Chapter 23
Authentication Protocol and Key Management
This Lecture

- Designing Authentication Protocol
- Key Management
  - Key Distribution Centre (KDC)
  - How the KDC distributes keys?
  - KDC vs. CA@PKI
Designing Authentication Protocol

1st attempt: Alice says “I am Alice”; but Bob can’t “see” Alice over a network

2nd: Alice says so w/ her source IP, but any one can fake the IP by spoofing

3rd: Alice says so w/ her password
   - Vulnerable to replay attack, attacker can easily steal the password
   - OK, let’s encrypt it for Bob
   - Still, replay attack, attacker can just replay the encrypted password
   - If only Bob is the verifier, perhaps he can also store the ctxt. he saw?
     - And rule out all those ciphertexts he has seen before?
   - It needs a probabilistic encryption scheme to create a “fresh” ctxt.
     - e.g., ElGamal
   - But ElGamal is publicly-randomizable, i.e., anyone can “refresh” a ctxt.
Authentication Protocol (cont.)

4th: Bob sends a *number used once* (nonce) \( N \), Alice encrypts it

- *i.e.*, Alice encrypts both password and the nonce
- If Bob receives the same number \( N \), concludes that it’s timely.
- We can use symmetric-key or public-key (of Bob) here.
- But Bob cannot encrypt to Alice. What if Bob wants to do that?

5th attempt: using the public-key

- Bob sends \( N \) to Alice, and also requests for Alice’s public-key \( pk \)
- Alice replies by sending \( pk \) and a signature on \( N \) with \( sk \)
- If verification succeeds, Bob can encrypt to Alice using \( pk \) now
Man-in-the-Middle Attack and Fix

**Attack**
- Eve sits in the middle, withheld Alice’s signature, sends her own signatures and her own public key instead.
- Eve decrypts using her own secret key, encrypts the message under Alice’s public key and sends her the ciphertext.
- Bob and Alice cannot really detect it since the conversation did go on smoothly, but Eve reads every single message!

**Fix (yes, we saw MitM attack before already)**
- The whole attack does not work if Bob can be assured that it is Alice’s real/authentic public key ➔ PKI
- *i.e.*, the last fix to have **certified** public keys available, e.g., under PKI.
- That’s *public key* infrastructure, can we do it in a “symmetric-key” way?
Getting Help from Trusted Party

**Symmetric-Key**
- How do two entities establish shared secret key over insecure network?
- Key Distribution Center (KDC) acting as intermediary between entities

**Public-Key**
- How does Alice know a public key is Bob’s (not Eve’s)?
- e.g. from website / email / disk
- Certification Authority (CA) issues digital certificate
Key Distribution Center (KDC): Settings

- KDC is a server which shares different secret key with each registered user
- There are many users, e.g., Alice and Bob
- Alice and Bob know their respective symmetric key for communicating with KDC, call them $K_{A-KDC}$, and $K_{B-KDC}$. 
- KGC distributes keys to pairs of users (hosts / processes / apps)
- Each user must share a *unique* key with the KDC
  - Called “master key” (of a user) here
  - May derive other “keys” to be used by the same user
  - Don’t confuse it with any system-wide secret of KDC
  - Distributed in some *non-cryptographic* (e.g., physical) ways
- Each user then uses this master key to communicate with KDC to get a temporary *session key*
  - for establishing a secure “session” with another user
KDC Operating Principles (in principle)

Notation:
- \( K\{m\} \rightarrow \text{use symmetric key } K \text{ to encrypt } m \)
- \( K_{AB} \rightarrow K_{AB} \text{ (Key for Alice and Bob), derived by KDC} \)

Here, we assumed that the KDC has already stored \( K_{Alice} \) and \( K_{Bob} \)
(And we do not use the trouble notation \( K_{Alice-KDC} \) and \( K_{Bob-KDC} \))
First Refinement of Key Distribution

Notation:

-> “ticket” = $K_{Bob}\{use \ K_{AB} \ for \ Alice\}$ // KDC no need to talk to Bob directly

As a general rule of thumb, one should include as much info. about the communication as possible; for example, Alice and Bob’s identity.

$K_{AB}$ implicitly refers to “Use $K_{AB}$ for Alice and Bob” from now on.
Needham-Schroeder Scheme

\(N_1\), Alice wants Bob

\(K_{Alice}\{N_1, \text{“Bob”}, K_{AB}, \text{ticket to Bob}\}\)

Recall: “ticket” = \(K_{Bob}\{\text{use } K_{AB} \text{ for Alice}\}\)

Invent \(K_{AB}\)

Bob

Alice

\(K_{AB}\{N_2\}\)

\(K_{AB}\{N_2 - 1, N_3\}\)

\(K_{AB}\{N_3 - 1\}\)
Inter-working btw. multiple KDC domains

- CIA's KDC generates $K_{\text{new}}$, then tells KGB's KDC
  “Alice from my domain (Alice_{CIA}) wants to talk to you using that key”

- KGB's KDC generates $K_{AB}$, then tells Boris
  “Alice from CIA’s domain wants to talk to you using that key”
### KDC vs. CA

**KDC**
- Keys must be kept secret
- Participating entities need frequent contact w/ the KDC
- KDC needs to be online
  - Single-point of failure
  - Performance bottleneck
  - Prime DDoS target
- KDC can impersonate anyone to anyone
- No “forward secrecy”
  - compromising KDC => attacker can attack “backward in time”, i.e., decrypt all recorded ctxt.

**CA**
- Certificates can be made public
- CA does not need to be online
  - Less vulnerable for network attack
  - (Revocation list/CRL-publishing can be delegated to other public server)
- If a CA crashed
  - no new users can be added
  - but existing members can still talk to each other
- What if CA is compromised?
  - Passive attack (i.e., decrypting without first impersonating) isn’t possible
  - Active attack can be mounted
Kerberos

- Trusted key server system from MIT
- Provides cryptographic authentication server(s) to authenticate users to servers and servers to users in a network environment
  - enable secure access control of networked resources
  - without needing to trust all workstations
  - relieve users/administrators the burden of managing accounts and passwords (potentially many)
Threats in using Kerberos

- Basic function: Users wish to access services on servers
- But attackers want to...
- pretend to be another user
- alter the network address of a workstation (*cf.*, .rhosts)
- eavesdrop on exchanges and use a replay attack

from [www.creativeuncut.com](http://www.creativeuncut.com)
Deploying Kerberos

- KDCs are “physically” secured.
- All Kerberos exchanges are protected against confidentiality and integrity attacks.
- Kerberos libraries are distributed on all nodes with users, applications, and other Kerberos-controlled resources.
- Kerberos-rized applications
  - telnet, rtools (rlogin, rcp, rsh), network file systems (NFS/AFS), others (pine, eudora, etc.)