Electronic Mail Security

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

Audio/Video recordings of this lecture are available at:
http://www.cse.wustl.edu/~jain/cse571-14/
Overview

1. Pretty Good Privacy (PGP)
2. S/MIME
3. DomainKeys Identified Mail (DKIM)

Email Security Enhancements

1. Confidentiality: Protection from disclosure
   - Authentication: Of sender of message
   - Message integrity: Protection from modification
   - Non-repudiation of origin: Protection from denial by sender
Pretty Good Privacy (PGP)

- Widely used de facto secure email
- Developed by Phil Zimmermann in 1991 for anti-nuclear movement private discussions ⇒ Criminal Investigation in 1993
- Selected the best available crypto algorithms and integrated into a single program
- On Unix, PC, Macintosh and other systems
- Originally free, now also have commercial versions available: Symantec Encryption Desktop and Symantec Encryption Server
- Published in 1995 as an OCRable book from MIT Press to allow export
- OpenPGP standard from IETF: Elliptic Curve Cryptography Digital Signature Algorithm (ECDSA) in RFC 6631, 2012

Ref: http://en.wikipedia.org/wiki/Pretty_Good_Privacy
PGP Operation – Authentication

1. Sender creates message
2. Make SHA-1 160-bit hash of message
3. Attached RSA signed hash to message
4. Receiver decrypts & recovers hash code
5. Receiver verifies received message hash
PGP Operation – Confidentiality

1. Sender forms 128-bit random session key
2. Encrypts message with session key
3. Attaches session key encrypted with RSA
4. Receiver decrypts & recovers session key
5. Session key is used to decrypt message
Confidentiality & Authentication

- Can use both services on same message
  - Create signature & attach to message
  - Encrypt both message & signature
  - Attach RSA/ElGamal encrypted session key
PGP Operation – Compression

- By default PGP compresses messages after signing but before encrypting
  - Uncompressed message & signature can be stored for later verification
- Compression is non-deterministic
  - Uses ZIP compression algorithm
PGP Operation – Email Compatibility

- PGP segments messages if too big
- PGP produces binary (encrypted) data & appends a CRC
- Email was designed only for text
  - Need to encode binary into printable ASCII characters
- Uses radix-64 or base-64 algorithm
- Maps 3 bytes to 4 printable chars: 26 upper case alphabets, 26 lowercase alphabets, 10 numbers, +, \n
![Table]

<table>
<thead>
<tr>
<th>Text content</th>
<th>M</th>
<th>a</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>77</td>
<td>97</td>
<td>110</td>
</tr>
<tr>
<td>Bit pattern</td>
<td>01001101</td>
<td>01100001</td>
<td>01101110</td>
</tr>
<tr>
<td>Index</td>
<td>19</td>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td>Base64-encoded</td>
<td>T</td>
<td>W</td>
<td>F</td>
</tr>
</tbody>
</table>

Ref: [http://en.wikipedia.org/wiki/Base64](http://en.wikipedia.org/wiki/Base64)
PGP Operation – Summary

(a) Generic Transmission Diagram (from A)

- \( X \leftarrow \text{file} \)
- \( \text{Signature required?} \) Yes \( \rightarrow \) generate signature \( X \leftarrow \text{signature} || X \)
- Compress \( X \leftarrow Z(X) \)
- \( \text{Confidentiality required?} \) Yes \( \rightarrow \) encrypt key, \( X \leftarrow E(\text{PU}_b, K_S) || E(K_S, X) \)
- convert to radix 64 \( X \leftarrow R64[X] \)

(b) Generic Reception Diagram (to B)

- \( \text{convert from radix 64} \ X \leftarrow R64^{D_1}[X] \)
- \( \text{Confidentiality required?} \) Yes \( \rightarrow \) decrypt key, \( X \leftarrow D(\text{PR}_b, E(\text{PU}_b, K_S)) \)
- \( X \leftarrow D(K_S, E(K_S, X)) \)
- \( \text{Decompress} \ X \leftarrow Z^D[X] \)
- \( \text{Signature required?} \) Yes \( \rightarrow \) strip signature from \( X \)
- verify signature
PGP Session Keys

- Need a session key of varying sizes for each message:
  - 56-bit DES,
  - 168-bit Triple-DES
  - 128-bit CAST (Carlisle Adams and Stafford Tavares)
  - IDEA (International Data Encryption Algorithm)

- Generated with CAST-128 using random inputs taken from previous uses and from keystroke timing of user

PGP Public & Private Keys

- Users are allowed to have multiple public/private keys
  ⇒ Need to identify which key has been used
    - Use a key identifier = Least significant 64-bits of the key
- Signature keys are different from encryption keys
  (Encryption keys may need to be disclosed for legal reasons)
PGP Message Format

Confirms correct key was used

Includes Signature, timestamp of signature, data

Does not include filename, timestamp of file
PGP Key Rings

- Private keys encrypted by a passphrase
- Public keys of all correspondents

### Private Key Ring

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Key ID*</th>
<th>Public Key</th>
<th>Encrypted Private Key</th>
<th>User ID*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_1$</td>
<td>$PU_i \mod 2^{64}$</td>
<td>$PU_i$</td>
<td>$E(H(P_j), PR_j)$</td>
<td>User $i$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Public Key Ring

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Key ID*</th>
<th>Public Key</th>
<th>Owner Trust</th>
<th>User ID*</th>
<th>Key Legitimacy</th>
<th>Signature(s)</th>
<th>Signature Trust(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_1$</td>
<td>$PU_i \mod 2^{64}$</td>
<td>$PU_i$</td>
<td>trust_flag$_i$</td>
<td>User $i$</td>
<td>trust_flag$_i$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* = field used to index table
PGP Message Generation

1. **Private Key Ring Selection**
   - ID_A selects a private key.

2. **Digest Creation**
   - The private key is used to encrypt a digest of the message with the private key (PR_A).

3. **Message Digest**
   - The message is digested (H).

4. **Session Key Generation**
   - The session key (K_s) is generated.

5. **Encryption**
   - The session key and the signature are encrypted with the public key (PU_b).

6. **Output**
   - The encrypted message and signature are output.
PGP Message Reception
Web of Trust

- There is no need to buy certificates from companies
- A user can sign other user’s certificates
- If you trust someone, you can trust users that they sign for.
- You can assign a level of trust to each user and hence to the certificate they sign for
- For example,
  - A certificate that is signed by a fully trusted user is fully trusted
  - A certificate signed by two half trusted users is fully trusted
  - A certificate signed by one half trusted user is half trusted
  - Some certificates are untrusted.

PGP Trust Model Example

DEFL are trusted
AB re half trusted
S is untrusted

? = unknown signatory
X → Y = X is signed by Y

= key's owner is trusted by you to sign keys
= key's owner is partly trusted by you to sign keys
= key is deemed legitimate by you
Certificate Revocation

- Owners can revoke public key by issuing a “revocation” certificate signed with the revoked private key
- New Web-of-trust certificates have expiry dates
S/MIME

- Secure/Multipurpose Internet Mail Extensions
- Original Internet RFC822 email was text only
- MIME for varying content types and multi-part messages
  - With encoding of binary data to textual form
- S/MIME added security enhancements
  - Enveloped data: Encrypted content and associated keys
  - Signed data: Encoded message + signed digest
  - Clear-signed data: Clear text message + encoded signed digest
  - Signed & enveloped data: Nesting of signed & encrypted entities
- Have S/MIME support in many mail agents
  - E.g., MS Outlook, Mozilla, Mac Mail etc
MIME Functions

- Types: Text/Plain, Text/Enriched, Multipart/Mixed, Image/jpeg, Image/gif, Video/mpeg, audio/basic, ...
- Encodings: 7bit, 8bit, binary, quoted-printable, base64

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="frontier"

This is a message with multiple parts in MIME format.
--frontier
Content-Type: text/plain

This is the body of the message.
--frontier
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

PGh0bWw+CiAgPGh1YWQ+CiAgPC9o2WFkPqogIDxib2R5PqogICAqPHA+VGhpcyBpcyB0aGUg
Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9ib2R5Pgo8L2h0bWw+Cg==
--frontier--
```

- Quoted-Printable: non-alphanumerics by =2 hex-digits, e.g., “=09” for tab, “=20” for space, “=3D” for =

S/MIME Cryptographic Algorithms

- Digital signatures: DSS & RSA
- Hash functions: SHA-1 & MD5
- Session key encryption: ElGamal & RSA
- Message encryption: AES, Triple-DES, RC2/40 and others
- MAC: HMAC with SHA-1
- Have process to decide which algorithms to use
**S/MIME Messages**

- S/MIME secures a MIME entity with a signature, encryption, or both
- Forming a MIME wrapped PKCS object (Public Key Cryptography Standard originally by RSA Inc Now by IETF)

<table>
<thead>
<tr>
<th>Type</th>
<th>Subtype</th>
<th>Smime parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multipart</td>
<td>Signed</td>
<td></td>
<td>clear msg w signature</td>
</tr>
<tr>
<td>Application</td>
<td>Pkcs7-mime</td>
<td>signedData</td>
<td>Signed entity</td>
</tr>
<tr>
<td>Application</td>
<td>Pkcs7-mime</td>
<td>envelopedData</td>
<td>Encrypted entity</td>
</tr>
<tr>
<td>Application</td>
<td>Pkcs7-mime</td>
<td>Degenerate signedData</td>
<td>Certificate only</td>
</tr>
<tr>
<td>Application</td>
<td>Pkcs7-mime</td>
<td>CompressedData</td>
<td>Compressed entity</td>
</tr>
<tr>
<td>Application</td>
<td>Pkcs7-signature</td>
<td>signedData</td>
<td>Signature</td>
</tr>
</tbody>
</table>

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

S/MIME Certificate Processing

- S/MIME uses X.509 v3 certificates
- Managed using a hybrid of a strict X.509 CA hierarchy and enterprise’s CAs
- Each client has a list of trusted CA’s certificates and his own public/private key pairs & certificates
- Several types of certificates with different levels of checks:
  - Class 1: Email and web browsing
  - Class 2: Inter-company email
  - Class 3: Banking, …
**S/MIME Enhanced Security Services**

- RFC2634 (1999) describes enhanced security services:
  - Signed receipts: Request a signed receipt
  - Security labels: Priority, which users (role) can access
  - Secure mailing lists: Request a list processor to encrypt
Domain Keys Identified Mail

- Emails signed by the enterprise, e.g. WUSTL rather than the sender
- Company’s mail system signs the message
- So spammers cannot fake that companies email addresses

Ref: http://en.wikipedia.org/wiki/DKIM
DKIM Functional Flow

RFC 5322 Message

Originating or Relaying ADMD: Sign Message with SDID

Private key store

Internet

Relaying or Delivering ADMD: Message signed?

yes

Verify signature

pass

Assessments

Reputation/accreditation information

Check signing practices

Message filtering engine

Local info on sender practices

no

Remote sender practices

(paired)

Public key store
1. Email can be signed, encrypted or both
2. PGP is a commonly used system that provides integrity, authentication, privacy, compression, segmentation, and MIME compatibility
3. PGP allows Web of trust in addition to CA certificates
4. S/MIME extends MIME for secure email and provides authentication and privacy
5. DKIM allows originating companies to sign all emails from their users
Homework 19

A. [19.4] The first 16 bits of the message digest in a PGP signature are transmitted in the clear. To what extent does this compromise the security of the hash algorithm?

B. [19.9] Encode the text “plaintext” using Radix-64 and quoted-printable
Acronyms

- AES: Advanced Encryption Standard
- ASCII: American Standard Code for Information Exchange
- CA: Certificate Authority
- CAST: Carlisle Adams and Stafford Tavares
- CRC: Cyclic Redundancy Check
- DCIM: Domain Key Identified Mail
- DES: Digital Encryption Standard
- DSS: Digital Signature Scheme
- ECC: Elliptic Curve Cryptography
- ECDSA: Elliptic Curve Cryptography Digital Signature Algorithm
- HMAC: Hybrid Message Authentication Code
- IDEA: International Data Encryption Algorithm
- IETF: Internet Engineering Task Force
- MAC: Message Authentication Code
- MD5: Message Digest 5
- MIME: Multipurpose Internet Mail Extension
Acronyms (Cont)

- MIT  Massachusetts Institute of Technology
- MS   Microsoft
- OCR  Optical Character Recognition
- PC   Personal Computer
- PGP  Pretty Good Privacy
- PKCS Public Key Cryptography Standard
- RC2  Ron's Code 2
- RFC  Request for Comments
- RSA  Rivest, Shamir, Adleman
- S/MIME Secure Multipurpose Internet Mail Extension
- SHA  Secure Hash Algorithm
- Triple-DES Triple Digital Encryption Standard
- WUSTL Washington University in Saint Louis