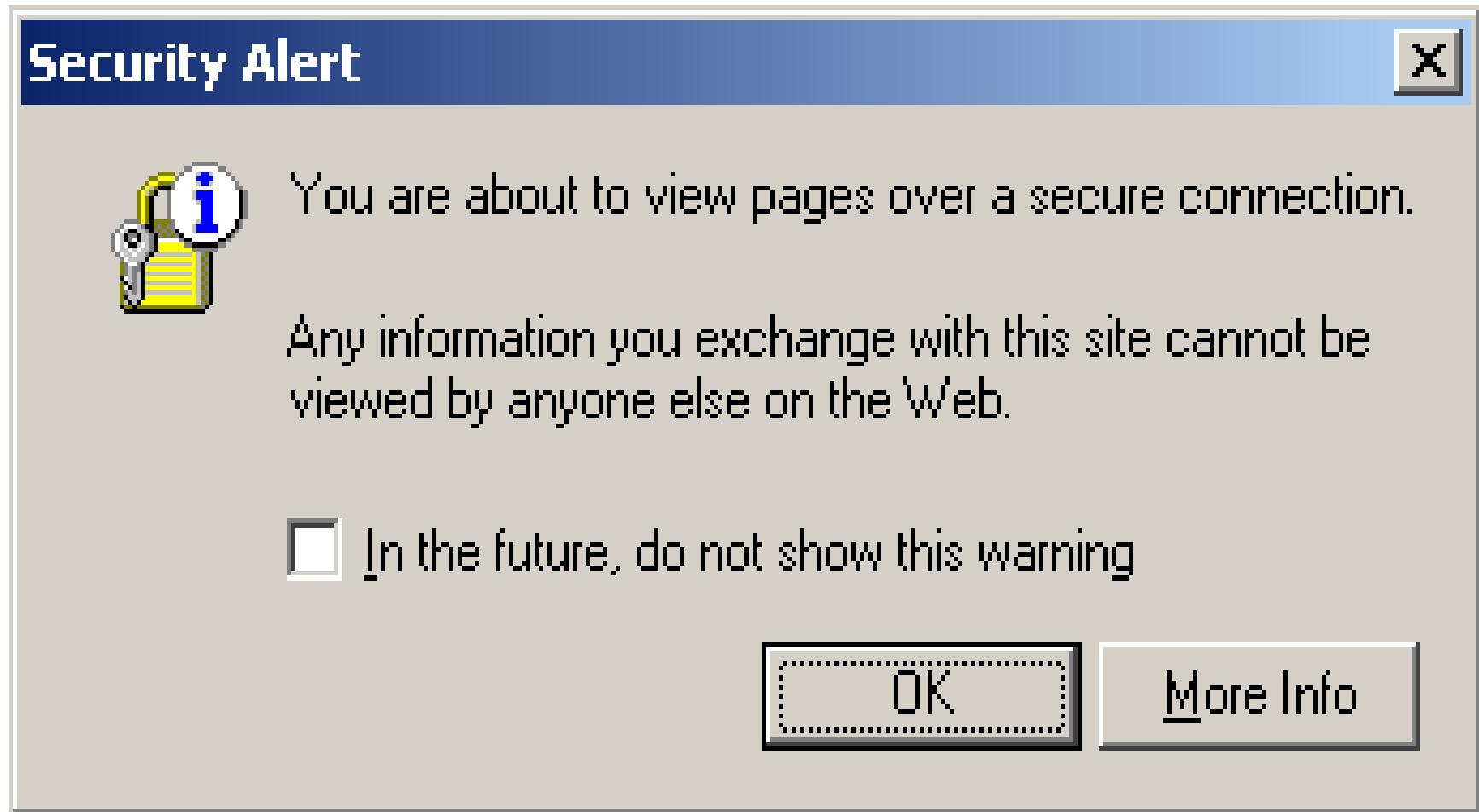
Restrictions/Endorsements

Please confirm you have read and accepted the [Fare Rules](#) by ticking the box:

Customer is warned that SSL communication is about to start



Customer is warned that SSL communication is about to start



Customer is being asked for credit card details protocol is now https; there is a lock on the status bar

Aer Lingus - Booking Information - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address https://www.fly aer lingus.com/cgi-bin/obel01im1/bookonline/loginLookupDispatchAction.do?BW_SessionID=@@@0163039272.1134718971@@@88 Go

Links Teaching addArticleAhram addArticleSolo addArticleSoloJour addAuthor AltaVista - Babel Fish Translation Amazon

Google "mortice lock" Search 2178 blocked AutoFill Options mortice lock

Card Information

Cardholder First Name *	<input type="text"/>
Cardholder Family Name *	<input type="text"/>
Card Type *	Select Card
Card Number (no spaces) *	<input type="text"/>
Switch Issue Number (where applicable)	<input type="text"/>
Card Expiry Month/Year *	<input type="button"/> / <input type="button"/>
CVV Number (Visa, Mastercard & American Express only)	<input type="text"/> What is this ?
Cardholder Address *	<input type="text"/>
City/Town (no punctuation) *	<input type="text"/>
State/County (no punctuation)	<input type="text"/>
Postal/Zip Code	<input type="text"/>
Country *	Select Country

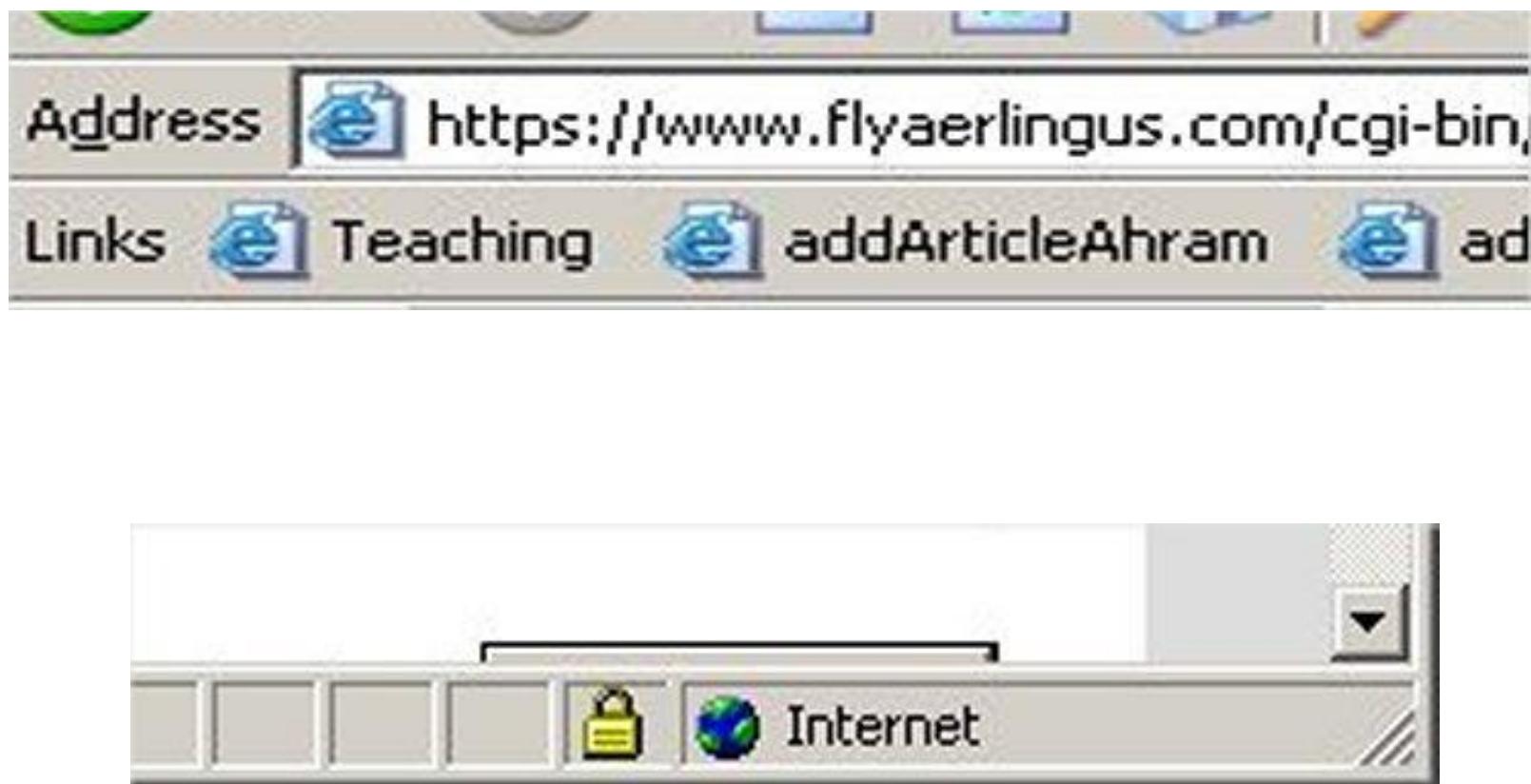
Please refer to our [Privacy Policy](#) for details on how we use your personal information.

Please ensure you read our [Passport/Visa Information](#) that can also be found in our Services and Frequent Flyer section.

We would like to send you occasional emails about our services, including our best online deals. If you would rather not receive this information then please tick this box.

Done  Internet

HTTPS and lock



What happened when user agreed to secure connection



- The Aer Lingus server sent its certificate to the user's browser
- The browser decided that the CA on the certificate was trustworthy and that the public key really was for Aer Lingus
- Henceforth, all information sent by the user would be encoded using the public key

Sometimes, a browser needs to receive secrets from a server

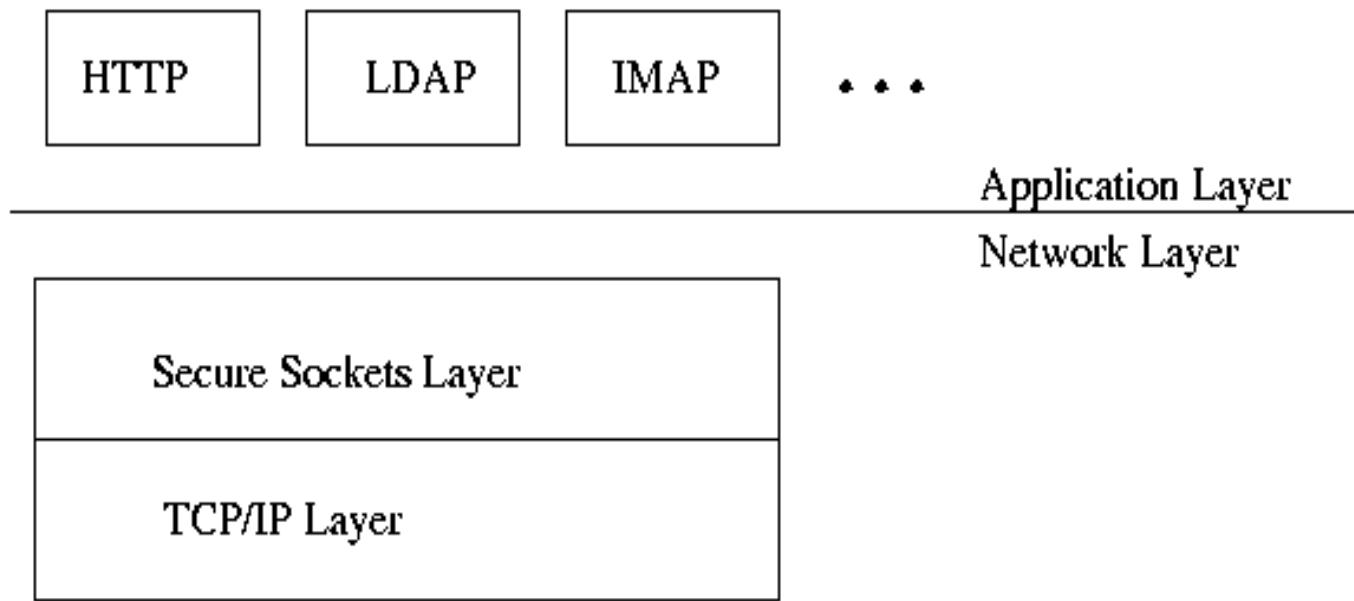
- If a browser needs to receive secrets from a server, the browser must be able to provide a public-key certificate to the server
- There are freely-available utilities, such as OpenSSL, which enable you to
 - select a public+private key combination
 - get a signed certificate for your public key
 - import the certificate into your browser, so that it can send this to servers who request it

SSL

- A protocol developed by Netscape.
- It is a whole new layer of protocol which operates above the Internet TCP protocol and below high-level application protocols.
- It intercepts web traffic and provides security between browser and server
- Encryption is used to guarantee secure communication in an insecure environment
- All security operations are transparent at both ends of the communication
- SSL uses public-key cryptography

What is SSL?

Figure 1 SSL runs above TCP/IP and below high-level application protocols



Internet message access protocol (IMAP) is one of the two most prevalent Internet standard protocols for e-mail retrieval, the other being the Post Office Protocol (POP).

Lightweight Directory Access Protocol, a set of protocols for accessing information directories - X.500.

SSL runs above TCP/IP and below higher-level protocols

- The Transmission Control Protocol/Internet Protocol (TCP/IP) governs the transport and routing of data over the Internet.
- The HTTP, LDAP, and IMAP run "on top of" TCP/IP in the sense that they all use TCP/IP to support typical application tasks such as displaying web pages or running email servers.
- The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP or IMAP.
 - It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

HTTPS

- Hypertext Transfer Protocol Secure (HTTPS) is a combination of the Hypertext Transfer Protocol with the SSL protocol to provide encrypted communication and secure identification of a network web server.
- HTTPS connections are often used for payment transactions on the World Wide Web and for sensitive transactions in corporate information systems.

What is SSL?

- Secure Socket Layer (SSL) transmits private documents via the Internet.
- The SSL Security protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection.
- SSL is built into all major browsers and web servers.
- Transport layer security service

SSL

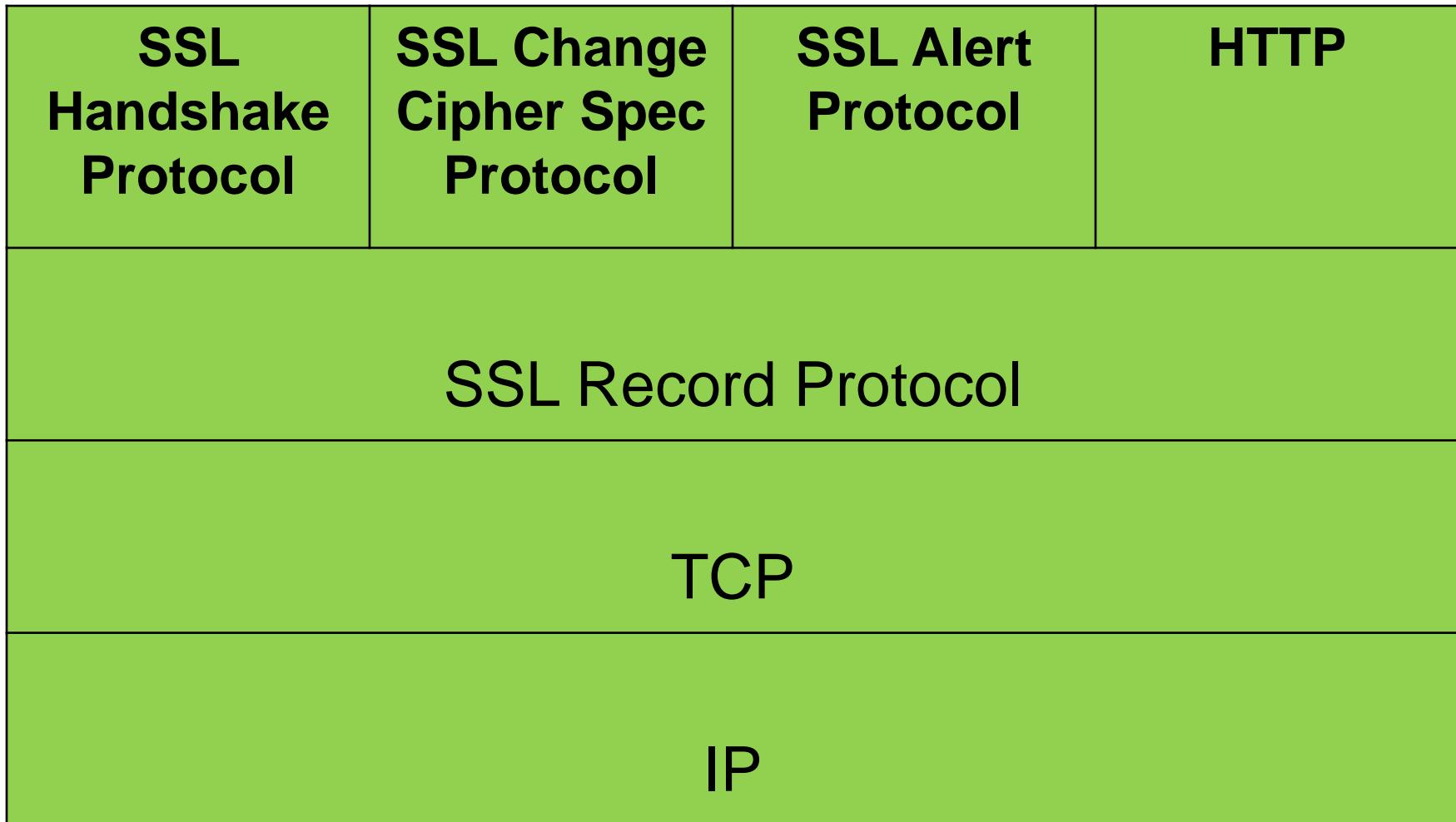
- The protocol uses a third party, a Certificate Authority (CA), to identify one end or both end of the transactions.
- The primary goal of SSL is to provide privacy and reliability between two communicating applications.

- **SSL server authentication** allows a user to confirm a server's identity.
- **SSL client authentication** allows a server to confirm a user's identity.
- **An encrypted SSL connection** requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality.

How it works

- A browser requests a secure page (usually https://).
- The web server sends its public key with its certificate.
- The browser checks that the certificate was issued by a trusted party, that the certificate is still valid and that the certificate is related to the site contacted.
- The browser then uses the public key, to encrypt a random symmetric encryption key and sends it to the server with the encrypted URL required as well as other encrypted http data.
- The web server decrypts the symmetric encryption key using its private key and uses the symmetric key to decrypt the URL and http data.
- The web server sends back the requested html document and http data encrypted with the symmetric key.
- The browser decrypts the http data and html document using the symmetric key and displays the information.

SSL Protocol Stack Table



SSL Protocol Stack

The SSL Protocol Stack is composed of two layers.

1. The first layer is the higher layer which is composed of SSL Handshake Protocol, SSL Change Cipher Spec Protocol, SSL Alert Protocol, and HTTP, which are used in the management of SSL exchanges.
2. The second layer is the lower layer composed of the SSL Record Protocol, TCP, and IP.

SSL components

SSL Handshake Protocol

- negotiation of security algorithms and parameters
- key exchange
- server authentication and optionally client authentication

SSL Record Protocol

- fragmentation
- compression
- message authentication and integrity protection
- encryption

SSL Alert Protocol

- error messages (fatal alerts and warnings)

SSL Change Cipher Spec Protocol

- a single message that indicates the end of the SSL handshake

SSL Record Protocol

- The SSL Record Protocol provides two services for SSL connections: *confidentiality*, by encrypting application data; and *message integrity*, by using a message authentication code (MAC)
- The Record Protocol takes an application message to be transmitted, fragments the data into blocks, compresses the data (optionally), applies a MAC, encrypts, adds a header and transmits the resulting unit.
- Received data is decrypted, verified, decompressed, and reassembled and then delivered to the calling application, such as the browser.
- The Record Protocol is a base protocol that can be utilized by some of the upper-layer protocols of SSL.
 - Ex: Handshake protocol which is used to exchange the encryption and authentication keys.

Steps of SSL Record Protocol

Steps:

- **Fragmentation:** The record layer fragments information blocks into SSLPlaintext records of 2^{14} bytes (16384 bytes) or less.
- **Compression:** All records are compressed using the compression algorithm. The compression algorithm translates an SSLPlaintext structure into an SSLCompressed structure.

Steps.....

- **Compute a MAC**
 - Compute a message authentication code over the compressed data
 - a shared secret key is used. In essence, the hash code (for example, MD5) is calculated over a combination of the message, a secret key, and some padding.
 - The receiver performs the same calculation and compares the incoming MAC value with the value it computes.
 - If the two values match, the receiver is assured that the message has not been altered in transit.

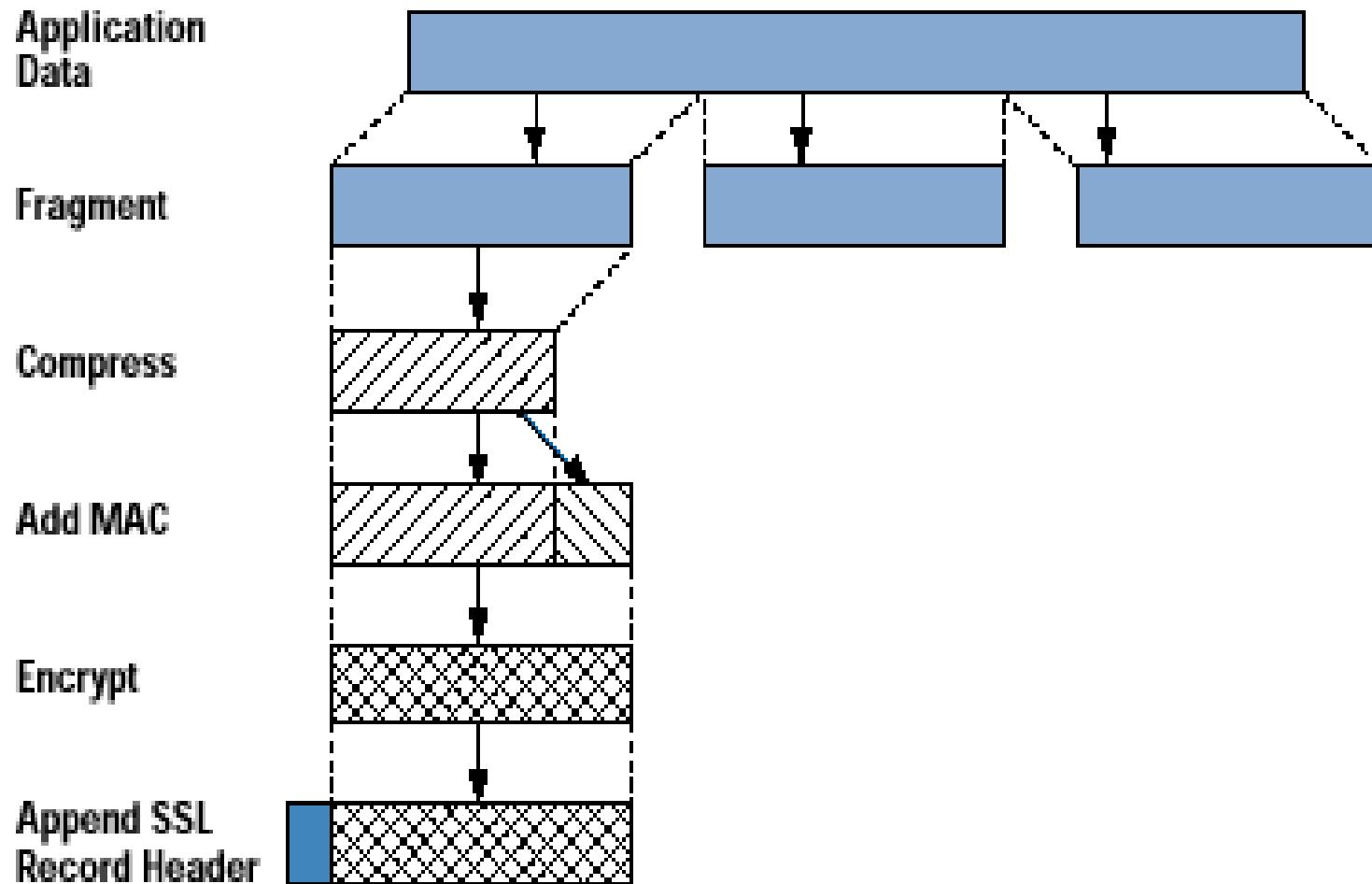
Steps.....

- **Encryption**
 - the compressed message plus the MAC are encrypted using symmetric encryption.
 - A variety of encryption algorithms may be used, including the Data Encryption Standard (DES) and triple DES.

Steps

- **Record header**
 - Append SSL record header
 - SSL record header consists of
 - *Content Type* (8 bits): The higher-layer protocol used to process the enclosed fragment.
 - The content types that have been defined are `change_cipher_spec`, `alert`, `handshake`, and `application_data` (payload from any application that would normally use TCP) .
 - *Major Version* (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.
 - *Minor Version* (8 bits): Indicates minor version in use. For SSLv3, the value is 0.
 - *Compressed Length* (16 bits): The length in bytes of the plain-text fragment (or compressed fragment if compression is used).

SSL Record Protocol Operation



Change Cipher Spec Protocol

- The protocol consists of a single message, which is encrypted and compressed under the current CipherSpec. The message consists of a single byte of value 1.
- The change cipher spec message is sent by both the client and server to notify the receiving party that subsequent records will be protected under the just-negotiated CipherSpec and keys.
- The client sends a change cipher spec message following handshake key exchange and certificate verify messages
 - the server sends one after successfully processing the key exchange message it received from the client.

Alert Protocol

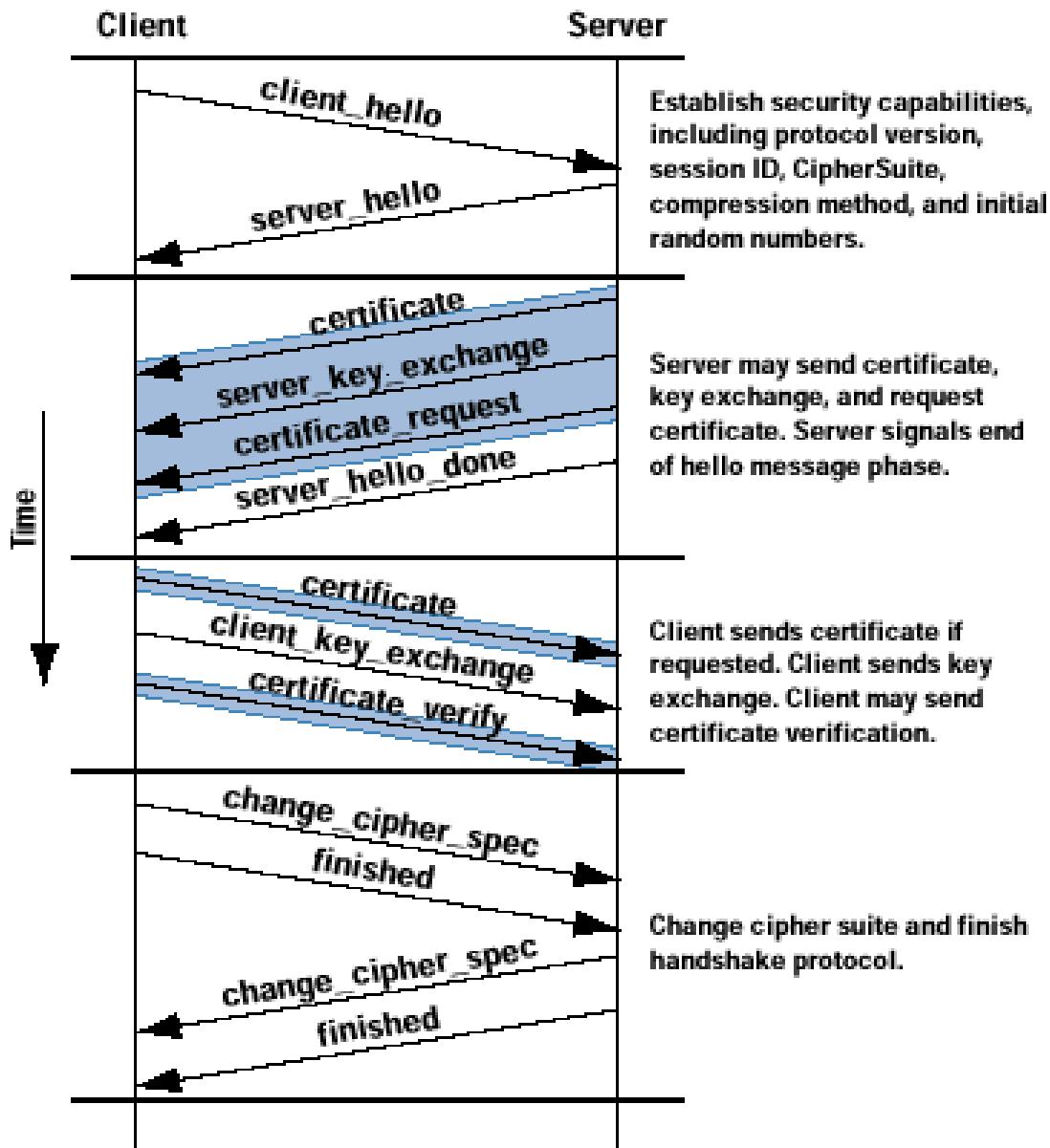
- The Alert Protocol is used to convey SSL-related alerts to the peer entity.
- Alert messages are encrypted and compressed, as specified by the current connection state.
- Each message in this protocol consists of two bytes. The first byte takes the value "warning" (1) or "fatal"(2) to convey the severity of the message.
- 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.

Alert Protocol

- The second byte contains a code that indicates the specific alert.
- An example of a fatal message is **illegal_parameter**.
 - An example of a warning message is **close_notify** (*notifies the recipient that the sender will not send any more messages on this connection*).

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 the client and the server.



How Handshake Protocol works

- Phase 1
- Phase 2
- Phase 3 and
- Phase 4

Handshake Protocol - Phase 1

- Phase 1 is used to initiate a logical connection and to establish the security capabilities that will be associated with it.
- The exchange is initiated by the client, which sends a **client_hello** message with the following parameters:
 - **Version**: The highest SSL version understood by the client.
 - **Random**: A client-generated random structure
 - serve as nonces and are used during key exchange to prevent replay attacks.
 - **Session ID**: A variable-length session identifier.
 - **CipherSuite**: A list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference.
 - **Compression Method**: A list of the compression methods the client supports.

Handshake Protocol - Phase 1

- After sending the client_hello message, the client waits for the server_hello message, which contains the same parameters as the client_hello message.
- The **Version field** contains the lower of the version suggested by the client and the highest version supported by the server.
- The **Random field** is generated by the server and is independent of the client's Random field.
- The **CipherSuite** field contains the single CipherSuite selected by the server from those proposed by the client.
- The **Compression field** contains the compression method selected by the server from those proposed by the client.

Handshake Protocol - Phase 1

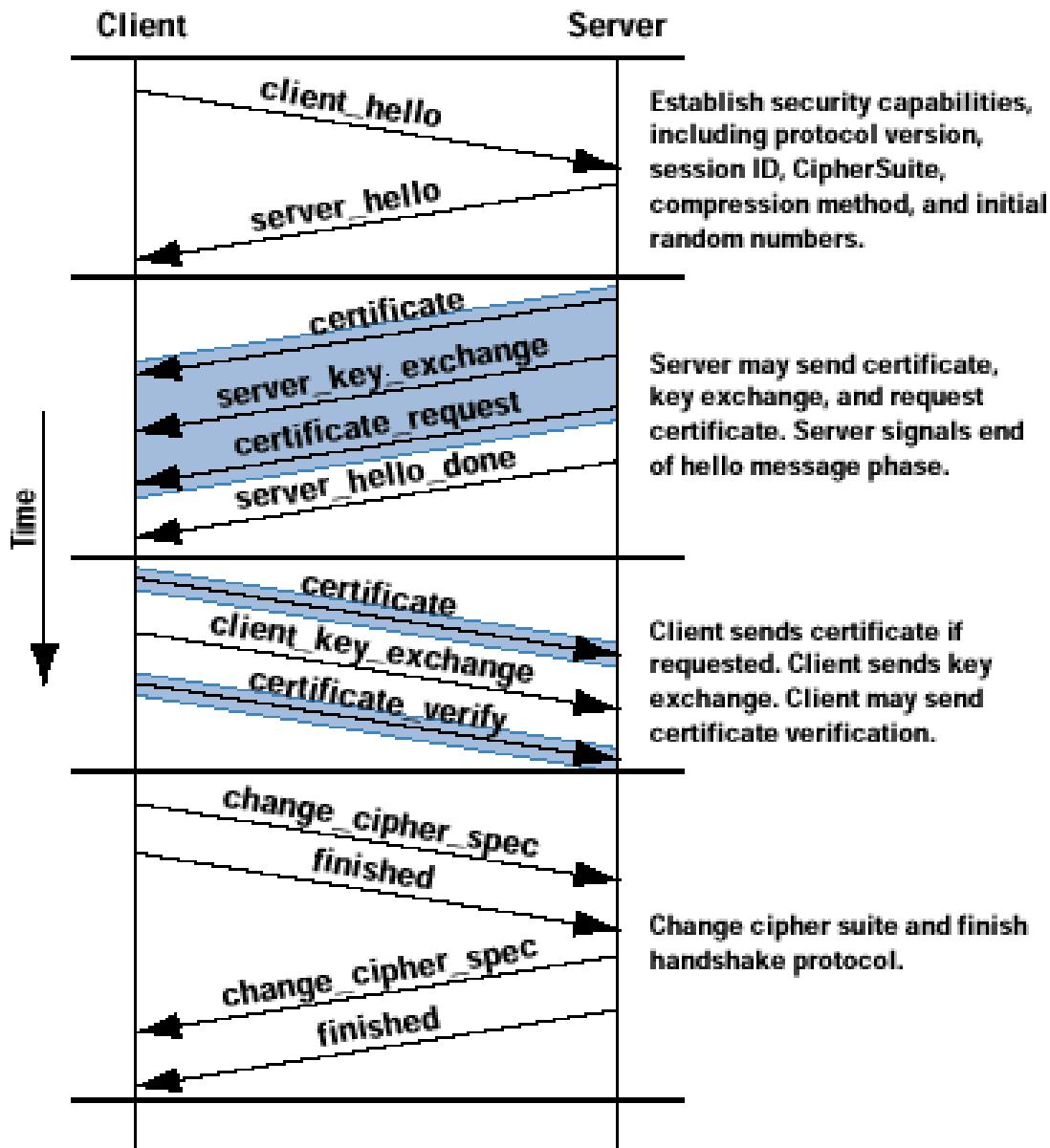
- The first element of the CipherSuite parameter is the key exchange method.
- The following key exchange methods are supported:
 - RSA
 - Fixed Diffie-Hellman
 - Ephemeral Diffie-Hellman
 - Anonymous Diffie-Hellman
- Following the definition of a key exchange method is the CipherSpec, which indicates the encryption and hash algorithms and other related parameters.

Handshake Protocol - Phase 1

- The following key exchange methods are supported:
 - **RSA**: The secret key is encrypted with the receiver's RSA public key. A public-key certificate for the receiver's key must be made available.
 - **Fixed Diffie-Hellman**: This is a Diffie-Hellman key exchange in which the server's certificate contains the Diffie-Hellman public parameters signed by the certificate authority (CA).
 - That is, the public-key certificate contains the Diffie-Hellman public-key parameters.
 - The client provides its Diffie-Hellman public key parameters either in a certificate, if client authentication is required, or in a key exchange message. .

Handshake Protocol - Phase 1

- **Ephemeral Diffie-Hellman:** This technique is used to create ephemeral (temporary, one-time) secret keys.
 - The Diffie-Hellman public keys are exchanged, and signed using the sender's private RSA or DSS key. The receiver can use the corresponding public key to verify the signature. Certificates are used to authenticate the public keys.
- **Anonymous Diffie-Hellman:** The base Diffie-Hellman algorithm is used, with no authentication.
 - Each side sends its public Diffie-Hellman parameters to the other, with no authentication.



Note: Shaded transfers are optional or situation-dependent messages that are not always sent

Handshake Protocol - Phase 2

- The server begins Phase 2 by sending its certificate.
- The certificate message is required for any agreed-on key exchange method except anonymous Diffie-Hellman.
- Next, a server_key_exchange message may be sent, if it is required.
 - It is not required in two instances: (1) The server has sent a certificate with fixed Diffie-Hellman parameters; or (2) RSA key exchange is to be used.

Handshake Protocol - Phase 2

- Next, the server (server not using anonymous Diffie-Hellman) can request a certificate from the client.
 - The certificate_request message includes two parameters:
 - certificate_type and certificate_authorities.
 - The certificate type indicates the type of public-key algorithm.
 - The second parameter in the certificate_request message is a list of the distinguished names of acceptable certificate authorities.

Handshake Protocol - Phase 2

- The final message in Phase 2, and one that is always required, is the server_done message, which is sent by the server to indicate the end of the server hello and associated messages.
- After sending this message, the server waits for a client response.
- This message has no parameters.

Handshake Protocol - Phase 3

- Upon receipt of the server_done message, the client should verify that the server provided a valid certificate, and check that the server hello parameters are acceptable.
- If all is satisfactory, the client sends one or more messages back to the server.
- If the server has requested a certificate, the client begins this phase by sending a certificate message.
- If no suitable certificate is available, the client sends a no_certificate alert instead.

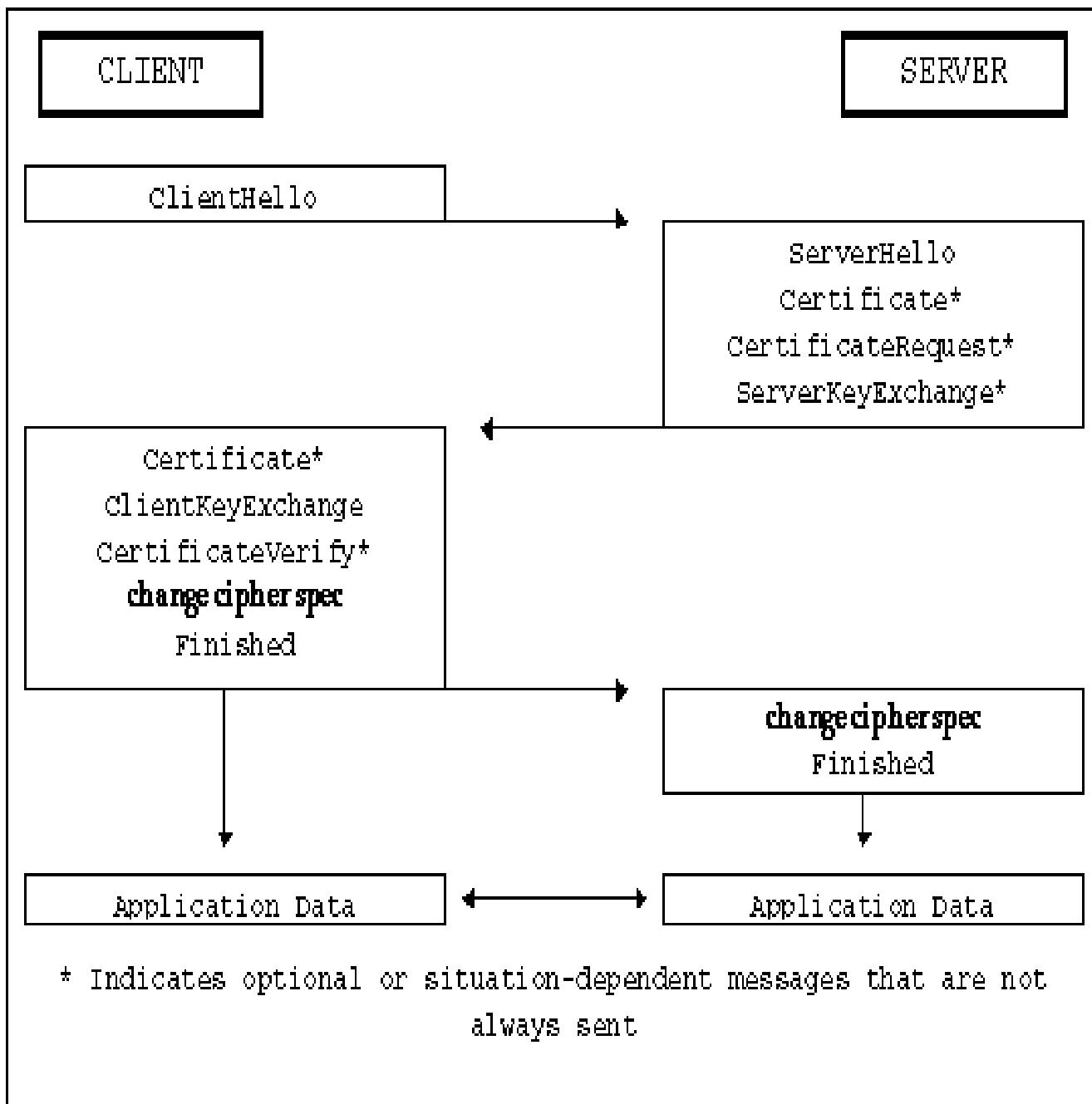
Handshake Protocol - Phase 3

- Next is the client_key_exchange message.
 - The content of the message depends on the type of key exchange.
- Finally, in this phase, the client may send a certificate_verify message to provide explicit verification of a client certificate.

Handshake Protocol - Phase 4

- Phase 4 completes the setting up of a secure connection. The client sends a change_cipher_spec message
- The server sends its own change_cipher_spec message,
- At this point the handshake is complete and the client and server may begin to exchange application layer data.

The change cipher spec message is sent by both the client and server to notify the receiving party that subsequent records will be protected under the just-negotiated CipherSpec and keys.



Secure Electronic Transactions

- An open encryption and security specification.
- Protect credit card transaction on the Internet.
- Companies involved:
 - MasterCard, Visa, IBM, Microsoft, Netscape, RSA, Terisa and Verisign
- Not a payment system.

- SET is a protocol for allowing secure transactions to take place on the Internet.
- It is based on the idea that the merchant and the end-user don't directly transfer funds, but they use a third party (payment gateway).
- It provides a set of protocols and formats that allow users to securely use the existing credit card payment infrastructure on the Internet.

- SET protocol satisfies the following:
 - Confidentiality of payment and ordering information
 - Integrity of transmitted data
 - Authentication that cardholder is a legitimate user of the card account
 - Authentication that a merchant can accept credit card transactions
 - Open, vendor-independent and interoperable to protect all participants

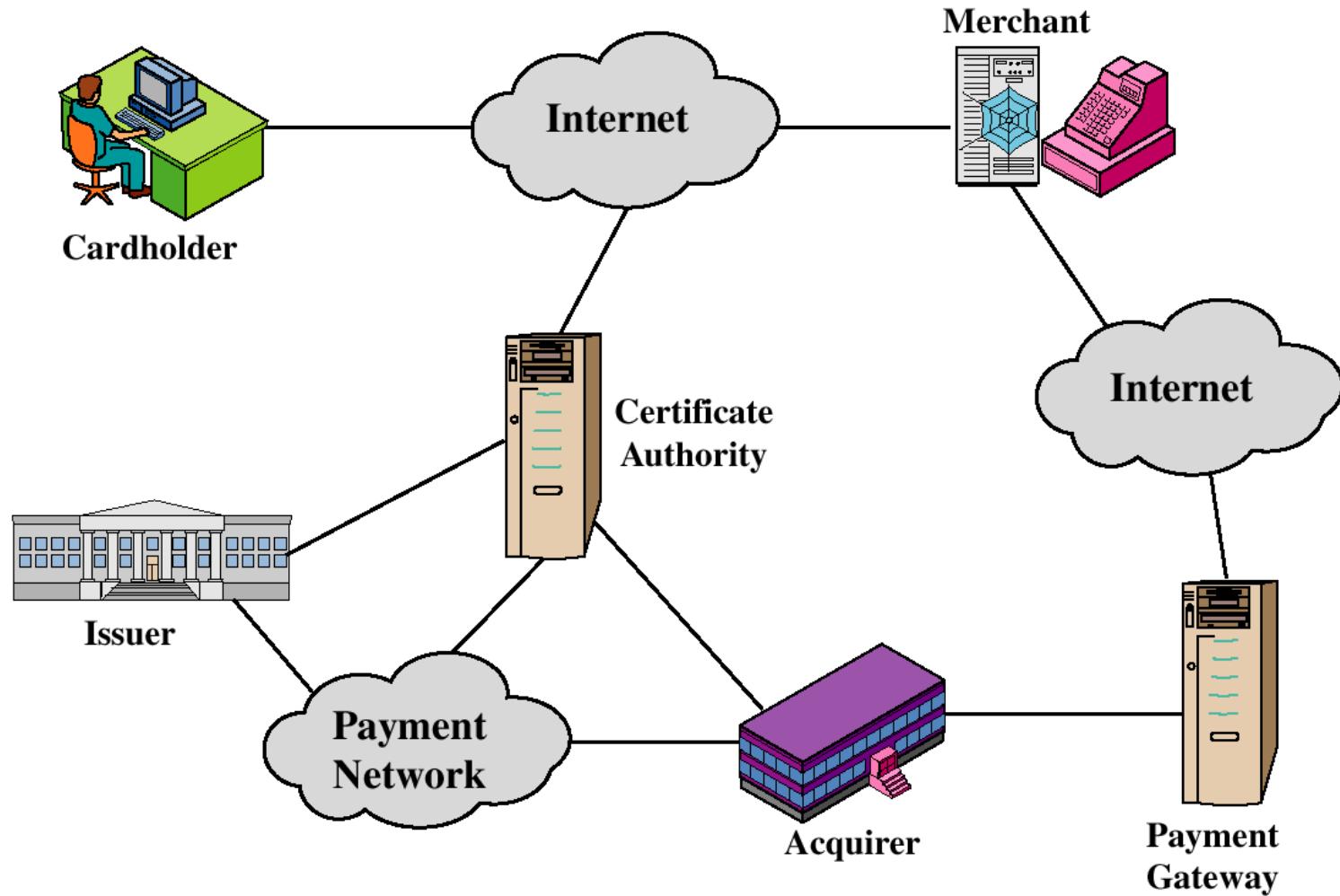
SET Participants

- Card Holder
 - In SET, the card holder is the same as any normal person wishing to use their card to purchase goods in a shop.
- Merchant
 - This is the business that wishes to sell goods online, but unlike a ‘normal’ shop the transaction would be electronic, with no customer physically present.
- Issuer
 - The issuer such as a bank is the organisation that provides the card holder with the payment card.

- Acquirer
 - A financial institution that establishes an account with a merchant and processes payment card authorizations and payments.
 - Merchants will usually accept more than one credit card brand but do not want to deal with multiple bankcard associations or with multiple individual issuers.
 - The acquirer provides authorization to the merchant that a given card account is active and that the proposed purchase does not exceed the credit limit.
 - Acquirer also provides electronic transfer of payments to the merchant's account.

- **Payment Gateway**
 - A function operated by the acquirer that processes merchant payment messages.
 - The payment gateway interface between SET and the existing bankcard payment networks for authorization and payment functions.
 - The merchant exchanges SET messages with the payment gateway over the Internet, while the payment gateway has some direct or network connection to the acquirer's financial processing system.
- **Certificate Authority**
 - The certification authority provides a trust model for all participants, providing a mechanism for identifying who users say they

SET Participants



Sequence of events for transactions

1. The customer opens an account.
2. The customer receives a certificate signed by the bank.
3. Merchants have their own certificates.
4. The customer places an order
browsing the merchant's site to select items and determine the price.

The list of items are sent to merchant and he returns the order form containing list of items, price, total price and an order number.

5. The merchant is verified

Merchant sends a copy of its certificate and customer verifies.

6. Customer sends the order and payment information to the merchant along with customers certificate.

The order confirms the purchase of items and the payment contains credit card details.

The payment information is encrypted so that merchant cannot read it. The customer's certificate enables merchant to verify the customer.

7. The merchant request payment authorization.

Merchant sends the payment information to the payment gateway, requesting authorization that the customer's available credit is sufficient for this purchase

8. The merchant sends confirmation of the order to the customer.

9. The merchant provides the goods or service.

10. The merchant requests payments.

The request is sent to the payment gateway which handles all of the payment processing.

Dual Signature for SET

- Concept: Link Two Messages Intended for Two Different Receivers:
 - Order Information (OI): Customer to Merchant
 - Payment Information (PI): Customer to Bank
- Goal: Limit Information to A “Need-to-Know” Basis:
 - Merchant does not need credit card number.
 - Bank does not need details of customer order.
 - Afford the customer extra protection in terms of privacy by keeping these items separate.
- This link is needed to prove that payment is intended for this order and not some other one.

Why Dual Signature?

- Suppose that customers send the merchant two messages:
 - The signed order information (OI).
 - The signed payment information (PI).
 - In addition, the merchant passes the payment information (PI) to the bank.
- If the merchant can capture another order information (OI) from this customer, the merchant could claim this order goes with the payment information (PI) rather than the original.

Dual Signature

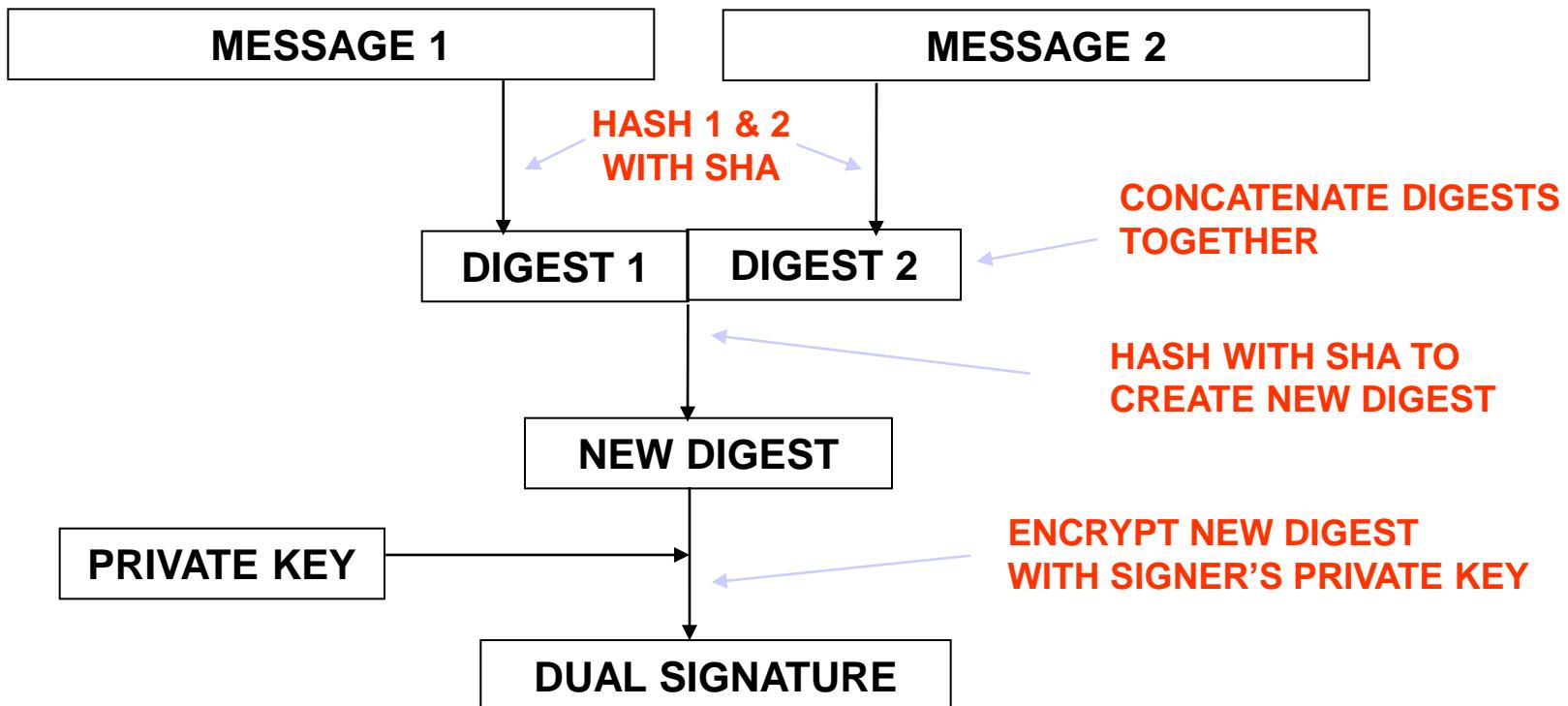
- use a dual signature
 - signed concatenated hashes of OI & PI

$$DS = E(KR_c, [H(H(PI) || H(OI))])$$

KR_c – customer's private key

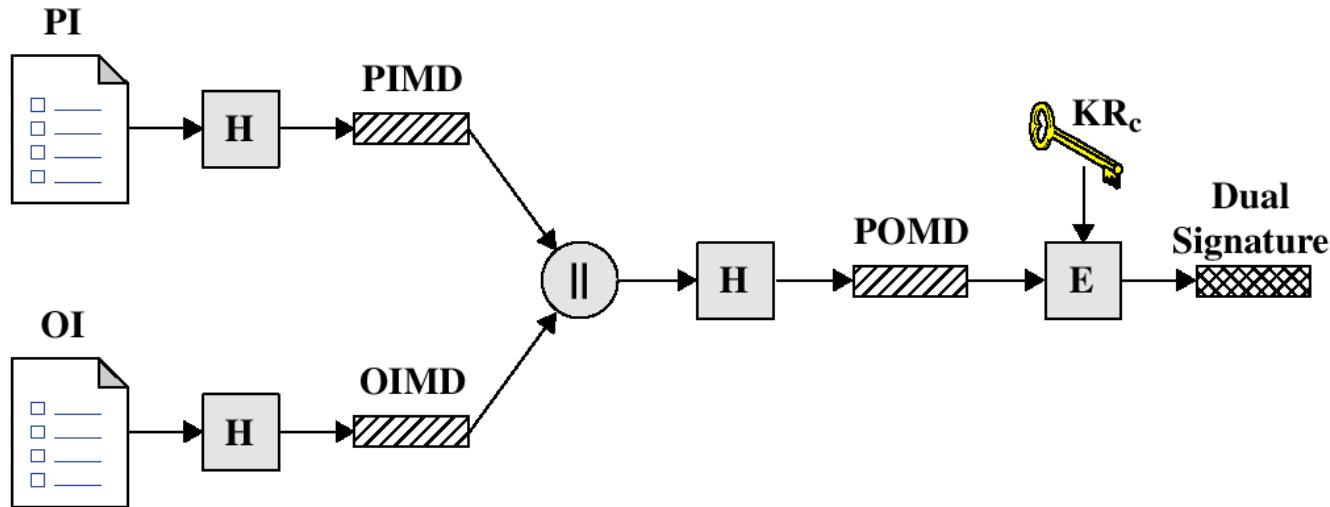
Dual Signatures

- Links two messages securely but allows only one party to read each.



Dual Signature

$$DS = E_{KR_c} [H(H(PI) \parallel H(OI))]$$



PI = Payment Information

OI = Order Information

H = Hash function (SHA-1)

\parallel = Concatenation

PIMD = PI message digest

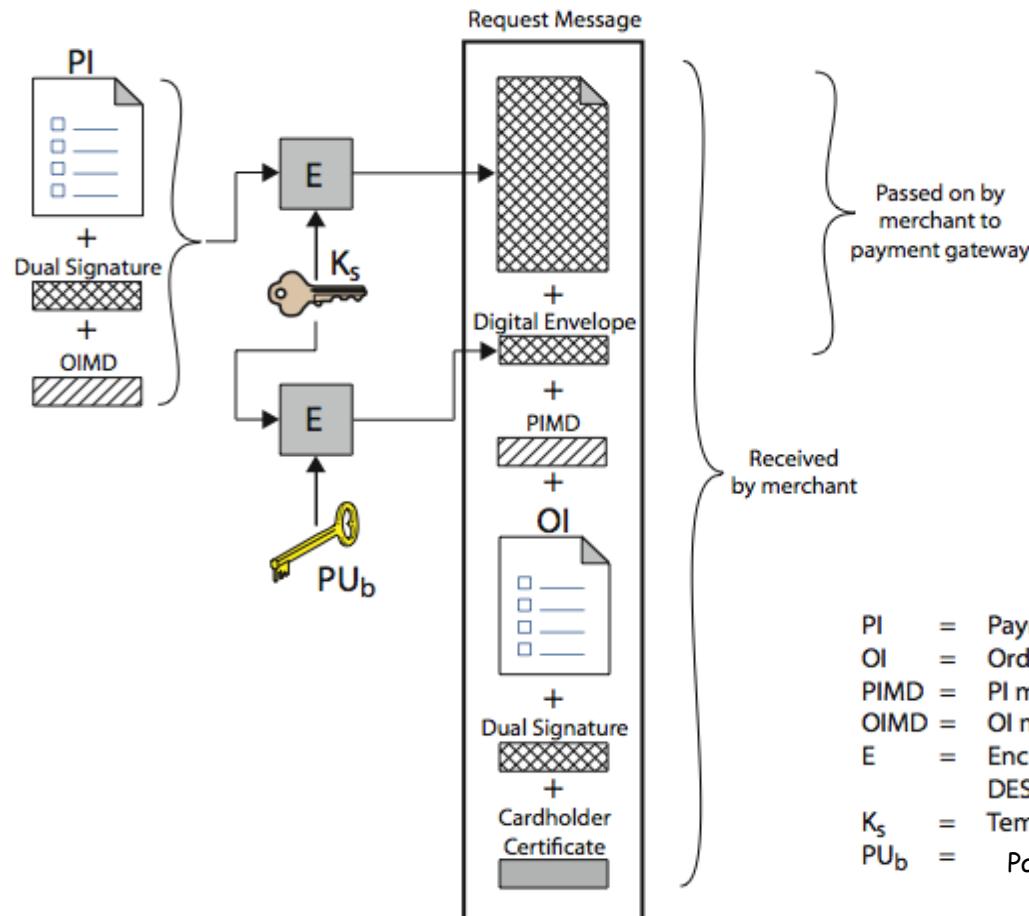
OIMD = OI message digest

POMD = Payment Order message digest

E = Encryption (RSA)

KR_c = Customer's private signature key

Purchase Request – Customer

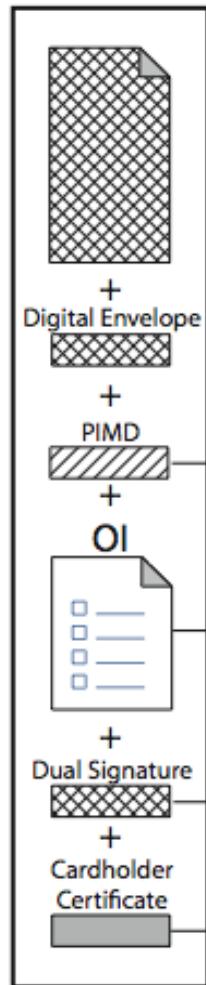


PI	= Payment Information
OI	= Order Information
PIMD	= PI message digest
OIMD	= OI message digest
E	= Encryption (RSA for asymmetric; DES for symmetric)
K_s	= Temporary symmetric key
PU_b	= Payment Gateway's Public Key

Cardholder sends Purchase Request

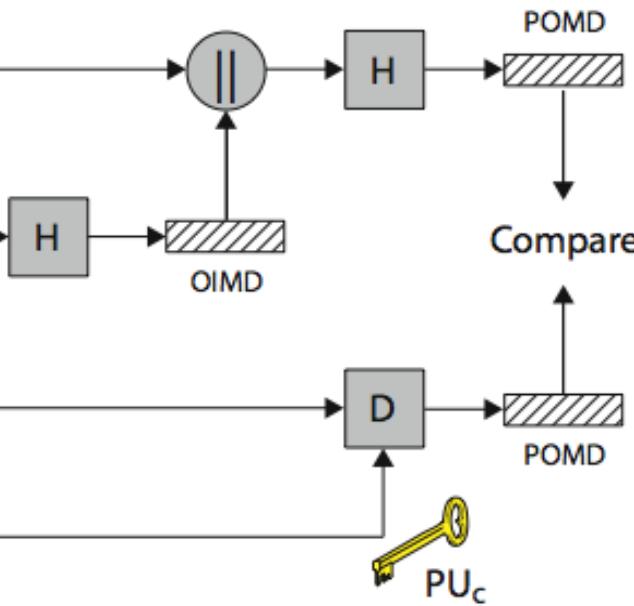
Merchant verifies the Purchase Request

Request Message



Passed on by
merchant to
payment gateway

OI = Order Information
OIMD = OI message digest
POMD = Payment Order message digest
D = Decryption (RSA)
H = Hash function (SHA-1)
PU_c = Customer's public signature key



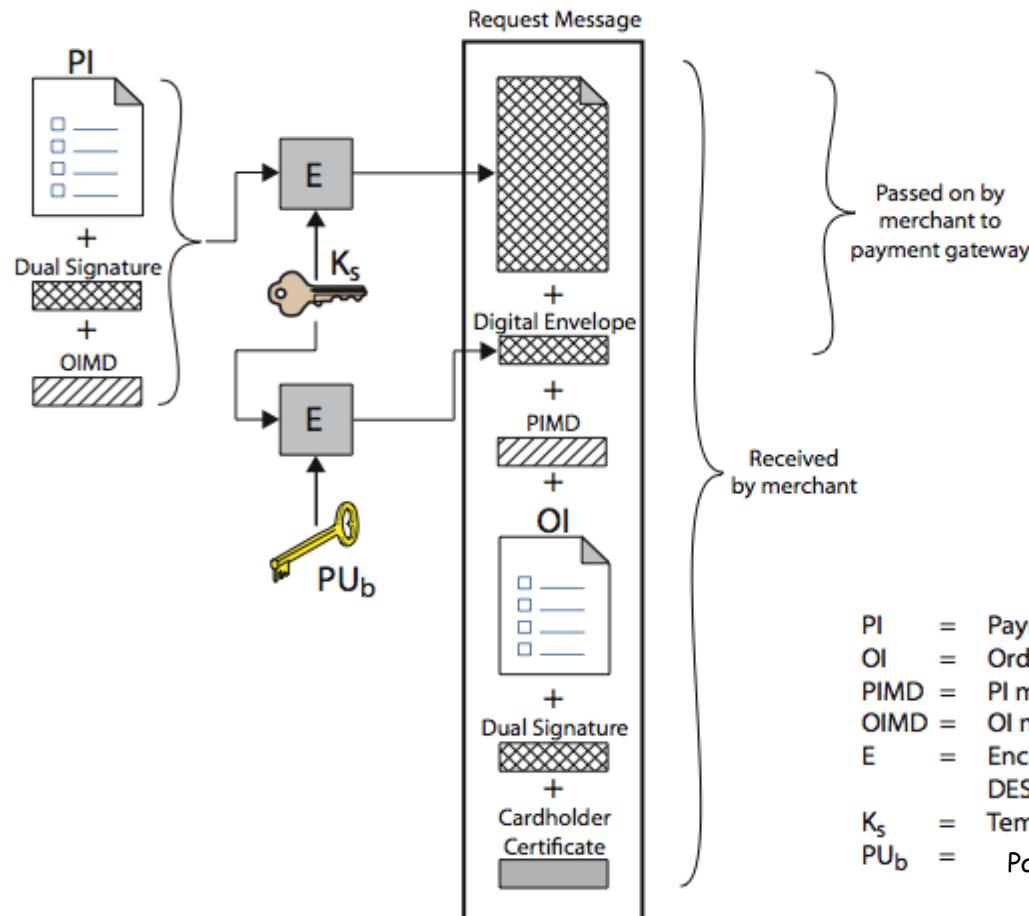
SET Purchase Request

- When the merchant receives the Purchase Request message
 - Verifies the card holder certificates by means of its CA signatures.
 - Verifies the dual signature using the customer's public key
 - Ensures that the order has not been tampered with in transit.
 - Processes the order and forwards the payment information to the payment gateway for authorization.
 - Sends a purchase response to the cardholder.
 - Response includes a response block that acknowledges the order and transaction number. The block is signed by the merchant's private key.

DS Verification by Merchant

- The merchant has the public key of the customer obtained from the customer's certificate.
- Now, the merchant can compute two values:
 $H(PIMD \parallel H(OI))$
 $D_{KUC}[DS]$
- Should be equal!

Purchase Request – Customer



Cardholder sends Purchase Request

Payment Gateway Authorization

1. Verifies all certificates
2. Decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. Verifies merchant's signature on authorization block
4. Decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. Verifies dual signature on payment block
6. Verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
7. Requests & receives an authorization from issuer
8. Sends authorization response back to merchant

Contains a signed **Capture Token** to be used for payment request. Merchant can't decrypt it.

DS Verification by Bank

- The bank is in possession of DS, PI, the message digest for OI (OIMD), and the customer's public key, then the bank can compute the following:
$$H(H(PI) \parallel OIMD)$$
$$D_{KUC} [DS]$$

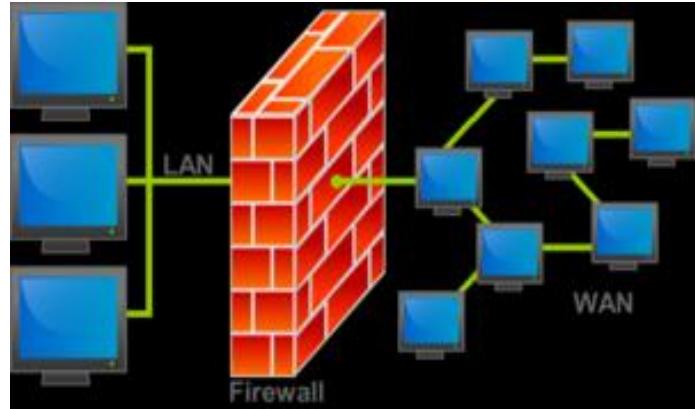
Payment Capture

- Merchant sends payment gateway a payment capture request
 - Includes payment amount , signed Capture Token, transaction ID etc..
- Gateway checks request
- Then causes funds to be transferred to merchants account
- Notifies merchant using capture response

Firewalls

- The Internet has made large amounts of information available to the average computer user at home, in business and in education.
- Yet connecting a private network to the Internet can expose critical or confidential data to malicious attack from anywhere in the world.
- Firewalls can protect both individual computers and corporate networks from hostile intrusion from the Internet.

What is a Firewall?



- A system designed to prevent unauthorized access to or from a private network.
- All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.
- Firewalls can be implemented in both hardware and software, or a combination of both.

Functions of Firewalls

- A firewall is a “choke point/guard box” of controlling and monitoring the network traffic.
- It allows interconnections between different networks with some level of trust.
- It imposes restrictions on network services (only authorized traffic is allowed).
- It enforces auditing and controlling access (alarms of abnormal behaviour can be generated).
- It provides perimeter defence.

- A firewall examines all traffic routed between the two networks to see if it meets certain criteria.
 - If it does, it is routed between the networks, otherwise it is stopped.
 - A firewall filters both inbound and outbound traffic.
 - It can also manage public access to private networked resources such as host applications.
 - It can be used to log all attempts to enter the private network and trigger alarms when hostile or unauthorized entry is attempted.

What Firewalls Can Do

- Service control
 - Determines the types of Internet services that can be accessed, inbound or outbound. Firewall filters traffic on the basis of IP address, TCP, UDP, port numbers, FTP protocols etc..
- Direction control
 - Determines the direction in which particular service requests are allowed to flow through the firewall.
- User control
 - Controls access to a service according to which user is attempting to access it.
- Behavior control
 - Controls how particular services are used (e.g. filter e-mail).

What Firewalls Cannot Do

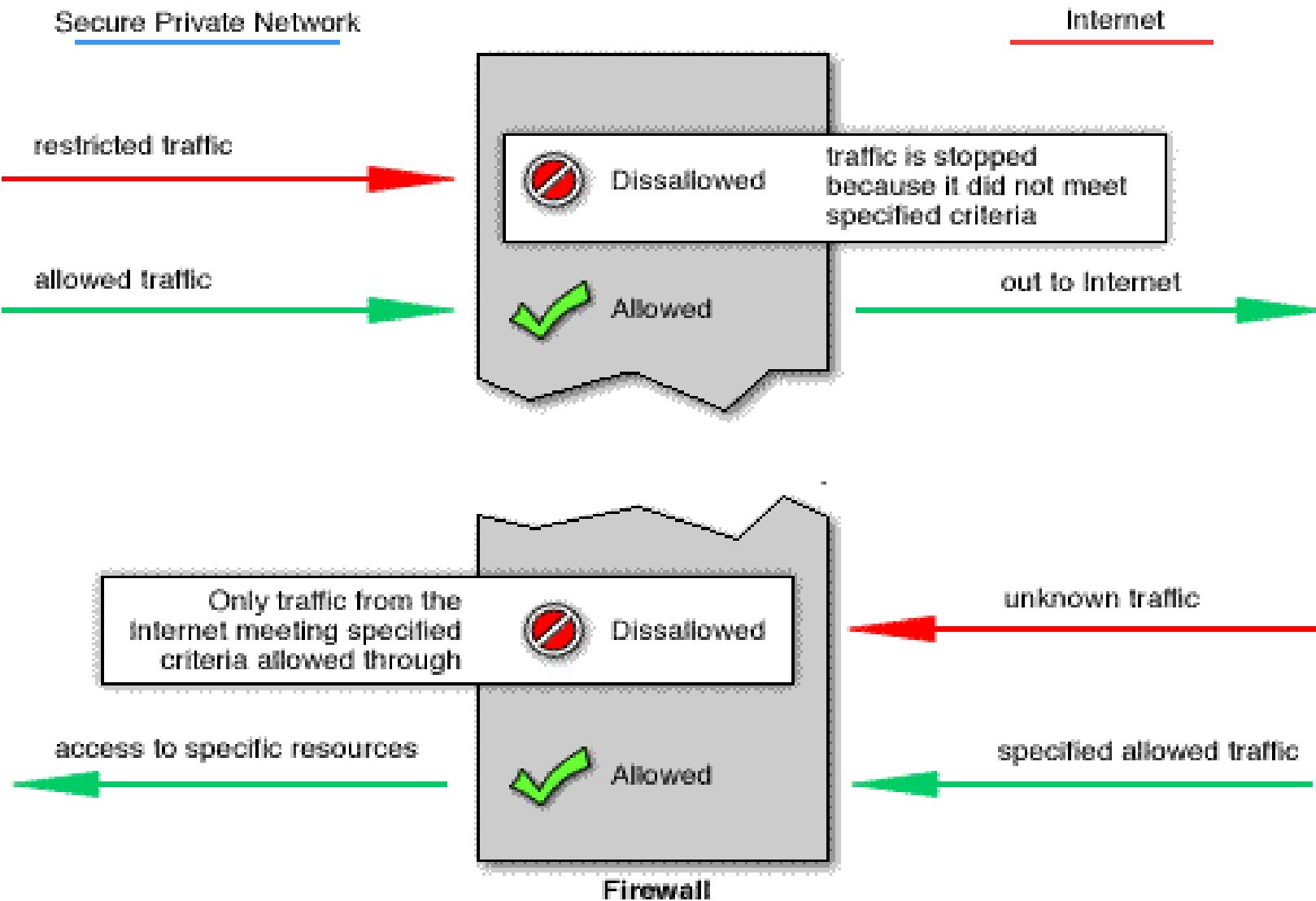
- Cannot protect from attacks bypassing it
 - e.g. trusted organisations, trusted services (e.g. SSL/SSH)
- Cannot protect against internal threats
 - e.g. disgruntled employee
- Cannot protect against transfer of all virus infected programs or files
 - because of huge range of OS and file types

How does a firewall work?

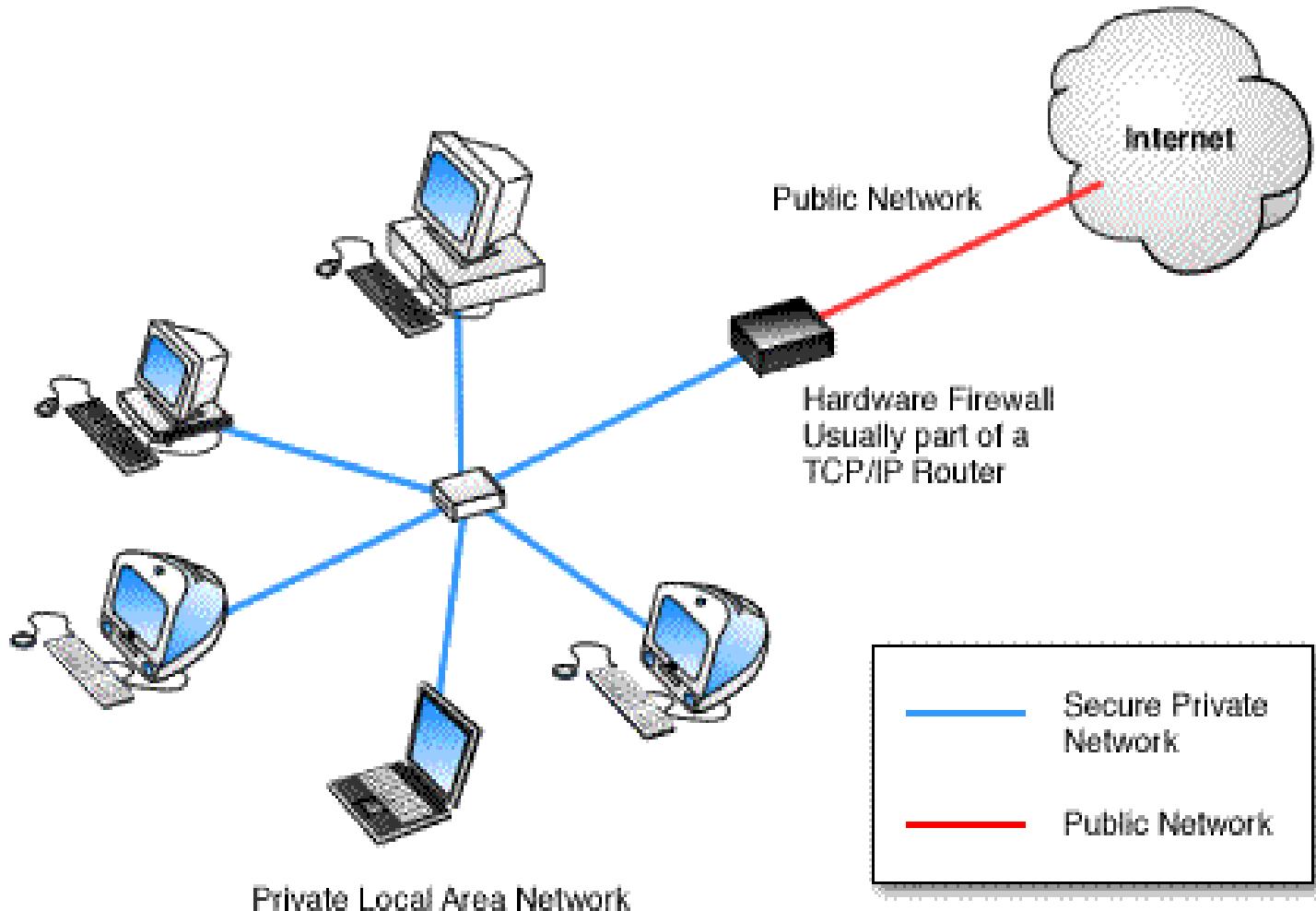
- There are two access denial methodologies used by firewalls.
 - A firewall may allow all traffic through unless it meets certain criteria, or it may deny all traffic unless it meets certain criteria.
 - The type of criteria used to determine whether traffic should be allowed through varies from one type of firewall to another.
 - How a firewall determines what traffic to let through depends on which network layer it operates at.

- Firewalls operate at different layers to use different criteria to restrict traffic.
- The lowest layer at which a firewall can work is layer three.
 - In the OSI model this is the network layer.
 - In TCP/IP it is the Internet Protocol layer.
 - This layer is concerned with routing packets to their destination. At this layer a firewall can determine whether a packet is from a trusted source, but cannot be concerned with what it contains or what other packets it is associated with.

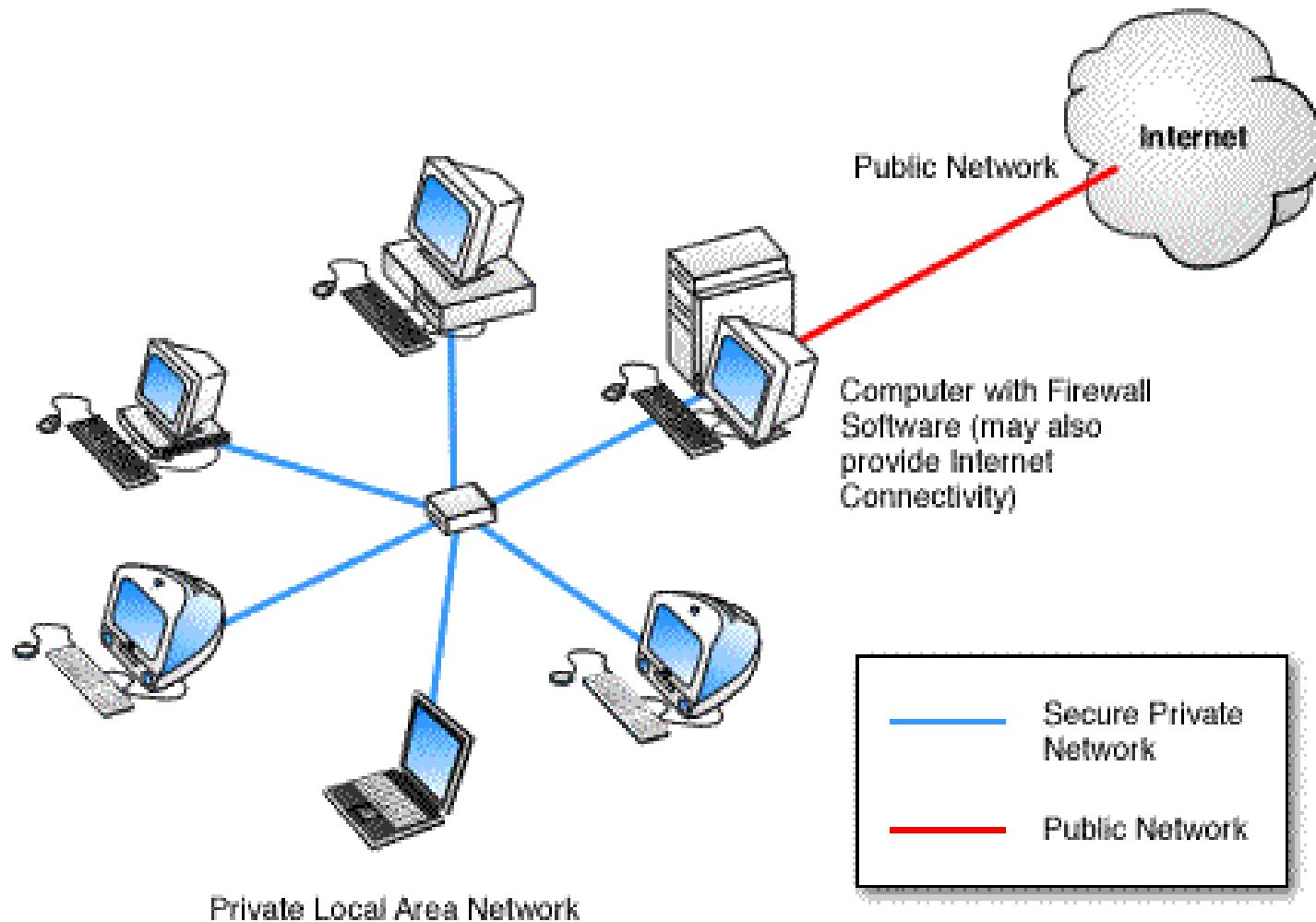
Basic Firewall Operation



Hardware Firewall



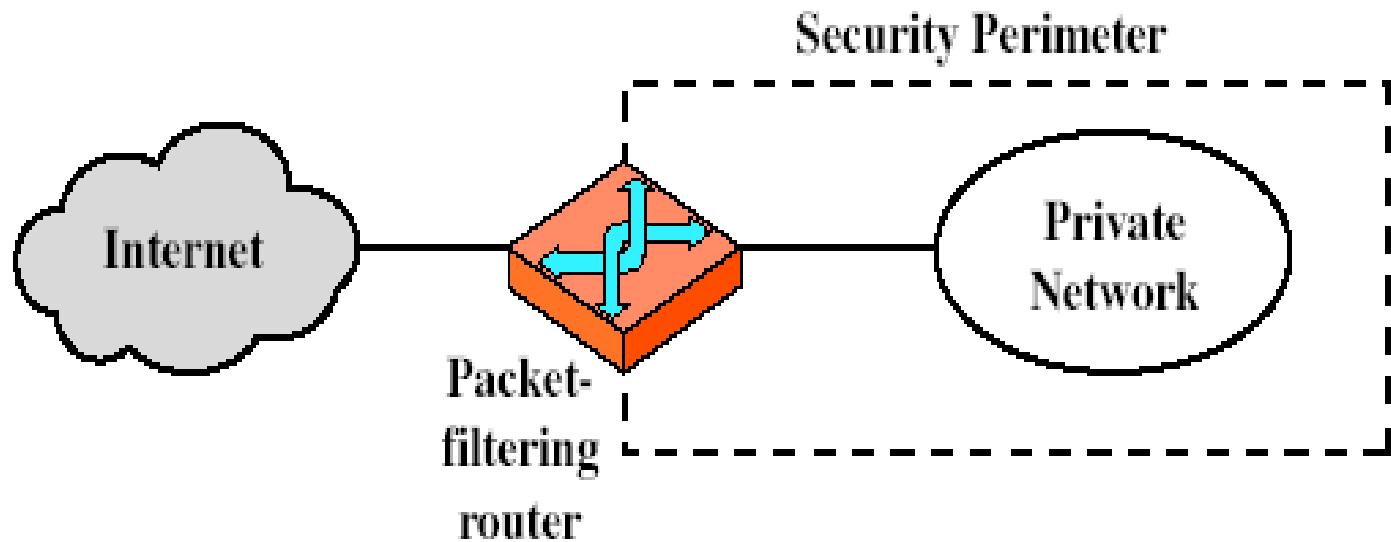
Computer with Firewall Software



Types of Firewalls

- Three common types
 - Packet-filtering router
 - Application-level gateway
 - Circuit-level gateway

Packet-filtering Router



(a) Packet-filtering router

Packet-filtering Router

- Packet filtering firewalls work at the network level of the OSI model, or the IP layer of TCP/IP. They are usually part of a router.
- In a packet filtering firewall each packet is compared to a set of criteria before it is forwarded.
 - Depending on the packet and the criteria, the firewall can drop the packet, forward it or send a message to the originator.
 - Rules can include source and destination IP address, source and destination port number and protocol used.
- The advantage of packet filtering firewalls is their low cost and low impact on network performance.
- Most routers support packet filtering.

A Simple Packet-filtering Router : Example

```
boolean allow (packet) {  
    if (! match (packet.source,  
                "130.194.*.*"))  
        return false;  
    /* Only allow packets from 130.194.*.* */  
    else if (match (packet.source,  
                    "140.194.225.*"))  
        return false;  
    /* Allow all packets from 130.194.*.*,  
       except from subnet 225.* */  
    else  
        return true;  
}
```

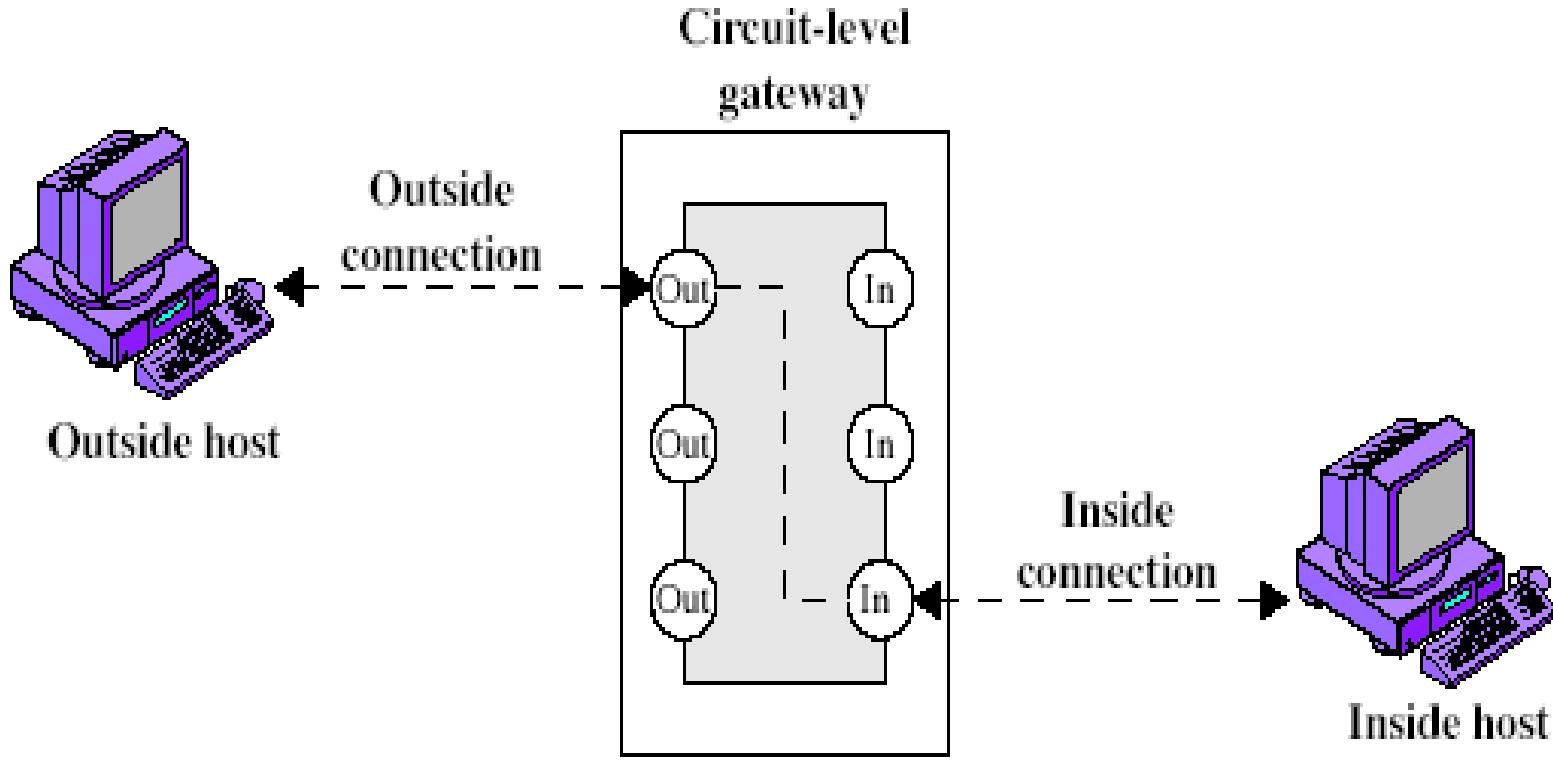
- Advantages:
 - Simplicity
 - Transparency to users
 - High speed
- Disadvantages:
 - Difficulty of setting up packet filter rules
 - Lack of Authentication

Attacks on Packet Filters

- IP address spoofing
 - fake source address to be trusted
 - add filters on router to block
- Source routing attacks
 - attacker sets a route other than default
 - block source routed packets
- Tiny fragment attacks
 - split header info over several tiny packets
 - either discard or reassemble before check

- **Source routing** is a method that can be used to specify the route that a packet should take through the network. In source routing the path through the network is set by the source or a device that tells the network source the desired path.
- **IP address spoofing** denotes the action of generating IP packets with fake source IP addresses in order to impersonate other systems.
- **Tiny Fragment Attack** is a class of attack on Internet firewalls taking advantage that it is possible to impose an unusually small fragment size on outgoing packets.
 - If the fragment size is made small enough to force some of a TCP packet's TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match. If the filtering implementation does not enforce a minimum fragment size, a disallowed packet might be passed because it didn't hit a match in the filter.

Circuit Level Gateway



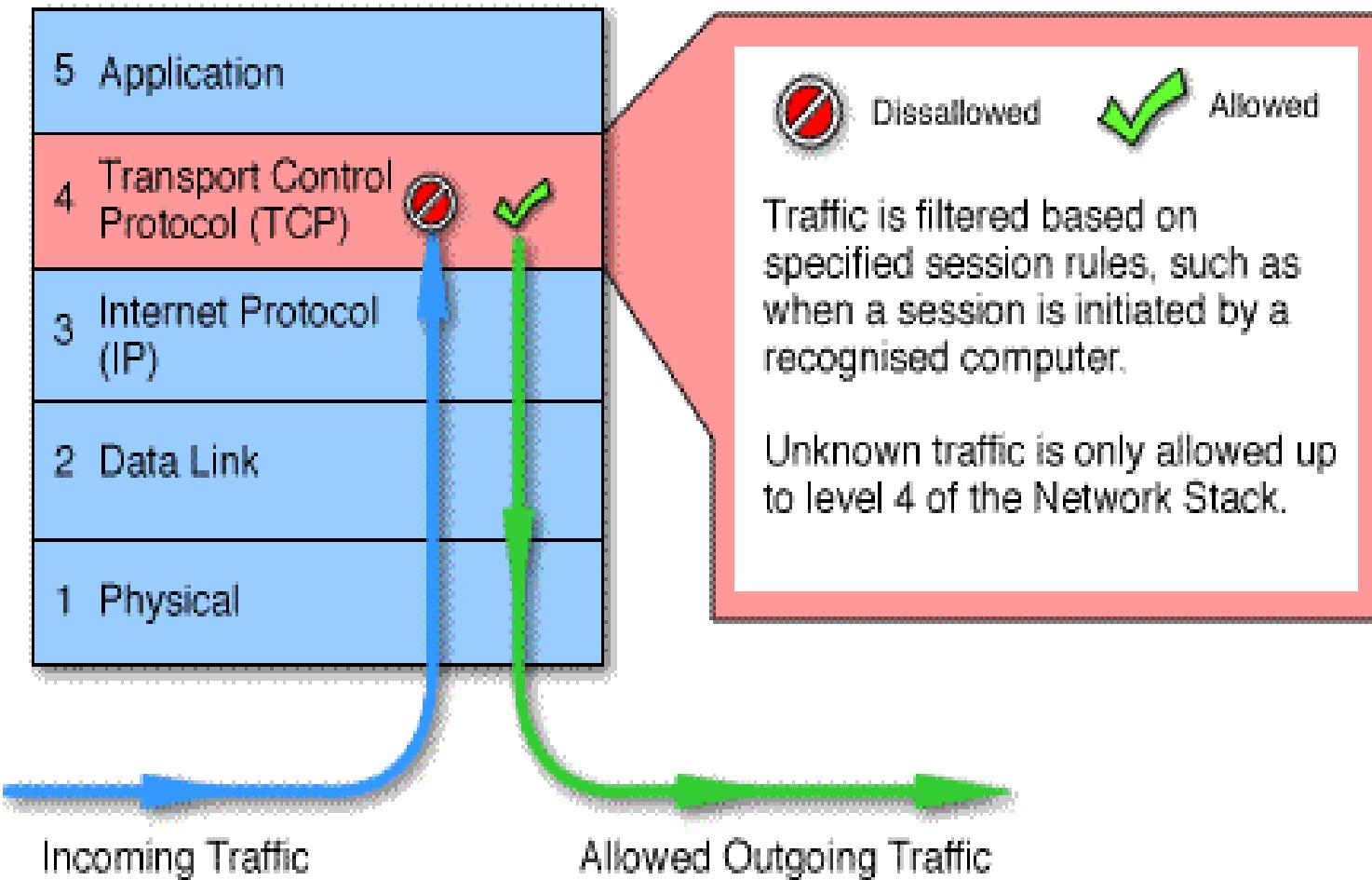
(c) Circuit-level gateway

Circuit Level Gateway

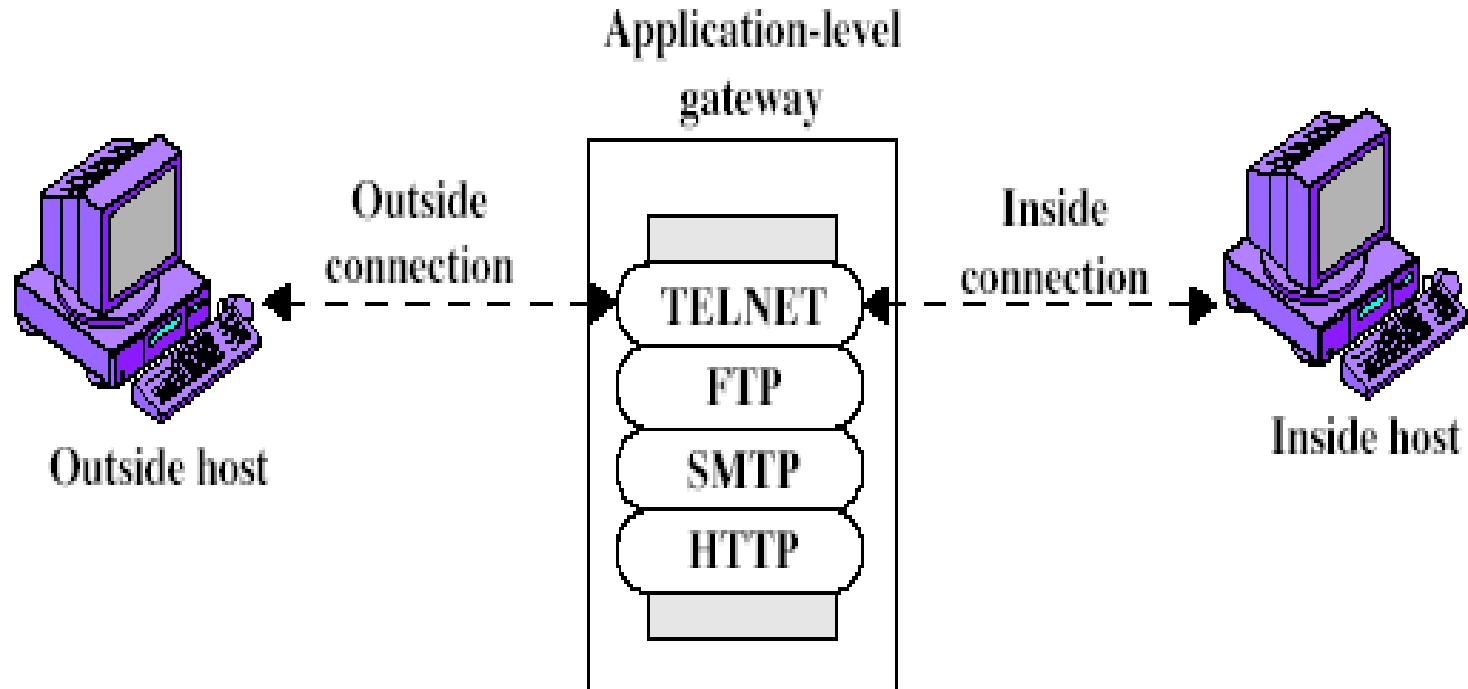
- The security function consists of determining which connections will be allowed.
- Relays two TCP connections without examining contents
 - Typically use is a situation in which the system administrator trusts the internal users.
- Circuit level gateways work at the session layer of the OSI model, or the TCP layer of TCP/IP.
- Circuit level gateways are relatively inexpensive and have the advantage of hiding information about the private network they protect. On the other hand, they do not filter individual packets.

- Whether a connection is valid may be based upon the predefined rules. Eg:
 - Valid destination IP address and/or port
 - Valid source IP address and/or port
 - Allowed time of day
 - Valid protocol

Circuit level Gateway



Application Level Gateway

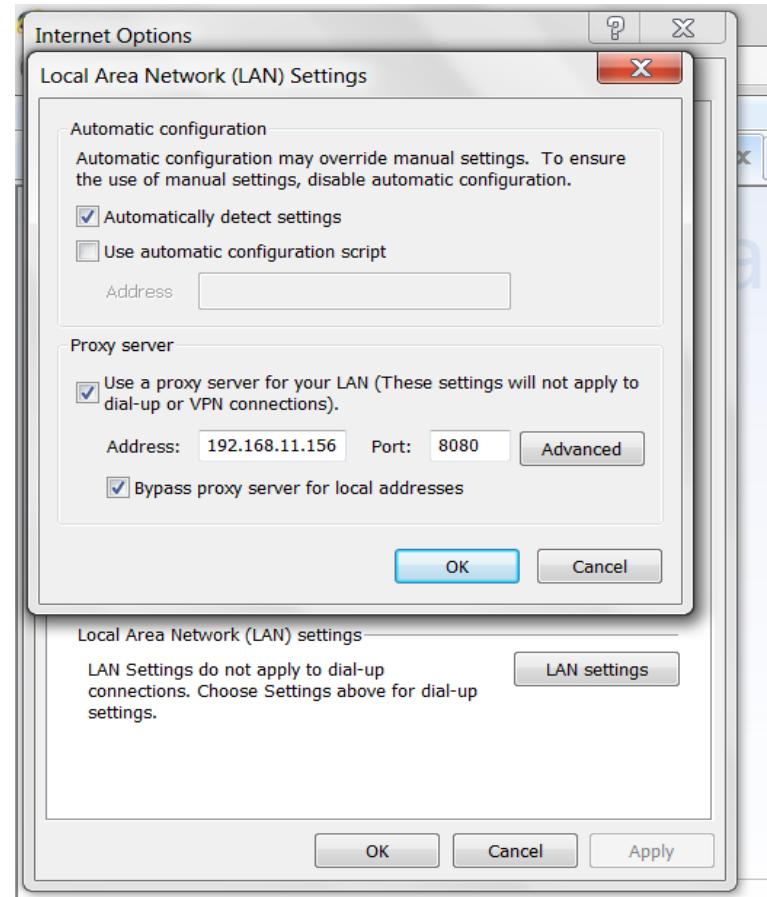


(b) Application-level gateway

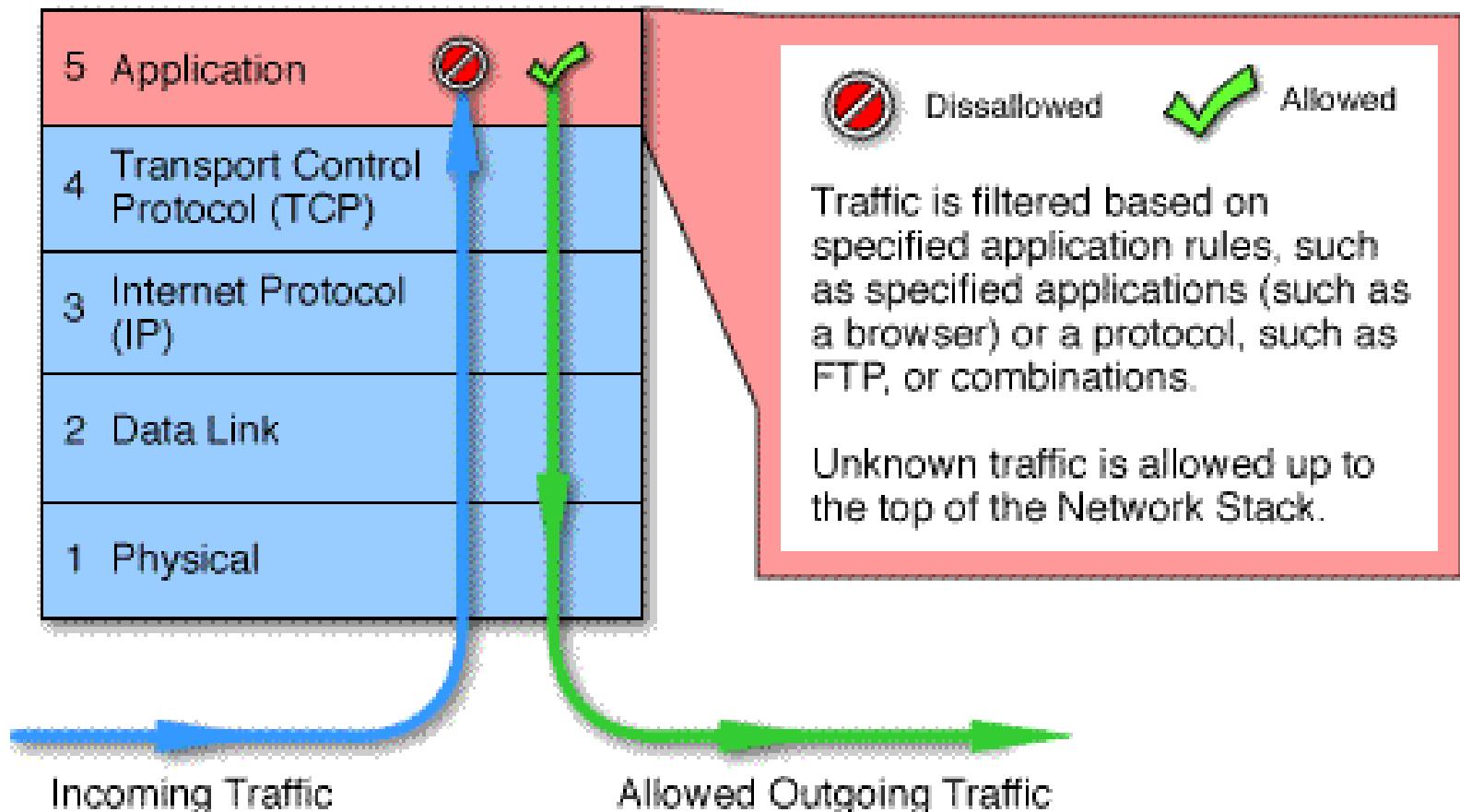
- Application level gateways, also called proxies, are similar to circuit-level gateways except that they are application specific.
- They can filter packets at the application layer of the OSI model.
- An application level gateway that is configured to be a web proxy will not allow any ftp, gopher, telnet or other traffic through.
 - Because they examine packets at application layer, they can filter application specific commands such as http:post and get, etc.

- When a user on the trusted networks wishes to connect to a service on the untrusted network such as the Internet, the application is directed to the proxy server on the firewall.
 - The proxy server evaluates the request and decides to permit or deny the request based on a set of rules that are managed for the individual network service.
- When packets from the outside arrive at the gateway, they are examined and evaluated to determine if the security policy allows the packet to enter into the internal network.
 - Not only does the server evaluate IP addresses, it also looks at the data in the packets to stop hackers from hiding information in the packets.

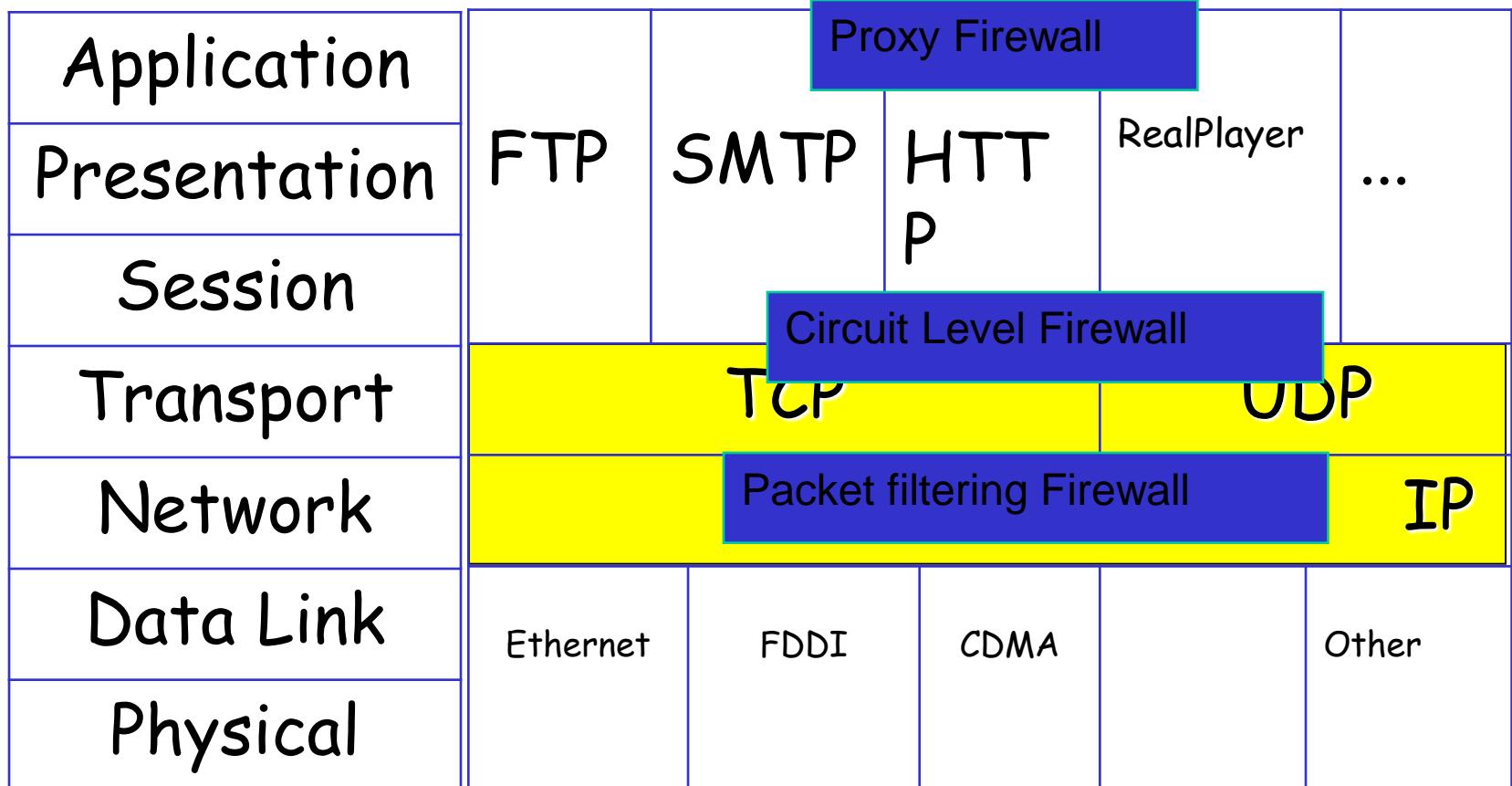
- Application level gateways can also be used to log user activity and logins.
- They offer a high level of security, but have a significant impact on network performance.
- They are not transparent to end users and require manual configuration of each client computer.



Application level Gateway



Firewalls at Different Layers



Bastion Host

- A system identified by the firewall administrator as a critical strong point in the network's security.
- The bastion host serves as a platform for an application-level or circuit-level gateway.
- The Bastion Host generally hosts a single application, for example a proxy server, and all other services are removed or limited to reduce the threat to the computer.

- All access to an intranet from the Internet will be required to come through the bastion host.
 - By concentrating all access in a single server, or a small group of servers, it's much easier to protect the entire intranet.
- The bastion host does not provide intranet services itself.
 - When it receives a request from the Internet for an intranet service, the host passes the request to the appropriate server. Subsequently, it takes the response and passes it back to the Internet.

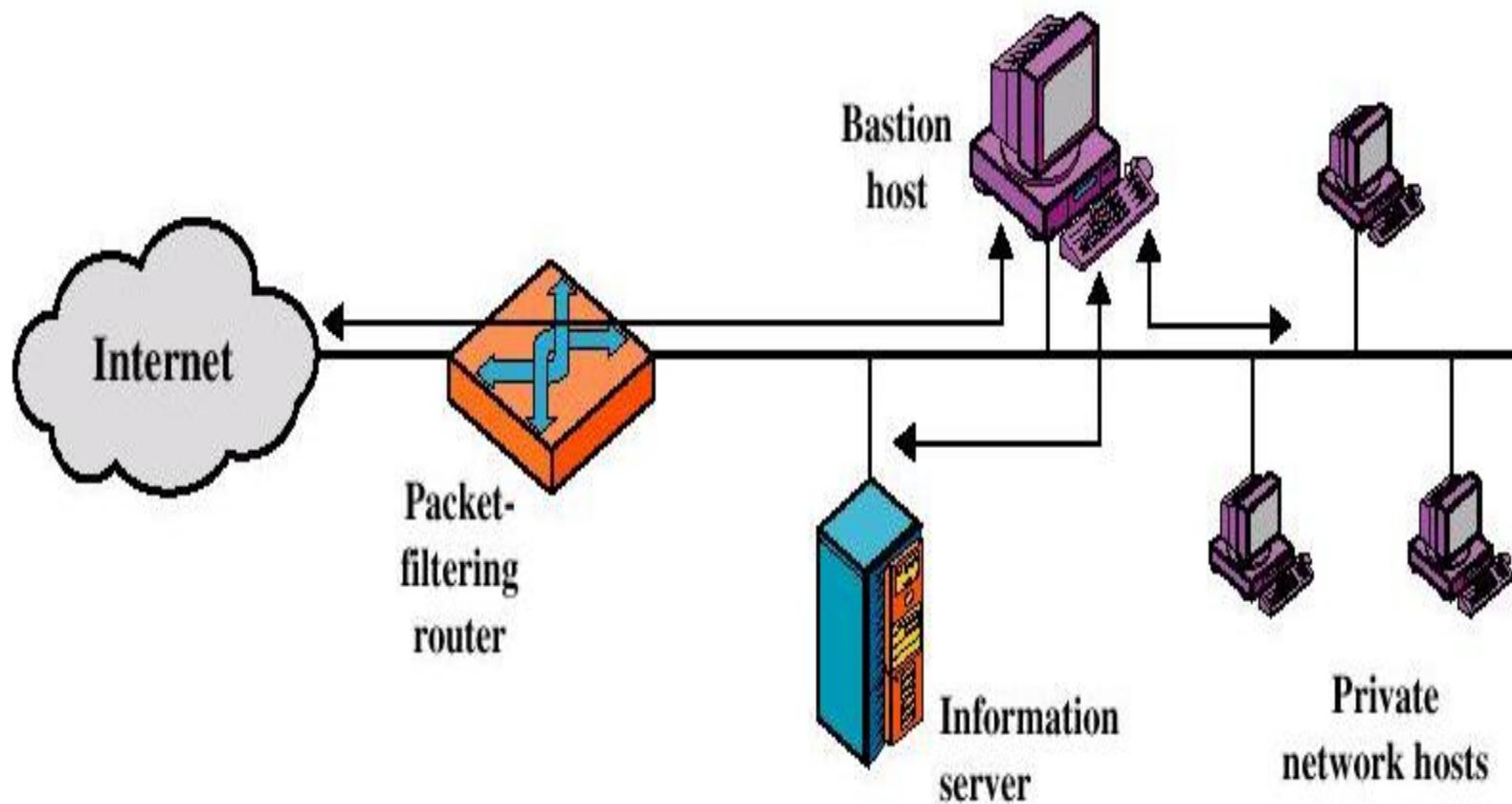
Firewall Configurations

- In addition to the use of simple configuration of a single system (single packet filtering router), more complex configurations are possible.

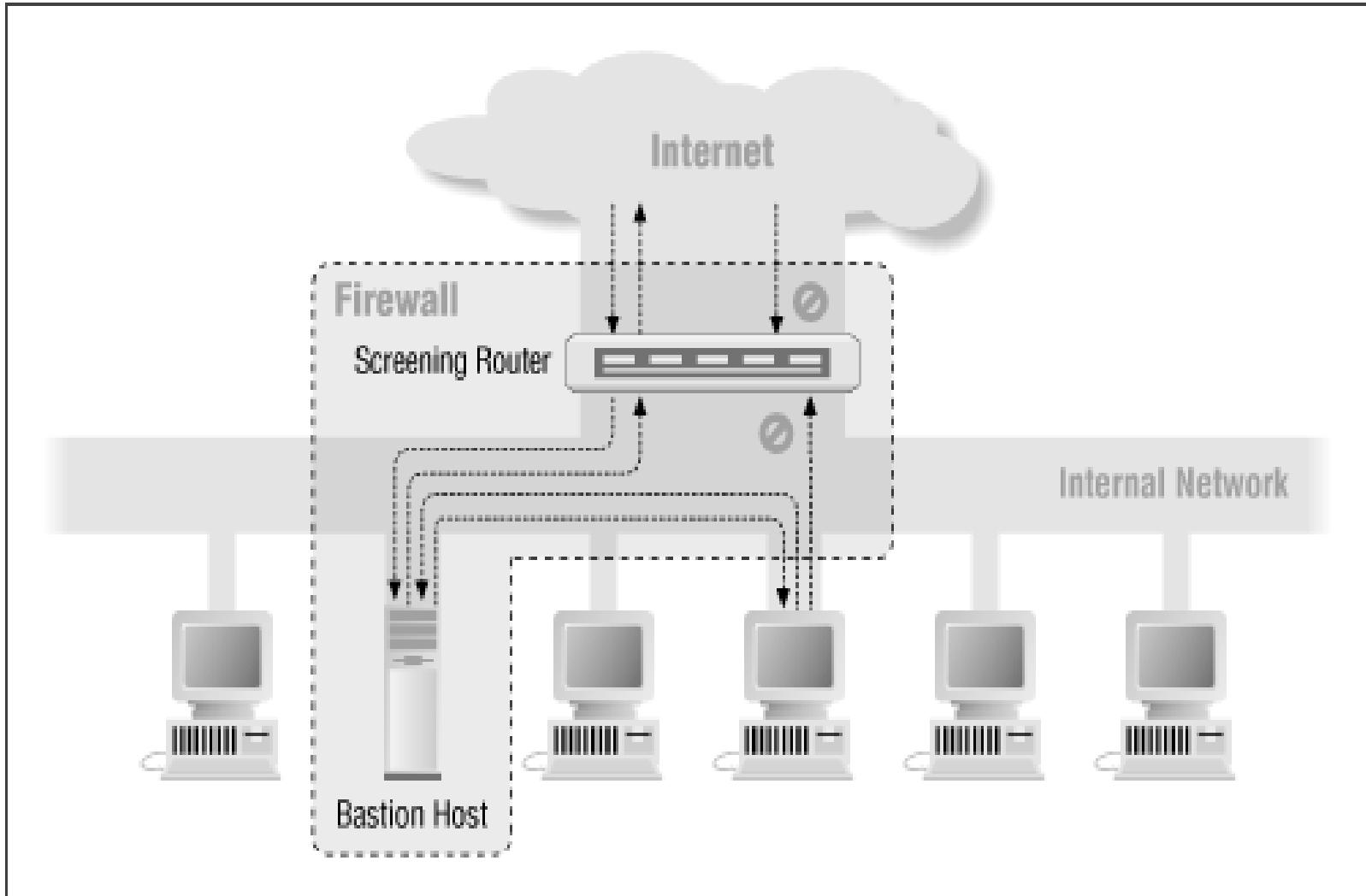
Three common configurations

- Screened host firewall system (single-homed bastion host)
- Screened host firewall system (dual-homed bastion host)
- Screened-subnet firewall system

Screened host firewall system (single-homed bastion host)

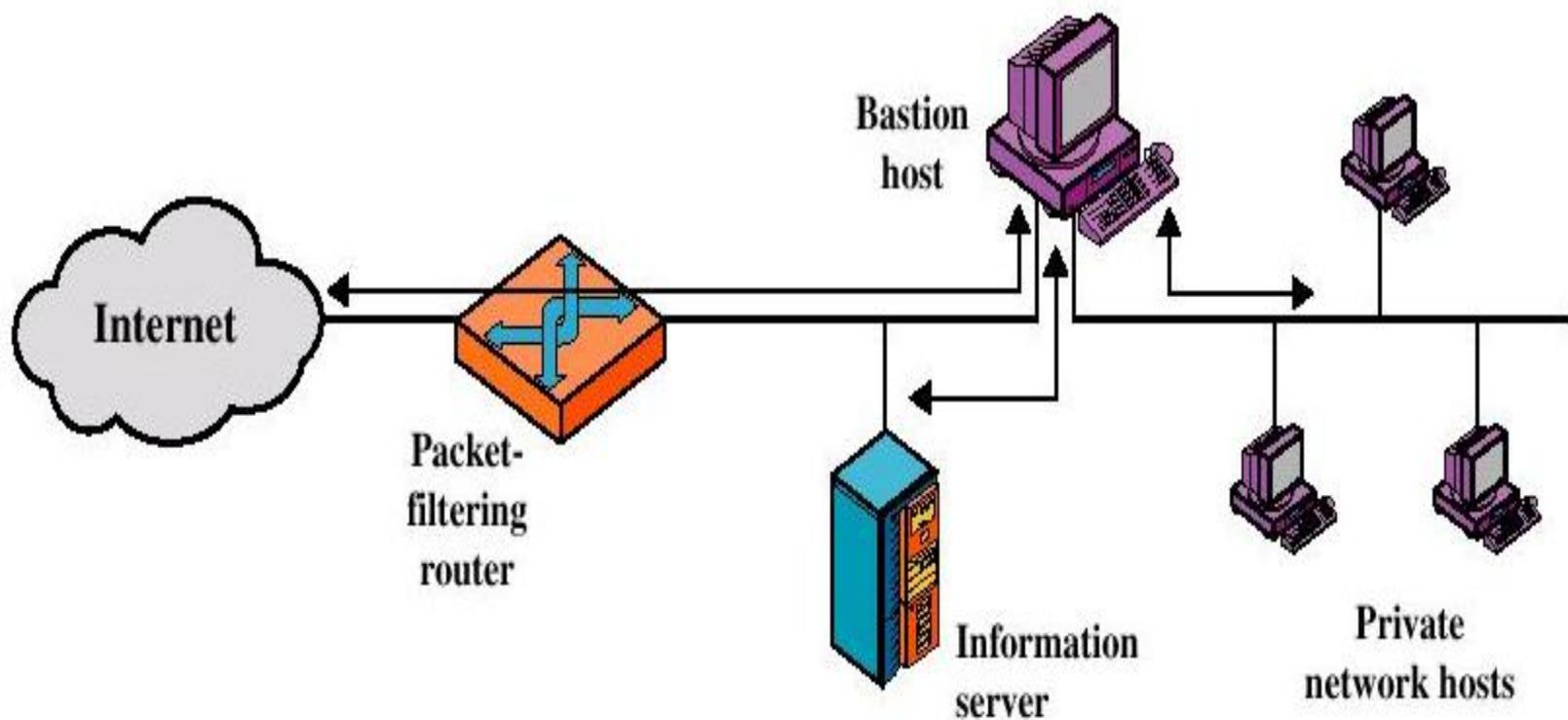


Screened host architecture

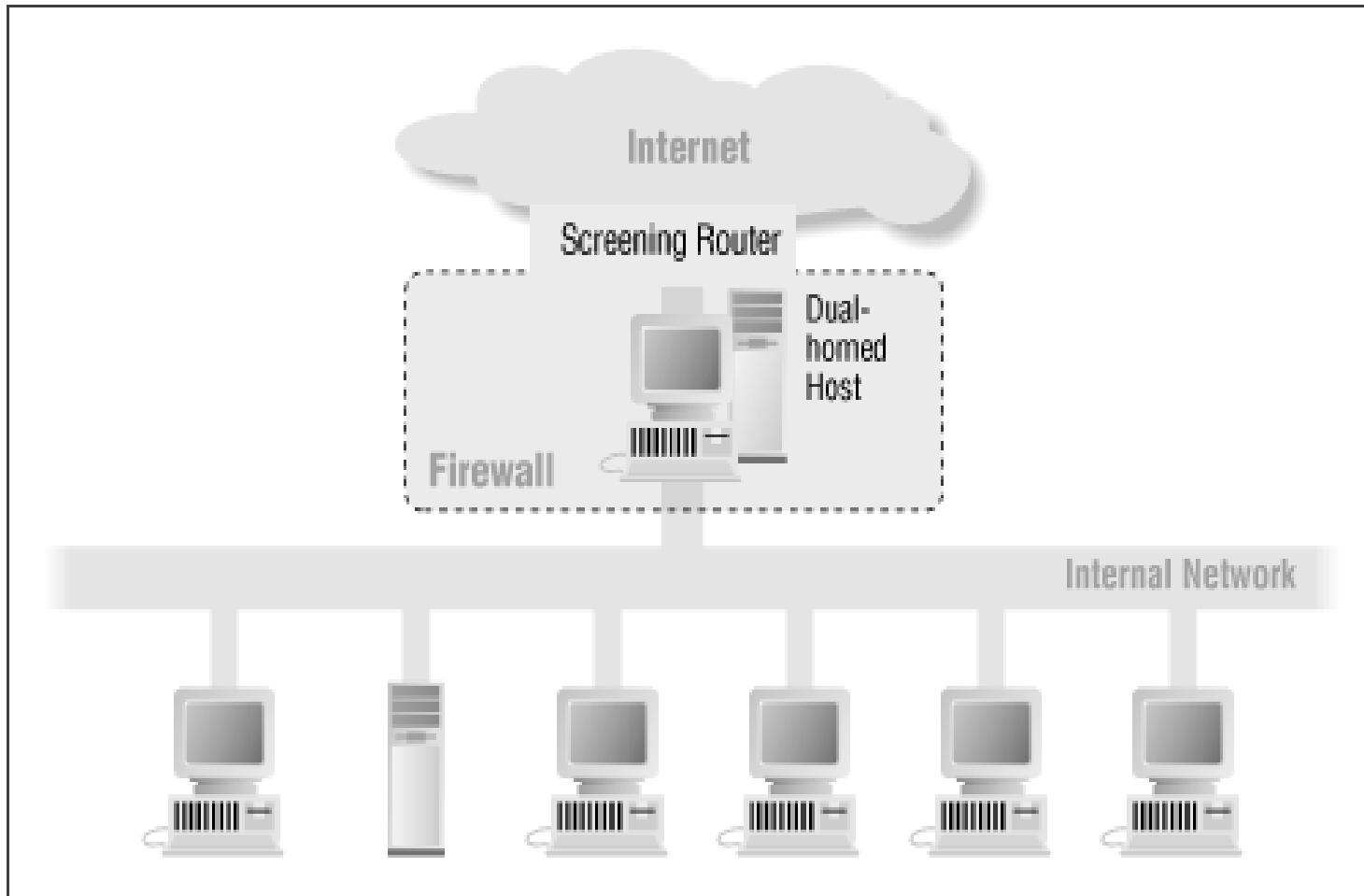


- The most common firewall configuration is a screened host gateway.
 - Implemented using a packet-filtering router and a bastion host.
 - Often the packet-filtering router is configured to block traffic to the bastion host on specific ports, permitting only a small number of services to communicate with it.
- The bastion host performs authentication and proxy functions.

Screened host firewall system (dual-homed bastion host)



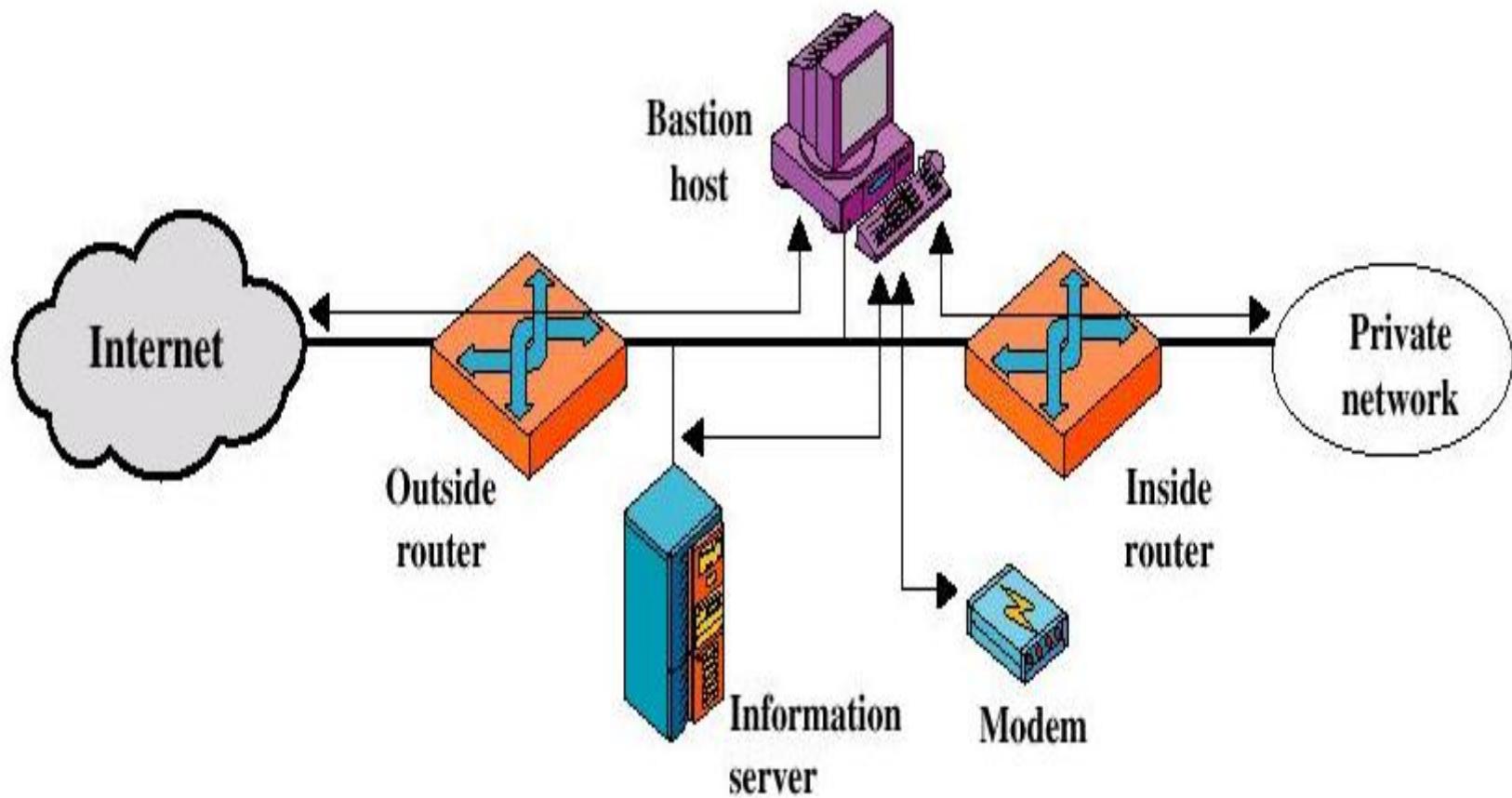
Dual-homed host architecture



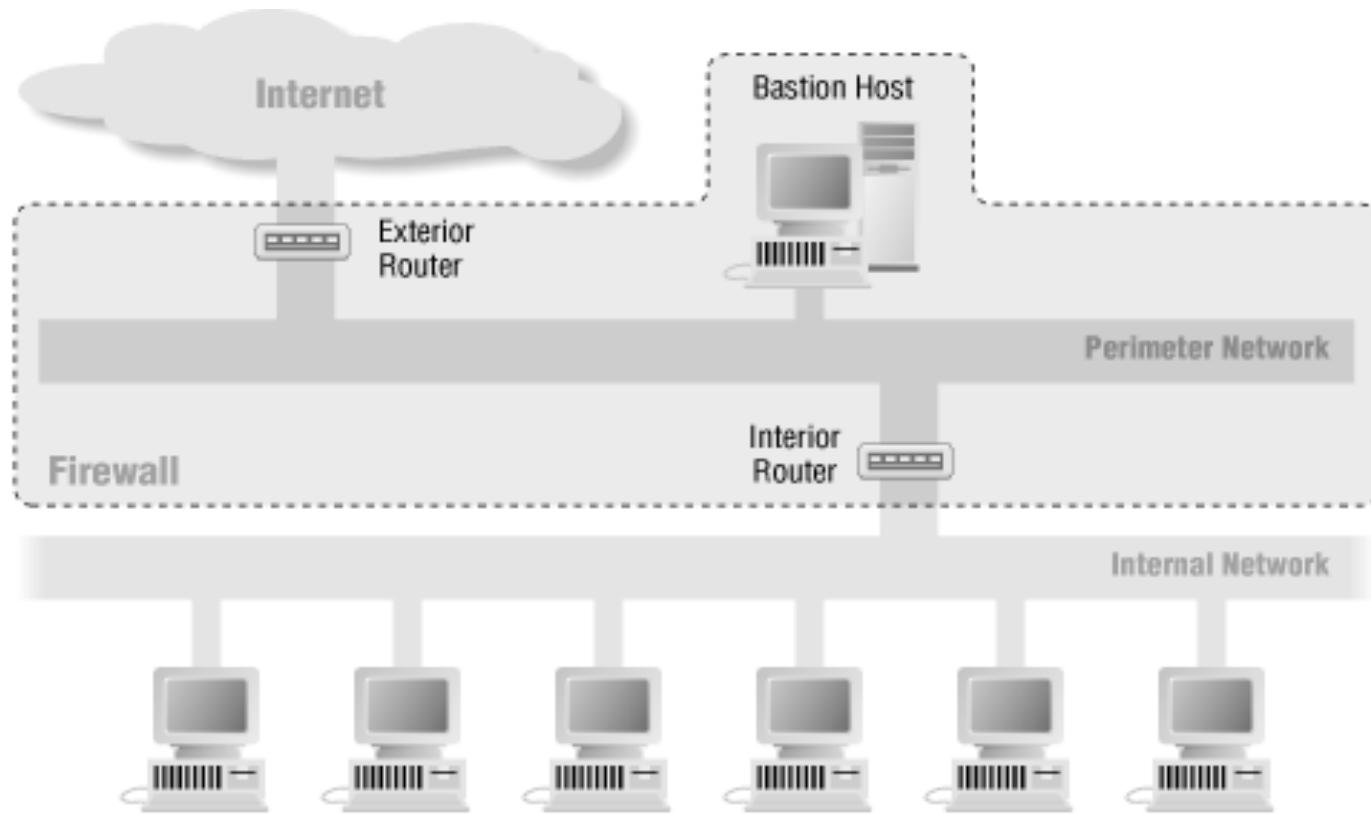
- Screened host firewall, dual-homed bastion configuration
 - Only packets from and to the bastion host are allowed to pass through the router.
 - Traffic between the Internet and other hosts on the private network has to flow through the bastion host.
 - The packet-filtering router is not completely compromised.

- Greater security than single configurations because of two reasons:
 - This configuration implements both packet-level and application-level filtering (allowing for flexibility in defining security policy).
 - An intruder must generally penetrate two separate systems.
 - This configuration also affords flexibility in providing direct Internet access (e.g. Web server).

Screened-subnet firewall system



Screened-subnet firewall system



There are two screening routers. One sits between the perimeter net and the internal network, and the other sits between the perimeter net and the external network (usually the Internet). Even if the attacker somehow broke in to the bastion host, he'd still have to get past the interior router. There is no single vulnerable point that will compromise the internal network.

- Screened subnet firewall configuration
 - Most secure configuration of the three.
 - Two packet-filtering routers are used.
 - Creation of an isolated sub-network.
- Advantages:
 - Three levels of defense to thwart intruders.
 - The outside router advertises only the existence of the screened subnet to the Internet (internal network is invisible to the Internet).
 - The inside router advertises only the existence of the screened subnet to the internal network (the systems on the inside network cannot construct direct routes to the Internet).

Java Security

JAVA

- A revolutionary application platform from Sun Microsystems.
- Like other operating systems, the Java platform provides developers with I/O, networking, windows and graphics capabilities and other facilities needed to develop and run sophisticated applications.
- The Java platform also provides an important capability not found in traditional operating systems.
 - Write Once/Run Anywhere executables, allows Java programs written on one type of hardware or operating system to run unmodified on almost any other type of computer.

- Write Once/Run Anywhere executables
 - Developers working on a Sun Ultra computer running the Solaris operating system can produce an executable which also runs on Windows PCs, Macintosh and many other types of computers without any porting.
 - This frees up development resources for other work and ensures that new applications and new versions of old applications are simultaneously available for all platforms in an organization.

Java Virtual Machine

- Java provides its Write Once/Run Anywhere capability through the Java Virtual Machine.
- The Virtual Machine is implemented on top of a machine's native operating system.
- Java applications run on top of the virtual machine.
- The virtual machine insulates the application from differences between underlying operating systems and hardware and ensures cross platform compatibility among all implementations of the Java platform

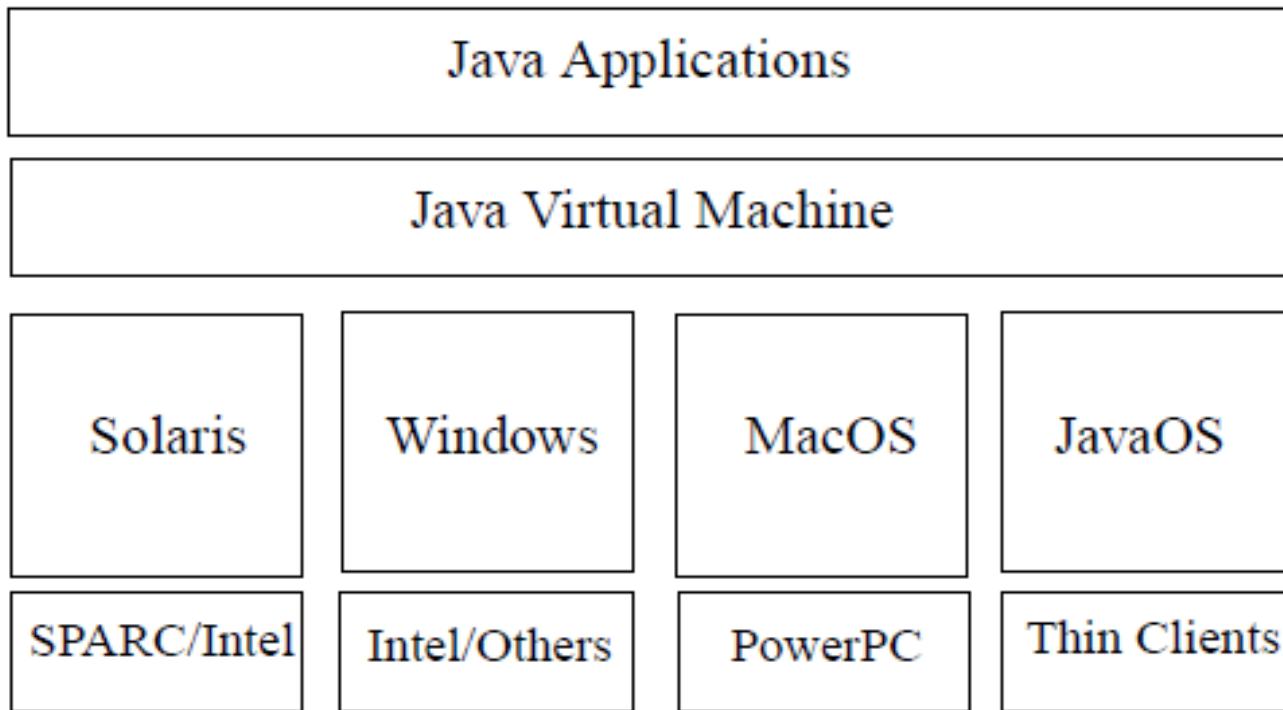


Fig. 1 The Java Virtual Machine sits between a native operating system and Java applications, allowing a single executable to run on many different systems.

Applets

- Web Applets are one of the most exciting uses of the Java Platform.
- Applets are small pieces of executable code which may be included in Web pages and which run inside of the user's browser.
- While traditional web pages have been limited to simple text and graphics, applets allow web publishers to include sophisticated, interactive applications in their pages.
 - For example, a stock broker might want to publish the results of a financial analysis model.
 - With applets, instead of publishing a simple graph showing the results of the model, the broker could publish the model itself, along with connections to live stock market data and the customer's portfolio.

Applet Security Implications

- Use of applets potentially adds a new security vulnerability.
 - An employee searching an external Web site for information might inadvertently load and execute an applet without being aware that the site contains executable code.
 - This automatic distribution of executables makes it very likely that software will be obtained from untrusted third parties.
 - Since the applet is imported into the user's web browser and runs locally, this software could potentially steal or damage information stored in the user's machine on a network file server.
 - Since this software is already behind the company's firewall, the applet could attack other unprotected machines on a corporate intranet.
 - *These attacks would not be stopped by traditional security measures.*

- Java protects its users from these dangers by placing strict limits on applets.
 - Applets cannot read from or write to the local disk.
 - Stand-alone windows created by applets are clearly labelled as being owned by untrusted software.
 - These limits prevent malicious applets from stealing information, spreading viruses, or acting as Trojan horses.
 - Applets are also prohibited from making network connections to other computers on the corporate intranet.
 - This prevents malicious applets from exploiting security flaws that might exist behind the firewall or in the underlying operating system.

The Sandbox (JDK 1.0)

- Java's security allows a user to import and run applets from the Web or an intranet without undue risk to the user's machine.
 - The applet's actions are restricted to its “sandbox”, an area of the web browser dedicated to that applet.
 - The applet may do anything it wants within its sandbox, but cannot read or alter any data outside of its sandbox.
 - The sandbox model is to run untrusted code in a trusted environment so that if a user accidentally imports a hostile applet, that applet cannot damage the local machine.

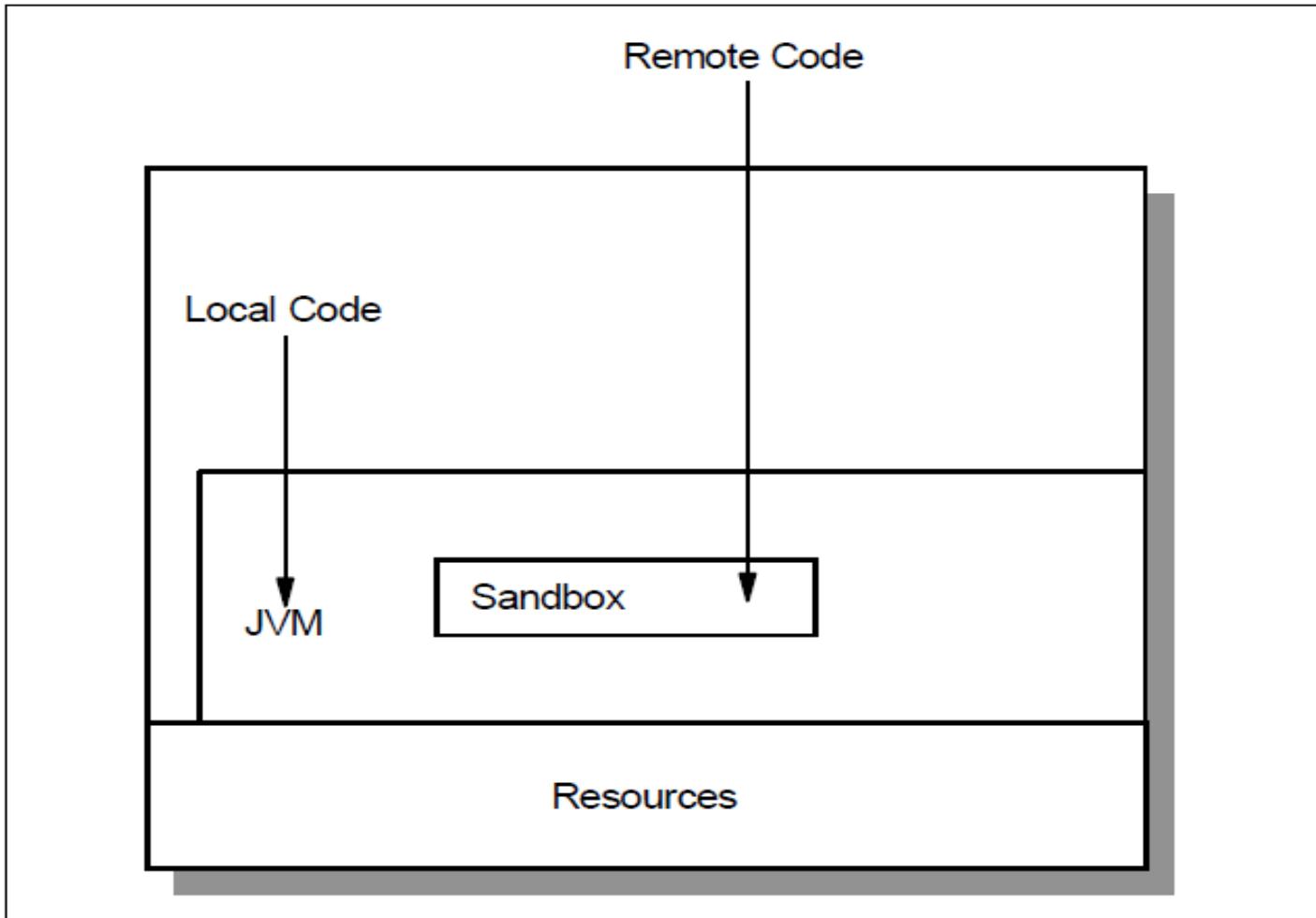


Figure 30. JDK 1.0 Sandbox Security Model

Sandbox vs. OS

- Because most operating systems allow applications broad access to the machine, especially in PCs where very little protection is provided by the operating system, the runtime environment cannot be trusted.
- To compensate for this lack, security policies often require *a level of trust to be established in the application* before it is executed.
 - For example, an organization might require that before an employee runs an application obtained from the web, that application must be checked for viruses and its source code examined for malicious code.

Sandbox vs. OS

- There are two problems with this approach.
 - First, the checks required to build trust in the application may be too complex and time-consuming to be practical.
 - Few employees will take the time to read the source code of an application and compile it locally to ensure that it takes no hidden hostile actions.
 - Second, virus checkers require constant maintenance in order to remain effective.
 - They must be updated with samples of newly discovered viruses and must be installed on each machine.
 - Also, many virus checkers can be turned off, either accidentally, as part of an installation procedure.
 - If the checker is accidentally left off, the machine and possibly the entire organization are at risk.

Sandbox vs. OS

- Java solves these problems by automatically confining applets to the sandbox.
- End-users do not have to take any special action in order to ensure the safety of the machine.
 - Because the sandbox prevents the actions required to spread a virus or steal information, instead of trying to identify a virus infected executable or potential attacker, the sandbox does not require periodic updates with new viruses.

Building The Sandbox

- The sandbox is made up of several different systems operating together:
 - Class Loader
 - Verifier
 - Security Manager

Class Loader

- When an applet is to be imported from the network, the web browser calls the applet *class loader*.
 - In addition to fetching an applet's executable code from the network, the class loader enforces the name space hierarchy.
 - A name space controls what other portions of the Java Virtual Machine an applet can access.
 - By maintaining a separate name space for trusted code which was loaded from the local disk, the class loader prevents untrusted applets from gaining access to more privileged, trusted parts of the system.
 - Applets downloaded from the net cannot create their own class loaders.
 - Downloaded applets are also prevented from invoking methods in the system's class loader.

Verifier

- Before running a newly imported applet, the class loader invokes the verifier.
 - The verifier checks that the applet conforms to the Java language specification and that there are no violations of the Java language rules or name space restrictions.
 - The verifier also checks for common violations of memory management, like stack underflows or overflows, and illegal data type casts, which could allow a hostile applet to corrupt part of the security mechanism or to replace part of the system with its own code.

Security Manager

- The security manager enforces the boundaries around the sandbox.
- Whenever an applet tries to perform an action which could corrupt the local machine or access information, the Java Virtual Machine first asks the security manager if this action can be performed safely.
- If the security manager approves the action
 - for example, a trusted applet from the local disk may be trying to read the disk, or an imported untrusted applet may be trying to connect back to its home server — the virtual machine will then perform the action.
 - Otherwise, the virtual machine raises a security exception and writes an error to the Java console.

Security Manager

- The security manager will not allow an untrusted applet to read or write to a file, delete a file, get any information about a file, execute operating system commands or native code, load a library, or establish a network connection to any machine other than the applet's home server.
- An application or a web browser can only have one security manager.
 - Assures that all access checks are made by a single security manager enforcing a single security policy.
- The security manager is loaded at start-up and cannot be extended, overridden or replaced.
- Applets can not create their own security managers.

Java Language Features

- Java has several language features which protect the integrity of the security system and which prevent several common attacks.
 - For example, Java programs are not allowed to define their own memory pointers or to access physical memory directly. This prevents an applet from accessing and modifying critical parts of the security system.
- The language tracks the type of newly created classes and objects so that an applet cannot forge its own class loader or security manager.
- Studies have shown that 40% to 50% of all bugs are caused by errors in memory management.
 - By automating memory management, Java eliminates a large class of bugs; this results in more stable and reliable code.

Security Through Openness (Not a Blackbox)

- Sun (?) published all the details of Java security model when Java was first released.
 - This included the design specifications for the language mechanisms and the sandbox, and a full source implementation.
 - This approach, dubbed security through openness, was intended to encourage security researchers to examine the Java model and to report any security flaws found; the flaws could be fixed before attacks based on those flaws could become endemic on the Web.

Trusted Code in JDK 1.1

- The security architecture in JDK 1.1 introduced the concept of **signed remote code**.
 - Remote codes (applets), signed by a trusted entity, were permitted access to several of the system resources.
 - A remote code with an appropriate digital signature was treated with the same respect as local code, and so it could be considered trusted.
 - An appropriate digital signature was one that was recognized as trusted by the client.
 - On the other hand, unsigned remote code or remote code signed with a digital signature not recognized as trusted by the client, was still confined to the sandbox.

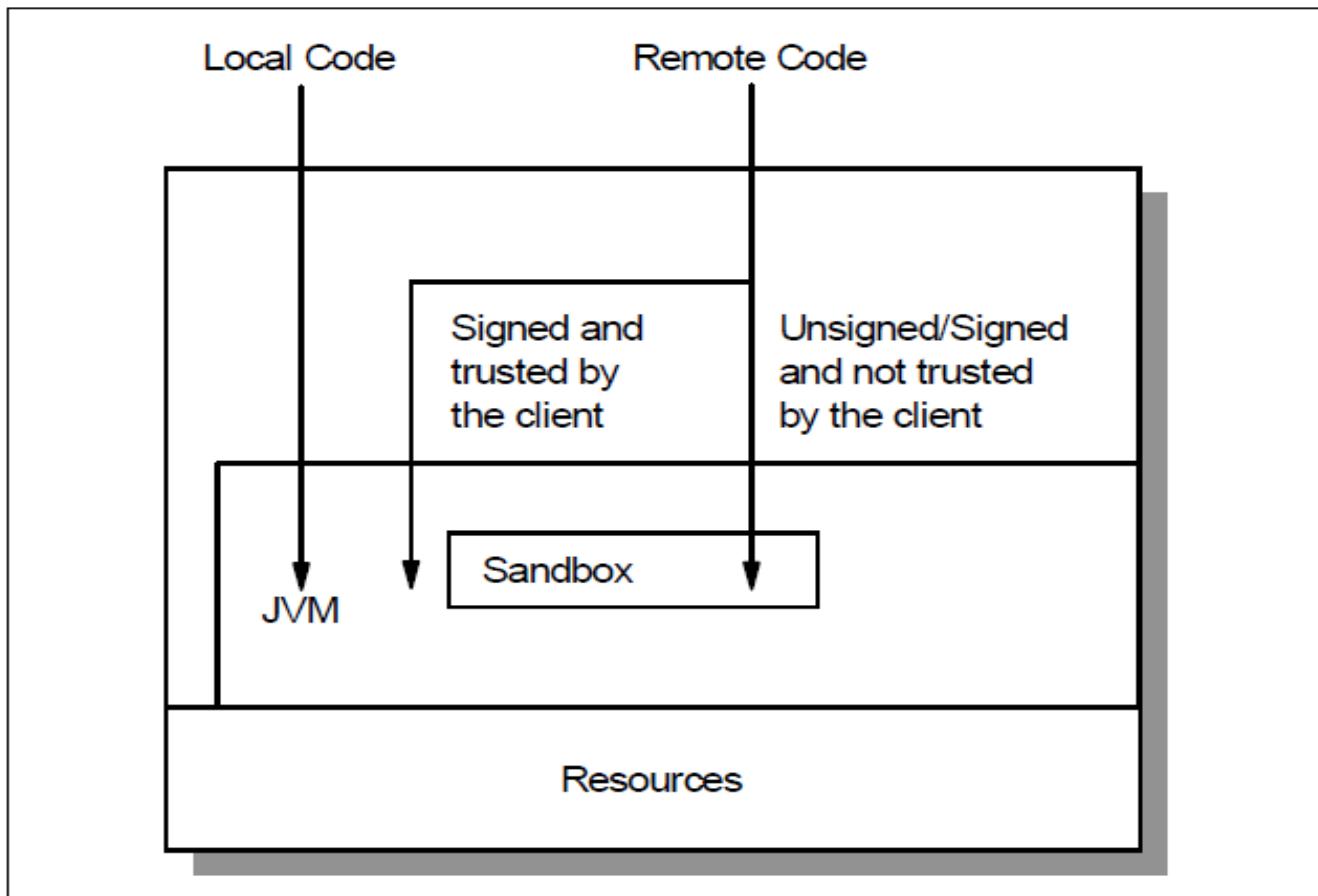


Figure 31. Trusted and Untrusted Code in JDK 1.1

- “Signed applets” give us the same level of confidence in network distributed software.
- *To sign an applet, the producer first bundles all the Java code and related files into a single file called a Java Archive, or JAR.*
- *The producer then creates a string called a digital signature based on the contents of the JAR.*

The Fine-Grained Access Control of Java 2

- Fine-grained access control is the ability to grant specific permissions (*say read and write permission on file x, but only read permissions on file y and no permissions on file z*) to a particular piece of code about accessing specific resources of the client depending on the signers of the code and/or the URL location from which the code was loaded.

The Fine-Grained Access Control of Java 2

- With this security model, all code, whether remotely downloaded or local, signed or unsigned, will have access to system resources based on what is defined in *a policy file*.
 - This allows the client to explicitly specify the permissions to be granted to different signatories of code and different sources.
 - This way the end user can download, install and run applications from the Web by granting them permissions for only those actions that are necessary.

The Fine-Grained Access Control – a scenario

- Consider for example the following scenario, based on the JDK 1.1 security model.
 - You download a little tic-tac-toe program from the Web.
 - It is signed by an entity you **trust**, and you are sure that it will not crash your system. For this reason, you accept to run it.
 - But, this code reads your address book, and sends all the e-mail addresses you have to the database of the nearest junk mailer.
- **With JDK 1.1, you do not have an option to restrict access to code to do only certain things. You either install the software, or you make do without it.**

The Fine-Grained Access Control – a scenario

- If you are running Java 2-enabled software, you can instruct the JVM, through modifications in a **policy file**, that code loaded from a particular URL (local or remote) and/or signed by a particular entity is restricted to specific local resources.
 - For example, you may specify in the policy file that the code in question may read files in one particular directory and can do nothing else – cannot open sockets, cannot write or delete any files, etc.

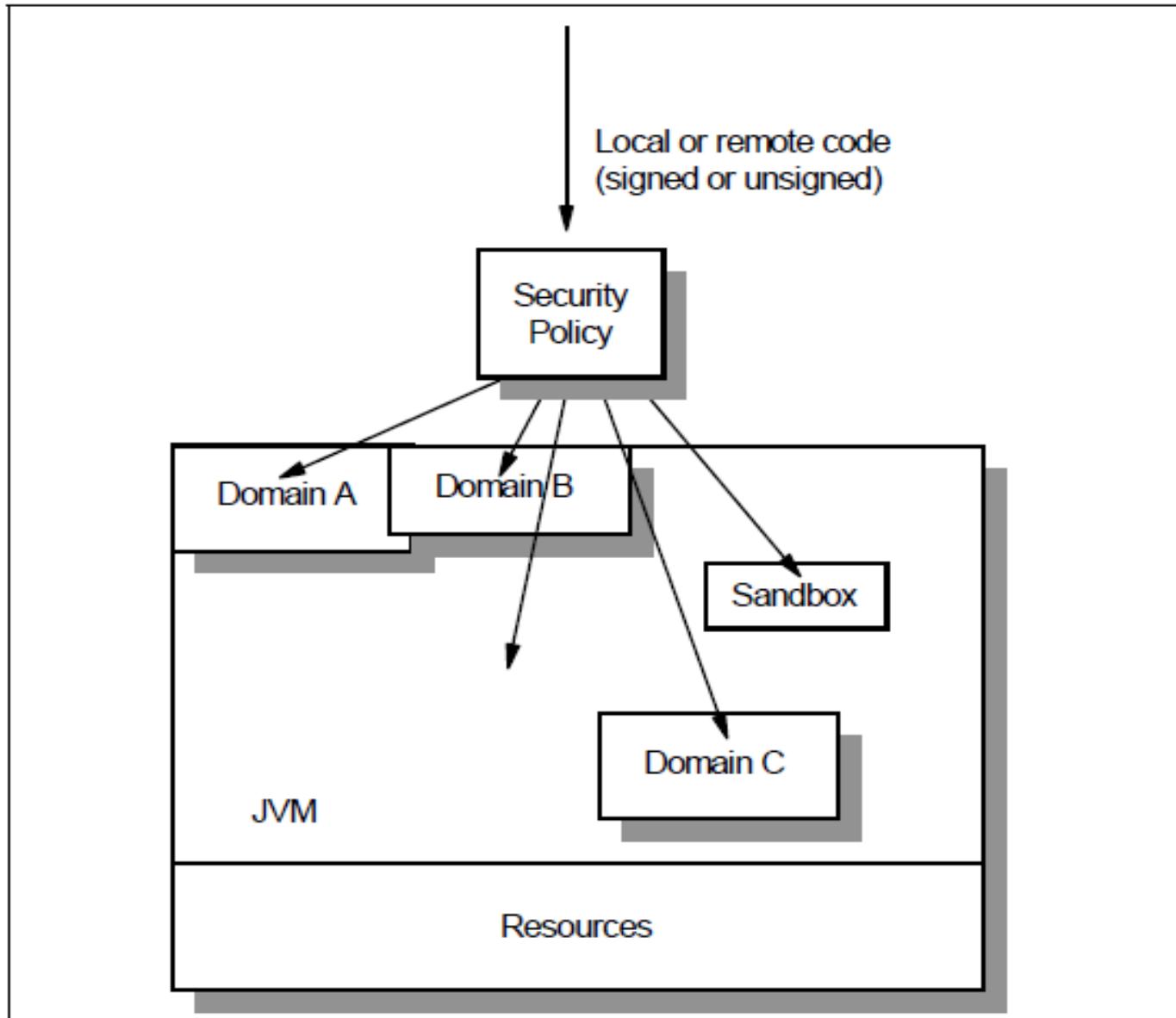


Figure 32. Fine-Grained Access Control Mechanism in Java 2 SDK

- Protection domains generally fall into two distinct categories: *system and application*.
- Each domain (system or application) may also implement additional protection of its internal resources within its own domain boundary.

Lexical Scoping of Privilege Modifications

- This refers to the availability of the option in the security architecture to temporarily grant more privileges to a specific piece of code in an execution thread, which are additional to the privileges the code would have enjoyed by itself.
- This facility is available only with Java 2 and achieved with the help of the doPrivileged() method.

- ## Java Security Manager?

- The Java Security Manager is a class that manages the external boundary of the Java Virtual Machine (JVM) sandbox, controlling how code executing within the JVM can interact with resources outside the JVM.
- When the Java Security Manager is activated the Java API checks with the security manager for approval before executing a wide range of potentially unsafe operations.
- The Security Manager **uses a security policy** to determine whether a given action will be permitted or denied.

- ## Security Policy?

- A set of defined permissions for different classes of code.
- The Java Security Manager compares actions requested by applications against the security policy.
- If an action is allowed by the policy, the Security Manager will permit that action to take place.
- If the action is not allowed by the policy, the Security Manager will deny that action.
- The security policy can define permissions based on the location of code or on the code's signature.

- The Security Manager and the security policy used are configured using the Java Virtual Machine options `java.security.manager` and `java.security.policy` .
- **java.security.manager**
 - Use a security manager, optionally specifying which security manager to use.
 - If no argument is supplied with this option the default JDK security manager, `java.lang.SecurityManager`, is used.
 - To use another security manager implementation, supply the fully qualified classname of a subclass of `java.lang.SecurityManager` with this option.

- **java.security.policy**

- Specifies a policy file to augment or replace the default security policy for the VM. This option takes two forms:
 - `java.security.policy=policyFileURL`
 - The policy file referenced by *policyFileURL* will *augment* the default security policy configured by the VM.
 - `java.security.policy==policyFileURL`
 - The policy file referenced by *policyFileURL* will *replace* the default security policy configured by the VM.
 - The *policyFileURL* value can be a URL or a file path.

Java Virtual Machine

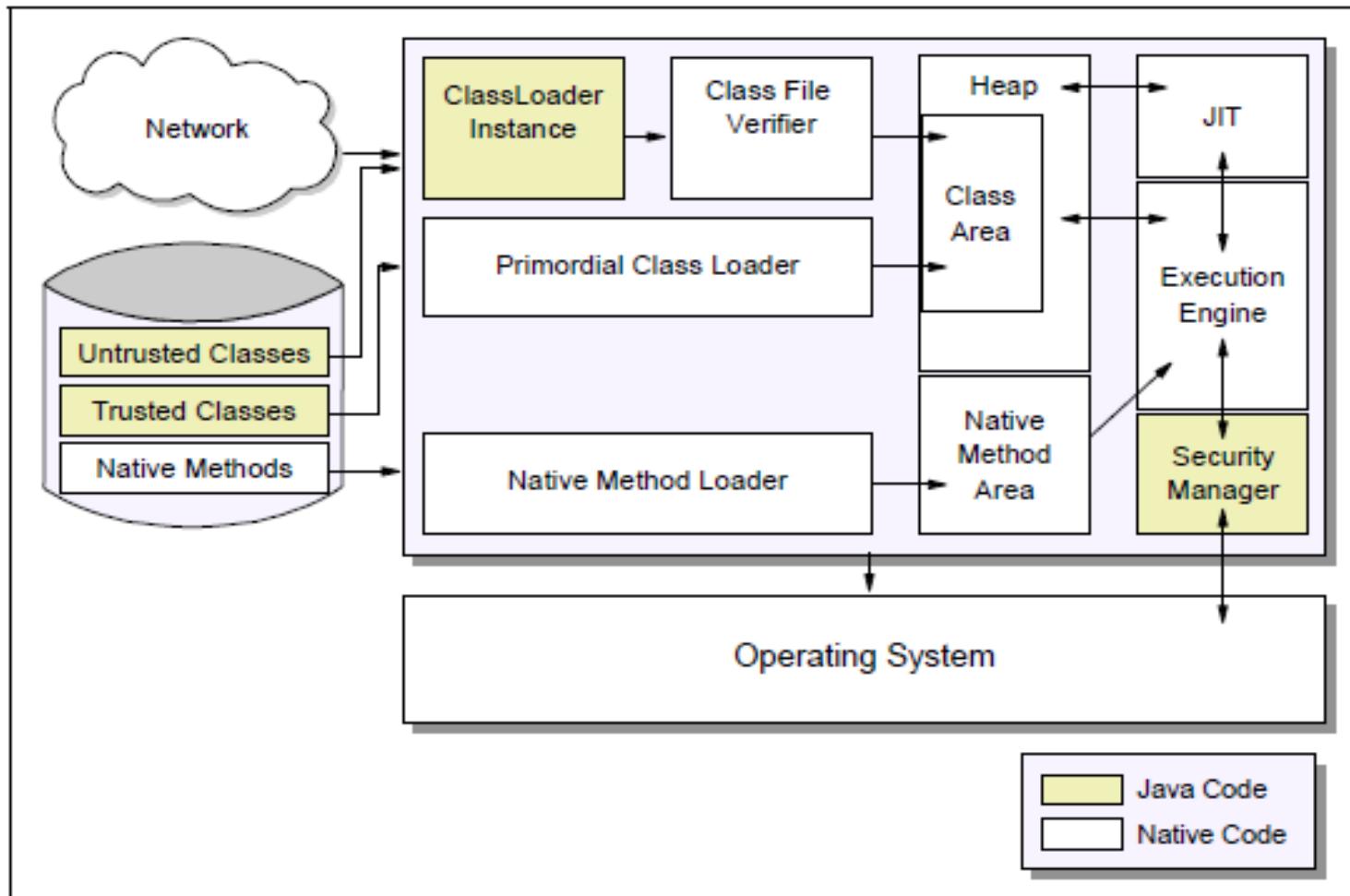


Figure 42. Components of the JVM

Table 1. Evolution of the Java Security Model

	JDK 1.0	JDK 1.1	Java 2 SDK
<i>Local unsigned code resource access</i>	Unconstrained	Unconstrained	Policy based
<i>Local signed code resource access</i>	Not available	Unconstrained if trusted	Policy based
		Constrained by the Java sandbox if untrusted	
<i>Remote unsigned code resource access</i>	Constrained by the Java sandbox	Constrained by the Java sandbox	Policy based
<i>Remote signed code resource access</i>	Not available	Unconstrained if trusted	Policy based
		Constrained by the Java sandbox if untrusted	
<i>Lexical scoping of privilege modification</i>	Not available	Not available	Stack annotation based with doPrivileged()
<i>Cryptographic services for data confidentiality/integrity</i>	Not available	Java Cryptography Extension 1.1	Java Cryptography Extension 1.2
<i>Digital signature services for code signing</i>	Not available	Java Cryptography Architecture DSA signature	Java Cryptography Architecture DSA signature

OS Security??