Admin: Project overviews due today
Quiz on Monday
(In class; coursework through today, your own notes & class notes (printed) are allowed & handouts.)

Outline:

- Key establishment
- Direct
  - Single server - Symm
    - Asymm
- Large-scale: X.509
  - SPKI/SDSI
- Cert revocation
Key management / key distribution

themes: crypto, keys, names, individuals, trust, identity, scaling, usability, certificates, PKI, trusted intermediaries

* keys need to be shared to be useful (at least PK must for PK crypt)
  * how is such sharing to be arranged?

* Directly (by physical mtg)
  
  Alice -------- Bob

  meet in private (no eavesdroppers)
  recognize each other (authentication)
  share PK's, or symmetric keys
  save in database: (when Alice has >1 contact...)

  Alice                  Bob
          ______        ______
         /        \      /        \
        /         \    /         \  
       /          \  /          \ 
      /            / /            /
     /             / /             /
    /              / /              /
   /                / /                /
  
  note appearance of names tied to entries...

privacy not needed if PK's are exchanged (as opposed to symm. keys)
Alice or Bob could be a computer (e.g. Alice installs key in computer Bob, or computer Alice gives user Bob her public key). Such direct meetings are necessary foundation of key mgt, as well see...

Indirect / Two-link / TTP (trusted third party) or server

```
A  \  S
   \ /
    B
```

Alice & Bob can't meet in person, but they have each met with trusted third party S and exchanged keys.

They can then request S to broker a "key-setup" operation so that A & B end up sharing a key, more or less as if they had actually met.

However, as we'll see, they need to trust S to be have properly set up protocol (aka "key exchange" needs to be well designed).

Needham & Schroeder proposed 2 such protocols: one for symmetric keys, and one for public keys.
NS Symmetric Protocol

K_{AS} = key shared between A & S \{ already set up
K_{BS} similar for B & S

"Nonce" means a "use once" value

N_A nonce generated by Alice
N_B " " " Bob

could be from a counter, or a long random value...

Used to protect against certain forms of "reply attack"...

K_{AB} : key that gets setup here between A, B (and S)

\{ M \}_{k} \text{ message } M \text{ encrypted with } k
\text{ (e.g. encrypt } M \text{ using AES in suitable mode, and by } k,
\text{ append } MAC_{k_{a}}(M), \text{ where } K_{1} \text{ & } K_{2} \text{ derived from } K)

(literature often is vague about properties of \{ M \}_{k})
Protocol:

1. $A \rightarrow S: A, B, N_A$
   - hi, I'm Alice, & I want to talk with Bob
   - $N_A$ is my "request nonce"

2. $S \rightarrow A: \{N_A, K_{AB}, B, \{K_{AB}, A\}\}$
   - Ok here's key $K_{AB}$ & blob to give $B$

3. $A \rightarrow B: \{K_{AB}, A^2K_{BS}\}$
   - $A$ knocks on $B$'s door
   - $B$ decrypts & checks blob

4. $B \rightarrow A: \{N_B\}$
   - $B$ challenges $A$ with $K_{AB}$

5. $A \rightarrow B: \{N_B^{-1}\}$
   - $A$ responds

- Who knows $K_{AB}$? $A$, $B$, $S$

- $S$ must be trusted: can pretend to be $A$ to $B$ or vise versa...

- Note roles of names: handles by which to identify parties
  - addresses to which msgs can be sent
  - text strings that can be included in messages

- If no nonces in 1, 2 & cut out 3, 5: could replay earlier session to $A$ or $B$
  (can do same if $K_{AB}$ later compromised... → fix w/ timestamps)
PK protocol

1. A → S: A, B

2. S → A: \{ K_{PB}, B^2 \}_{K_{SS}}
   signed "cert" for S's PK

3. A → B: \{ N_A, A^2 \}_{K_{PB}}
   knock
   encrypted, bound together
   "post! I'm A, and here's N_A... till then!"
   PK req

3' B → S: B, A
3" S → B: \{ K_{PA}, A^2 \}_{K_{SS}}
   signed "cert" for A's PK

4. B → A: \{ N_A, N_B \}_{K_{PA}}
   encrypted, bound together
   (non-malleable...)

5. A → B: \{ N_B^2 \}_{K_{PB}}
   yep, I'm really here...

- 3' & 3" could be replaced by including blob/"cert" \{ K_{PA}, A^2 \}_{K_{SS}} in 2
- at end only A & B know N_A & N_B; eavesdroppers don't...
Attack! (Gavin Lowe) 17 yrs later!

automated analysis

intruder I gets A to initiate communication with I
then passes hack \( E_{N_A, A^3} \) on to B (after re-encrypt with \( K_{AB} \))

- B responds with \( E_{N_A, N_B^3} \), which I sends to A
- A sends \( E_{N_B^3} \) to I, I decrypts it & gets \( N_B \)
- I sends \( E_{N_B^3} \) to B

Now B thinks he is sharing \( N_A \) & \( N_B \) only with A, but I knows \( N_B \), WRONG

Fix: (4) \( B \rightarrow A: E_{N_A, N_B, B^3} \)

Moral: Be explicit in protocols!

(e.g. give session id both ways, & identities;

give hash of shared transcript in each message
(i.e. of all previous messages...)

Huge literature on such key-establishment protocols...