ENEE 459-C
Computer Security

Digital signatures and security protocols

UNIVERSITY OF
MARYLAND

The Big Picture

	Secret Key Setting	Public Key Setting
Secrecy / Confidentiality	Stream ciphers Block ciphers + encryption modes: AES, DES	Public key encryption: RSA, El Gamal, etc.
Authenticity / Integrity	Message Authentication Code: SHA-2	Digital Signatures: RSA, etc.

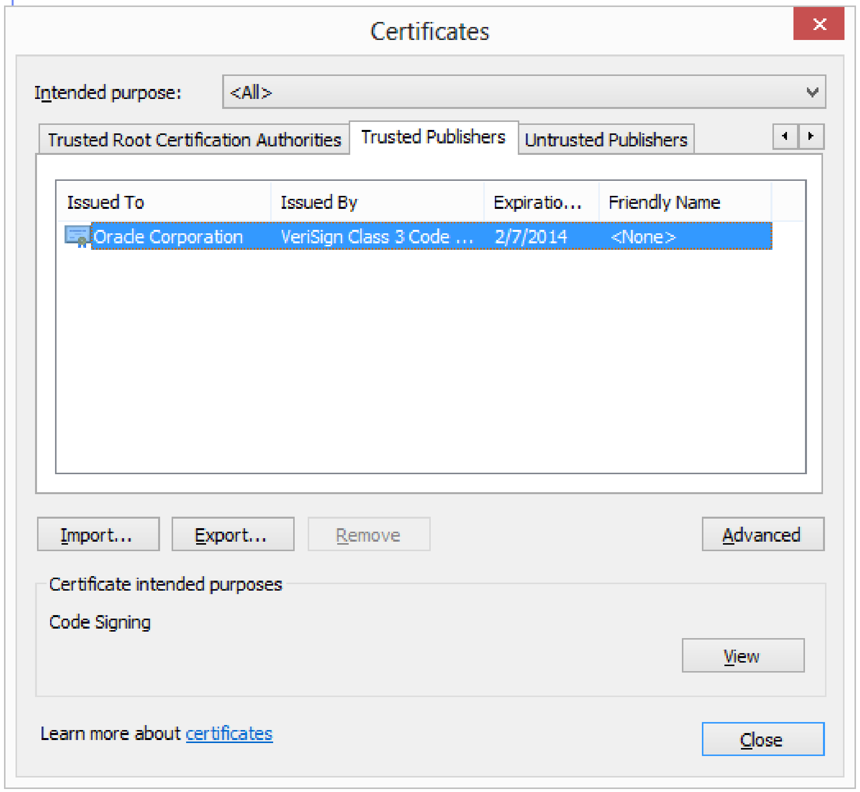
Do you trust your public key?

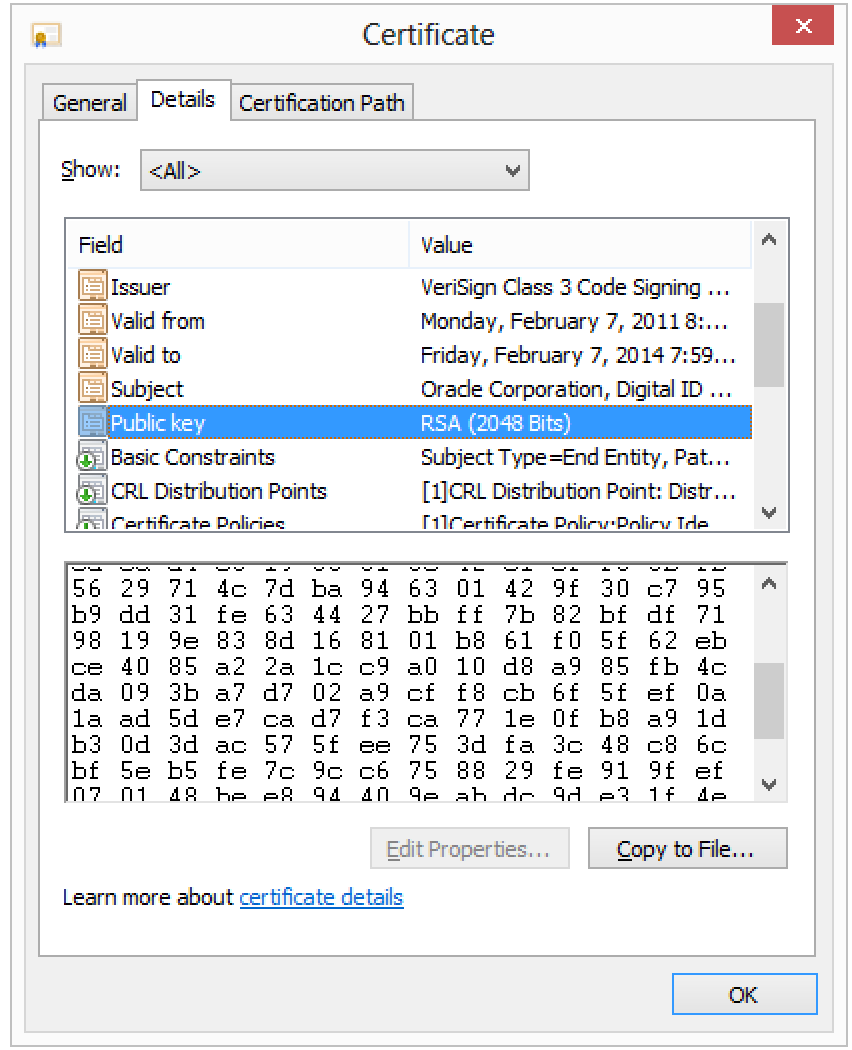
- Impostor Claims to be a True Party
  - True party has a public and private key
  - Impostor also has a public and private key
- Impostor sends impostor’s own public key to the verifier
  - Says, “This is the true party’s public key”
  - This is the critical step in the deception

X.509 Certificates

- 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
- Example: see certificates accepted by your browser

Example: Oracle's certificate

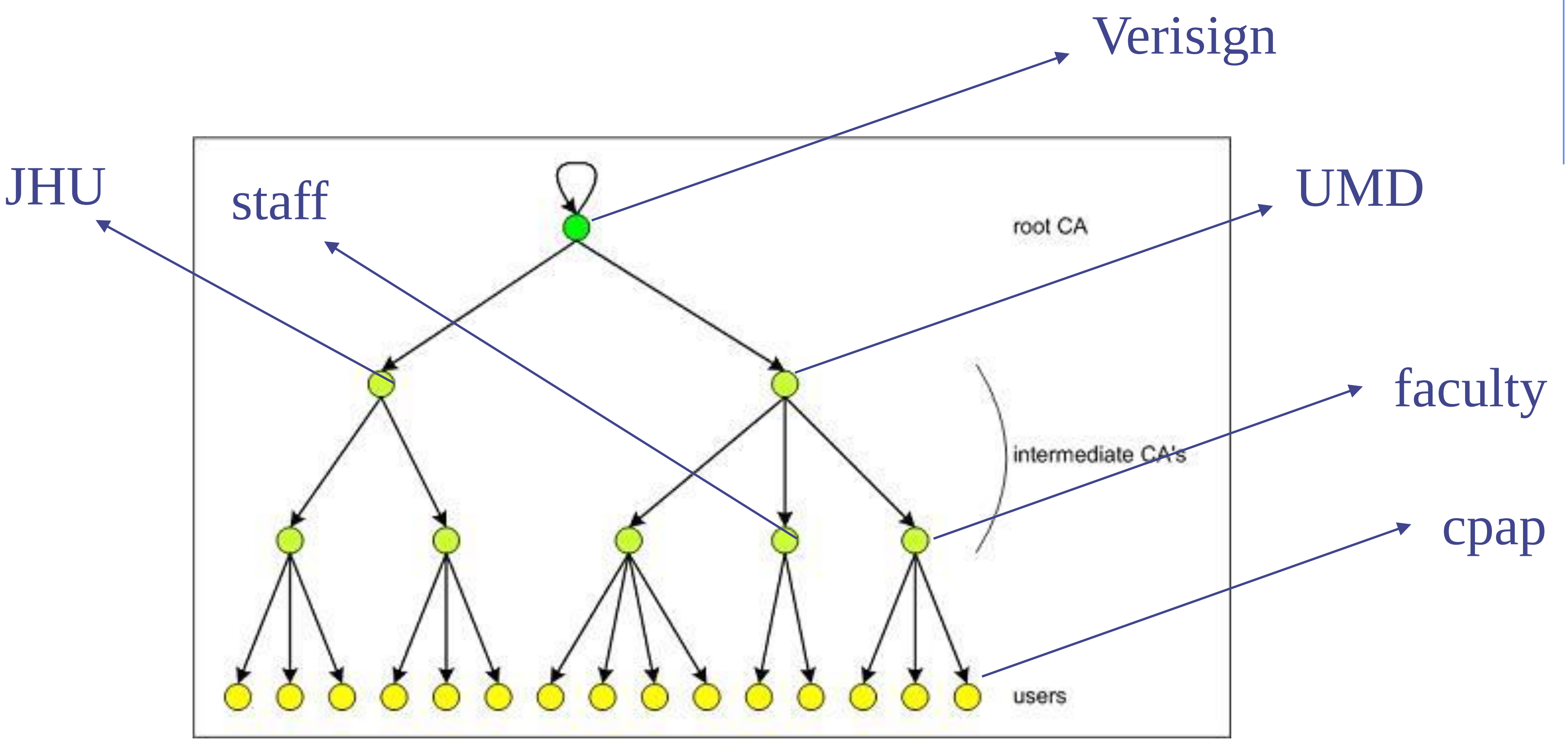




Certificate Hierarchy

- Single CA certifying every public key is impractical
- Instead, use trusted root authorities
- Root CA signs certificates for intermediate CAs, they sign certificates for lower-level CAs, etc.
  - Certificate "chain of trust"
    - $sig_{Verisign}$("UMD", PK_{UMD})
    - sig_{UMD}("faculty", $PK_{faculty}$)
    - $sig_{faculty}$("cpap", PK_{cpap})

Example



What bad things can happen if the root CA system is compromised?

Certificate Revocation

- Revocation is <u>very</u> important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to this CA and CA no longer wishes to certify him
  - CA's certificate has been compromised!
- Expiration is a form of revocation, too
  - Many deployed systems don't bother with revocation
  - Re-issuance of certificates is a big revenue source for certificate authorities

Integrated Security System

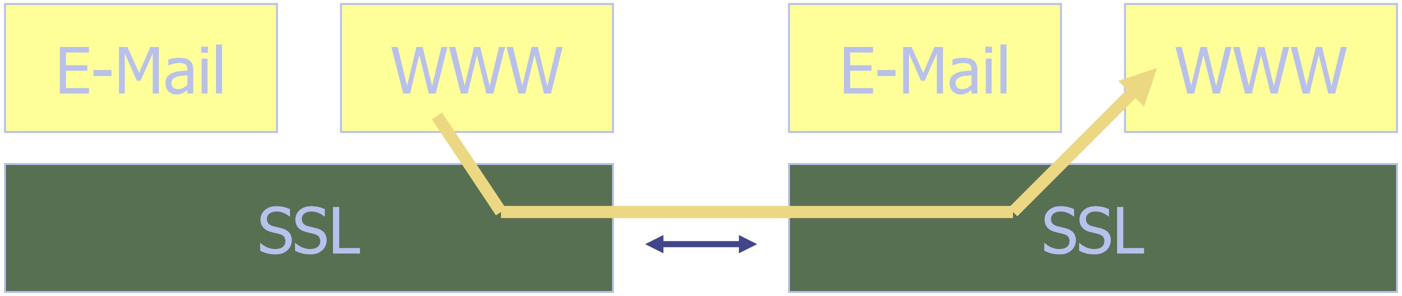
- When two parties communicate …
  - Their software usually handles the details
  - First, negotiate security methods
  - Then, authenticate one another
  - Then, exchange symmetric session key
  - Then can communicate securely using symmetric session key and message-by-message authentication

SSL Integrated Security System

- *SSL*
  - *Secure Sockets Layer*
  - Developed by Netscape
- TLS (now)
  - Netscape gave IETF (Internet Engineering Task Force) control over SSL
  - IETF renamed it TLS (Transport Layer Security)
  - Usually still called SSL

Location of SSL

- Below the Application Layer
  - Protects all application exchanges
  - Not limited to any single application
    - WWW transactions, e-mail, etc.



Protocols: Key agreement

Key Agreement among Multiple Parties

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

- Solution: Uses a central authority, a.k.a., Trusted Third Party (TTP)
  - Every party shares a key with a central server.
  - In an organization with many users, often times already every user shares a secret with a central TTP, e.g., password for an organization-wide account

A simple 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 (1)

$A \leftarrow T$: $E[K_{AT}]$ (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 is the problem here?

A more secure 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 (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)

With this modification, A is sure he has a fresh key.
Are we done?

Needham-Schroeder 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:

$$A \rightarrow B: \quad A \qquad (1)$$

$$B \rightarrow A: \quad E[K_{BT}](A,N'_B) \qquad (2)$$

$$A \rightarrow T: \quad A, B, N_A, E[K_{BT}](A,N'_B) \qquad (3)$$

$$A \leftarrow T: \quad E[K_{AT}]\ (N_A, B, k, E[K_{BT}](k,A,N'_B)) \qquad (4)$$

$$A \rightarrow B: \quad E[K_{BT}](k,A,N'_B) \qquad (5)$$

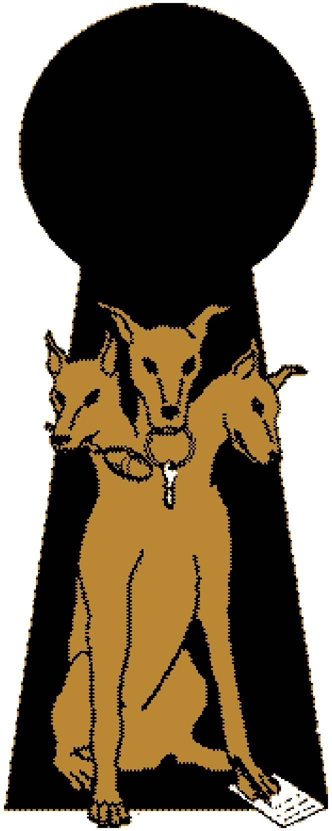
$$A \leftarrow B: \quad E[k]\ (N_B) \qquad (6)$$

$$A \rightarrow B: \quad E[k]\ (N_B-1) \qquad (7)$$

With this modification, step 5 cannot be compromised

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 Drawback

- Single point of failure:
  - requires online Trusted Third Party: Kerberos server
- Useful primarily inside an organization
  - Does it scale to Internet? What is the main difficulty?