CS 426 (Fall 2010)

Key Distribution & Agreement

Outline

- Key agreement without using public keys
- Distribution of public keys, with public key certificates
- Diffie-Hellman Protocol
  - Correction: Also discovered earlier in GCHQ, by **Malcolm J. Williamson in 1974.**

Key Agreement in Symmetric Crypto

- For a group of N parties, every pair needs to share a different key
  - Needs to establish N(N-1)/2 keys

- Solution: Uses a central authority, a.k.a., Trusted Third Party (TTP)
  - Every party shares a key with a central server.
  - How to achieve that in an organization with many users?

Needham-Schroeder Shared-Key Protocol: Use Trusted Third Party

- Parties: A, B, and trusted server T
- Setup: A and T share K_{AT}, B and T share K_{BT}
- Goal: Mutual entity authentication between A and B; key establishment
- Messages:

$A \rightarrow T$: A, B, N_A (1)

$A \leftarrow T$: $E[K_{AT}]$ (N_A, B, k, $E[K_{BT}]$(k,A)) (2)

$A \rightarrow B$: $E[K_{BT}]$ (k, A) (3)

$A \leftarrow B$: E[k] (N_B) (4)

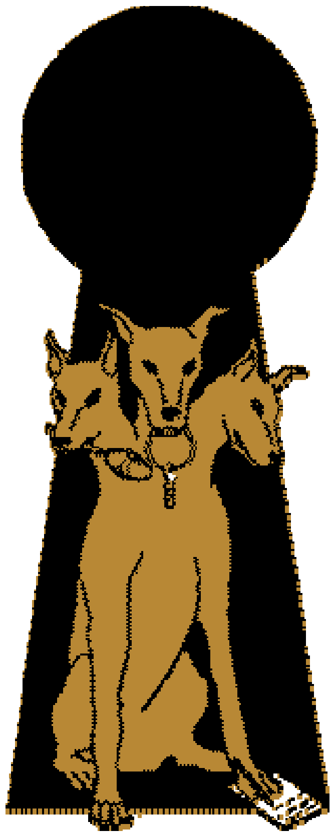
$A \rightarrow B$: E[k] (N_B-1) (5)

What bad things can happen if there is no N_A?

Another subtle flaw in Step 3.

Kerberos

- Implement the idea of Needham-Schroeder protocol
- Kerberos is a **network authentication protocol**
- Provides authentication and secure communication
- Relies entirely on **symmetric cryptography**
- Developed at MIT: two versions, Version 4 and Version 5 (specified as RFC1510)
- http://web.mit.edu/kerberos/www
- Used in many systems, e.g., Windows 2000 and later as default authentication protocol



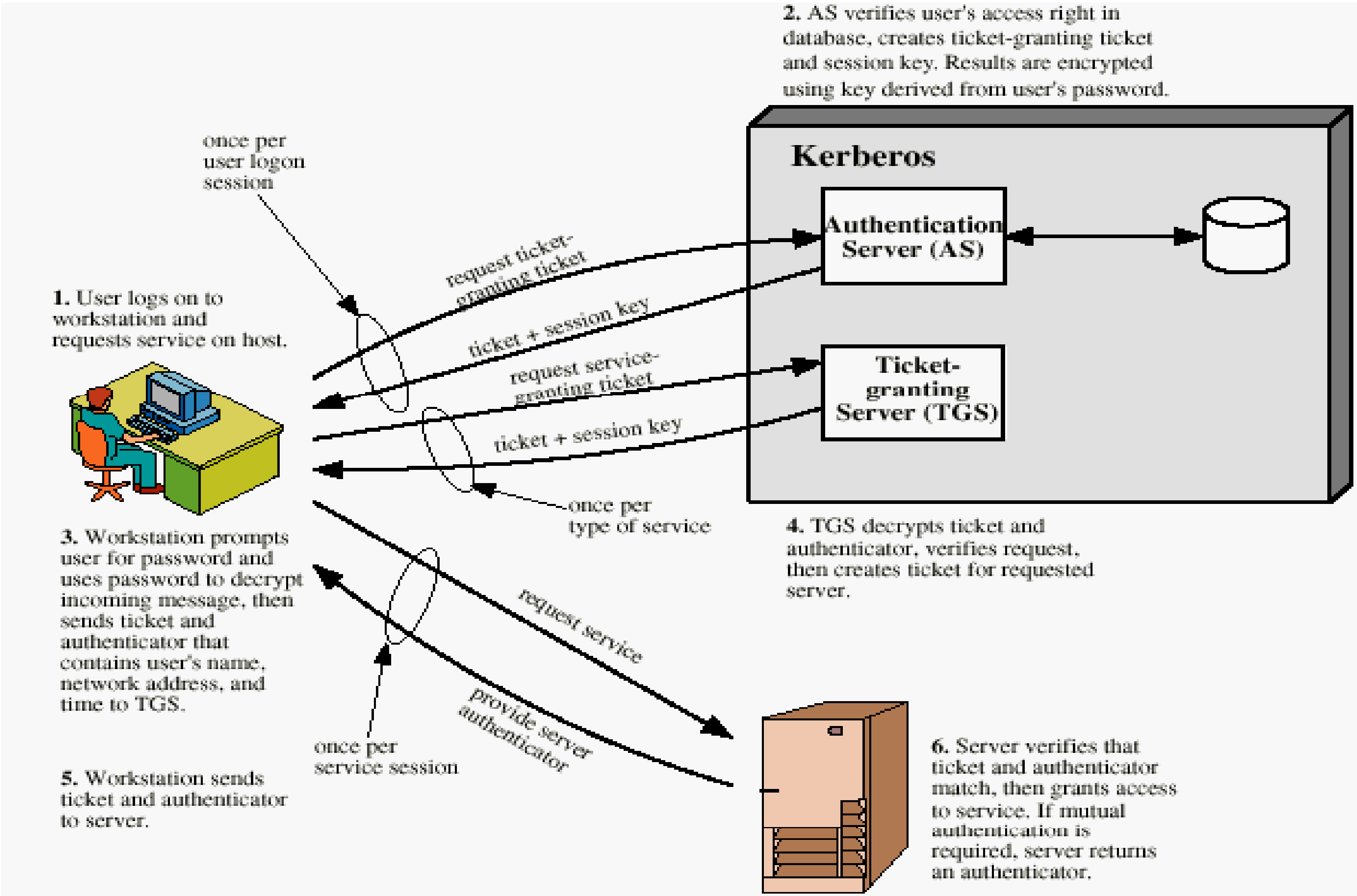
Kerberos Overview

- One issue of Needham-Schroeder
  - Needs the key each time a client talks with a service
- Solution: Separates TTP into an AS and a TGT.
- The client authenticates to AS using a long-term *shared secret* and receives a TGT.
  - supports single sign-on
- Later the client can use this TGT to get additional tickets from TGS without resorting to using the shared secret. These tickets can be used to prove authentication to SS.

AS = Authentication Server   TGS = Ticket Granting Server

SS = Service Server   TGT = Ticket Granting Ticket

Overview of Kerberos



Kerberos Drawback

- Single point of failure:
  - requires online Trusted Third Party: Kerberos server
- Security partially depends on tight clock synchronization. Convenience requires loose clock synchronization
  - Use timestamp in the protocol
  - The default configuration requires synchronization to with 10 minutes.
- Useful primarily inside an organization
  - Does it scale to Internet? What is the main difficulty?

Public Keys and Trust

- Public Key: P_A
- Secret key: S_A

- Public Key: P_B
- Secret key: S_B

How are public keys stored?

How to obtain the public key?

How does Bob know or 'trusts' that P_A is Alice's public key?

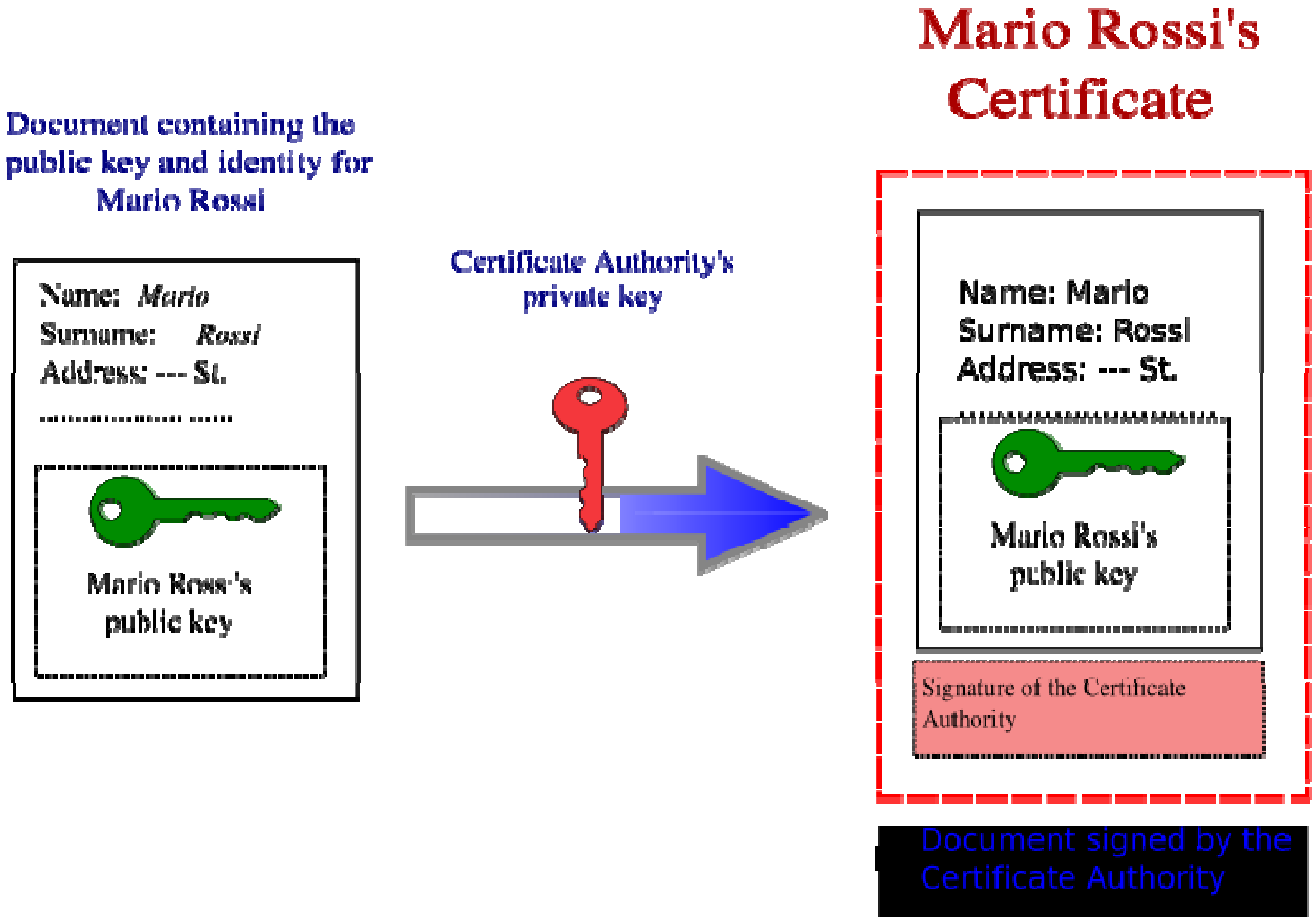
Distribution of Public Keys

- **Public announcement**: users distribute public keys to recipients or broadcast to community at large
- **Publicly available directory**: can obtain greater security by registering keys with a public directory
- Both approaches have problems, and are vulnerable to forgeries

Public-Key Certificates

- A certificate binds identity (or other information) to public key
- Contents digitally signed by a trusted Public-Key or Certificate Authority (CA)
  - Can be verified by anyone who knows the public-key authority's public-key
- For Alice to send an encrypted message to Bob, obtains a certificate of Bob's public key

Public Key Certificates

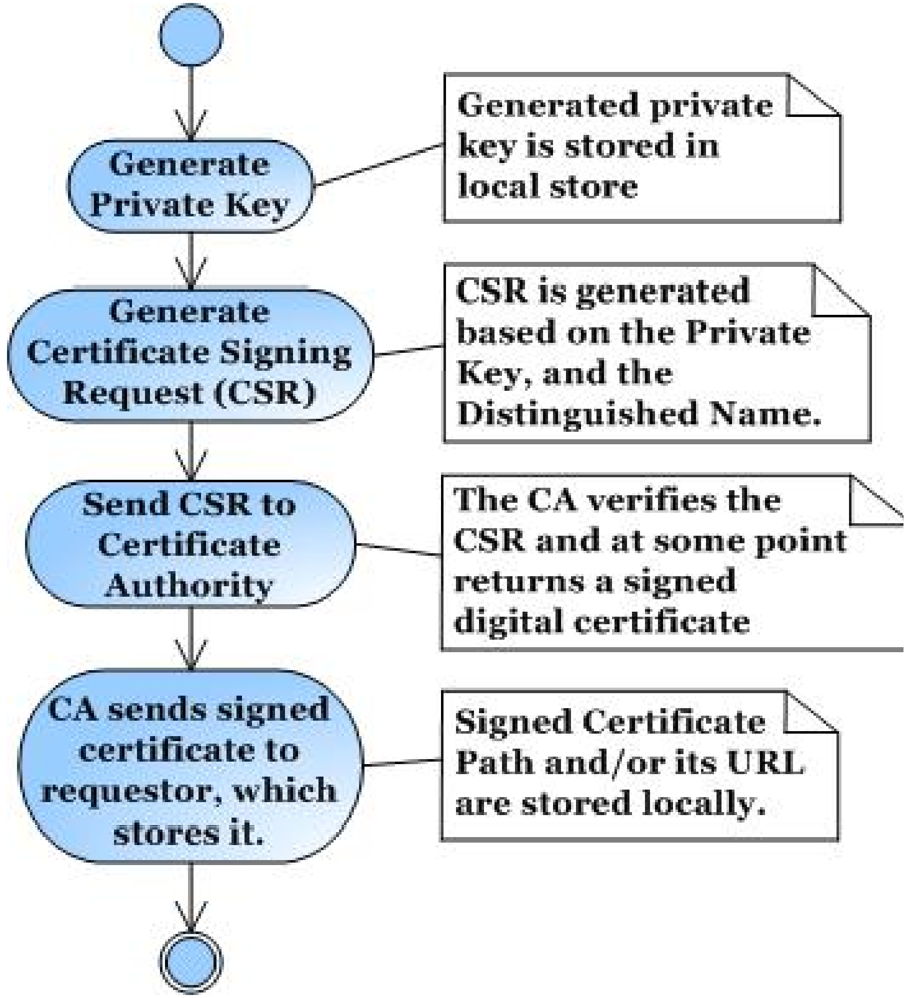


X.509 Certificates

- Part of X.500 directory service standards.
  - Started in 1988
- Defines framework for authentication services:
  - Defines that public keys stored as **certificates** in a public directory.
  - Certificates are **issued and signed** by an entity called **certification authority (CA).**
- Used by numerous applications: SSL, IPSec, SET
- Example: see certificates accepted by your browser

How to Obtain a Certificate?

- Define your own CA (use openssl or Java Keytool)
  – Certificates unlikely to be accepted by others
- Obtain certificates from one of the vendors: VeriSign, Thawte, and many others



CAs and Trust

- Certificates are trusted if signature of CA verifies
- Chain of CA's can be formed, head CA is called root CA
- In order to verify the signature, the public key of the root CA should be obtain.
- TRUST is centralized (to root CA's) and hierarchical
- What bad things can happen if the root CA system is compromised?
- How does this compare with the TTP in Needham/Schroeder protocol?

Key Agreement: Diffie-Hellman Protocol

Key agreement protocol, both A and B contribute to the key

Setup: p prime and g generator of Z_p^*, p and g public.

$g^a \bmod p$ →

← $g^b \bmod p$

Pick random, secret a

Compute and send $g^a \bmod p$

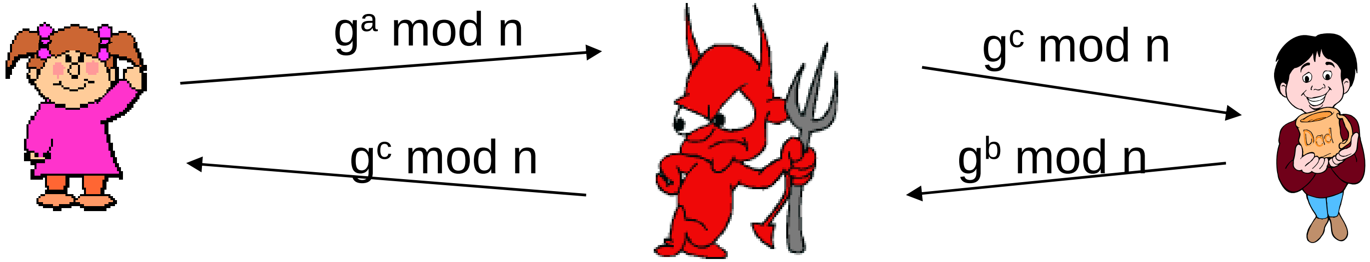
$K = (g^b \bmod p)^a = g^{ab} \bmod p$

Pick random, secret b

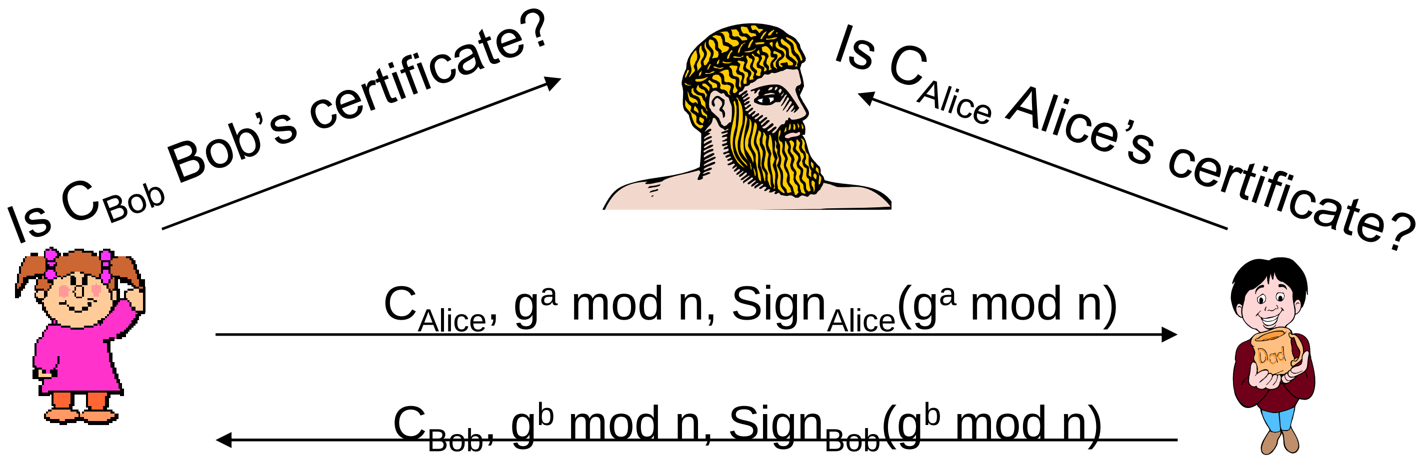
Compute and send $g^b \bmod p$

$K = (g^a \bmod p)^b = g^{ab} \bmod p$

Authenticated Diffie-Hellman

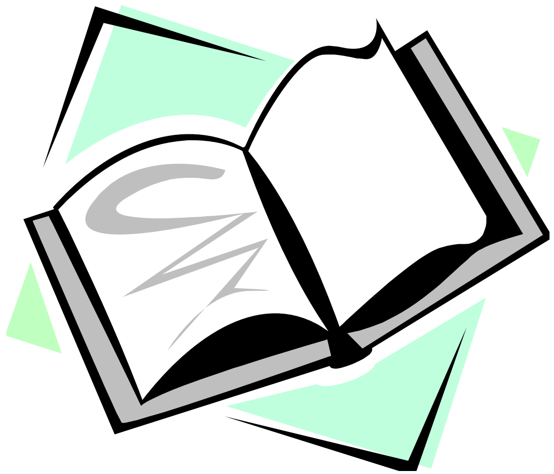


- Alice computes g^{ac} mod n and Bob computes g^{bc} mod n !!!



Readings for This Lecture

- On Wikipedia
  - **Needham-Schroeder protocol (only the symmetric key part)**
  - Public Key Certificates



Coming Attractions …

- Network Security