Key Management
Chapter 10

Key Management
• Distribution of cryptographic keys
• Binding identity to a key
• Generation, maintenance, revoking

Session Keys
• More the same symmetric key used, more likely to be broken
• Generate and use a symmetric key for use during a specific communication for data only
• PK schemes good for encrypting random data; not good otherwise - especially if possible plaintext is from a small set
  – Forward search attack
  – Encrypt all possibilities with public key

Communicating Securely
How can Alice send Bob a symmetric key to enable communication?
If sent in the clear, Eve can see
Use a *Trusted* 3rd Party
Cathy shares a unique key with all users

Cathy generates session keys when users want to communicate between themselves

I want to communicate with Bob. Please give me a key.

Here is a session key

Needham-Schroeder

• Added authentication to thwart replay
• Random numbers used
1. Alice ⊲ Cathy: {Alice || Bob|| rand₁}
2. Cathy ⊲ Alice: 
   {Alice || Bob|| rand₁|| ksession || Alice|| ksession || Bob|| ksession || kAlice}
3. Alice ⊲ Bob: {Alice || ksession || Bob}
4. Bob ⊲ Alice: {rand₂} ksession
5. Alice ⊲ Bob: {rand₂+1} ksession

Basic Exchange

• Vulnerable to replay by Eve
• Eve copies Bob’s encrypted key and messages encrypted under the sessions key
  – Eve can’t “read” the messages, but Bob will still read (and possibly redo something) it
  – “deposit $500”
N-S Analysis

• Can’t repeat random numbers
  – AKA nonces
• What if session key is known by Eve?

Denning & Sacco Analysis

• If clocks are too fast or too slow, vulnerable
Public Key Key Exchange

- Here interchange keys known
  - $e_A, e_B$ Alice and Bob’s public keys known to all
  - $d_A, d_B$ Alice and Bob’s private keys known only to owner
- Simple protocol
  - $k_s$ is desired session key

\[ \text{Alice} \rightarrow \{ k_s \} e_B \rightarrow \text{Bob} \]

Problem and Solution

- Vulnerable to forgery or replay
  - Because $e_B$ known to anyone, Bob has no assurance that Alice sent message
- Simple fix uses Alice’s private key
  - $k_s$ is desired session key

\[ \text{Alice} \rightarrow \{ \{ k_s \} d_A \} e_B \rightarrow \text{Bob} \]

Notes

- Can include message enciphered with $k_s$
- Assumes Bob has Alice’s public key, and vice versa
  - If not, each must get it from public server
  - If keys not bound to identity of owner, attacker Eve can launch a man-in-the-middle attack (next slide; Cathy is public server providing public keys)
    - Solution to this (binding identity to keys) discussed later as public key infrastructure (PKI)

\[ \text{Man-in-the-Middle Attack} \]

\[ \text{Alice} \xrightarrow{\text{send Bob’s public key}} \text{Eve} \xrightarrow{\text{Eve intercepts request}} \text{Cathy} \]

\[ \text{Alice} \xleftarrow{\text{Alice}} \xrightarrow{e_E} \text{Eve} \xleftarrow{\text{Eve}} \xrightarrow{e_B} \text{Cathy} \]

\[ \text{Alice} \rightarrow \{ k_s \} e_E \rightarrow \text{Eve} \xrightarrow{\text{Eve intercepts message}} \text{Bob} \]

\[ \text{Eve} \rightarrow \{ k_s \} e_E \rightarrow \text{Bob} \]
Key Generation

- **Goal:** generate difficult to guess keys
- **Problem statement:** given a set of $K$ potential keys, choose one randomly
  - Equivalent to selecting a random number between 0 and $K-1$ inclusive
- **Why is this hard:** generating random numbers
  - Actually, numbers are usually pseudorandom, that is, generated by an algorithm

What is “Random”?

- **Sequence of cryptographically random numbers:** a sequence of numbers $n_1, n_2, \ldots$ such that for any integer $k > 0$, an observer cannot predict $n_k$ even if all of $n_1, \ldots, n_{k-1}$ are known
  - Best: physical source of randomness
    - Random pulses
    - Electromagnetic phenomena
    - Characteristics of computing environment such as disk latency
    - Ambient background noise

What is “Pseudorandom”?

- **Sequence of cryptographically pseudorandom numbers:** sequence of numbers intended to simulate a sequence of cryptographically random numbers but generated by an algorithm
  - Very difficult to do well
    - Linear congruential generators [$a_t = (an_{t-1} + b) \mod n$] broken
    - Polynomial congruential generators [$a_t = (a_1n_{t-1} + \ldots + a_0n_{t-1} a_0) \mod n$] broken too
  - Here, “broken” means next number in sequence can be determined

Best Pseudorandom Numbers

- **Strong mixing function:** function of 2 or more inputs with each bit of output depending on some nonlinear function of all input bits
  - Examples: DES, MD5, SHA-1
  - Use on UNIX-based systems:
    
    (date; ps gaux) | md5

    where “ps gaux” lists all information about all processes on system
Cryptographic Key Infrastructure

- Goal: bind identity to key
- Classical: not possible as all keys are shared
  - Use protocols to agree on a shared key (see earlier)
- Public key: bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name

Certificates

- Create token (message) containing
  - Identity of principal (here, Alice)
  - Corresponding public key
  - Timestamp (when issued)
  - Other information (perhaps identity of signer)
  - signed by trusted authority (here, Cathy)

\[ C_A = \{ e_A \| \text{Alice} \| T \} d_C \]

Use

- Bob gets Alice’s certificate
  - If he knows Cathy’s public key, he can decipher the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice’s public key
- Problem: Bob needs Cathy’s public key to validate certificate
  - Problem pushed “up” a level
  - Two approaches: Merkle’s tree, signature chains

Key Escrow

- **Key escrow system** allows authorized third party to recover key
  - Useful when keys belong to roles, such as system operator, rather than individuals
  - Business: recovery of backup keys
  - Law enforcement: recovery of keys that authorized parties require access to
- Goal: provide this without weakening cryptosystem
- Very controversial
Escrowed Encryption Standard

- EES; Capstone; Skipjack
- SymmetricSkipjack cipher, 80 bit
- 1993
- LEAF
  - Law Enforcement Access Field
  - Allowed agencies access to keys to also decrypt
- 2-party systems/protocols are hard enough, 3rd party even more difficult to secure

EES

- LEAF vulnerability
  - Can generate a LEAF and insert into traffic
- Most banks, companies stuck with DES rather than a ‘broken’ scheme
- Key escrow
  - LEAF keys held by a Trusted Third Party

PGP ADK

- 1997, Additional Encryption Key
- In even employee no longer had their private key, let company unlock messages
  - Corporate customers wanted this feature
- Alternative to key escrow
- Original implementation allowed users to attach an ADK to their key
- Bug in August 2000 (now fixed)
  - Was not digitally signed
  - Someone else could potentially attach a different ADK to your key

Copyright 2002, 2003 Julie M. Starr. All Rights Reserved.