COMPUTER SECURITY CS 419

KEY DISTRIBUTION & AGREEMENT, SECURE COMMUNICATION

READINGS FOR THIS LECTURE

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

OUTLINE AND OBJECTIVES

- Key distribution among multiple parties
- Kerberos
- Distribution of public keys, with public key certificates
- Diffie-Hellman Protocol
- TLS/SSL/HTTPS

KEY AGREEMENT AMONG MULTIPLE PARTIES

- For a group of N parties, every pair needs to share a different key
  - What is the number of keys?
- Solutions
  - Symmetric Encryption - Use a central authority, a.k.a. (TTP).
  - Asymmetric Encryption – PKI.

NEEDHAM-SCHROEDER SHARED-KEY PROTOCOL

- 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:

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

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

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

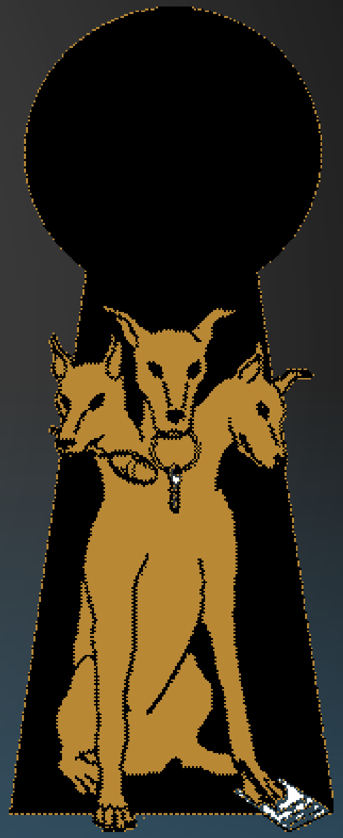
$$A \leftarrow B: \ E[k]\,(N_B) \qquad (4)$$

$$A \rightarrow B: \ E[k]\,(N_B\text{-}1) \qquad (5)$$

What bad things can happen if there is no NA.

KERBEROS

- Implements 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: http://web.mit.edu/kerberos/www
- Used in many systems, e.g., Windows 2000 and later as default authentication protocol

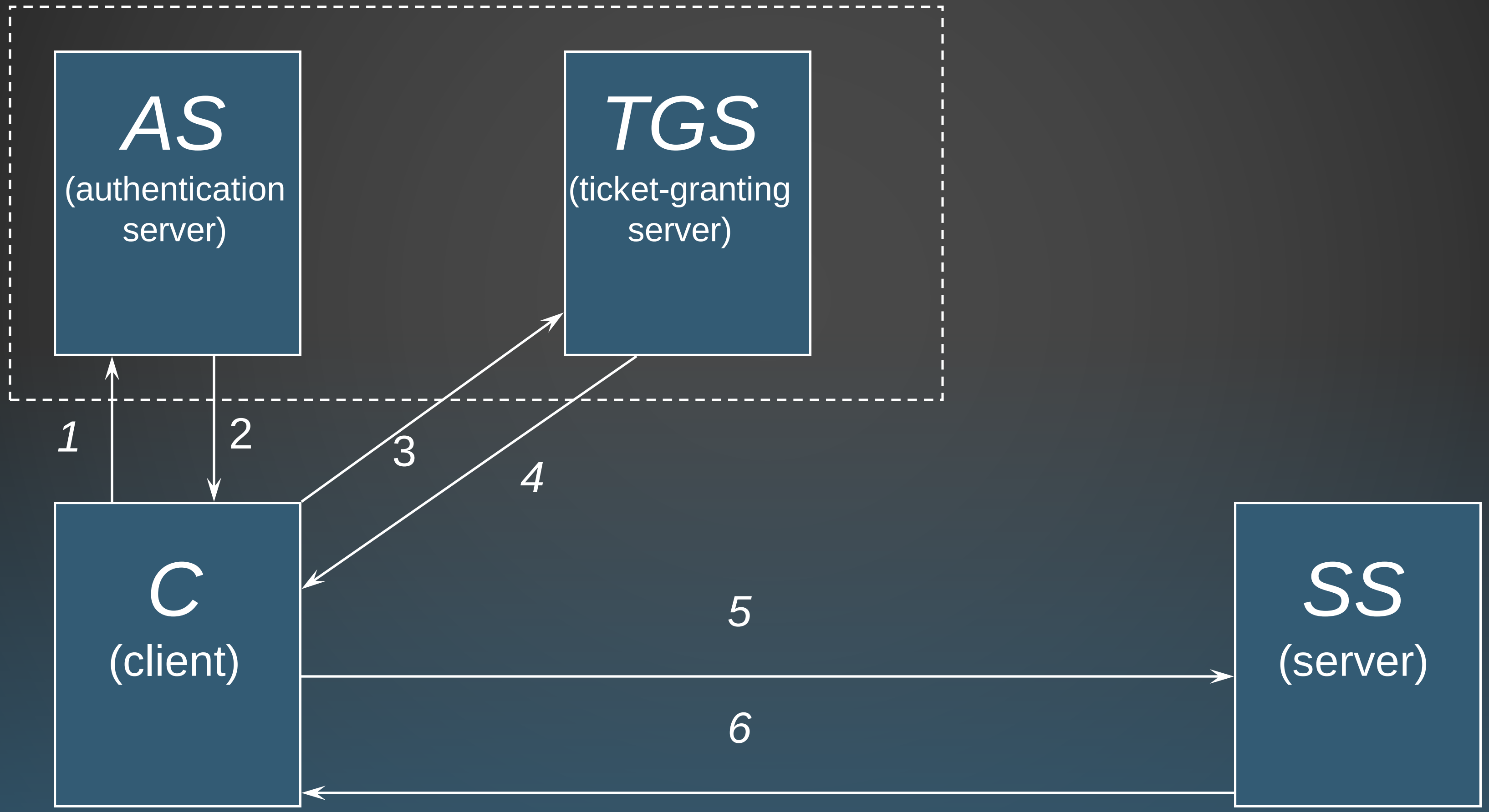


KERBEROS OVERVIEW

- One issue of Needham-Schroeder – Needs [K_{AT}] for every communication.
- Kerberos solution: Separates into an AS and a TGS.
- The client authenticates to AS using a long-term *shared secret* and receives a TGT.
- Use this TGT to get additional tickets from TGS without resorting to using the shared secret.

AS = Authentication Server
TGS = Ticket Granting Server
SS = Service Server
TGT = Ticket Granting Ticket

KERBEROS PROTOCOL - 1



KERBEROS PROTOCOL – 2 (SIMPLIFIED)

1. $C \rightarrow AS: TGS \,||\, N_C$
2. $AS \rightarrow C: \{K_{C,TGS} \,||\, C\}_{K_{AS,TGS}} \,||\, \{K_{C,TGS} \,||\, N_C \,||\, TGS\}_{K_{AS,C}}$

   (Note that the **first** part of message 2 is the **ticket granting ticket (TGT)** for the TGS)
3. $C \rightarrow TGS: SS \,||\, N'_C \,||\, \{K_{C,TGS} \,||\, C\}_{K_{AS,TGS}} \,||\, \{C||T_1\}_{K_{C,TGS}}$
4. $TGS \rightarrow C: \{K_{C,SS} \,||\, C\}_{K_{TGS,SS}} \,||\, \{K_{C,SS} \,||\, N'_C \,||\, SS\}_{K_{C,TGS}}$

   (Note that the **first** part in message 4 is the **ticket** for the server S).
5. $C \rightarrow SS: \{K_{C,SS} \,||\, C\}_{K_{TGS,SS}} \,||\, \{C \,||\, T_2\}_{K_{C,SS}}$
6. $SS \rightarrow C: \{T_3\}_{K_{C,SS}}$

KERBEROS DRAWBACK

- Single point of failure
- 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?

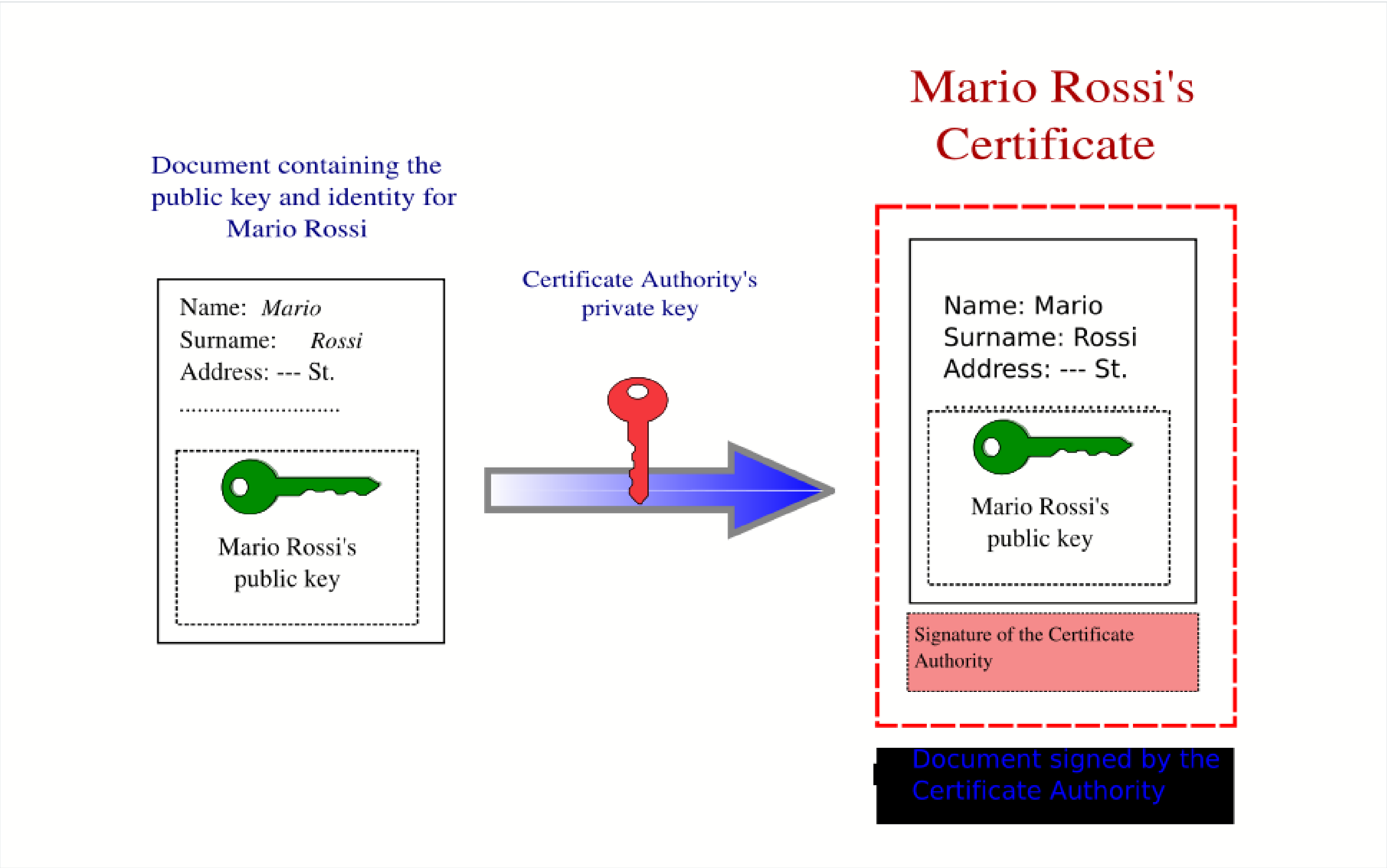
DISTRIBUTION OF PUBLIC KEYS

- 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

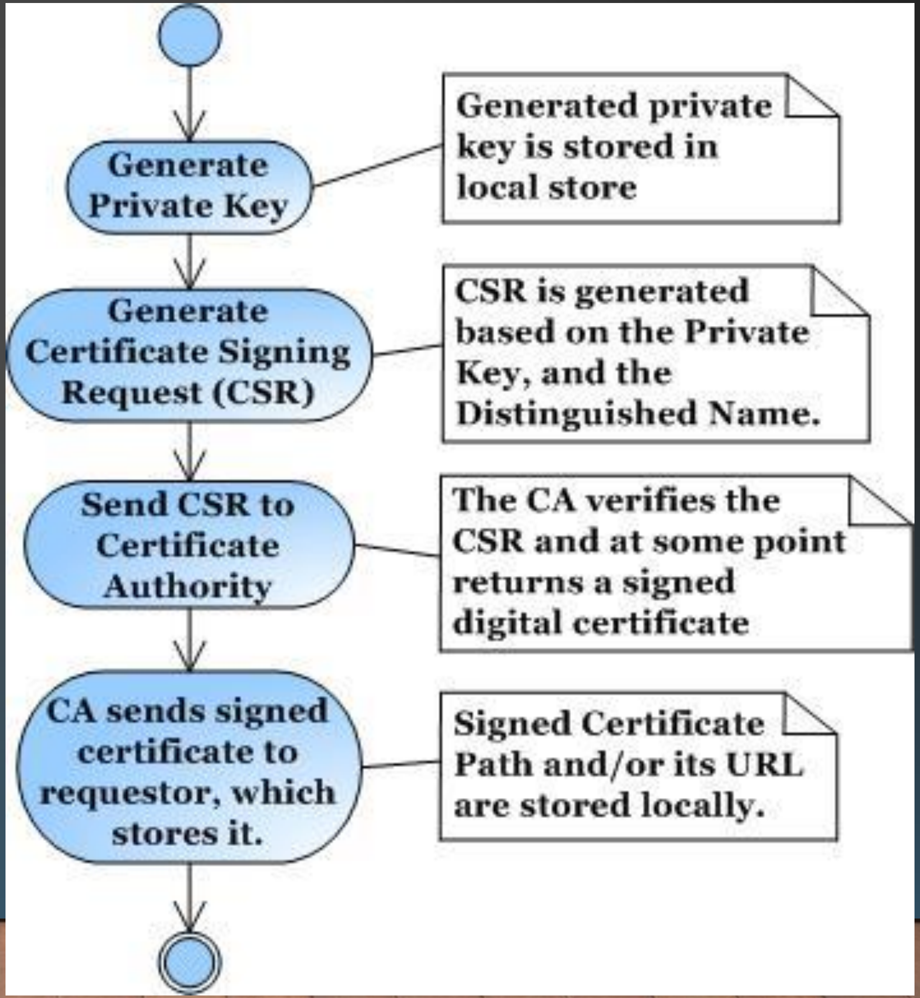


X.509 CERTIFICATES

- 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.

$g^a \bmod p \longrightarrow$

$\longleftarrow g^b \bmod p$

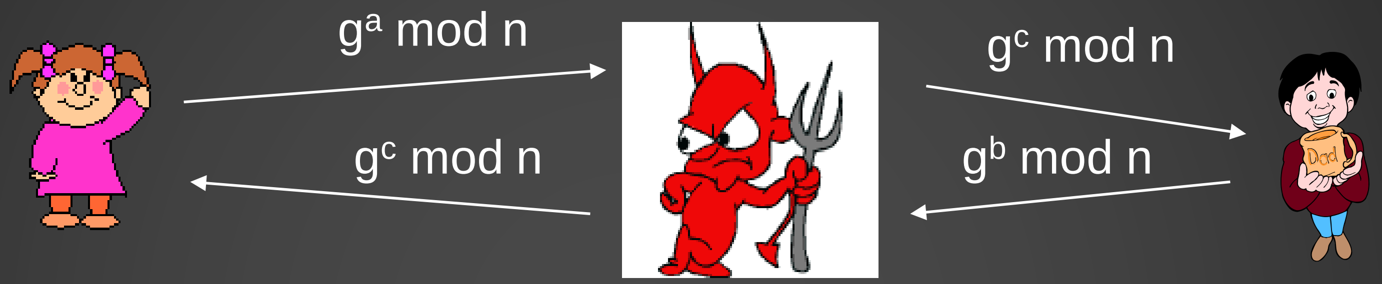
Pick random, secret (a)
Compute and send $g^a \bmod p$

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

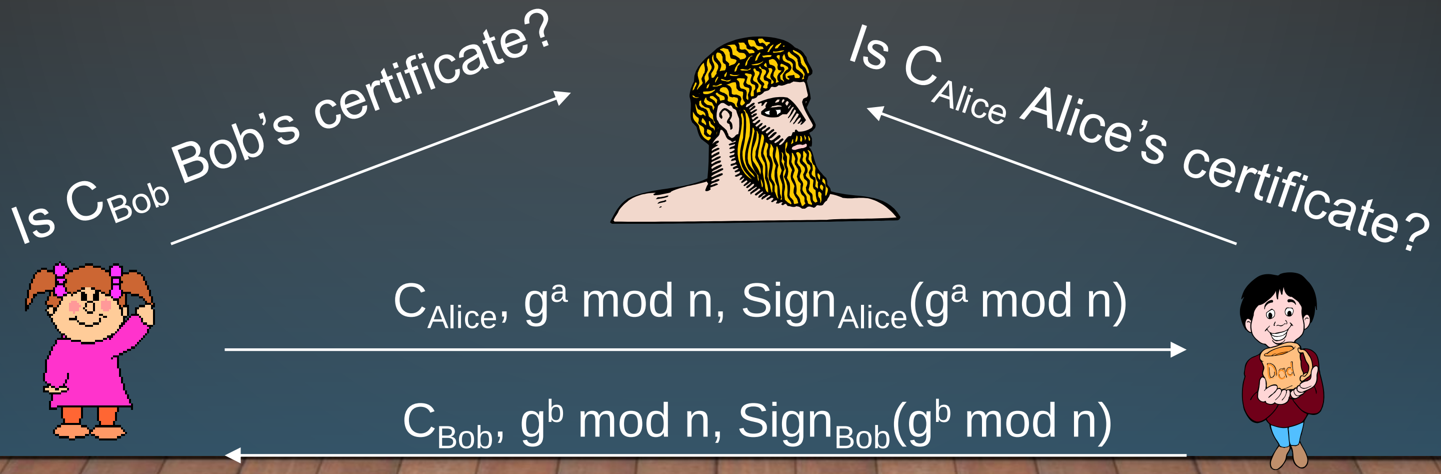
Pick random, secret (b)
Compute and send $g^b \bmod p$

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

AUTHENTICATED DIFFIE-HELLMAN



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



TRANSPORT LAYER SECURITY (TLS)

- Predecessors: Secure socket layer (SSL): Versions 1.0, 2.0, 3.0
- TLS 1.0 (SSL 3.1); Jan 1999
- TLS 1.1 (SSL 3.2); Apr 2006
- TLS 1.2 (SSL 3.3); Aug 2008
- Standard for Internet security
  - Originally designed by Netscape
  - Goal: "... provide privacy and reliability between two communicating applications"
- Two main parts
  - Handshake Protocol
    - Establish shared secret key using public-key cryptography
    - Signed certificates for authentication
  - Record Layer
    - Transmit data using negotiated key, encryption function

USAGE OF SSL/TLS

- Applied on top of transport layer (typically TCP)
- Used to secure HTTP (HTTPS), SMTP, etc.
- One or both ends can be authenticated using public key and certificates
  - Typically only the server is authenticated
- Client & server negotiate a cipher suite, which includes
  - A key exchange algorithm, e.g., RSA, Diffie-Hellman, SRP, etc.
  - An encryption algorithm, e.g., RC4, Triple DES, AES, etc.
  - A MAC algorithm, e.g., HMAC-MD5, HMC-SHA1, etc.

VIEWING HTTPS WEB SITES

- Browser needs to communicate to the user the fact that HTTPS is used
  - Check some common websites
  - When users correctly process this information, can defeat phishing attacks
  - Security problems exist
    - People don’t know about the security indicator
    - People forgot to check the indicator
    - Browser vulnerabilities enable incorrect indicator to be shown
    - Use confusing URLs
    - Stored certificate authority info may be changed

NEXT CLASS

- This is (mostly) all for crypto!
- System and Software Security