

ISA 766

Internet Security Protocols

Secure Email, Virus, Worm

ISA 666

Duminda Wijesekera based on .ppts
of F. Wang+J. Kurose

1

Objectives of Lecture

- Understand how e-mail systems operate over networks.
- Classify the threats to the security of e-mail.
- Study how S/MIME and PGP can be used to add security to e-mail systems.
- Examine what other security measures are needed to ensure security for e-mail systems.

ISE at George Mason University

ISA 666 Duminda Wijesekera based on
.ppts of F. Wang+J. Kurose

2

Contents

- Why study e-mail security?
- E-mail – what it is and how it works.
- E-mail security threats.
- Secure e-mail standards and products - PGP and S/MIME.
- E-mail security beyond PGP and S/MIME.
- Virus and Worm

Why Study E-mail Security?

- After web browsing, e-mail is the most widely used network-reliant application.
- Mail servers, after web servers, are the most often attacked Internet hosts.
- Basic e-mail offers little security, counter to public perception.
- Good technical solutions are available, but not widely used.
 - If we understand why this is so, we might understand something about why security is 'hard'.
- Examine some open problems related to E-mail

What It Is and How It Works

- What is an e-mail?
 - RFC 822 and MIME
- How are e-mails transported, accessed and stored?
 - mUAs and mTAs
 - SMTP, POP3 and IMAP

Internet apps: their protocols and transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NSF	TCP or UDP
Internet telephony	proprietary (e.g., Vocaltec)	typically UDP

Transport service requirements of common apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	loss-tolerant	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video: 10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps up	yes, 100's msec
financial apps	no loss	elastic	yes and no

RFC 822

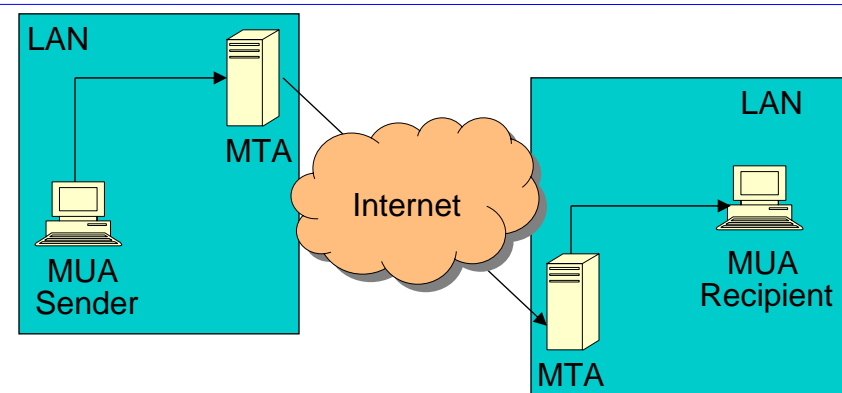
- An e-mail is a message made up of a string of ASCII characters in a format specified by RFC 822 (dating from 1982).
- Two parts, separated by blank line:
 - The **header**: sender, recipient, date, subject, delivery path,...
 - The **body**: containing the actual message content.
- Use of ASCII causes problems for non-ASCII message bodies, e.g. attachments – more later.

An Example RFC 822 Message

```
From: somebody@gmu.edu
To:   santa@wonderland.net
Cc:   angel@heaven.net
Subject: RFC 822 example
Date: Fri, 15 Nov 2002 13:58:49
```

This is a test message illustrating RFC 822. It's not very long and it's not very exciting. But you get the point.

How Are E-mails Transported?

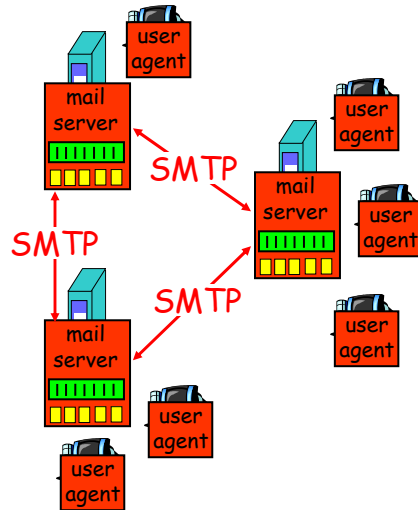


- MUA= Mail User Agent, *aka* Mail Client
- MTA=Mail Transport Agent, *aka* Mail Server

Electronic Mail: mail servers

Mail Servers

- **mailbox** contains incoming messages (yet to be read) for user
- **message queue** of outgoing (to be sent) mail messages
- **smtp protocol** between mail servers to send email messages
 - client: sending mail server
 - “server”: receiving mail server



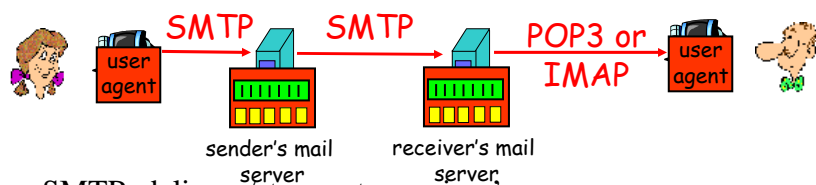
ISE at George Mason University

ISA 666

Duminda Wijesekera based on
ppts of F. Wang+J. Kurose

11

Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
 - POP: Post Office Protocol [RFC 1939]
 - authorization (agent <-->server) and download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - more features (more complex)
 - manipulation of stored msgs on server
 - HTTP: Hotmail , Yahoo! Mail, etc.

ISE at George Mason University

ISA 666

Duminda Wijesekera based on
ppts of F. Wang+J. Kurose

12

Electronic Mail: smtp [RFC 821]

- Uses tcp to reliably transfer email msg from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
 - handshaking (greeting)
 - transfer of messages
 - closure
- command/response interaction
 - **commands**: ASCII text
 - **response**: status code and phrase
- messages must be in 7-bit ASCII

Composition and Delivery – 1

- MUA = Mail client is a program running on Sender's machine, e.g. Microsoft Outlook or Netscape Messenger.
- Sender supplies `To :` and `Subject :` fields and message body.
- MUA translates into RFC 822 message and connects across LAN to MTA = Mail server.
- MUA instructs MTA using a protocol called SMTP (or a proprietary alternative) and sends RFC 822 message.

Composition and Delivery – 2

- Sender's MTA uses DNS (Domain Name Service) to find IP address of recipient's MTA (could be local) based on To: field.
- Sender's MTA opens connection to Recipient's MTA and uses SMTP (Simple Mail Transfer Protocol) to instruct remote MTA and transfer RFC 822 message
 - often across public Internet.
- Intermediate MTAs may be involved.
- Recipient's MTA may deliver to Recipient's MUA or may store message locally for later retrieval across LAN.

Simple Mail Transfer Protocol

- Basic SMTP is defined in RFC 821, widely used for MUA-MTA and MTA-MTA conversations.
- SMTP uses TCP on port 25 for connections, so SMTP traffic carried over LAN and Internet and is (largely) unprotected.
- 'Skilled' user can talk SMTP directly over a telnet connection to remote MTA, supplying From: field of choice.
- So forging e-mail is nearly trivial (though mail headers usually give away source IP address).

Sample smtp interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Try smtp interaction for yourself!

- **telnet servername 25**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client
(reader)

SMTP: final words

- smtp uses persistent connections
- smtp requires that message (header & body) be in 7-bit ascii
- certain character strings are not permitted in message (e.g., CRLF . CRLF). Thus message has to be encoded (usually into either base-64 or quoted printable)
- smtp server uses CRLF . CRLF to determine end of message

ISE at George Mason University

Comparison with http

- http: pull
- email: push
- both have ASCII command/response interaction, status codes
- http: each object is encapsulated in its own response message
- smtp: multiple objects message sent in a multipart message

ISA 666

Duminda Wijesekera based on
.pts of F. Wang+J. Kurose

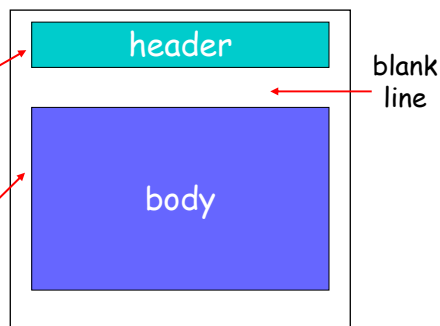
19

Mail message format

smtp: protocol for exchanging email msgs

RFC 822: standard for text message format:

- header lines, e.g.,
 - To:
 - From:
 - Subject:*different from smtp commands!*
- body
 - the “message”, ASCII characters only



ISE at George Mason University

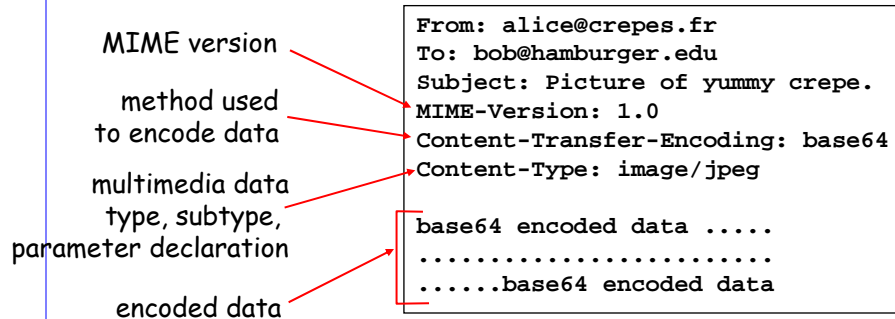
ISA 666

Duminda Wijesekera based on
.pts of F. Wang+J. Kurose

20

Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type



MIME

MIME = Multipurpose Internet Mail Extensions

- Extends the capabilities of RFC 822 to allow e-mail to carry non-textual content, non-ASCII character sets, long messages.
- Uses extra header fields in RFC 822 e-mails to specify form and content of extensions.
- Supports a variety of content types, but e-mail still ASCII-coded for compatibility.
- Specified in RFCs 2045-2049.

MIME types

Content-Type: type/subtype; parameters

Text

- example subtypes: **plain, html**

Image

- example subtypes: **jpeg, gif**

Audio

- example subtypes: **basic** (8-bit mu-law encoded), **32kadpcm** (32 kbps coding)

Video

- example subtypes: **mpeg, quicktime**

Application

- other data that must be processed by reader before “viewable”
- example subtypes: **mword, octet-stream**

MIME headers

MIME specifies 5 new e-mail header fields:

- **MIME-Version** (must be 1.0)
- **Content-Type**
- **Content-Transfer-Encoding**
- **Content-ID** -
optional
- **Content-Description** -
optional

MIME Content-Type

- Seven major content types with 15 sub-types.
- Multipart content type has 4 subtypes.
- Most important is Multipart/mixed, indicating that the body contains multiple parts.
- Each part can be a separate MIME message – hence nesting of MIME messages to any level.
- Parts separated by a boundary string defined in Content-Type field.

Multipart Type

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.....base64 encoded data
--98766789--
```

An Example MIME Message

```
From: santa@wonderland.net
To: somebody@gmu.edu
Subject: That document
Date: Wed, 3 Nov 2004 19:55:47 -0000
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="---next part"
-----next part
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

Hello, here's that document I said I'd send. Regards, Santa
-----next part
Content-Type: application/x-zip-compressed; name="report.zip"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="report.zip"

rfvbnj756tbGHUSISyuhssia9982372SHHS3717277vsgGJ77JS77HFyt6GS8
-----next part--
```

Content-Transfer-Encoding

- RFC 822 e-mails can contain only ASCII characters.
- MIME messages intended to transport arbitrary data.
- The Content-Transfer-Encoding field indicates how data was encoded from raw data to ASCII.
- base64 is a common encoding:
 - 24 data bits (3 bytes) at a time encoded to 4 ASCII characters.

POP3 protocol

authorization phase

- client commands:
 - **user**: declare username
 - **pass**: password
- server responses
 - **+OK**
 - **-ERR**

transaction phase

- client:
- **list**: list message numbers
 - **retr**: retrieve message by number
 - **dele**: delete
 - **quit**

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Where's The E-mail?

- UNIX systems often transfer e-mail from MTA to files in local client file system.
 - Use elm, pine, xmail to read e-mail on client machine.
 - UNIX username and password controls access to client mailbox.
 - Thus security of mail system partly relies on user account security.

Where's The E-mail?

- Can also store e-mail on mail server rather than on client machine.
- Two common protocols for mail client-mail server interaction:
 - POP=Post Office Protocol (RFC 1939, v3)
 - IMAP=Internet Message Access Protocol (RFC 2060, v4rev1).
 - Other proprietary protocols also exist.
- Username and password required before mail can be accessed
 - often sent over network in clear.
- Secure extensions to POP and IMAP also exist:
challenge/response based on user password.

Web-based Access

- Useful for users with web browser but no mail client, e.g. user on the road.
- Username/password combination to control access.
- Now entire client-server interaction over HTTP instead of POP/IMAP.
- Possibly protected using SSL.

E-mail Security Threats

We distinguish two kinds of threats to the security of e-mail:

Threats to the security of e-mail itself

Threats that are enabled through e-mail.

- Almost all firewalls let e-mails pass through!
- An attractive vehicle for attackers to launch attack

Other classifications are possible!

Not an exhaustive list of threats!

Threats to E-mail

- Loss of confidentiality.
 - *E-mails are sent in clear over open networks.*
 - *E-mails stored on potentially insecure clients and mail servers.*
- Loss of integrity.
 - *No integrity protection on e-mails; message body can be altered in transit or on mail server.*

Threats to E-mail

- Lack of data origin authentication.
 - *Is this e-mail really from the person named in the From: field?*
 - *Recall SMTP directly over telnet allows forgery of all e-mail fields!*
 - *E-mail could also be altered in transit.*
 - *Even if the From: field looks fine, who was logged in as John.Smith when the e-mail was composed?*
 - *Sharing of e-mail passwords common.*

Threats to E-mail

- Lack of non-repudiation.
 - *Can I rely and act on the content? (integrity)*
 - *If so, can the sender later deny having sent it? Who is liable if I have acted?*
- Lack of notification of receipt.
 - *Has the intended recipient received my e-mail and acted on it?*
 - *A message locally marked as 'sent' may not have been delivered.*

Threats Enabled by E-mail

- Disclosure of sensitive information.
 - *It's much easier to distribute information by e-mail than it is by paper and snail mail.*
 - *Disclosure may be deliberate (and malicious) or unintentional.*
 - *Disclosure may be internal or external (e-mail crosses LANs as well as the Internet).*
 - *Disclosure may be of inappropriate, sensitive or proprietary information.*
 - *Can lead to loss of reputation and ultimately dismissal of staff.*

Threats Enabled by E-mail

- Exposure of systems to malicious code.
 - *Today, e-mail is the main vector by which computer viruses and worm spread.*
 - *Self-replicating code embedded in e-mail, exploits features/vulnerabilities of e-mail client.*
 - *Visual basic script;*
 - *Javascript in html formatted e-mail;*
 - *.exe attachments of dancing pigs.*
 - *Trojan horse code embedded in e-mail*

Threats Enabled by E-mail

- Exposure of systems to denial of service attacks.
 - *E-mail server attached to network, may be vulnerable to DoS attacks.*
 - *More relevant with increasing dependence on e-mail as the communications tool.*
 - *DoS on mail server may compromise other network services too.*

Threats Enabled by E-mail

- Exposure of *individuals* to denial of service attacks.
 - *Mail bombing and excessive spam.*
 - *Individuals get so swamped by incoming e-mail that they stop reading it.*
 - *Switch to other communications channels (usually around the “you have 1000 unread messages” mark).*

Threats Enabled by E-mail

- Spam
 - *Misconfiguration of relaying capability allows mail server to be exploited for spamming, i.e. bulk distribution of unsolicited e-mail.*
 - *Guilty server can end up on Open Relay Blacklist.*
 - *Result is that **all** e-mail from that server gets blocked by mail servers using blacklist.*
 - *Spam wastes bandwidth and decreases productivity.*
 - *Hotmail and other free e-mail systems are particularly victimised by spammers.*
 - *50% and more of all e-mail is now spam.*

Threats Enabled by E-mail

- Unauthorized access to systems.
 - *Mail servers (OS and application) can have many security vulnerabilities; they are also attached to external networks.*
 - *Perfect target for hacker.*
 - *Mail servers are being used as attack platform on other systems (your own and other peoples').*
 - *Consequent loss of reputation and potential damages claim.*

Threats Enabled by E-mail

- Any more threats?

Secure E-mail Standards and Products

- We focus on S/MIME and PGP.
- Other now defunct standards: PEM (privacy enhanced mail), X.400.
 - Parts of these persist: PEM introduced base64 encoding, X.400 led to X.509 certificate standards.
- Lots of commercial products:
 - Hushmail (www.hushmail.com), XenoMail, identity-based secure e-mail (www.voltagesecurity.com),...

S/MIME

- Originated from RSA Data Security Inc. in 1995.
- Further development by IETF S/MIME working group at:
www.ietf.org/html.charters/smime-charter.html.
- Version 3 specified in RFCs 2630-2634.
- Allows flexible client-client security through encryption and signatures.
- Widely supported, e.g. in Microsoft Outlook, Netscape Messenger, Lotus Notes.

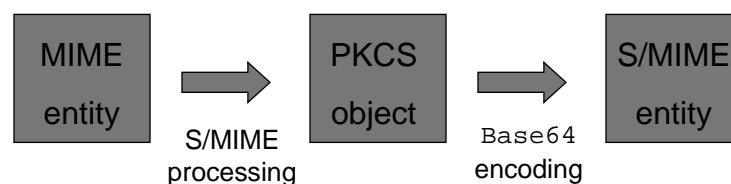
S/MIME Message Formats

- As the name suggests, S/MIME adds security features by extending MIME.
- S/MIME adds 5 new content type/subtype combinations, including:
 - application/pkcs7-mime;
smime-type=enveloped-data
 - application/pkcs7-mime;
smime-type=signed-data
 - multipart/signed

S/MIME Processing

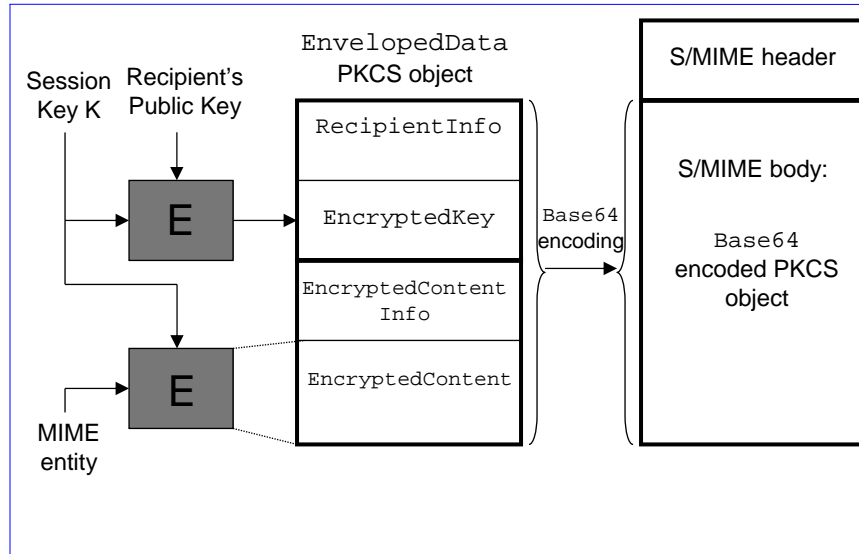
- S/MIME processing can be applied to any MIME entity:
 - One part of a MIME multipart message, perhaps one that is itself of S/MIME Content-Type.
 - End result of S/MIME processing is always another MIME entity, of S/MIME Content-Type.
 - Hence encryption and signature can be applied one after another, and in either order.

S/MIME Processing – Sender



- Initial S/MIME processing produces a PKCS object.
- PKCS=Public Key Cryptography Standard.
- PKCS object includes information needed for processing by recipient as well as the original content.
- But PKCS objects are in binary format, hence need for further base64 encoding to produce final result MIME object of S/MIME content-type.
- Recipient performs steps in reverse.

S/MIME enveloped-data



S/MIME enveloped-data

An example message (from RFC 2633):

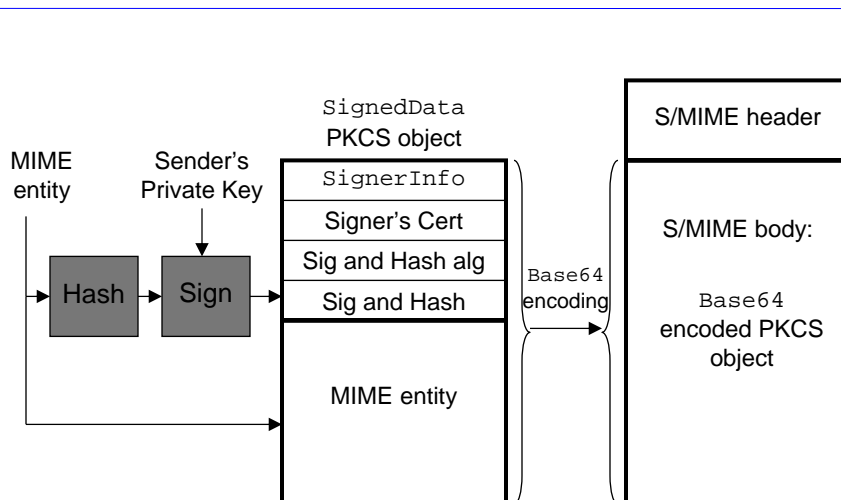
```
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GI
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTTrfvbnjT6jHd
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHh6
```

S/MIME enveloped-data

- S/MIME enveloped-data type gives data confidentiality service through encryption.
- S/MIME header contains original To:, From: and Subject: fields, so protection not complete.
- Symmetric algorithm with session key for efficient bulk encryption and asymmetric encryption using recipient's public key to protect session key.
- Recipient reverses steps: obtain K using private key, then use K to decrypt EncryptedContent.
 - Algorithms needed are specified in RecipientInfo and EncryptedContentInfo blocks.

S/MIME signed-data



S/MIME signed-data

An example message (from RFC 2633):

```
Content-Type: application/pkcs7-mime;  
    smime-type=signed-data; name=smime.p7m  
Content-Transfer-Encoding: base64  
Content-Disposition: attachment; filename=smime.p7m
```

```
567GhIGfHfYT6ghyHhHUujpFyF4f8HHGTrfvhJhjH776tbB97  
7n8HHGT9HG4VQpFyF467GhIGfHfYT6rfvbnj756tbBgHyHhHU  
HUujhJh4VQpFyF467GhIGfHfYGTTrfvbnjT6jH7756tbB9H7n8
```

S/MIME signed-data

- S/MIME signed-data type gives **integrity, authenticity and non-repudiation** services using sender signatures.
- Multiple signers supported – prepare a `SignerInfo` block for each one.
- Recipient checks signature using MIME entity embedded in PKCS object and public (verification) key of sender.
- Recipient without S/MIME capability cannot read the original message (even if he doesn't care about signatures).

S/MIME Clear Signing

- Uses MIME multipart/signed content type.
- First part contains MIME entity to be signed.
- Second part contains S/MIME application/pkcs7-signature entity, created as for signed-data type.
- Recipients who have MIME but not S/MIME capability can still read message contents.
- Recipients who have S/MIME capability use first part as MIME object in S/MIME signature verification.

S/MIME Clear Signing

```
Content-Type: multipart/signed;  
  protocol="application/pkcs7-signature";  
  micalg=sha1; boundary=boundary42  
--boundary42  
Content-Type: text/plain  
  
This is a clear-signed message.  
--boundary42  
Content-Type: application/pkcs7-signature;  
  name=smime.p7s  
Content-Transfer-Encoding: base64  
Content-Disposition: attachment; filename=smime.p7s  
  ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF4674  
  VQpfyF467GhIGfhfYT6jH77n8HHGghyHhHUujhJh756tb6  
--boundary42--
```

S/MIME Algorithms

- Symmetric encryption:
 - DES, 3DES, RC2 with 40 and 64 bit keys.
- Public key encryption:
 - RSA, ElGamal.
- Hashing:
 - SHA-1, MD5.
- Signature:
 - RSA, Digital Signature Standard (DSS).

PGP

- PGP=“Pretty Good Privacy”
- First released in 1991, developed by Phil Zimmerman, provoked export control and patent infringement controversy.
- Freeware: OpenPGP and variants:
 - www.openpgp.org, www.gnupg.org
- Commercial: formerly Network Associates International, now PGP Corporation at www.pgp.com
- OpenPGP specified in RFC 2440 and defined by IETF OpenPGP working group.
 - www.ietf.org/html.charters/openpgp-charter.html
- Available as plug-in for popular e-mail clients, can also be used as stand-alone software.

PGP

- Functionality similar to S/MIME:
 - encryption for confidentiality.
 - signature for non-repudiation/authenticity.
- One level of processing only, so less flexible than S/MIME.
- Sign before encrypt, so signatures on unencrypted data.
 - Sigs can be detached and stored separately.
- PGP-processed data is base64 encoded and carried inside RFC822 message body.

PGP Algorithms

Broad range of algorithms supported:

- Symmetric encryption:
 - DES, 3DES, AES and others.
- Public key encryption of session keys:
 - RSA or ElGamal.
- Hashing:
 - SHA-1, MD-5 and others.
- Signature:
 - RSA, DSS, ECDSA and others.

PGP Key Rings

- PGP supports multiple public/private keys pairs per sender/recipient.
- Keys stored locally in a *PGP Key Ring* – essentially a database of keys.
- Private keys stored in encrypted form; decryption key determined by user-entered passphrase.
- So security once again depends on users remembering passwords!

Key Management for PGP and S/MIME

- PGP and S/MIME use
 - public keys for encrypting session keys / verifying signatures.
 - private keys for decrypting session keys / creating signatures.
- Where do these keys come from and on what basis can they be trusted?

S/MIME Key Management

- S/MIME uses public-key certificates and certificate chains to validate public keys.
- Certificates comply with ISO/ITU-T X.509v3 public key certificate standard.
- Same standard as used to define certificates in SSL/TLS and IPsec.

X.509 Certificate Format

An X.509 certificate is a data structure including the following fields:

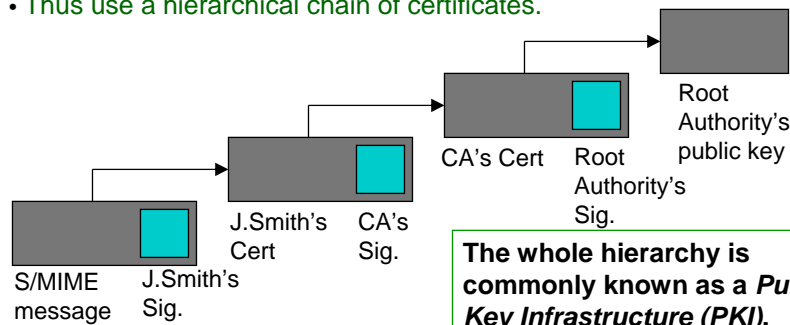
- Version number (1, 2, 3 or 4).
- Serial number of certificate.
- Issuer name.
- Validity period.
- Subject name – a “Distinguished Name”.
- Subject’s public key info: algorithms (eg RSA); parameters (eg size); the public key itself.
- Extension fields.
- The Issuer’s signature on all the above fields.

Use of X.509 Certificates

- Issuer commonly called a Certification Authority (CA).
- Third party can check validity of Issuer's signature in certificate.
- Certificate can therefore vouch that subject is in possession of the private key corresponding to the public key in the certificate.
- But first need authentic copy of Issuer's public key!

X.509 Certificate Chains

- Repeat the checking process on Issuer's certificate,... until *root of trust* is reached
 - a certificate embedded in browser or e-mail client from a root authority whose public key is implicitly trusted.
- Thus use a hierarchical chain of certificates.



X.509 and S/MIME

- Subject's public key can be for signature verification or for encryption – specified in an X.509 extension field.
- X.509 Subject name must be a distinguished name, e.g. “c=US, o=company, ou=sales, cn=John Smith”
- So use another X.509 extension field “Alternative Name” to include e-mail address in certificate.

S/MIME Key Management Issues

Some issues:

- **Interpretation:** End-user is asked: “Do you trust this certificate?” How should a security-unaware user interpret this?
- **Scale:** How to manage large populations of users?
- **Revocation:** How to communicate to all users that a certificate is no longer valid?
- **Liability:** How much liability (if any) does the Issuer accept? Maybe OK if Issuer is your employer.
- **Private key storage:** End-user's desktop most likely, maybe password protected.
- **Certificate issuance procedures** (aka registration): Is this really J. Smith? OK, which J. Smith?

PGP Key Management

- PGP adopts a completely different trust model – the *web of trust*.
- No centralised authority like a root of trust in X.509.
- Individuals sign one another's public keys, these "certificates" are stored along with keys in key rings.
- PGP computes a *trust level* for each public key in key ring
 - Complex formula based on number of signatures on that key, and level of trust in each signature.
- Users interpret trust level for themselves.

PGP Key Management Issues

- Original intention was that all e-mail users would contribute to web of trust.
- Reality is that this web is sparsely populated.
- How should security-unaware users assign and interpret trust levels?
- Later versions of PGP support X.509 certs.
- PGP fine for small groups and out-of-band public key distribution (eg floppy).

Open Problems in E-mail Security

- PGP and S/MIME counter the basic threats to confidentiality, integrity and authenticity of e-mail quite well (assuming good key management).
- They don't protect against other threats
 - Virus, virus hoax
 - Worm
 - Trojan horse
 - Spam
 - DDoS
 - Traffic analysis
 - Unauthorized use

Computer Virus

- **What is a virus?**

A computer virus is a small program written to alter the way a computer operates, without the permission or knowledge of the user. A virus must meet two criteria:

 - **It must execute itself.** It will often place its own code in the path of execution of another program.
 - **It must replicate itself.** For example, it may replace other executable files with a copy of the virus infected file. Viruses can infect desktop computers and network servers alike.

Computer Virus (Cont'd)

- **Five recognized types of viruses** (according to Symantec)
 - **File infector viruses**
 - These viruses normally infect executable code, such as .com and .exe files
 - **Boot sector viruses**
 - Boot sector viruses infect the system area of a disk--that is, the boot record on floppy disks and hard disks.
 - **Master boot record viruses**
 - Master boot record viruses are memory resident viruses that infect disks in the same manner as boot sector viruses.
 - **Multi-partite viruses**
 - Multi-partite (also known as polypartite) viruses infect both boot records and program files. These are particularly difficult to repair.
 - **Macro viruses**
 - These types of viruses infect data files.

Worm

- **What is worm?**
 - Computer program that **replicates itself** from system to system **without the use of a host file**.
 - This is in contrast to viruses, which requires the spreading of an infected host file.
 - Although worms generally exist inside of other files, often Word or Excel documents, there is a difference between how worms and viruses use the host file.
 - **Worm propagates extremely fast**
 - It is possible for a worm to infect 90% of the susceptible hosts in minutes
 - Worm has growth pattern similar to real-world biological virus.
 - The propagation of worm fits well with simple epidemiological mathematical models

Worm (Cont'd)

- Notorious worms
 - CodeRed (2001)
 - MS-Windows NT/2000
 - Nimda (2001)
 - MS-Windows 95/98/ME/NT/2000
 - Slammer (2003)
 - MS SQL Server 2000
 - MS Desktop Engine (MSDE) 2000
 - Blaster (2003)
 - MS-Windows NT/2000/XP/Server 2003
 - Mydoom (2004)

Trojan Horse

- **What is a Trojan horse?**

Trojan Horses are impostors--files that claim to be something desirable but, in fact, are malicious.

 - Trojan horse does not replicate itself.
 - For a Trojan horse to spread, you must, invite these programs onto your computers
 - for example, by opening an email attachment or downloading and running a file from the Internet.
- Trojan horse example
 - PWSteal.Trojan

Detection of Computer Virus and Worms

- Based on predefined attack signatures
 - Attack signatures won't be available before the attack happens and causes damage
 - Signature updates can be overwhelming
- Based on contact tracing and transmission chain identification
 - “borrow” the idea of classical epidemiological method that uses contact investigation to identify the transmission chain
 - Do not need attack signatures, and could potentially detect new (previously-unknown) virus or worm

Decidability Issue on E-mail Virus and Worm Detection

- Two important theoretical results
 - Fred Cohen has demonstrated that **there is no algorithm that could detect all computer viruses**
 - F. Cohen. Computer Viruses: Theory and Experiments. *Computers & Security*, Vol. 6(1), Pages 22–35, 1987
 - David Chess and Steve White have added the bad news and pointed out that **there exist computer viruses that no algorithm can detect**
 - D. M. Chess and S. R. White. An Undetectable Computer Virus. In *Virus Bulletin Conference 2000*.
<http://www.research.ibm.com/antivirus/SciPapers/VB2000DC.pdf>

Current Defences against Virus and Worm

- Install Anti-virus software
 - Overwhelming the customer with frequent signature updates
- Install patches (as fast/much as you can)
 - Overwhelming the customer with
 - Compatibility issue
 - Regression test effort
- Automatic patching
 - How about letting Microsoft automatically updating your Windows OS?
- Active Immunization (just like vaccine in real-world)
 - Anti-worm (new method proposed in WORM'04)
 - What else?