Key Management (1)

Week 4, Tuesday

What’s To Address

Key distribution (today)
• Key generation
• Key maintenance
• Key revocation

Notations

• $X \square Y: \{Z\} k$
  $X$ sends $Y$ a message $Z$ enciphered with key $k$

Classical cryptosystem[] symmetric cryptosystem
• $k$: a secret key
  – $k_{AB}$: Alice’s key
  – $k_{Bob}$: A key shared by Alice and Bob
Public key cryptosystem[] asymmetric cryptosystem
• $e$: a public key
• $d$: a private key

Session Key

• A key used for a particular communication session
  – Set up when the session starts
  – Discarded when the session ends

Interchange Key

• A cryptographic key that is bound to a specific principal
  – Think it as lifetime binding

• Different from a session key
  – Which is only to be used during a session

Key Exchange

• Goal: enable Alice and Bob to communicate secretly using a shared key

• Requirements
  – The shared key cannot be transmitted in the clear
  – Alice and Bob may decide to trust a third party
  – Cryptosystems and protocols are publicly known
Classical Cryptographic Key Exchange

- Alice cannot send the key to Bob in the clear
  - Eve may eavesdrop
- Ask help from Cathy
  - Alice and Cathy share a secret key: $k_{AC}$
  - Bob and Cathy also share a (different) secret key: $k_{BC}$

(1) Alice $\rightarrow$ Cathy : \{request for session key to Bob\} $k_{AC}$
(2) Cathy $\rightarrow$ Alice : $\{k_s\}k_{AC} \parallel \{k_s\}k_{BC}$
(3) Alice $\rightarrow$ Bob : $\{k_s\}k_{BC}$
(4) Both Alice and Bob have the session key $k_s$

Can This Scheme Be Attacked?

- Let’s say in step (4) Alice told Bob on Tuesday “Today’s security class is cancelled”
- Eve replays (3) and (4) on Thursday
- Problem: Bob does not know whom he is talking to

So, This is a Replay Attack

- How to prevent it?

Needham-Schroeder Protocol

(1) Alice $\rightarrow$ Cathy : \{Alice$\|Bob\|\text{rand}_1\}$
(2) Cathy $\rightarrow$ Alice :
  \{Alice$\|Bob\|\text{rand}_1\}\|k_s\}k_{AC} \parallel \{Alice\}k_{BC} \parallel k_{AC}$
(3) Alice $\rightarrow$ Bob : $\{Alice\}k_s\parallel k_{BC}$
(4) Bob $\rightarrow$ Alice : $\{\text{rand}_2\}k_s$
(5) Alice $\rightarrow$ Bob : $\{\text{rand}_2-1\}k_s$

Analysis

- How does Bob ensure that he is talking with Alice?
  - Step 5 can verify
  - Well, only Alice knows the session key (in addition to Bob)
- How does Alice ensure that she is talking with Bob?
  - Step 2 is for sure from Cathy in response to step 1
  - Step 3 ensures only Bob knows the session key (in addition to Alice)
- Can Eve do a replay attack?
But What If Session Keys Are Predictable (or Stolen)?

- In practice, session keys are generated pseudorandomly
  - Thus may be possible to predict next session key
- Denning and Sacco: the protocol breaks if Eve obtains a session key

(1) EveÆBob : {Alice || $k_s$, {Alice || $T$ || $k_s$}}$_B,C$
(2) BobÆAlice : {rand$_2$}$_B,C$ [intercepted by Eve]
(3) EveÆBob : {rand$_2$-1}$_s$

Bob thinks he is talking to Alice, but actually Eve!

Revised Needham-Schroeder Protocol

- Use timestamp for replay prevention

(1) AliceÆCathy : {Alice||Bob||rand$_1$}
(2) CathyÆAlice :
  {Alice||Bob||rand$_1$, {Alice || T || $k_s$}$_B,C$, $k_{A,C}$}
(3) AliceÆBob :
  {Alice || T || $k_s$}$_B,C$
  [Bob rejects the message if too old]
(4) BobÆAlice :
  {rand$_1$}$_s$
(5) AliceÆBob :
  {rand$_2$-1}$_s$

Requires synchronized clocks!

Otway-Rees Protocol

(1) AliceÆBob : num||Alice||Bob, {rand$_1$ || num || Alice||Bob}$_A,C$
(2) BobÆCathy :
  num||Alice||Bob, {rand$_1$ || num || Alice||Bob}$_A,C$
  {rand$_2$ || num || Alice||Bob}$_B,C$
(3) CathyÆBob :
  num, {rand$_1$ || $k_s$}$_A,C$
  {rand$_2$ || $k_s$}$_B,C$
(4) BobÆAlice :
  num, {rand$_1$ || $k_s$}$_A,C$

Analysis

- Replay attack is difficult
  - For every new session, Alice picks a new num
  - Alice checks num consistency at step 4

- No timestamps are used, no clock synchronization is needed

Kerberos

- Uses revised Needham-Schroeder protocol
  - A client: Alice
  - A service provider: Gutterberg
  - A Kerberos authentication server: Cerberus
  - A ticket server: Barnum
Steps

1. Alice \[\rightarrow\] Cerberus: Alice || Barnum

2. Cerberus \[\rightarrow\] Alice: \{k_{A,B}\} \ k_A,C \ || \ T_{A,B}

[Ticket] \ T_{A,B} = \text{Bil}(\text{Alice’s address} || \text{valid time} || k_{A,B}) \ k_B

3. Alice \[\rightarrow\] Barnum: Guttenberg || A || T_{A,B}

(Authenticator) \ A_{A,B} = \text{Bil}(\text{generation time} || k_t) \ k_{A,B}

Public Key Cryptography Key Exchange

• Very easy
  - Alice \[\rightarrow\] Bob: \{k_s\} e_{Bob}
  
  – Alice and Bob then can securely communicate using a symetric cryptosystem

• Flaw: Bob does not know who sent the message
  – Could be Eve forging such a message

Fix the Flaw: Sign \ k_s

Alice \[\rightarrow\] Bob: Alice, \{k_s\} d_{Alice} \ e_{Bob}

• Question: What if Alice does not have Bob’s public key?

Announcement: Seventh Annual UO Programming Competition

- Date: Saturday morning, May 3.
- Registration deadline: April 28.
- Open to all UO students.
- There will be grad and undergrad divisions; teams in the undergraduate division will consist of three undergraduate students and teams in the graduate division will consist of two graduate students.
- Send email with the names and email addresses of your team members to hrs@cs.uoregon.edu.
- Links to previous contest problem sets can be found at http://www.cs.uoregon.edu/resources/progteam/dept_contest/