CS 370: Introduction to Security

04.27: Digital certificate, Diffie-hellman

Tu/Th 4:00 – 5:50 PM

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

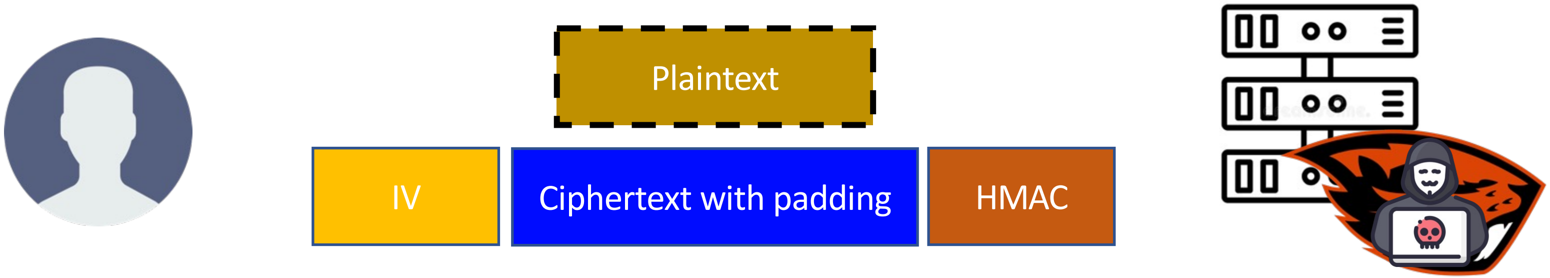
SAIL
Secure AI Systems Lab

Topics for today

- Digital certificate
  - What is it?
  - What problem does it solve?
  - How to create a digital certificate?
  - How does it make the Internet secure?
- Diffie-Hellman
  - What is it?
  - What problem does it solve?
  - What is the weakness of DH?
  - How can we address the weakness?

Digital certificate: motivation

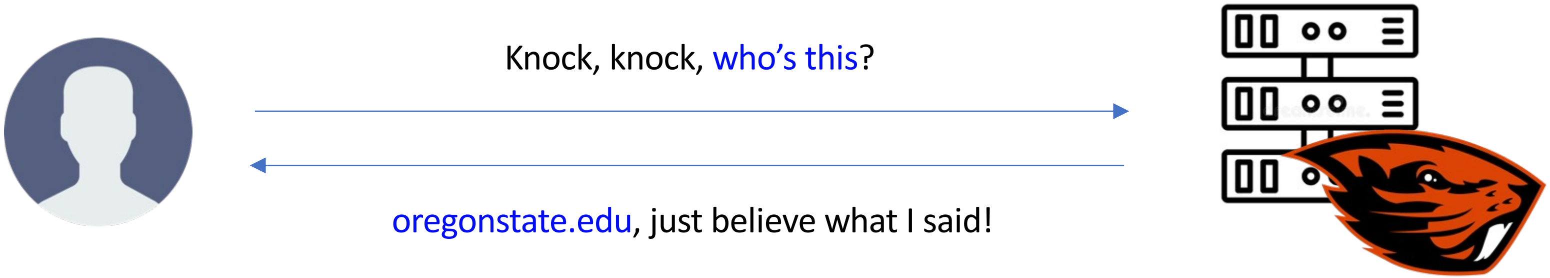
- An example scenario:
  - Suppose the oregonstate.edu server has the public/private key
  - You want to connect to the website securely



  - Confidentiality: comes from the Block Cipher that we will use
  - Integrity: comes from HMAC
- Where's authenticity?
  - How do you know the other end is oregonstate.edu?

How can we check the authenticity?

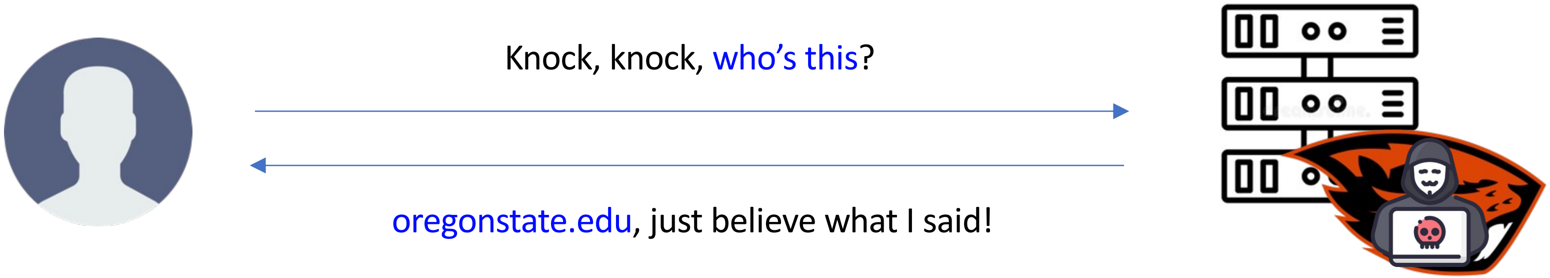
- Can we check the other end is the one that we want to talk with?



We Need Some Ways to Check If They Are OSU (Authenticity)!

How can we check the authenticity?

- Can we check the other end is the one that we want to talk with?



We Need Some Ways to Check If They Are OSU (Authenticity)!

How do we do that in the real-life?

How can we do this for online communication?

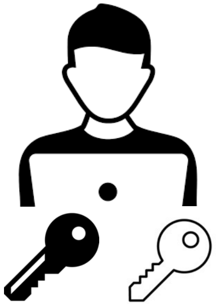
- Intuition
  - Need an identification mechanism
  - Need information that we can use to verify the sender

- Solution
  - Let’s do this with RSA cryptography algorithm
  - Let “oregonstate.edu” publicize the public key
  - Let “oregonstate.edu” share their info. and signed by their private key

Recap: RSA and Digital Signature

- Digital signature
  - A mathematical scheme for verifying the authenticity of digital messages
  - RSA can be used for "signing"

- Encryption and decryption for "signing"
  - Encryption is applying the private exponent to a plaintext: $C = M^d \bmod N$
  - Decryption is applying the public exponent to a ciphertext: $M = C^e \bmod N$



Recap: RSA and Digital Signature

- Digital signature
  - A mathematical scheme for verifying the authenticity of digital messages
  - RSA can be used for "signing"

- Encryption and decryption for "signing"
  - Encryption is applying the private exponent to a plaintext: $C = M^d \bmod N$
  - Decryption is applying the public exponent to a ciphertext: $M = C^e \bmod N$

Recap: RSA and Digital Signature

- Digital signature
  - A mathematical scheme for verifying the authenticity of digital messages
  - RSA can be used for "signing"

- Encryption and decryption for "signing"
  - Encryption is applying the private exponent to a plaintext: $C = M^d \bmod N$
  - Decryption is applying the public exponent to a ciphertext: $M = C^e \bmod N$

Recap: RSA and Digital Signature

- Digital signature
  - A mathematical scheme for verifying the authenticity of digital messages
  - RSA can be used for “signing”
- Encryption and decryption for digital signature
  - Encryption is applying the private exponent to a plaintext: $C = M^d \bmod N$
  - Decryption is applying the public exponent to a ciphertext: $M = C^e \bmod N$

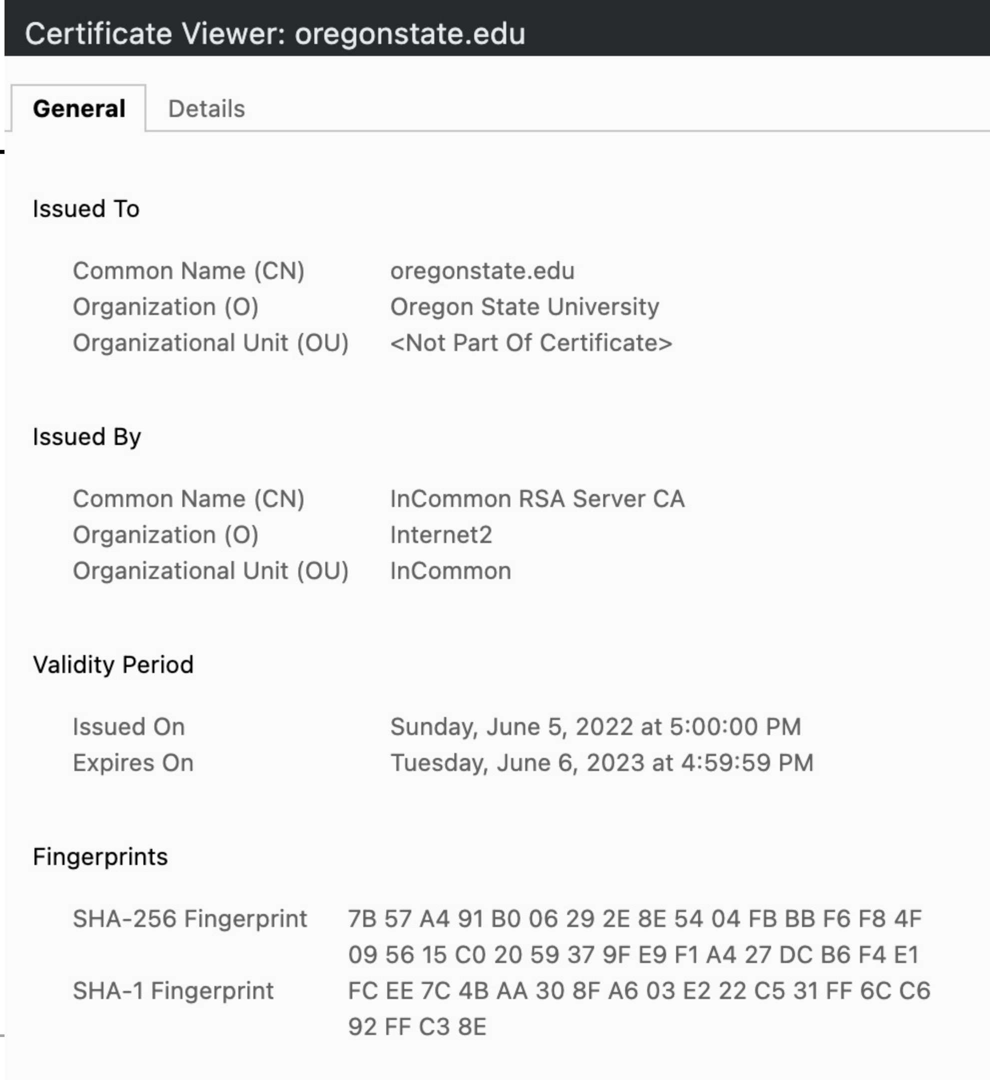
How can we do this for online communication?

- Intuition
  - Need an identification mechanism
  - Need information that we can use to verify the sender

- Solution: Public Key Infrastructure (PKI)
  - Let's do this with RSA cryptography algorithm
  - Let "oregonstate.edu" publicize the public key
  - Let "oregonstate.edu" share their info. and signed by their private key (= we create a digital certificate)

THE INFO: DIGITAL CERTIFICATE

- A file that contains
  - Entity info (CN)
  - Issuer info (CN)
  - Public key
  - Signature



How to create a digital certificate?

- Requester prepares a certificate request
  - Entity information
  - Public key
  - Signature (proving that I have the public key)

Certificate
CN: oregonstate.edu
Will use for:
  *.oregonstate.edu

Public Key: 0x112233445566778899aabbccddeeff….
  (beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
  (using beaver's private key)

How to create a digital certificate?

- Requester prepares a certificate request
  - Entity information
  - Public key
  - Signature (proving that I have the public key)

Get SHA256 sum of this part

Sign it with the private key

Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x112233445566778899aabbccddeeff....
(beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
(using beaver's private key)

How to create a digital certificate?

- Requester prepares a certificate request
  - Entity information
  - Public key

- Issuer verifies the requester information, and digitally sign the cert
  - Verify the entity information
  - Get a SHA-256 fingerprint of the certificate
  - Sign the fingerprint (with issuer’s private key)

```
RSA_encrypt(private_key, SHA-256(certificate))
```

How to create a digital certificate?

- Issuer verifies the requester information, and digitally sign the cert
  - Verify the entity information
  - Get a SHA-256 fingerprint of the certificate
  - Sign the fingerprint (with issuer's private key)

```
RSA_encrypt(private_key, SHA-256(certificate))
```

Get SHA256 sum of this part

Sign it with the private key

Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x112233445566778899aabbccddeeff....
(beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
(using beaver's private key)

How to create a digital certificate?

- Requester prepares a certificate request
  - Entity information
  - Public key

- Issuer verifies the requester information, and digitally sign the cert
  - Verify the entity information
  - Get a SHA-256 fingerprint of the certificate
  - Sign the fingerprint (with issuer’s private key)

```
RSA_encrypt(private_key, SHA-256(certificate))
```

- Anyone with the public key can verify the result
  - Get issuer’s public key from their certificate

Certification creation details: step 1

- The certificate requesting entity fills
  - Entity information
  - Public Key

- Entity:
  - For google, its *.google.com
  - Can be your website address

CN = oregonstate.edu

- *.secure-ai.systems
  - also has a certificate

Certificate
CN: oregonstate.edu
Will use for:
    *.oregonstate.edu

Public Key: 0x112233445566778899aabbccddeeff….
    (beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
    (with beaver's private key)

Certification creation details: step 2

- The issuer receives the certificate request and verifies:
  - Entity
    - Their identification
    - Owning the target domain name
    - Owning the public key
  - The signature
    - Decrypt the signature with public key
    - It must be the same as SHA256 sum
    - It proves their holding the private key

CN = oregonstate.edu

Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x112233445566778899aabbccddeeff….
(beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
(with beaver's private key)

Certification creation details: step 2

- The issuer receives the certificate request and verifies:
  - Entity:
    - Their identification
    - Owning the target domain name
    - etc…
  - Then, fill issuer information
    - Issuer information
    - Issuer public key

InCommon®

CN = oregonstate.edu

Certificate
CN: oregonstate.edu
Will use for:
    *.oregonstate.edu

Public Key: 0x112233445566778899aabbccddeeff….
    (beaver’s public key)

Issuer: InCommon RSA
Public Key: 0x22334455667788990011aabbccddeeff

Certification creation details: step 2

- The issuer receives the certificate request and verifies:
  - Entity:
    - Their identification
    - Owning the target domain name
    - etc…
  - Then, fill issuer information
    - Issuer information
    - Issuer public key
  - and then, sign the certificate
    - Get SHA-256 of the certificate
    - Attach it as a signature!

InCommon®

CN = oregonstate.edu

Certificate
CN: oregonstate.edu
Will use for:
    *.oregonstate.edu

Public Key: 0x112233445566778899aabbccddeeff….
    (beaver's public key)

Issuer: InCommon RSA
Public Key: 0x22334455667788990011aabbccddeeff
Signature: 0xffeeddccbbaa00112233445566778899
    (InCommon RSA's private key)

The certificate issued

- Now InCommon RSA verified
  - oregonstate.edu is owned by
  - Oregon State University
  - With a specific Public Key

Subject Public Key Info

Subject Public Key Algorithm

Subject's Public Key

Field Value

Modulus (2048 bits):
C8 7D 2D A8 EB 12 59 6B 90 6D 4F 71 1E 4C FA C2
F7 A1 EC F6 E6 0E 39 52 FF 69 C0 36 CD A9 74 6E
60 72 C8 34 AF CC F7 6F 8E 66 D0 C5 0D E9 9C 66
F0 B2 D1 D8 75 A7 B9 82 E5 E8 C3 3F 13 35 1E 1E
71 F1 92 B4 40 07 EA 27 BE F9 9B AF E8 D2 E3 71

Certificate Viewer: oregonstate.edu

General | Details

Issued To

Common Name (CN)	oregonstate.edu
Organization (O)	Oregon State University
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	InCommon RSA Server CA
Organization (O)	Internet2
Organizational Unit (OU)	InCommon

Validity Period

Issued On	Sunday, June 5, 2022 at 5:00:00 PM
Expires On	Tuesday, June 6, 2023 at 4:59:59 PM

Fingerprints

SHA-256 Fingerprint	7B 57 A4 91 B0 06 29 2E 8E 54 04 FB BB F6 F8 4F 09 56 15 C0 20 59 37 9F E9 F1 A4 27 DC B6 F4 E1
SHA-1 Fingerprint	FC EE 7C 4B AA 30 8F A6 03 E2 22 C5 31 FF 6C C6 92 FF C3 8E

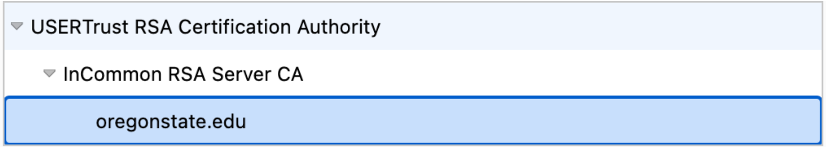
Recap: OSU certificate

- OSU owns “oregonstate.edu”
  - Verified by InCommon RSA

- Verification of the certificate
  - Use InCommon RSA’s public key
  - Where is it? It is written in InCommon RSA’s certificate

- But InCommon RSA, who will verify their identity?
  - InCommon RSA verifies “oregonstate.edu”
  - Who will verify InCommon RSA?

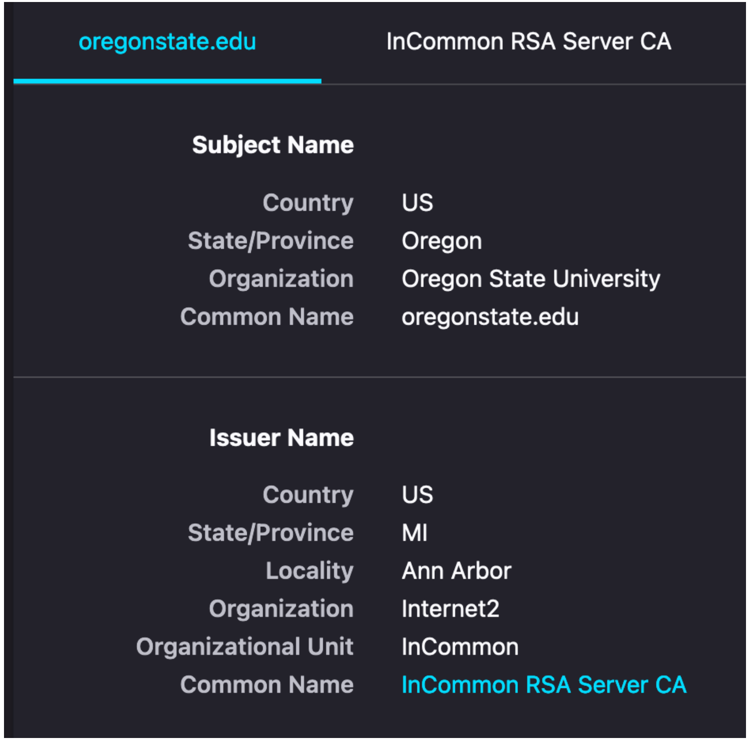
Let's see it from the browser



- "oregonstate.edu"
  - Verified by InCommon RSA Server CA
- InCommon RSA Server CA
  - Verified by USERTrust RSA Certificate Authority
- USERTrust RSA CA
  - Verified by self

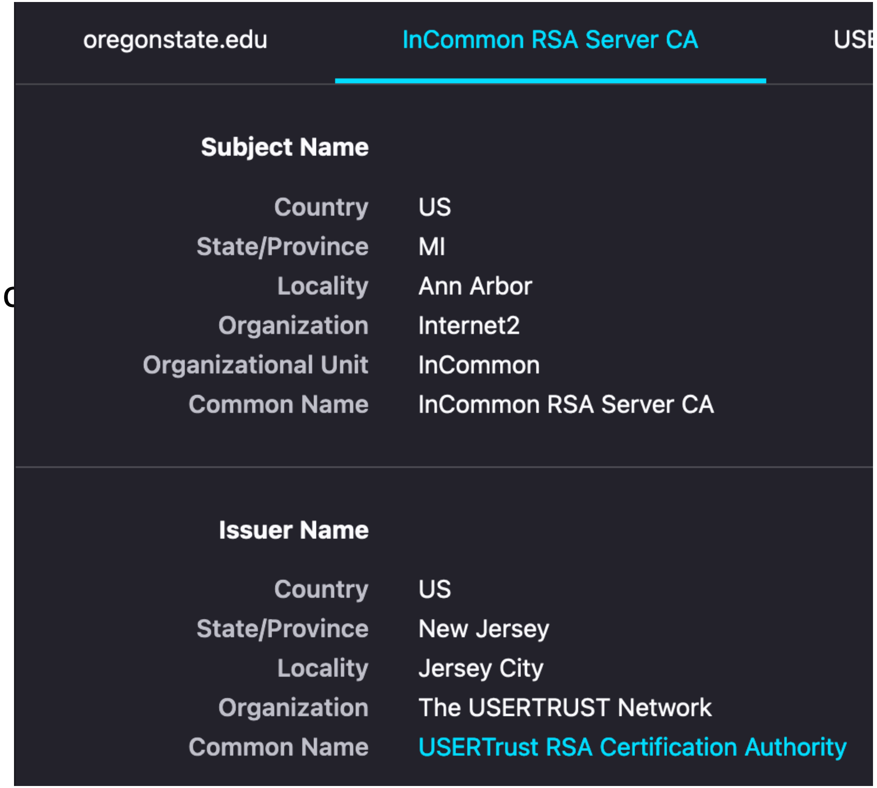
Trust chain

- "oregonstate.edu"
  - Verified by InCommon RSA Server CA
- InCommon RSA Server CA
  - Verified by USERTrust RSA Certificate Authority
- USERTrust RSA CA
  - Verified by self



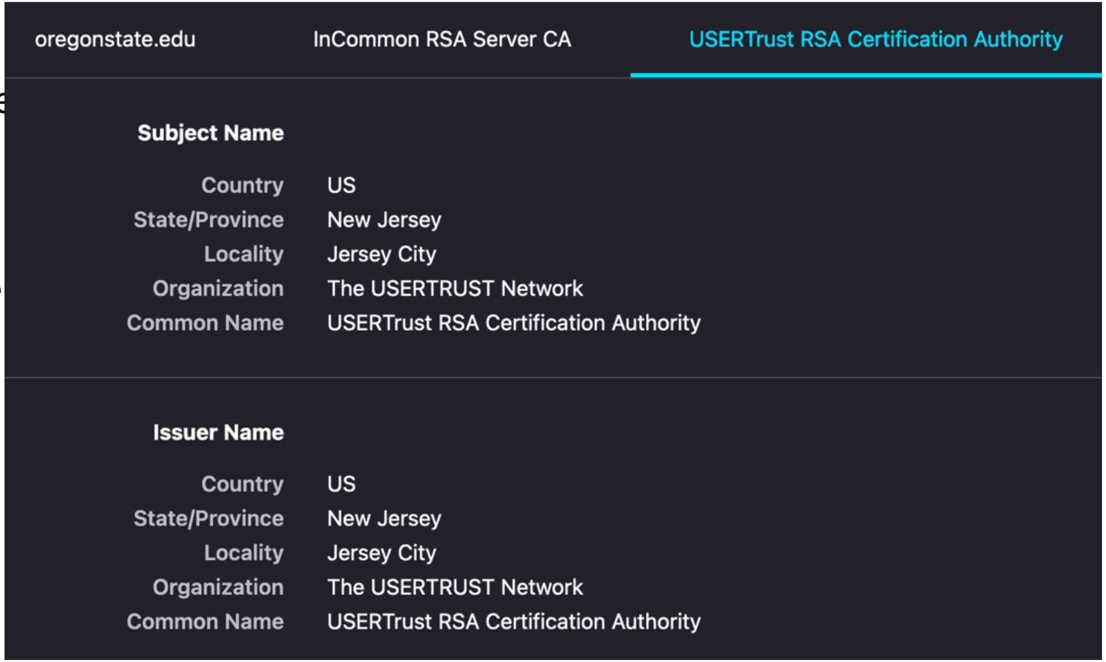
Trust chain – cont'd

- "oregonstate.edu"
  - Verified by InCommon RSA Server CA
- InCommon RSA Server CA
  - Verified by USERTrust RSA Certificate Autho
- USERTrust RSA CA
  - Verified by self



Trust chain – cont'd

- "oregonstate.edu"
  - Verified by InCommon RSA Se
- InCommon RSA Server CA
  - Verified by USERTrust RSA Ce
- USERTrust RSA CA
  - Verified by self



Trust chain in real-life

- An example:
  - Student
  - Oregon resident
  - U.S. Citizen

- When issuing the student ID
  - We verify your Oregon ID…

Trust chain in real-life

- An example:
  - Student
  - Oregon resident
  - U.S. Citizen

- When issuing the student ID
  - Verify your Oregon ID…

- When issuing the Oregon Driver's License
  - Require either one of your birth certificate, previous Driver's License, or U.S. passport

Trust chain in real-life

- An example:
  - Student
  - Oregon resident
  - U.S. Citizen

- When issuing the student ID
  - Verify your Oregon ID…

- When issuing the Oregon Driver's License
  - Require either one of your birth certificate, previous Driver's License, or U.S. passport

- When issuing the U.S. passport
  - Require your birth certificate or previously issued passport..

Trust chain in real-life

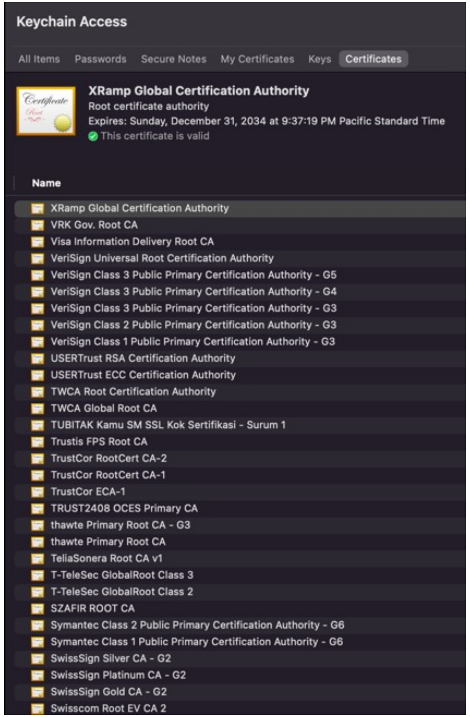
- An example:
  - Student
  - Oregon resident
  - U.S. Citizen

We need someone to verify the originality of the proving document…

- When issuing the student ID
  - Verify your Oregon ID…
- When issuing the Oregon Driver's License
  - Require either one of your birth certificate, previous Driver's License, or U.S. passport
- When issuing the U.S. passport
  - Require your birth certificate or previously issued passport..

Root certificate authority (Root CA ≈ US in prev. example)

- Define small set of trustworthy certificate authorities
  - Private companies are authorized by some jurisdiction to run the CA company
    - Google Trust Service (GTS CA)
    - DigiCert
    - Verisign
    - etc..
- Trust their self-signed certificate
  - Stored in almost every computer machines

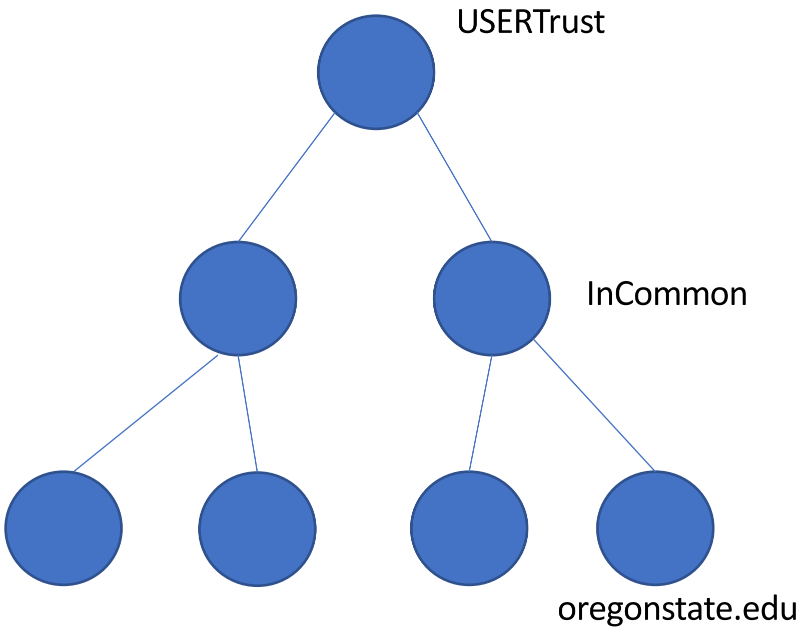


Public key infrastructure (PKI)

- An Infrastructure that provides public key with certificate chain

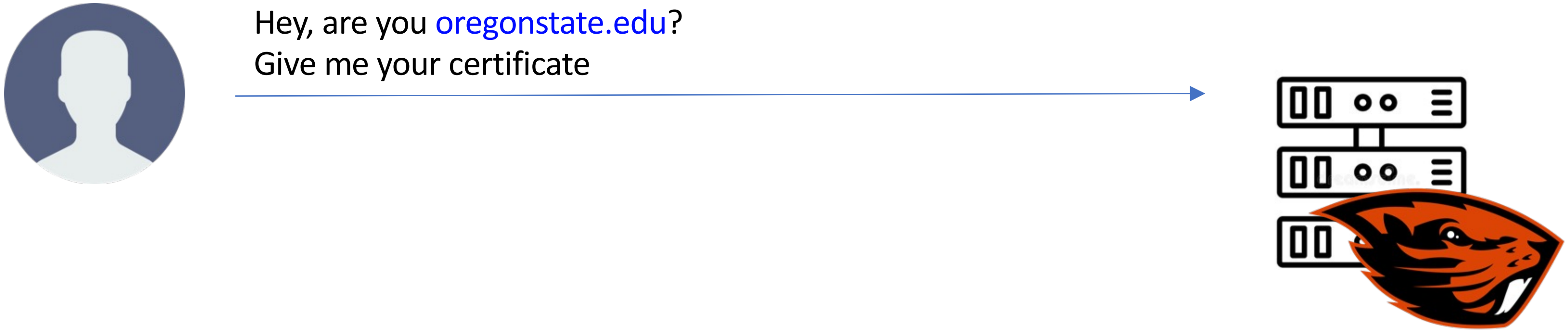
- Trust anchor: Root CA
  - Set a small set of entities use self-signed cert

- Verify the certificate chain!
  - Must verify the entire chain



Let's verify oregonstate.edu

- Using the digital certificate!

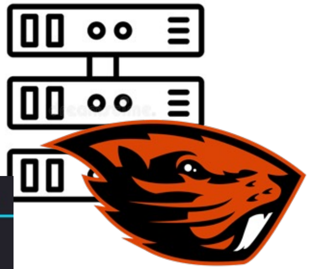


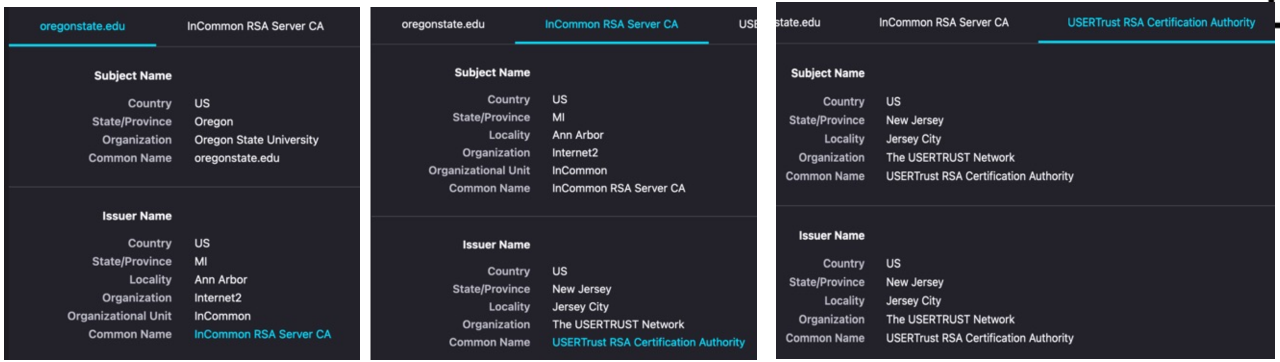
Let's verify oregonstate.edu

- Using the digital certificate!

Hey, are you oregonstate.edu?
Give me your certificate

Yes, I am oregonstate.edu!
Here's my cert (certificate)



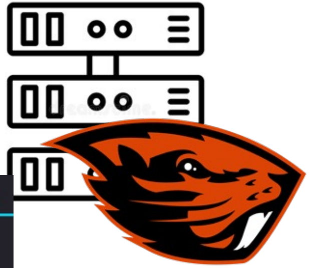


Let's verify oregonstate.edu

- Using the digital certificate!

Hey, are you oregonstate.edu?
Give me your certificate

Yes, I am oregonstate.edu!
Here's my cert



oregonstate.edu | InCommon RSA Server CA

Subject Name	
Country	US
State/Province	Oregon
Organization	Oregon State University
Common Name	oregonstate.edu

Issuer Name	
Country	US
State/Province	MI
Locality	Ann Arbor
Organization	Internet2
Organizational Unit	InCommon
Common Name	InCommon RSA Server CA

oregonstate.edu | InCommon RSA Server CA | USE

Subject Name	
Country	US
State/Province	MI
Locality	Ann Arbor
Organization	Internet2
Organizational Unit	InCommon
Common Name	InCommon RSA Server CA

Issuer Name	
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

state.edu | InCommon RSA Server CA | USERTrust RSA Certification Authority

Subject Name	
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

Issuer Name	
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

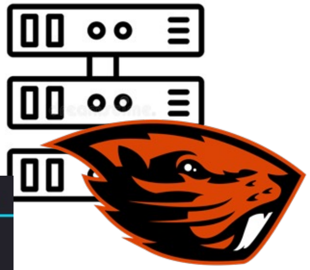
Oregonstate verified by InCommon RSA

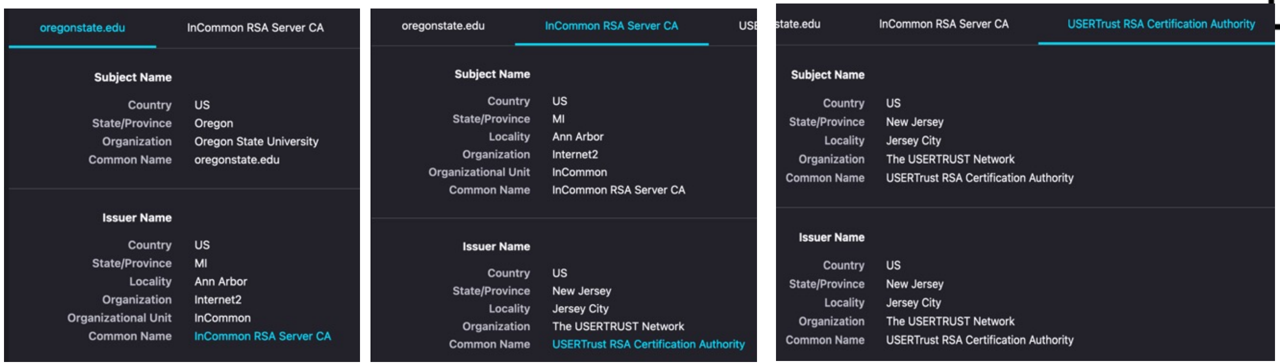
Let's verify oregonstate.edu

- Using the digital certificate!

Hey, are you oregonstate.edu?
Give me your certificate

Yes, I am oregonstate.edu!
Here's my cert





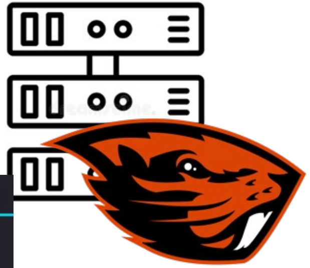
InCommon RSA verified by USERTrust RSA

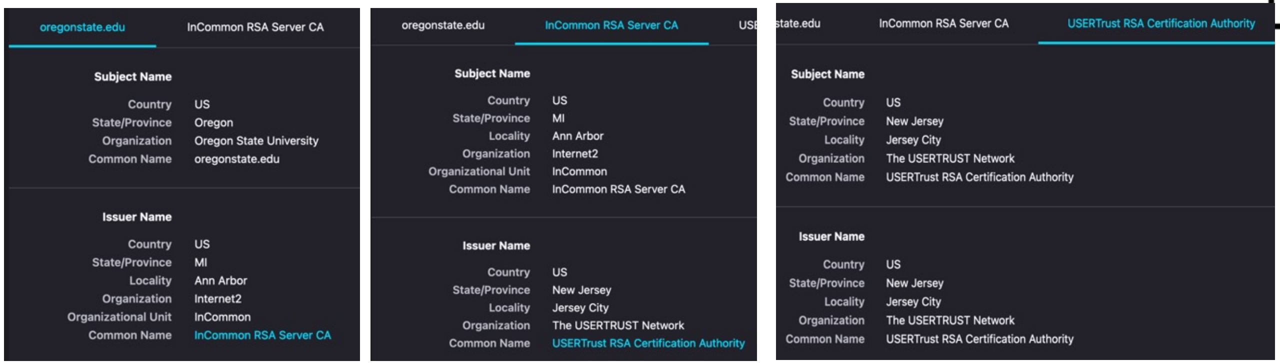
LET'S VERIFY OREGONSTATE.EDU

- Using the digital certificate!

Hey, are you oregonstate.edu?
Give me your certificate

Yes, I am oregonstate.edu!
Here's my cert





oregonstate.edu | InCommon RSA Server CA

Subject Name	
Country	US
State/Province	Oregon
Organization	Oregon State University
Common Name	oregonstate.edu

Issuer Name	
Country	US
State/Province	MI
Locality	Ann Arbor
Organization	Internet2
Organizational Unit	InCommon
Common Name	InCommon RSA Server CA

oregonstate.edu | **InCommon RSA Server CA** | USE

Subject Name	
Country	US
State/Province	MI
Locality	Ann Arbor
Organization	Internet2
Organizational Unit	InCommon
Common Name	InCommon RSA Server CA

Issuer Name	
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

state.edu | InCommon RSA Server CA | **USERTrust RSA Certification Authority**

Subject Name	
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

Issuer Name	
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

USERTrust RSA is self-verified (ROOT CA)

Topics for today

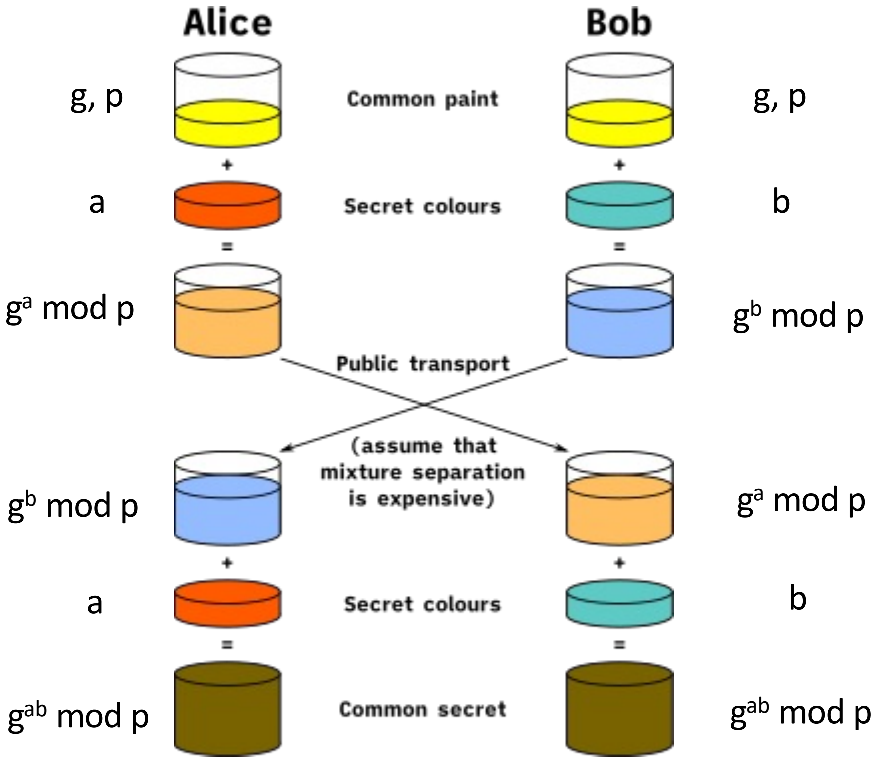
- Digital certificate
  - What is it?
  - What problem does it solve?
  - How to create a digital certificate?
  - How does it make the Internet secure?
- Diffie-Hellman
  - What is it?
  - What problem does it solve?
  - What is the weakness of DH?
  - How can we address the weakness?

Diffie-Hellman key exchange

- Diffie-Hellman
  - A method of securely exchanging cryptographic keys over a public channel
  - Two parties can establish a shared secret (private) key over an insecure channel

- Security:
  - Based on the difficulty of mathematical problem of discrete logarithm

Diffie-Hellman key exchange in graphics



Diffie-Hellman key exchange

- Diffie-Hellman
  - A method of securely exchanging cryptographic keys over a public channel
  - Two parties can establish a shared secret (private) key over an insecure channel

- Security:
  - Based on the difficulty of mathematical problem of discrete logarithm
  - Example:
    - Given g, a, b, A, B, where
    - $g^a \bmod p = A$
    - $g^b \bmod p = B$
    - Can you compute $g^{ab} \bmod p$?

Diffie-Hellman key exchange

- User A & User B agrees on g and p, where g and p are primes

- User A secretly chooses a, send $A = g^a \bmod p$
- User B secretly chooses b, send $B = g^b \bmod p$

- User A receives B, compute $B^a = (g^b)^a \bmod p = g^{ab} \bmod p$
- User B receives A, compute $A^b = (g^a)^b \bmod p = g^{ab} \bmod p$

- $g^{ab} \bmod p$ is our secret

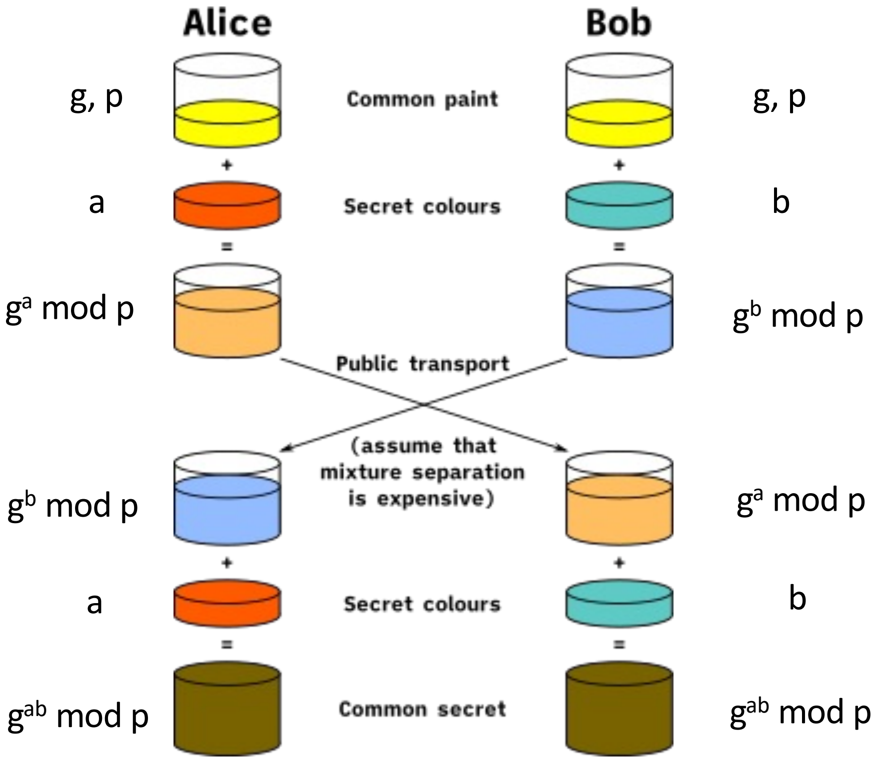
Diffie-Hellman key exchange

- g^{ab} mod p is our secret

- Suppose:
  - Attacker knows g, p, $A = g^a$ mod p and $B = g^b$ mod p
  - $A+B = (g^a + g^b)$ mod p
  - $AB = g^{(a+b)}$ mod p

- Security:
  - Hard to compute g^{ab} from those values
  - Discrete logarithm; can you guess a from $A = g^a$ mod p

Diffie-Hellman key exchange in graphics



Diffie-Hellman key exchange example

- $g = 5, p = 23$

- A chooses $a = 4$
  - $A = 5^4 \bmod 23 = 625 \bmod 23 = 4$
- B chooses $b = 3$
  - $B = 5^3 \bmod 23 = 125 \bmod 23 = 10$

- $B^4 = 10^4 \bmod 23 = 10000 \bmod 23 = 18$
- $A^3 = 4^3 \bmod 23 = 64 \bmod 23 = 18$
- $5^{(4*3)} = 5^{12} \bmod 23 = 18$

Diffie-Hellman key exchange: implications

- Users are agreeing on two prime numbers
  - g, p

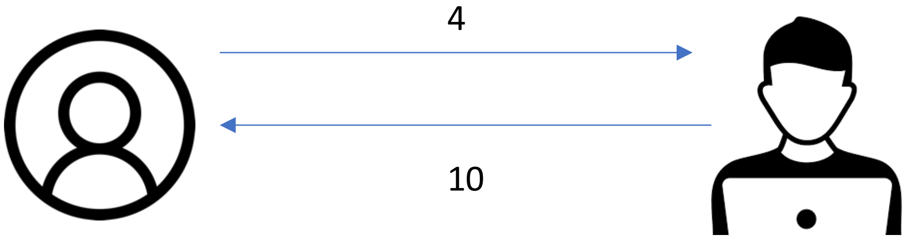
- User A chooses any integer a, nobody knows it
- User B chooses any integer b, nobody knows it

- By sharing g^a mod P and g^b mod p
  - Both shares g^{ab} mod P without leaking a nor b

Two entities can interactively share a secret without directly leaking the secrets to others

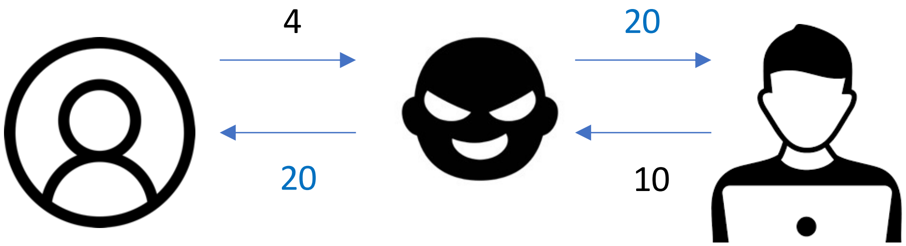
Diffie-Hellman weakness: man-in-the-middle

- Suppose A and B wants to share a secret
  - g = 5, p = 23
  - A chooses a = 4
    - A = 5^4 mod 23 = 625 mod 23 = 4
  - B chooses b = 3
    - B = 5^3 mod 23 = 125 mod 23 = 10



Diffie-Hellman weakness: man-in-the-middle

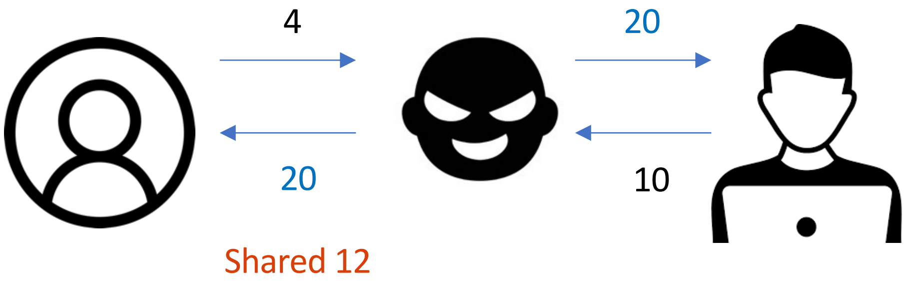
- Suppose C intercepts communication between A and B
  - A chooses a = 4
    - A = 5^4 mod 23 = 625 mod 23 = 4
  - B chooses b = 3
    - B = 5^3 mod 23 = 125 mod 23 = 10
  - C chooses c = 5
    - C = 55 mod 23 = 3125 mod 23 = 20



- C sends 20 to both A and B

Diffie-Hellman weakness: man-in-the-middle

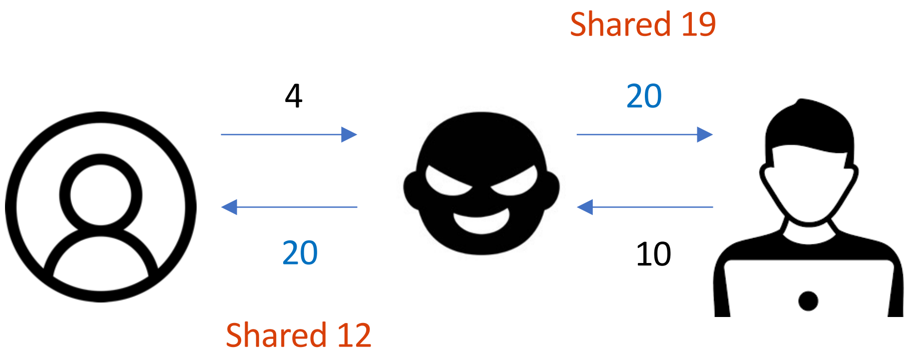
- A chooses a = 4
  - $A = 5^4 \bmod 23 = 625 \bmod 23 = 4$
  - $C^a = 20^4 \bmod 23 = 160000 \bmod 23 = 12$
- C chooses c = 5
  - $C = 5^5 \bmod 23 = 3125 \bmod 23 = 20$
  - $A^c = 4^5 \bmod 23 = 1024 \bmod 23 = 12$



- C shares a secret of 12 with A

Diffie-hellman weakness: man-in-the-middle

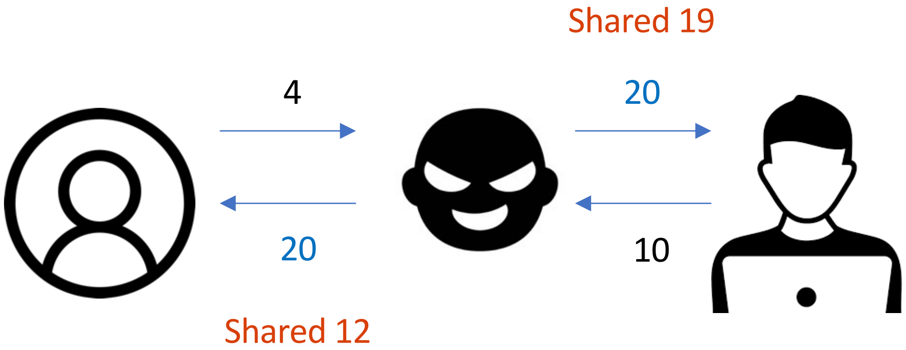
- B chooses b = 3
  - $B = 5^3 \bmod 23 = 125 \bmod 23 = 10$
  - $C^b = 20^3 \bmod 23 = 8000 \bmod 23 = 19$
- C chooses c = 5
  - $C = 5^5 \bmod 23 = 3125 \bmod 23 = 20$
  - $B^c = 10^5 \bmod 23 = 100000 \bmod 23 = 19$
- C shares a secret of 19 with B



Diffie-Hellman weakness: man-in-the-middle

- Whenever A sends a message
  - Decrypt with 12, read it!
  - Encrypt with 19, send to B!

- Whenever B sends a message
  - Decrypt with 19, read it!
  - Encrypt with 12, send to A!



Diffie-Hellman is susceptible to the Man-in-the-Middle (MITM) attack!

Summary: secure internet communication

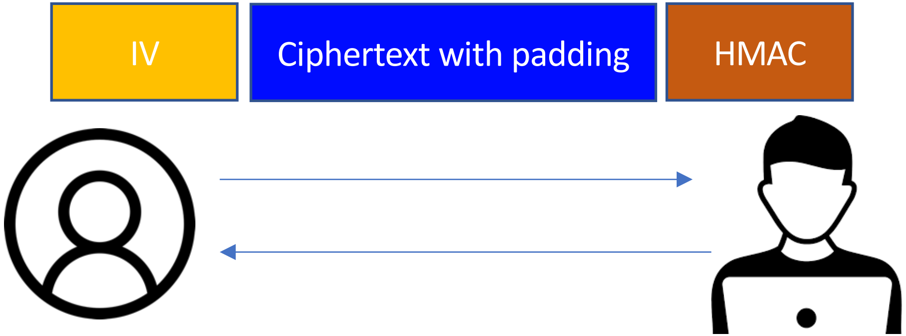
- Authentication
  - Get the certificate of each entity
  - Verify their public key
  - Using certificate trust chain!

- Key-exchange
  - A computes g^a mod p, and sign that with A's private key
  - B computes g^b mod p, and sign that with B's private key
  - Both can verify the identity of each and then share
    - g^{ab} mod p

We Can Defeat MITM

Summary: secure internet communication

- Confidentiality
  - Run SHA-256('cipher key' + g^{ab} mod p)
  - Use that as the key for the block cipher
  - e.g., AES-256-CBC



- Integrity
  - Run SHA-256('mac key' + g^{ab} mod p)
  - Use that as the key for HMAC

A Communication Channel with
Authenticity, Confidentiality, and Integrity
Has Been Established :)

Micro-labs (week 4)

- raw-rsa
- raw-dh

Thank You!

Tu/Th 4:00 – 5:50 PM

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab