On Authentication

“I was so frightened, I couldn't remember my own name!”

White Queen in *Through the Looking Glass*, by Lewis Carroll
Password authentication

• User authenticates herself by entering a password -- who is checked against the server’s database
  – Pros:
    • Supported by almost every system
    • Users familiarized with the process
  – Cons:
    • Good password management requires more of the user than other methods
Attacks on password systems

• Dictionary attacks
  • Use a likely password dictionary against every user in the system; or
  • Use public information about a particular user to generate a customized password dictionary

– On-line (insider/outsiders)
  • Access the login process. Feasible for outsiders

– Off-line (insider/outsider after undue disclosure)
  • Obtain leaked database of encoded passwords. Compute the encoding of all possible passwords until a match is found.
Mitigation strategies

• Force users to choose good passwords
  – Specify need to include lower and upper-case alphabet characters, digits and non-alphanumeric keys.
  – Checking them against dictionary attacks
• Make users to change passwords often
• Use salted passwords
Eavesdropping the password

- Often users must authenticate over network connections
- The connection must be separately secured
- Alternative: Use strong authentication techniques
  - Overcome password weakness problems
  - Prevent eavesdropping
“How am I to get in?” asked Alice again, in a louder tone. “ARE you to get in at all?” said the Footman. “That's the first question, you know.”

From *Alice in the Wonderland*
First Steps

• How can we make this protocol secure?

• Note that Bob knows sensitive information about Alice, namely her password. What if use a cryptographic key instead?
Cryptographic Authentication

Alice, knock-knock

Challenge R

$X = F(K, R)$

K = shared key between Alice and Bob
Distributing keys

• In a large network, it is infeasible to assume that each pair of nodes share a secret key.

• Idea: Use a central server to distribute keys, the **Key Distribution Center (KDC)**.
  – Analogy: In a LAN, often the password database is stored in a single server
Using a KDC

• Entity $\alpha$ contacts the KDC and uses $K_\alpha$ (shared key) to authenticate herself and request a key for use with entity $\beta$.
• KDC generates a session key $R_{\alpha\beta}$
• KDC encrypts the session key for $\alpha$ and for $\beta$
• KDC gives both copies to $\alpha$
• $\alpha$ decrypts the session key and contacts $\beta$, giving him the encryption of the session key computed by the KDC
Multiple KDCs

Alice

May I talk to KDC₂?

K₁₂ {Kₙₑw}, K₁₂ {Kₙₑw}

Kₐ{l}(Kₙₑw), K₁₂ {Kₙₑw}

May I talk to Bob? K₁₂ {Kₙₑw}

Kₙₑw{K_AB}, K_B {K_AB}

Hello Bob: K_B {K_AB}

Bob's home KDC₁

K₁₂ {Kₙₑw}

Bob's home KDC₂

Bob
A Global KDC infrastructure

- If each organization wishes to set up a KDC, it becomes infeasible for each to have a key shared with all the others
- KDCs need to authenticate other KDCs
  - Hierarchical infrastructure, with root KDC
  - Trust graphs, with varying trust levels
    - KDC1 does NOT trust KDC2
    - KDC1 trusts KDC2 to authenticate KDC2 users
    - KDC1 trusts KDC2 to authenticate any users
    - KDC1 trusts KDC2 to introduce trusted KDCs
Session Keys

- Session keys should be used for securing channels, with long-term key only for authentication
  - Reduces amount of information encrypted under the same key/ make cryptanalysis difficult
  - Prevents replay attacks
- Known-key attack:
  - Long-term keys derivable from session transcript + session key (BAD!)
More on Session Keys

• Leak of a session-key should NOT enable discovery of further session keys
  – Otherwise as bad as leaking the long-term key
• Forward secrecy:
  – Previous session keys undiscoverable after compromise of a session key
  – Previous session keys undiscoverable after compromise of a long-term key
    • This last level of security requires techniques from public key cryptography