

# Digital Signature



Xinwen Fu, Ph.D

Professor

Department of Computer Science  
University of Massachusetts Lowell



# Outline

---

- Introduction to digital signature
- Introduction to certificate
- Hands-on labs



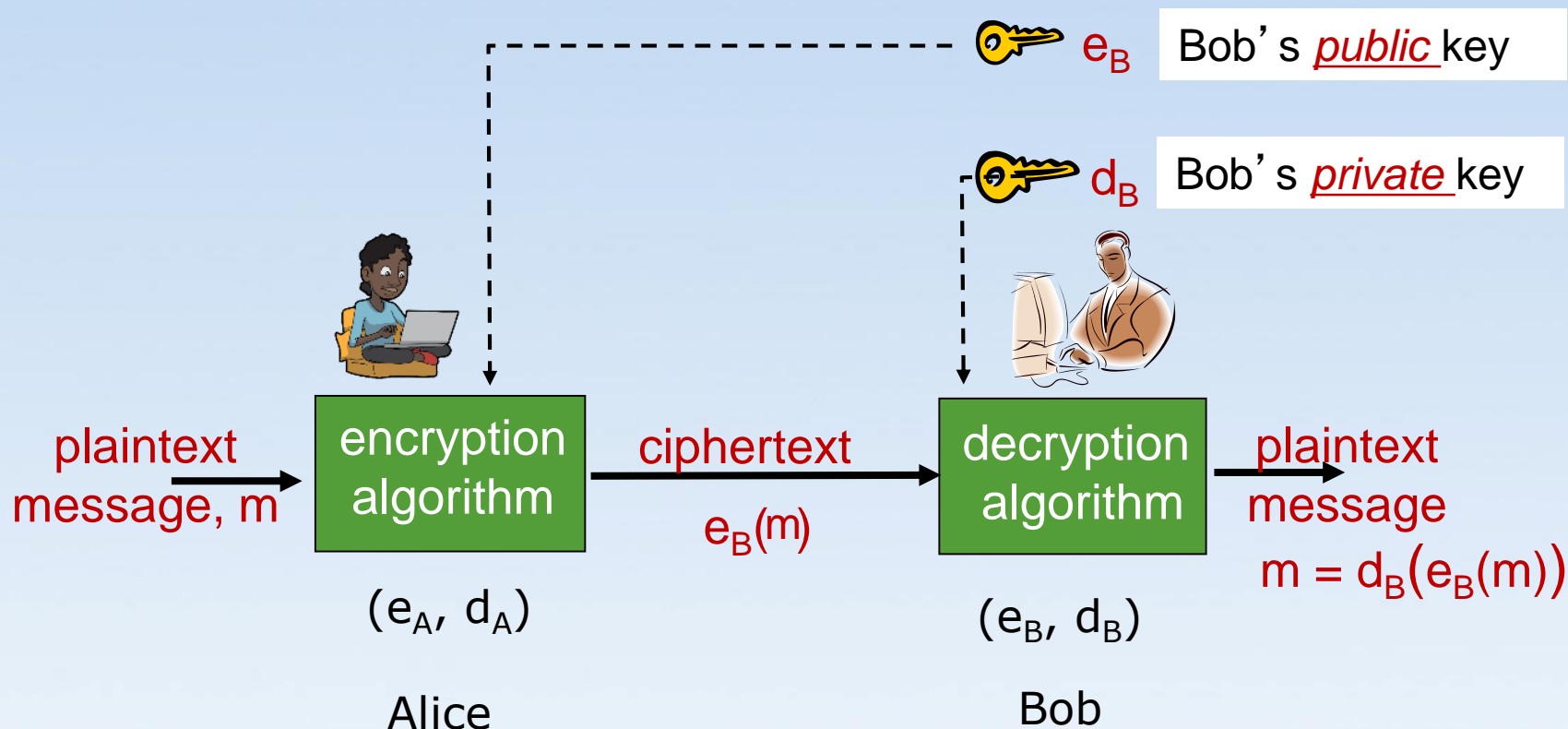
# Conventional Signature

---

- Verifies that the signer approves the writing
- How do we sign a digital document like a pdf file?



# Review of Public Key Cryptography



# Generate Public and Private Key Pair

---

- Generate public and private key pair
  - openssl genpkey -out privkey.pem -algorithm rsa
  - Note: privkey.pem contains both public and private keys
- Extract public key from privkey.pem and save it in pubkey.pem
  - openssl rsa -in privkey.pem -outform PEM -pubout -out pubkey.pem



# Encryption with RSA

---

- Create a file
  - echo "Welcome to LinuxCareer.com" > encrypt.txt
- Encryption
  - openssl rsautl -encrypt -inkey pubkey.pem -pubin -in encrypt.txt -out encrypt.dat
- Encode the binary ciphertext with base64
  - openssl enc -base64 -in encrypt.dat -out encrypt.dat.base64
  - Note: encoding is not necessary, but needed for sending the ciphertext through our chat server



# Decryption with RSA

---

- Decode *encrypt.dat.base64* and get the binary ciphertext
  - openssl enc -base64 -d -in encrypt.dat.base64 -out encrypt.dat
- Decryption
  - openssl rsautl -decrypt -inkey privkey.pem -in encrypt.dat -out new\_encrypt.txt



# Public Key Encryption Properties

---

- ① need  $e(.)$  and  $d(.)$  such that

$$d_B(e_B(m)) = m$$

- ② given public key  $e_B$ , it should be impossible to compute private key  $d_B$

**RSA:** Rivest, Shamir, Adelson algorithm





# Another important property

---

The following property will be very useful

$$e_B(d_B(m)) = m = d_B(e_B(m))$$

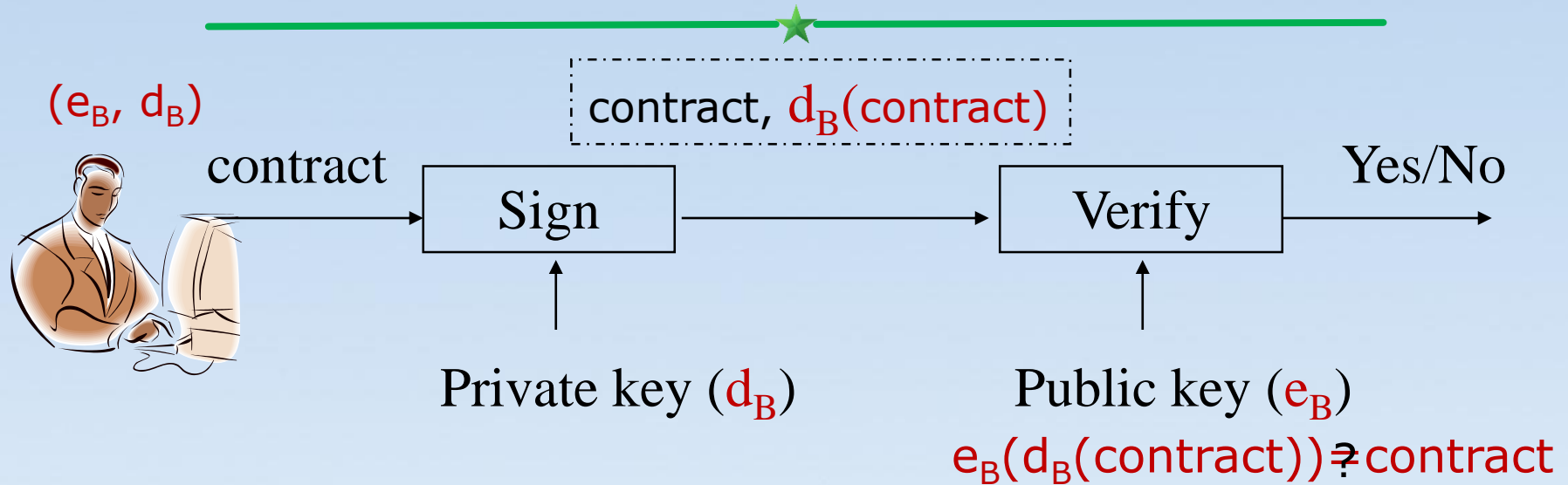
use public key first,  
followed by  
private key

use private key  
first, followed by  
public key

*result is the same!*

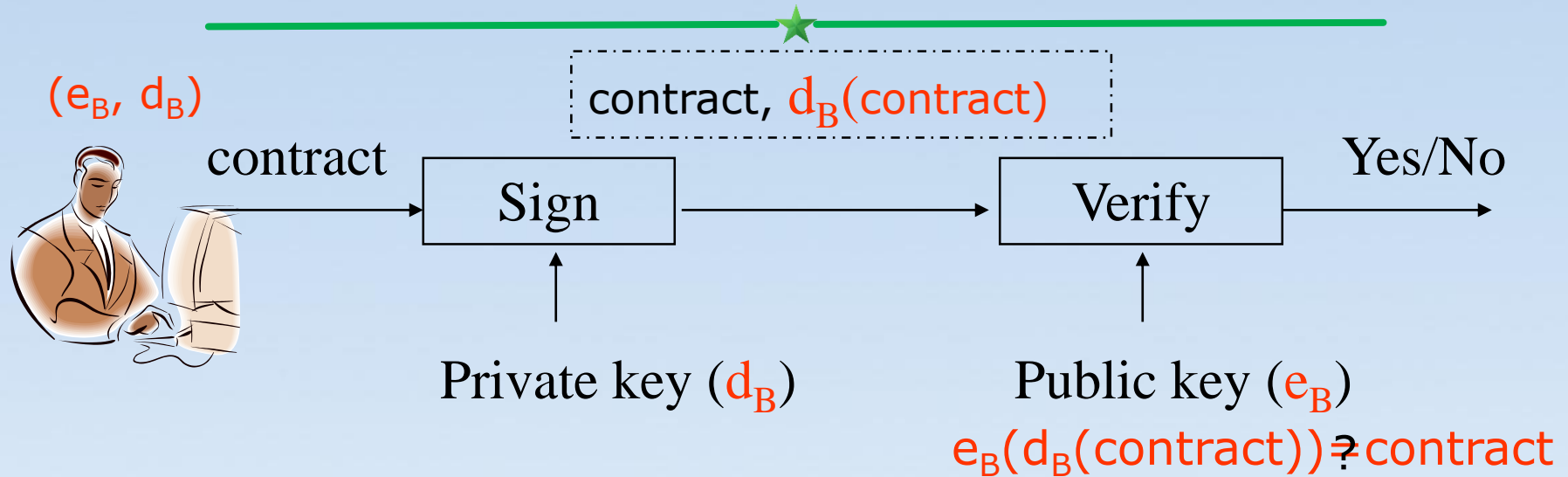


# Naive Digital Signature



- Bob encrypts the contract with his private key
  - $d_B(\text{contract})$  is the signature
- Those who have Bob's public key can verify the signature
  - Decrypt the signature
  - Compare the decrypted contract with the contract sent over

# Digital Signature Properties



- The party with private key can create digital signature
- Anyone who knows the public key can verify digital signature
- The signer cannot deny that he/she has done so

# Problem of Naive Digital Signature?

---

- contract,  $d_B(\text{contract})$

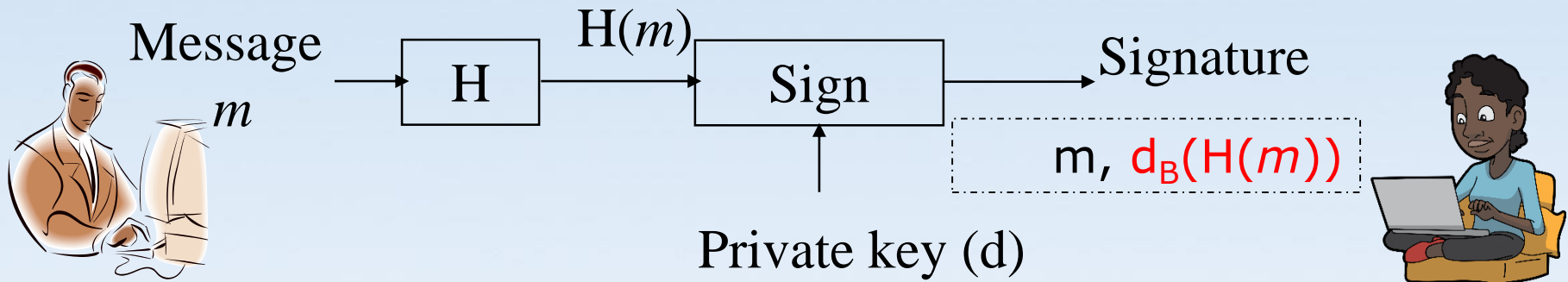
**Signature too long!**



# Signing of true Digital Signature

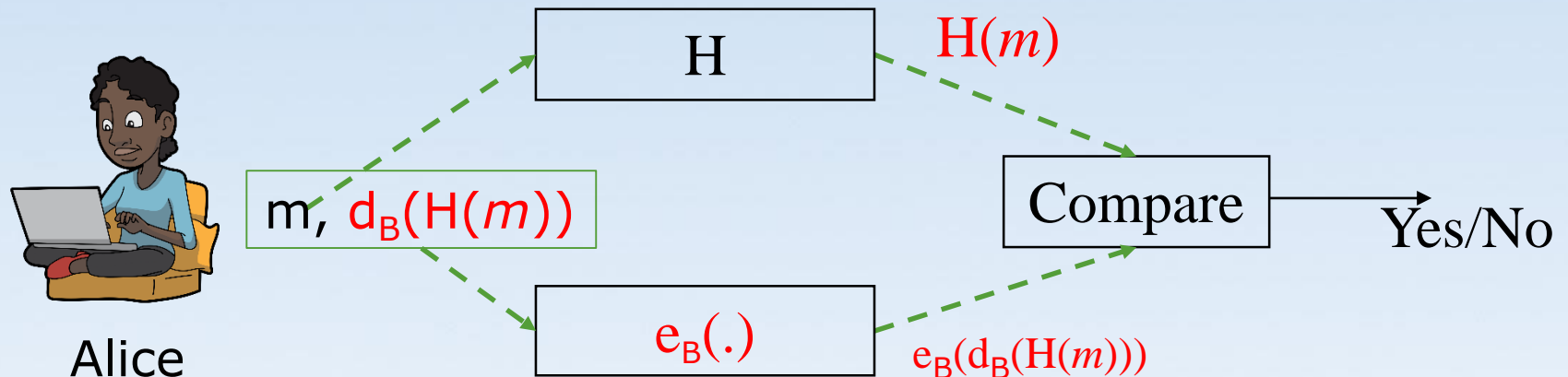
---

- Generate message hash
- Encrypt the hash with the private key



# Verification of True Digital Signature

1. Hash the message  $m$  that was sent over to get  $H(m)$
2. Decrypt digital signature  $d_B(H(m))$  with Bob's public key
3. Compare  $H(m)$  and  $e_B(d_B(H(m)))$



# Outline

---

- Introduction to digital signature
- Introduction to certificate
- Hands-on labs



# Question

- How can you safely give your public key to everyone?

$(e_C, d_C)$



Cody

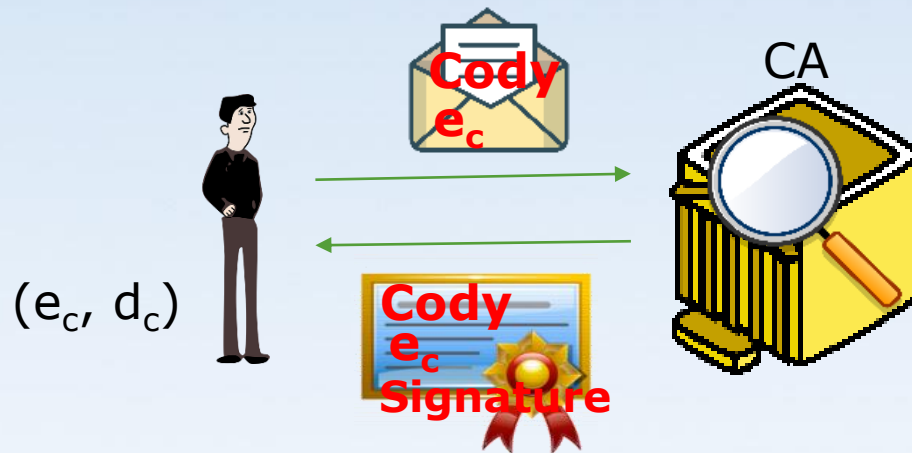
$e_C$





# Obtain a Certificate Issued by a certificate authority (CA)

1. Cody generates his key pair ( $e_c$ ,  $d_c$ )
2. He goes to a CA face-to-face
3. CA verifies his information: email, ID, address, etc.
4. CA issues Cody a certificate, which contains
  - Cody's identity, other information and his public key  $e_c$
  - Signed by CA



# Verify a Certificate

- Cody can get a certificate from a certificate authority (CA)
- Every computer is shipped with the CA's public key in the format of a certificate
- When Cody sends its certificate to a computer, how can the computer verify Cody's public key is genuine?



# Terms for Certificates

---

- The **owner of a certificate** is called a **subject**
- Common name is the identity of the owner
- Subject's *CommonName* can be:
  - An explicit name, e.g. cs.uml, or
  - A name with a wildcard character, e.g. \*.uml.edu



# Windows 10 Certificate Store

- *Type here to search -> certmgr.msc*

certmgr - [Certificates - Current User\Trusted Root Certification Authorities\Certificates]

File Action View Help


Certificates - Current User

- Personal
- Trusted Root Certification Authorities
  - Certificates
- Enterprise Trust
- Intermediate Certification Authorities
- Active Directory User Objects
- Trusted Publishers
- Untrusted Certificates
- Third-Party Root Certificates
- Trusted People
- Client Authentication Issuance
- Other People
- CurrentUser
- Local NonRemovable Certificates
- LocalMachine
- OpenVPN Certificate Store
- Smart Card Trusted Root Certificates

Issued To	Issued By	Expiration Date	Intended Purpose	Friend
Microsoft Root Certificate ...	Microsoft Root Certificate Au...	3/22/2036	<All>	Micr
Microsoft Time Stamp Root ...	Microsoft Time Stamp Root ...	10/22/2039	<All>	Micr
Network Solutions Certifica...	Network Solutions Certificate...	12/31/2030	Client Authentic...	Netv
NO LIABILITY ACCEPTED, (...)	NO LIABILITY ACCEPTED, (c)...	1/7/2004	Time Stamping	Veri!
PA_Root	PA_Root	5/19/2031	<All>	<Nc
QuoVadis Root CA 2	QuoVadis Root CA 2	11/24/2031	Client Authentic...	Quo
QuoVadis Root CA 2 G3	QuoVadis Root CA 2 G3	1/12/2042	Client Authentic...	Quo
QuoVadis Root Certificatio...	QuoVadis Root Certification ...	3/17/2021	Client Authentic...	Quo
SecureTrust CA	SecureTrust CA	12/31/2029	Client Authentic...	Trus
Security Communication R...	Security Communication Roo...	9/30/2023	Client Authentic...	SECI
Security Communication R...	Security Communication Roo...	5/29/2029	Client Authentic...	SECI
Starfield Class 2 Certificatio...	Starfield Class 2 Certification ...	6/29/2034	Client Authentic...	Star
Starfield Root Certificate A...	Starfield Root Certificate Aut...	12/31/2037	Client Authentic...	Star
StartCom Certification Aut...	StartCom Certification Autho...	9/17/2036	Client Authentic...	Star
SwissSign Gold CA - G2	SwissSign Gold CA - G2	10/25/2036	Client Authentic...	Swis
SwissSign Silver CA - G2	SwissSign Silver CA - G2	10/25/2036	Client Authentic...	Swis
Symantec Enterprise Mobil...	Symantec Enterprise Mobile ...	3/14/2032	Code Signing	<Nc
Thawte Premium Server CA	Thawte Premium Server CA	12/31/2020	Code Signing, Se...	thav
thawte Primary Root CA	thawte Primary Root CA	7/16/2036		
thawte Primary Root CA - ...	thawte Primary Root CA - G3	12/1/2037	Client Authentic...	thav
Thawte Timestamping CA	Thawte Timestamping CA	12/31/2020	Time Stamping	Thav
T-TeleSec GlobalRoot Class 2	T-TeleSec GlobalRoot Class 2	10/1/2033	Client Authentic...	T-Te
TWCA Global Root CA	TWCA Global Root CA	12/31/2030	Code Signing, Se...	TWC
TWCA Root Certification A...	TWCA Root Certification Aut...	12/31/2030	Client Authentic...	TWC
TWCA Root Certification A...	TWCA Root Certification Aut...	12/31/2030	Client Authentic...	TWC
USERTrust ECC Certification...	USERTrust ECC Certification ...	1/18/2038	Client Authentic...	Sect
USERTrust RSA Certification...	USERTrust RSA Certification A...	1/18/2038	Client Authentic...	Sect
UTN-USERFirst-Object	UTN-USERFirst-Object	7/9/2019	Encrypting File S...	Sect
VeriSign Class 3 Public Pri...	VeriSign Class 3 Public Prima...	7/16/2036	Client Authentic...	Veri!
VeriSign Universal Root Cer...	VeriSign Universal Root Certif...	12/1/2037	Client Authentic...	Veri!
www.pharos.com	www.pharos.com	2/9/2030	<All>	<Nc

Certificate

General Details Certification Path

 **Certificate Information**

**This certificate is intended for the following purpose(s):**

- Proves your identity to a remote computer
- Ensures software came from software publisher
- Protects software from alteration after publication
- Protects e-mail messages
- Ensures the identity of a remote computer
- All issuance policies

**Issued to:** thawte Primary Root CA

**Issued by:** thawte Primary Root CA

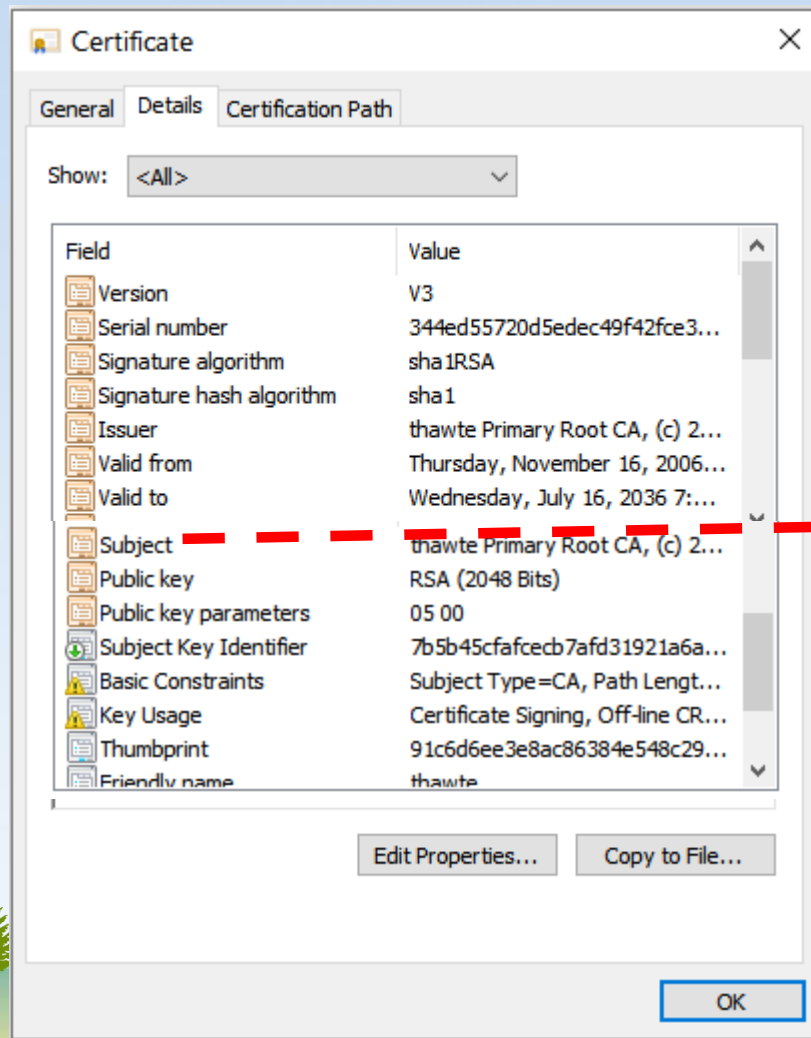
**Valid from** 11/16/2006 **to** 7/16/2036

Issuer Statement

OK

# Example Certificate Details

- Click subject



CN = thawte Primary Root CA  
OU = (c) 2006 thawte, Inc. - For authorized use only  
OU = Certification Services Division  
O = thawte, Inc.  
C = US

# Kali Certificate Store

---

- /etc/ssl/certs
- Command to view a certificate
  - openssl x509 -in certificate-name.pem -text



# Outline

---

- Introduction to digital signature
- Introduction to certificate
- Hands-on labs
  - Windows 10 certificate store
  - Sign and verify a file locally
  - Sign and verify a message through chat server



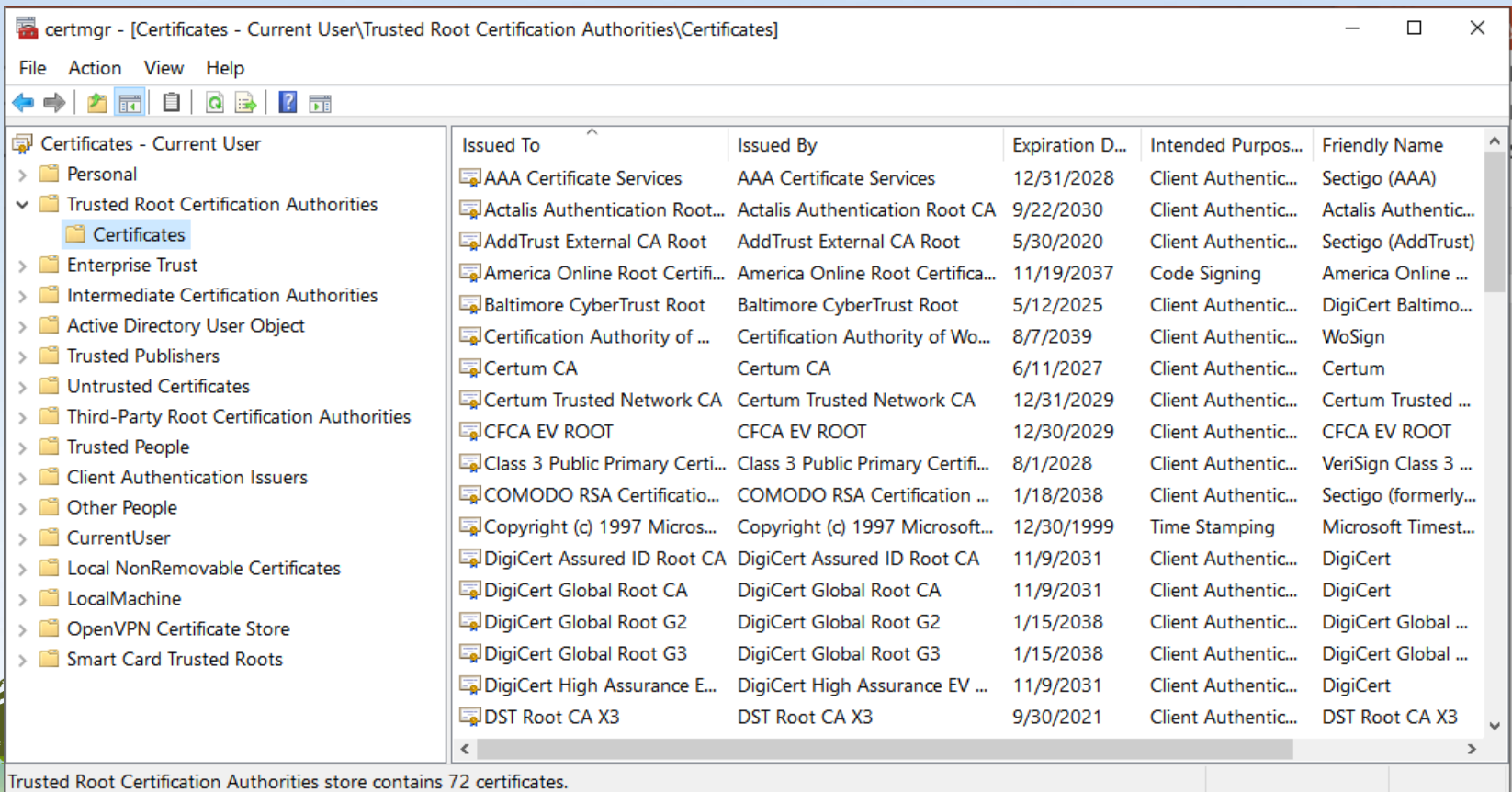
# Certificate Store





# Windows 10 Certificate Store

- TA finds the Windows 10 certificate store
- All people check content of a certificate from Trusted Root Certificate Authorities
- Group discussion: why are these certificates needed?



The screenshot shows the Windows 10 Certificate Store (certmgr) window. The title bar reads "certmgr - [Certificates - Current User\Trusted Root Certification Authorities\Certificates]". The menu bar includes "File", "Action", "View", and "Help". The left pane shows the "Certificates - Current User" tree with "Trusted Root Certification Authorities" expanded, and "Certificates" selected. The right pane displays a list of certificates in the Trusted Root Certification Authorities store. The list has columns for "Issued To", "Issued By", "Expiration D...", "Intended Purpos...", and "Friendly Name".

Issued To	Issued By	Expiration D...	Intended Purpos...	Friendly Name
AAA Certificate Services	AAA Certificate Services	12/31/2028	Client Authentic...	Sectigo (AAA)
Actalis Authentication Root...	Actalis Authentication Root CA	9/22/2030	Client Authentic...	Actalis Authentic...
AddTrust External CA Root	AddTrust External CA Root	5/30/2020	Client Authentic...	Sectigo (AddTrust)
America Online Root Certifi...	America Online Root Certifica...	11/19/2037	Code Signing	America Online ...
Baltimore CyberTrust Root	Baltimore CyberTrust Root	5/12/2025	Client Authentic...	DigiCert Baltimo...
Certification Authority of ...	Certification Authority of Wo...	8/7/2039	Client Authentic...	WoSign
Certum CA	Certum CA	6/11/2027	Client Authentic...	Certum
Certum Trusted Network CA	Certum Trusted Network CA	12/31/2029	Client Authentic...	Certum Trusted ...
CFCA EV ROOT	CFCA EV ROOT	12/30/2029	Client Authentic...	CFCA EV ROOT
Class 3 Public Primary Certi...	Class 3 Public Primary Certifi...	8/1/2028	Client Authentic...	VeriSign Class 3 ...
COMODO RSA Certificatio...	COMODO RSA Certification ...	1/18/2038	Client Authentic...	Sectigo (formerly...
Copyright (c) 1997 Micros...	Copyright (c) 1997 Microsoft...	12/30/1999	Time Stamping	Microsoft Timest...
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA	11/9/2031	Client Authentic...	DigiCert
DigiCert Global Root CA	DigiCert Global Root CA	11/9/2031	Client Authentic...	DigiCert
DigiCert Global Root G2	DigiCert Global Root G2	1/15/2038	Client Authentic...	DigiCert Global ...
DigiCert Global Root G3	DigiCert Global Root G3	1/15/2038	Client Authentic...	DigiCert Global ...
DigiCert High Assurance E...	DigiCert High Assurance EV ...	11/9/2031	Client Authentic...	DigiCert
DST Root CA X3	DST Root CA X3	9/30/2021	Client Authentic...	DST Root CA X3

Trusted Root Certification Authorities store contains 72 certificates.

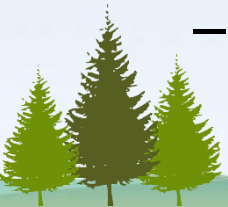
# Digital Signature



# Generate Public and Private Key Pair

---

- If done before, no need
- Generate public and private key pair
  - *openssl genpkey -out privkey.pem -algorithm rsa*
  - Note: privkey.pem can be used as the private key although it contains the public key
- Extract the public key from privkey.pem
  - *openssl rsa -in privkey.pem -outform PEM -pubout -out pubkey.pem*
- Publish your *pubkey.pem*, e.g. to our chat server
  - Never share *privkey.pem*



# Sign a file at Your Computer

---



- Sign a file called changelog or messages saved in a file
  - `openssl dgst -sha256 -sign privkey.pem -out sign.sha256 changelog`
  - The output sign.sha256 is binary
- Encode the binary signature with base64
  - `openssl enc -base64 -in sign.sha256 -out sign.sha256.base64`
  - Not really needed. It is needed here since we can send the base64 encoded message to our chat server
- Send both message and base64 encoded signature to our chat server



# Verify the Signature

---

- Save received signature into a file, e.g., called `sign.sha256.base64`
- Decode `sign.sha256.base64` and get the binary signature
  - *`openssl enc -base64 -d -in sign.sha256.base64 -out sign.sha256`*
- Verify the signature with the public key
  - *`openssl dgst -sha256 -verify pubkey.pem -signature sign.sha256 changelog`*



# Message Signature via Chat Server

---



- Sender generates public and private key pair (no need to do it if already done)
- Sender sends the public key (content of pubkey.pem) to Receiver
  - Receiver saves the public key into a file, e.g. called pubkey.pem
- Sender sends a message to receiver and saves the message into a file called, e.g., msg
  - Receiver saves the received message into a file called, e.g., msg
- Sender signs its msg, generate the signature file and encodes it with base64, e.g. called *sign.sha256.base64*
- Sender sends the content of *sign.sha256.base64* to Receiver
  - Receiver saves it into a file, e.g., called *sign.sha256.base64*
- Receiver decodes *sign.sha256.base64* into the binary signature file and verifies it



# Message under MITM

---

- Assume only the message is changed by MITM
  - That is, hahaha... is added to the received message
- Repeat the same procedure
- Is the verification ok this time?

