# CS530
# Key Management & Distribution

Bill Cheng

*http://merlot.usc.edu/cs530-s10*



## Using Cryptography

- Provides foundation for security services
  - touched upon one form of key exchange
- But can it bootstrap itself?
  - must establish shared key
  - straightforward plan
    - one side generates key
    - transmits key to other side
    - but how?



## Two Problems

- Peer-to-peer key sharing
- Prob 1: known peer, insecure channel
- Prob 2: secure channel, unknown peer



## Man in the Middle of DH

- DH provides key exchange, but no authentication
  - you don't really know you have a secure channel
  - man-in-the-middle
  - you exchange a key with eavesdropper (man-in-the-middle), who exchanges key with the person you think are you talking to directly
  - eavesdropper relays all messages, but observes or changes them in transit
  - solutions
    - published public values
    - authenticated DH (signed or encrypt DH value)
    - encrypt the DH exchange
    - subsequently send hash of DH value, with secret



## Security Through Obscurity?

- Caesar ciphers
  - very simple permutation
  - only 25 different cases
  - relies strictly on no one knowing the method
  - key exchange is really method exchange

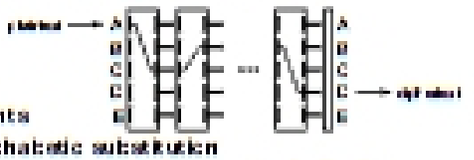


## Passwords

- Reduces permutation space to key space
  - Caesar cipher: one-letter "key"
  - 10-letter key for MSC reduces 26! ($\approx 4 \times 10^{26}$) to $26^{10}$
    - but hard to remember 10-letter key
  - 8-byte key for DES reduces $2^{64}$ ($\approx 10^{19}$) to $2^{56}$ ($\approx 10^{17}$)
- But key is more compact and perhaps more readily exchanged out of band
  - in person
  - by telephone (especially for public keys)
- Most security depends on some out of band bootstrap
  - exceptions? are they really exceptions?
    - DH provided key exchange, but not authentication

## The German Enigma Machine

- *Rotor*-based
  - rotors are wired codewheels
  - a rotor implements a fixed mono-alphabetic substitution
  - *polyalphabetic substitution* (with a long period) - the encipherment of each plaintext character causes various rotors to move, like an odometer (but not exactly)




- Broken first by Polish, then by English
  - not as easily as widely regarded
- Weaknesses in key distribution
  - day keys plus scramblers (using subkeys)
  - "session keys" encrypted in duplicate
  - Enigma did not use CFB/CFB



# Secret Key Distribution

**Bill Cheng**

***http://merlot.usc.edu/cs530-s10***



## Peer-to-Peer Distribution

- Technically easy
  - by hand
  - or have a day key
- But it doesn't scale
  - hundreds of servers...
  - times thousands of users...
  - yields ~ million keys
- Centralized key server
  - building up to the *Needham-Schroeder* approach

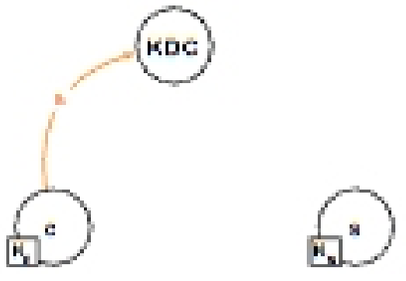


## Needham and Schroeder - Basic Idea

- User sends request to KDC (A)
- KDC generates a random key $K_{c,s}$
  - encrypted twice, each with a different key
    - $\{K_{c,s}, A\}K_c$, $\{K_{c,s}, A\}K_s$
    - $\{K_{c,s}, A\}K_c$ is the *credentials* (contains session key)
    - $\{K_{c,s}, A\}K_s$ is the *ticket*
    - ticket is *opaque* to the client, it is meant to be forwarded with application request
- No keys ever traverse the net in the clear

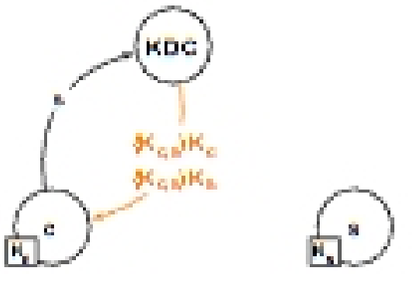


## KDC

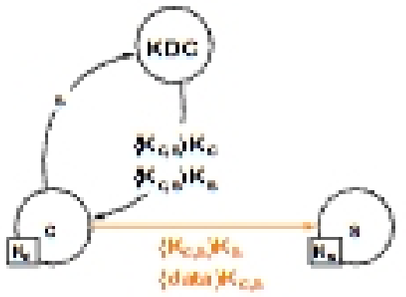





## KDC (Cont...)



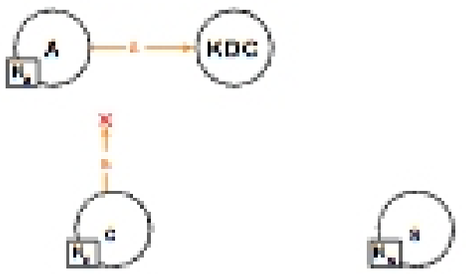
## KDC (Cont...)



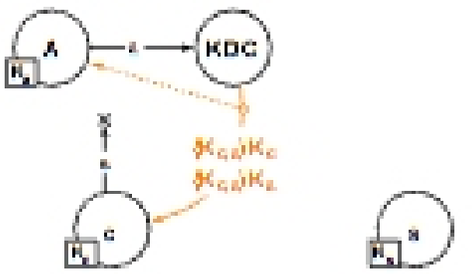

## Problem #1

- How does user know session key is encrypted for the server? And vice versa?
- Attacker intercepts initial request, and substitutes own name for server
  - can now read all c's user's messages intended for server

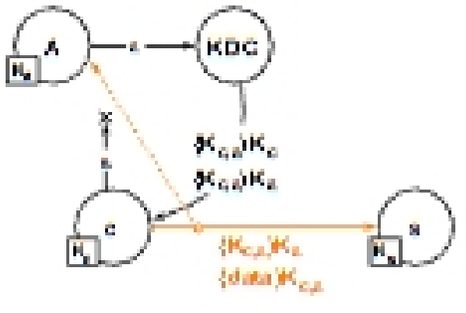
## Problem #1 (Cont...)




## Problem #1 (Cont...)




## Problem #1 (Cont...)




## Solution #1

- Add names to ticket, credentials
  - request looks like (c, s)
  - $\{K_{c,s}, s\}K_c$ and $\{K_{c,s}, c\}K_s$ respectively
- Both sides can verify intended target for key sharing
- This is basic *Needham-Schroeder*