Electronic Mail Security

Slide 1

Characteristics

File transfer, except…

- sender, receiver may not be present at the same time
- diversity (character sets, headers, …)
- not a transparent channel (8 bit data, CRLF)
- often not within a common realm

Slide 2

Distribution Lists

1. send to list site, which distributes:
   - unknown membership (except for bounces. . .)
   - geographical locality
   - size of list
   - avoid need for tree expansion
2. get list from maintainer and send

- “list of lists” – at list server or at receiver (warning!)
- can’t distinguish individuals from lists

Slide 3

Mail Forwarding

MUA: user agent – may disappear temporarily

MTA: message transfer agent – retries, route

- corporate MTA (security gateway)
- protocol translation (X.400, SMTP, Lotus Notes, . . .)

location: MX, manual

routing: DNS

Slide 4

Internet Email

- protocol: SMTP (RFC 821) ➠ ASCII commands, responses
- addresses: RFC 822
- separate: headers (message), envelope (commands: from, to)
- TCP, port 25
- DNS `MX` (mail exchange) records: domain $\rightarrow$ MTA(s)
- binary content, structure ➠ MIME (Multipurpose Internet Mail Extensions)

Slide 5

Security Services

- privacy
- authentication
- integrity
- non-repudiation
- proof of submission
- proof of delivery
- message flow confidentiality (did Alice sent Bob a message?)
- anonymity
- containment (leakage)
- audit

Slide 6

- accounting
- self destruct
- message sequence integrity

Slide 7

Establishing Public Keys

- email: often no prior meeting of principals
- ➠ use (chain of) certificates: x's public key is y, signed “Verisign”
- selection of certificates – not complete trust or felon!
- easily delivered with mail (but: size)

Slide 8

Privacy

- multiple recipients ➠ repeated encryption of long message
- ➠ only encrypt session key for each recipient
- list exploder: get session key, re-encrypt for each recipient
- local list: need key for each recipient

Slide 9

Email Faking

```
host -t mx whitehouse.gov
whitehouse.gov mail is handled (pri=100) by storm.eop.gov

telnet storm.eop.gov 25
Trying 198.137.241.51...
Connected to storm.eop.gov.
Escape character is '^]'.
220 Storm.EOP.GOV -- Server ESMTP (PMDF V5.1-7 #6879)
helo erlang.cs.umass.edu
250 Storm.EOP.GOV OK, [128.59.27.35].
mail from: hgs@somewhere.org
250 2.5.0 Address Ok.
rcpt to: hgs@cs.columbia.edu
250 2.1.5 hgs@cs.columbia.edu OK.
data
354 Enter mail, end with a single ".".
a test
.
```

Slide 10

```
250 2.5.0 Ok.
quit
```

Slide 11

Email Tracing

```
Received: from cs.columbia.edu (cs.columbia.edu [128.59.10.13]) by
  opus.cs.columbia.edu (8.8.5/8.6.6) with ESMTP id PAA07654 for
  <hgs@opus.cs.columbia.edu>; Thu, 10 Apr 1997 15:30:03 -0400 (EDT)
Received: from Storm.EOP.GOV (SYSTEM@storm.eop.gov [198.137.241.51])
  by cs.columbia.edu (8.8.5/8.6.6) with ESMTP id PAA16005
  for <hgs@cs.columbia.edu>; Thu, 10 Apr 1997 15:29:58 -0400 (EDT)
Received: from erlang.cs.umass.edu ([128.59.27.35]) by STORM.EOP.GOV
  (PMDF V5.1-7 #6879) with SMTP id <01IHJN1HAVHE000TEO@STORM.EOP.GOV> for
  hgs@cs.columbia.edu; Thu, 10 Apr 1997 15:29:42 EDT
From: hgs@somewhere.org
Date: Thu, 10 Apr 1997 15:29:42 -0400 (EDT)
Date-warning: Date header was inserted by STORM.EOP.GOV
To: hgs@opus.cs.columbia.edu
Message-ID: <01IHJN3GBO8Q000TEO@STORM.EOP.GOV>
MIME-version: 1.0
Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
Content-Length: 8
```

Slide 12

```
a test
```

Slide 13

Source Authentication

Address spoofing:

- telnet to almost any SMTP server
- some don’t insert appropriate `Received From:` header
- one receiver or list: sign with public key
- but: private key ➠ needs to authenticate/sign with exploder

Slide 14

Message Integrity

- authentication always with message integrity
- integrity without authentication: ransom note ➠ no system exists

Slide 15

Non-Repudiation

- Alice cannot deny having sent message to Bob
- may want plausible deniability

public key: non-repudiable source authentication easy

secret key: repudiable source authentication easy

Slide 16

Plausible Deniability with Public Keys

- Bob knows message m from Alice
- Bob can't prove it to anyone else

1. Alice: picks secret S just for m
2. $\{S\}_{\mathrm{Bob}}$
3. $[\{S\}_{\mathrm{Bob}}]_{\mathrm{Alice}}$
4. use S to compute MIC of m: DES CBC residue
5. Alice $\rightarrow$ Bob: MIC(S), $[\{S\}_{\mathrm{Bob}}]_{\mathrm{Alice}}$, m (separately . . .)

➠ Bob knows that message was from Alice (MIC)
Bob can construct any message he likes

Slide 17

Non-Repudiation with Secret Keys

- Bob prove to judge that Alice sent message
- need notary N with secret S_N, trusted by Bob, judge
- N authenticates Alice
- N: MIC with S_N ➠ *seal* MD("Alice", m or MD, S_N)
- sent m, seal to Bob
- Bob verify message: share key with N or ask N
- judge asks N if seal is valid

Slide 18

Proof of Submission

- *certified* mail (proof of delivery) or *certificate of mailing* (evidence of mailing)
- *registered*: + insurance
- sign message digest, time-of-day

Slide 19

Proof of Delivery

- *certified, return receipt requested*
- requires cooperation of last MTA or receiver
- can’t do receipt if and only if recipient got message (drop or refuse)

Slide 20

Message Flow Confidentiality and Anonymity

- eavesdropper can't tell
- intermediary: anonymous remailer (`anon.penet.fi`↓, `mary.indigo.ie`)
- random delay
- chop into pieces, hide size
- remailer chains, with layers of encryption
- if replies allows ➠ store mappings
- mappings interoperate badly with mailing lists

Slide 21

Containment

- limit distribution of email
- security classes

Slide 22

Mail Transport Issues

- Mail is *almost 8-bit clean* ➠ ESMTP
- if you thought the USPS was mutilating mail…
  - end-of-line: CR, LF, CRLF
  - 8th bit: choke, clear
  - EBCDIC (rare)
  - X.400
  - white space removal
  - long lines
- data transfer
- signatures break
- SMTP: assume text; MIME: arbitrary data

Slide 23

Disguising Data as Text

- canonicalization
- encoding: binary into smaller character set
  - `uuencode`: 3 octets (24 bits) $\rightarrow$ 4 characters (32 bits) from 6-bit set (0x20 [space] to 0x5f [_]), 60 characters per line
  - `base64`: 3 octets (24 bits) $\rightarrow$ 4 characters: A, B, …, Z, a, …, z, 0, …, 9, +, /
  - quoted-printable (if mostly ASCII): `=A0` (hex digits)

Slide 24

Names and Addresses

receiving mailbox: for SMTP (foo@bar.com) “RFC 822”

users: X.500 DN (/C=US/O=CIA/OU=drugs/PN=’Manuel Noriega’/)

- PEM: translate RFC 822 based on messages received to X.500
- PGP: familiar names or `name <email address>`

Slide 25

Old Messages

- is old message still valid (given key revocation, changes, ...)?
- problem: renege on old commitments by strategic key loss
- ➠ notary signs
- prove that message was generated after some date (why?)
- include lottery number

Slide 26

S/MIME

- RFC 2633: *S/MIME Version 3 Message Specification*
- also: PGP (various versions), OpenPGP
- uses CMS (cryptographic message syntax), RFC 2630, derived from PKCS#7
- SHA-1 (and MD5) for digests, DH for key encryption

Slide 27

S/MIME

```
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

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
```

Slide 28

```
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

Slide 29

S/MIME

```
SignedData ::= SEQUENCE {
  version CMSVersion,
  digestAlgorithms DigestAlgorithmIdentifiers,
  encapContentInfo EncapsulatedContentInfo,
  certificates [0] IMPLICIT CertificateSet OPTIONAL,
  crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
  signerInfos SignerInfos }

DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

SignerInfos ::= SET OF SignerInfo
```

Slide 30

S/MIME

```
SignerInfo ::= SEQUENCE {
  version CMSVersion,
  sid SignerIdentifier,
  digestAlgorithm DigestAlgorithmIdentifier,
  signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature SignatureValue,
  unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }
SignerIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
  subjectKeyIdentifier [0] SubjectKeyIdentifier }
SignedAttributes ::= SET SIZE (1..MAX) OF Attribute
UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute
Attribute ::= SEQUENCE {
  attrType OBJECT IDENTIFIER,
  attrValues SET OF AttributeValue }
```

Slide 31

```
AttributeValue ::= ANY
SignatureValue ::= OCTET STRING
```

Slide 32