CS 393/682 – Network Security

Nasir Memon – Polytechnic University
Module 5 – Secure Email
Course Logistics

- HW 3 due?
- Midterm week after “week after spring break - April 2, Wednesday 6 to 8:00.
- Reading list and sample midterms will be posted.
- There will be only one question on crypto.
Where do we provide security?

✓ Firewall
  ■ Application layer
    ■ S-MIME, PGP, Kerberos etc.
  ■ Transport layer
    ■ SSL, TSL.
  ■ Network layer
    ■ IPSEC
Application level security

<table>
<thead>
<tr>
<th>Kerberos</th>
<th>S/MIME</th>
<th>PGP</th>
<th>SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>SMTP</td>
<td>TCP</td>
<td>HTTP</td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- We look at examples of application level security mechanisms - secure email. PGP and S/MIME.
Secure Email – Issues

- May seem straightforward – encrypt and/or sign. But problem is richer than that!
- Types of security services that may be required include:
  - Privacy – May seem simple but problems include
    - How to establish keys?
    - What if single email multiple recipients?
      - Include session key encrypted with public key of each recipient.
  - What is mailing list?
    - Same as above but need to trust list maintainer.
  - What if Alice’s employer wants to be able to read her email?
    - Key Escrow
Secure Email Issues – Authentication and Integrity

- Source authentication could be provided by
  - Public key – message digest and digital signature. Same if multiple recipients
  - Private key – Use (private) key based cryptographic hash – MAC.
    - If multiple recipients then first do message digest and then MAC with different private keys for different recipients

- Message Integrity
  - Authentication mechanisms serve for integrity as well. But what if you want to send a ransom note?!!!
Secure Email Issues – Non-Repudiation and Repudiation

- **Non-Repudiation** –
  - Alice to Bank – “Please transfer a million dollars from my account to Bob’s.”

- **Repudiation** –
  - President to Henchmen – “Go ahead and sell arms to Iran to fund the Contras.”

- Non-repudiation based on public key cryptography – easy. Digital Signature.
  - What if after sending the message, Alice claims her private key was compromised?!
    - Undeniable signatures.
Protocol for Non-Repudiable Email – Private Key Version

- Needs Notary N trusted by Bob and Judge.
- Alice sends message M to N.
- N does some computation based on M and Alice’s name using $S_N$ (a random secret generated for this exchange) and produces a “seal” $S_N(M, Alice)$.
- Alice sends M and “seal” to Bob.
- If later Bob wants to prove to Judge that Alice sent him M, then he sends to Judge M and “seal”.
- Judge sends “seal” to notary who verifies the seal.
- Bob could not have produced the seal as he does not know $S_N$. 
Protocol for Repudiable Email – Public Key Version

- Alice picks secret key $K$ which will be used to encrypt message.
- Alice encrypts $S$ with Bob’s public key – $E_{BOB}[K]$.
- She “signs” $E_{BOB}[K]$ with private key getting $S_{ALICE}[E_{BOB}[K]]$.
- She uses $K$ to compute key based cryptographic hash $H_K[M]$, of message $M$.
- She sends $S_{ALICE}[E_{BOB}[K]]$, $H_K[M]$, and $M$ to Bob. ($M$ can be optionally encrypted).
- Bob can verify (decrypt) $M$ and be assured that message is from Alice.
- But once he has done that he can modify $M$ to any $M'$ and produce $H_K[M']$!
- So all he can prove to a 3rd party is that Alice sent some message but not what message.
Secure Email – Other Issues.

- Think about how you would provide the following:
  - Proof of delivery
    - ???
  - Proof of receipt
    - What if recipient refuses last packet?!! Send encrypted message and then get acknowledgement of receipt of this and only then decryption key!!
  - Message flow confidentiality
  - Anonymity
  - (Last two will be addressed a couple of weeks later under the title of anonymity and privacy.)
Secure Email Systems

- PGP – Pretty Good Privacy –
  - Provides confidentiality and authentication service that can be used in email and file storage application.
  - Public domain and open source. Uses well known and established crypto primitives like RSA, 3DES, SHA-1, Diffie-Hellman etc.
  - Phil Zimmerman – almost sent to jail for writing it.

- S/MIME – Secure/Multipurpose Internat Mail Extension.
  - Security enhancement of MIME based on technology from RSA. In RFS 822.
PGP – Services

- Authentication
  - SHA-1 for hash. RSA signature

- Confidentiality
  - CAST, IDEA or 3DES. Diffie Hellman or RSA for session key exchange.

- Compression
  - Zip

- E-mail compatibility
  - Radix 64.

- Segmentation
Notation: \textbf{KR}_a - Private key of user A. \textbf{KU}_a - Public key of user A. 
\textbf{Z}^{-1} - Decompression. \mid - Concatenation 
\textbf{EP} - Public Key encryption. \textbf{DP} - Private key decryption.
PGP - Confidentiality

**Notation:**
- \( KR_b \) - Private key of user B.
- \( KU_b \) - Public key of user B.
- \( H \) - Hash function.
- \( M \) - Message.
- \( Z \) - Compression.
- \( Z^{-1} \) - Decompression.
- \( | \) - Concatenation
- \( EC \) - Private Key encryption.
- \( DC \) - Private key decryption.
- \( EP \) - Public Key encryption.
- \( DP \) - Private key decryption.
- \( Ks \) - Session key.
**PGP - Authentication and Confidentiality**

**Notation:**
- \( K_{R_b} \) - Private key of user B.
- \( K_{U_b} \) - Public key of user B.
- \( H \) - Hash function.
- \( M \) - Message.
- \( Z \) - Compression.
- \( Z^{-1} \) - Decompression.
- \( H \) - Hash function.
- \( E_{K_{R_a}} \) - Private Key encryption.
- \( D_{P} \) - Private key decryption.
- \( E_{K_{U_b}} \) - Public Key encryption.
- \( D_{C} \) - Private key decryption.
- \( E_{K_{R_a}[H(M)]} \) - Private key decryption.
- \( K_s \) - Session key.

(c) Confidentiality and authentication
PGP - Compression

- Zip used.
- Signature generated before compression because:
  - Would need to store compressed version for later verification.
  - Compression algorithm not deterministic. Various parameter settings produce different forms.
- Encryption done after compression for:
  - Cryptanalysis is made more difficult as compressed message has less redundancy.
  - Encryption output is random which cannot be compressed.
PGP – Email compatibility

- Many email systems only allow ASCII text.
- Encrypted bytes may arbitrary.
- PGP performs Radix 64 conversion – three bytes mapped to 4 printable ASCII bytes.
- Message gets expanded by 33%.
- Compression more than compensates for that.
PGP - Segmentation

- Messages larger than a given amount are automatically broken up.
- Headers enable reassembly at other end.
- Segmentation done only at end so session key etc. only appear once.
PGP - Flow

(a) Generic Transmission Diagram (from A)

X ← file

Signature required?

Yes

generate signature
X ← signature || X

Confidentiality required?

Yes

encrypt key, X
X ← E_{K_{ub}}[K_{s}] || E_{Ks}[X]

Confidentiality required?

No

Compress
X ← Z(X)

convert to radix 64
X ← R64[X]

(b) Generic Reception Diagram (to B)

convert from radix 64
X ← R64^{-1}[X]

Confidentiality required?

Yes

decrypt key, X
K ← D_{K_{Rb}}[K_{s}]; X ← D_{Ks}[X]

Confidentiality required?

No

Decompress
X ← Z^{-1}(X)

Signature required?

Yes

strip signature from X
verify signature

No

No
PGP – Session Key Generation

- Uses complex techniques for generating random numbers and pseudo-random numbers for key generation (Read appendix of chapter 5).
  - The importance of this step cannot be overemphasized. Scheme can be broken if “random” numbers can be guessed.
- True random numbers generated from monitoring user keystrokes. Time and keystroke information used to generate key which is used to encrypt current true random buffer.
- Pseudorandom numbers generated from using encryption technique in CFB mode.
PGP – Random numbers use

- (True) Random numbers used to
  - Generate RSA key pairs
  - Initial seed for pseudorandom number generation
  - Additional input for pseudorandom number generation

- Pseudorandom numbers used to
  - Generate session keys
  - Generate IV for CFB mode encryption.
PGP - Key Identifiers

- Allows user to have multiple public key/private key pairs.
  - Why?
- Specific key used for a message identified by assigning key identifier.
- Key identifier is simply 64 lsb’s of key.
  - In principle you could have two different keys with same identifier but very very unlikely.
PGP – Message Format
PGP Key Rings

### 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>$T_i$</td>
<td>$KU_i \mod 2^{64}$</td>
<td>$KU_i$</td>
<td>$E_{H(P_i)}[KR_i]$</td>
<td>$User_i$</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>$T_i$</td>
<td>$KU_i \mod 2^{64}$</td>
<td>$KU_i$</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>
PGP Message Generation

Diagram:
- Passphrase
- Private key ring
- Encrypted private key
- Public key ring
- Key ID
- Message
- Message digest
- Encrypted signature

Steps:
1. Select ID_A
2. Select ID_B
3. Select public key
4. Select Key ID
5. Select session key
6. Select message
7. Select encrypted message
8. Output

PGP Message Reception

Diagram: Flow of message reception with key exchange and verification.
PGP – Assigning Trust to Keys

? = 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
**MIME**

- The Simple Mail Transfer Protocol (SMTP) (RFC 822) used to exchange electronic mail across the Internet.
- SMTP conveys the message content as plain ASCII text.
- The Multipurpose Internet Mail Extensions (MIME) (RFC 1521) extends format of SMTP to allow multipart text and binary data message bodies to be exchanged without loss of information.
MIME (Contd)

- MIME extends SMTP by adding 5 new header fields:
  1. MIME version
  2. Content-type
  3. Content transfer encoding
  4. Content-ID
  5. Content-description

- 4 and 5 are optional and may be ignored by recipient.
S/MIME Functionality

- Functionality added by S/MIME includes
  - Enveloped data
  - Signed data
  - Clear-signed data
  - Signed and enveloped data
  - Signed receipts
  - Security label
  - Secure mailing lists
S/MIME v3

- The S/MIME v3 standard consists of five parts:
  - Cryptographic Message Syntax (RFC 2630)
  - S/MIME Version 3 Message Specification (RFC 2633)
  - S/MIME Version 3 Certificate Handling (RFC 2632)
  - Diffie-Hellman Key Agreement Method (RFC 2631)
  - Certificate Distribution Specification (draft-ietf-smime-certdist)

- Enhanced Security Services for S/MIME (RFC 2634) - set of extensions to allow signed receipts, security labels, and secure mailing lists.
Further Reading

- For PGP and S/MIME read chapter 5 of text.
- For PGP - The International PGP Home Page at http://www.pgpi.org/
- The PGP Attack FAQ at http://www.stack.nl/~galactus/remailers/attack-faq.html