CS 370
Introduction to Security

Digital Certificates, PKI, and Diffie-Hellman Key Exchange

Yeongjin Jang

We Will have Asynchronous Lecture on 10/13 and 10/18

- We will have Quiz 1 on 10/20
- We will have Quiz Prep video on 10/18
- We will have video (asynchronous, no in-person/synchronous lecture)
  - On 10/13
  - SSL/TLS

Recap: Block Cipher

- The block cipher itself cannot protect encrypted data modified by attackers
  - ECB, we can substitute blocks to known-plaintext-encrypted-block
  - CBC, we can apply XOR to the ciphertext that is one-block before the plaintext
  - CTR, we can apply XOR to the ciphertext then the result will show on the plaintext

- Why?
  - Block Cipher gives us data confidentiality
  - Not data integrity

Recap: Create a MAC (Message Authentication Code)

- What I will do:
  - f(“secret-key” + encrypted_data) = message_authentication_code
  - MAC = f(“secret-key” + | IV | Block 0 | Block 1 |)

Put this at the end

IV	Block 0	Block 1	MAC

Recap: Checking a MAC

- When I read the encrypted data

IV	Block 0	Block 1	MAC

- MAC = f(“secret-key” + | IV | Block 0 | Block 1 |)

- MAC should be equal to | MAC |

Recap: What Attackers Can Do?

- What if they edited data?

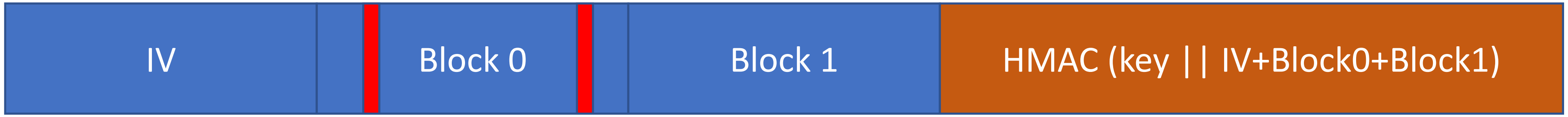
IV	Block 0	Block 1	MAC

- MAC’ = f(“secret-key” + IV Block 0 Block 1)
- MAC’ != MAC

- Suppose we have a function f(x) that
  - A slight value change in x for f(x) = y will result in drastic change in y

Recap: Decrypt Data with CBC

- Suppose you have a hash key = ‘asdf’
  - HMAC = SHA256(SHA256(‘asdf’) || encrypted_data)
  - = `7624e1f89ce009f8ec7e6e39781a42c0a27fa38f94db4f05f78b0f301007e06a`
- Suppose the attacker changed the encrypted_data



  - HMAC = SHA256(SHA256(‘asdf’) || **encrypted_data**)
  - = `389205904d6c7bb83fc676513911226f2be25bf1465616bb9b29587100ab1414`
- Mismatch with HMAC!

Recap: HMAC

- Block Cipher modes lets attacker play with ciphertext freely
  - They cannot be secure as we proved in challenges
- That's because Block Cipher protects only the data confidentiality
- Data Integrity left unprotected
- To protect data integrity, we can use cryptographic hash function
  - One way, it is hard to find an inverse of the result…
- HMAC, running cryptographic hash function with key on the data can protect data integrity…

Recap: RSA Summary

- Public/Private key Scheme
  - We can publish the public key – encryption key
  - We must hide the private key – decryption key
- Based on the difficulty of prime factorization
  - You cannot correlate the private key from the public key unless
  - You can factor a big number (a multiple of 2 big prime numbers)
- Anyone can encrypt message to the private key owner
  - Enc(public_key, message)
- Only the private key owner can decrypt message
  - Dec(private_key, encrypted_message)

Recap: RSA Summary

- Encryption with private key could be a ‘digital-signature’
  - Signed_message = Enc(private_key, message)
  - Message = Dec(public_key, signed_message)
- The correctly decrypted message using public key means that the private key holder have endorsed (‘encrypted’) the data
  - Anyone can verify this using the public key

Topic for Today

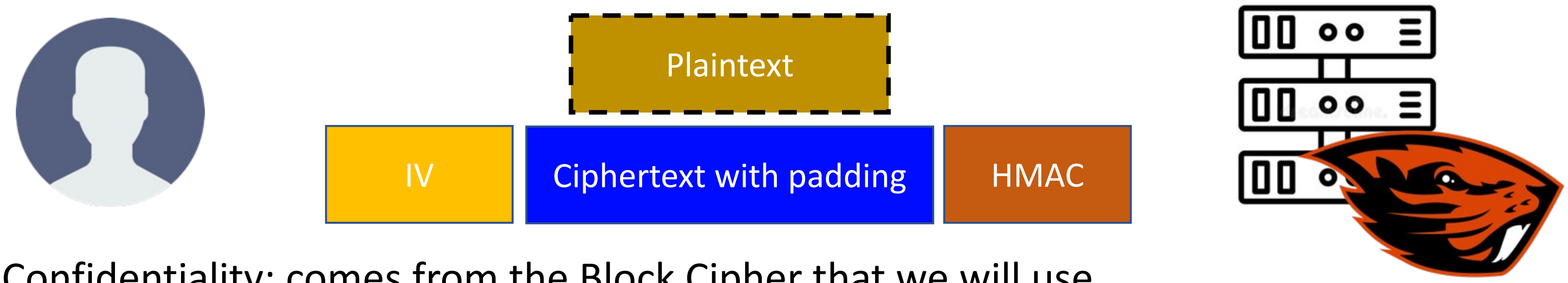
- Digital Certificate
  - Use RSA as a digital signature algorithm
  - Proves the authenticity of the data (authentication)

- Diffie-Hellman (DH) Key exchange algorithm
  - An algorithm to securely exchange a shared secret
  - The shared secret can later be used as a key for block cipher
  - i.e., key exchange using DH, and then, use AES-256 with the shared key

- How can we make the Internet communication secure?
  - Digital Signature (authenticity)
  - Key Exchange
  - Block Cipher (confidentiality)
  - HMAC (integrity)

Digital Certificate

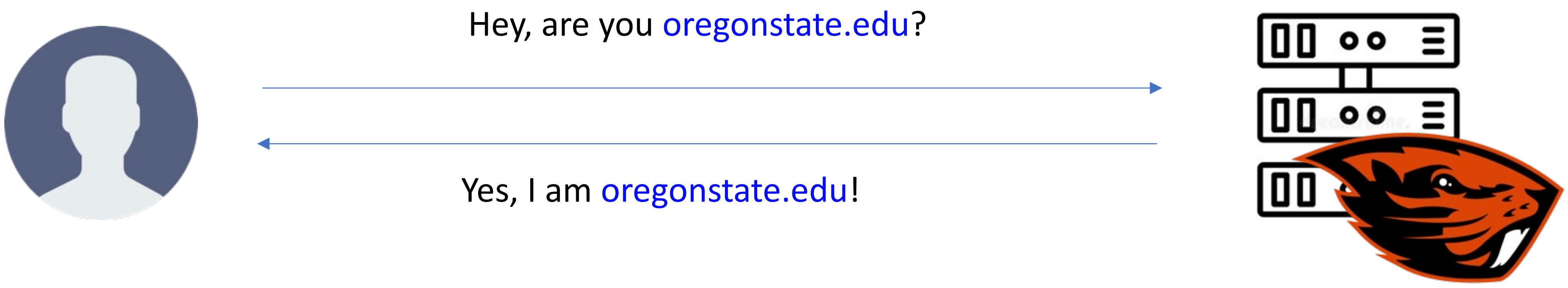
- The need
  - Suppose the oregonstate.edu server has the public/private key
  - You want to connect to the website securely



  - 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?

Authenticity

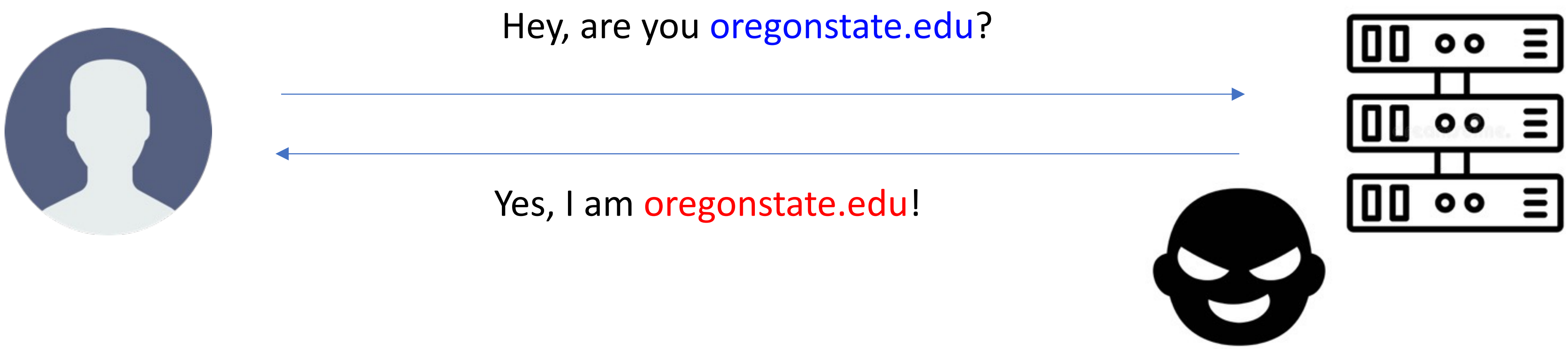
- Can we check the other end is the one that we want to talk with?



Is there any way that we can reliably check the other end's authenticity?

Authenticity

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



Is there any way that we can reliably check the other end's authenticity?

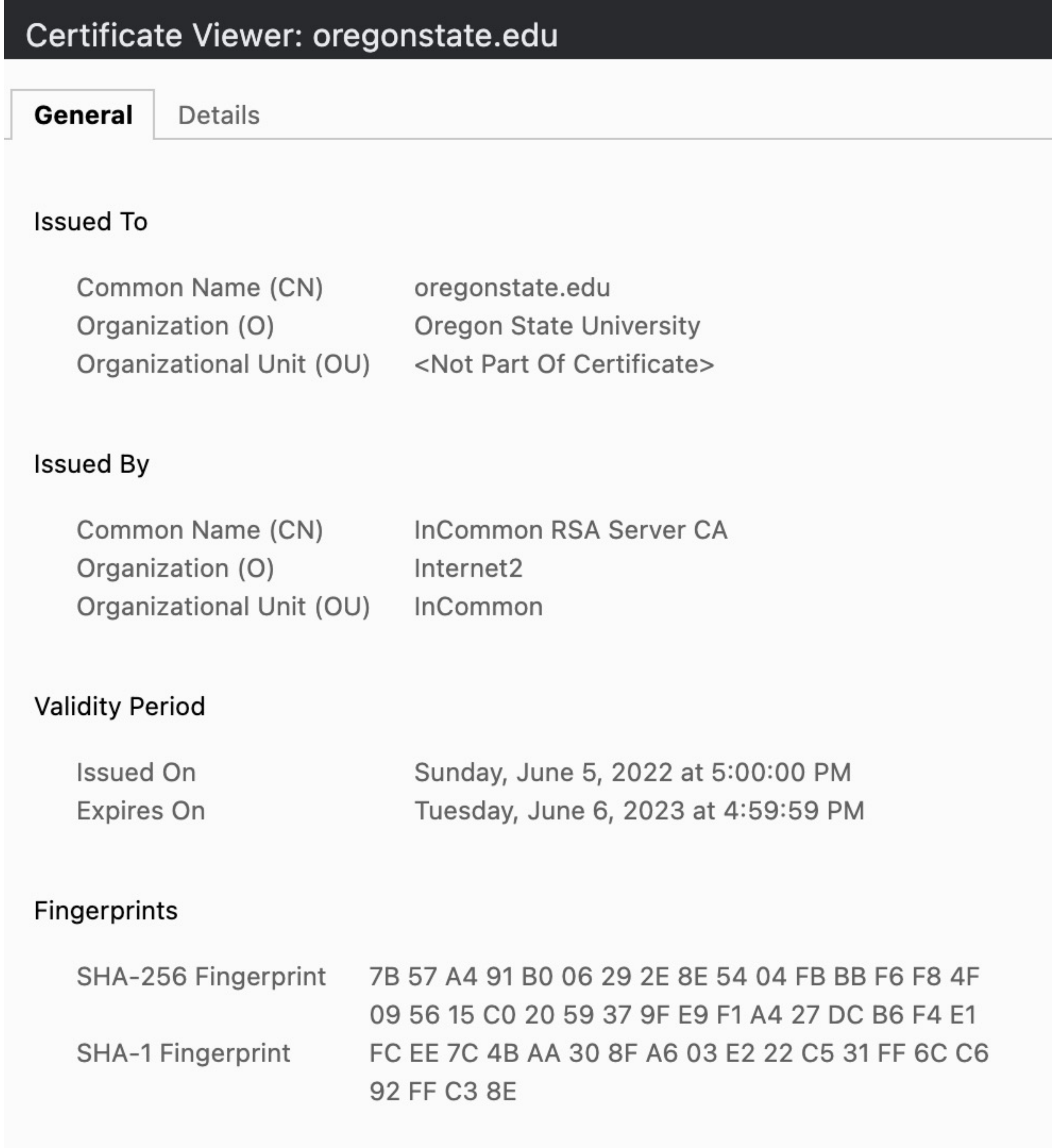

OREGON
DRIVER LICENSE
USA
4d NO A123456
1,2
8
www.oregonstate.edu
0x83823787832a87b
876e67fe67e6da
4b EXP 12/12/2026
4A ISS 03/16/2018
10 FIRST 03/16/2018
5 DD ZA0000089
9 CLASS C
9a END M
12 REST BD
15 SEX M
16 HGT 6’-02”
17 WGT 250 lb
18 EYES BRO
Signature
3 DOB 12/12/1979
VETERAN

Public Key Infrastructure (PKI)

- We need an identification method for the key and the real entity
  - We need an online ID card for crypto keys…
- With RSA, we can use public key cryptosystem
  - We can announce the public key
- Let anyone can access and verify it
- How?
  - Where can we publish this and verify it?
  - PKI resolves the problem…

Digital Certificate

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



Creating a Digital Certificate

- 1. 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)

Creating a Digital Certificate

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

Certificate

Get SHA256 sum of this part

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

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

Sign it with the private key

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

Creating a Digital Certificate

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

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

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

Creating a Digital Certificate

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

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

Certificate

Get SHA256 sum of this part

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

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

Sign it with the private key

Signature: 0xffeeddccbbaa00112233445566778899
(with Issuer's private key)

Creating a Digital Certificate

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

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

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

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

Digital Certificate Creation Step 1

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

CN = oregonstate.edu

- Entity can be anyone
  - For google, its *.google.com
  - Can be your website address
- *.unexploitable.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)

Digital Certificate Creation Step 2

- The issuer receives the certificate request
- Verifies the entity for
  - Their identification
  - Owning the target domain name
  - Owning the public key
- Verify the signature
  - Decrypt the signature with public key
  - It must be the same as SHA256 sum
- Verification proves holding of 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)

Digital Certificate Creation Step 2

- The issuer receives the certificate request
- Verifies the entity for
  - Their identification
  - Owning the target domain name
  - etc…
- Then, fill issuer information
  - Issuer information
  - Issuer public key

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

Digital Certificate Creation Step 2

- The issuer receives the certificate request
- Verifies the entity for
  - 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 fingerprint of the certificate
  - Attach it as a signature!

CN = oregonstate.edu

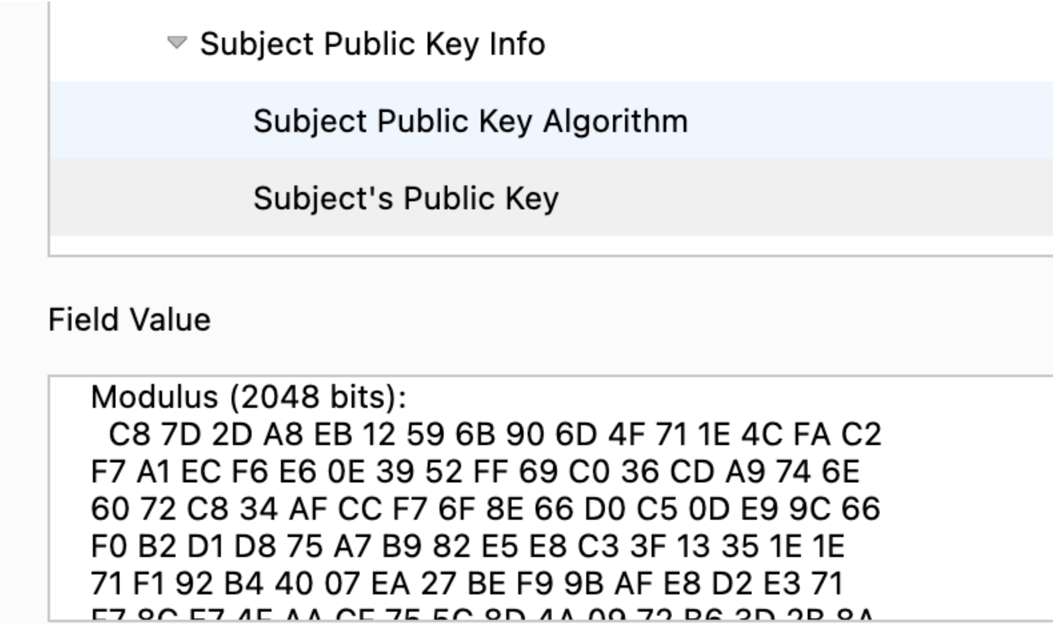
Certificate
CN: oregonstate.edu
Will use for:
        *.oregonstate.edu

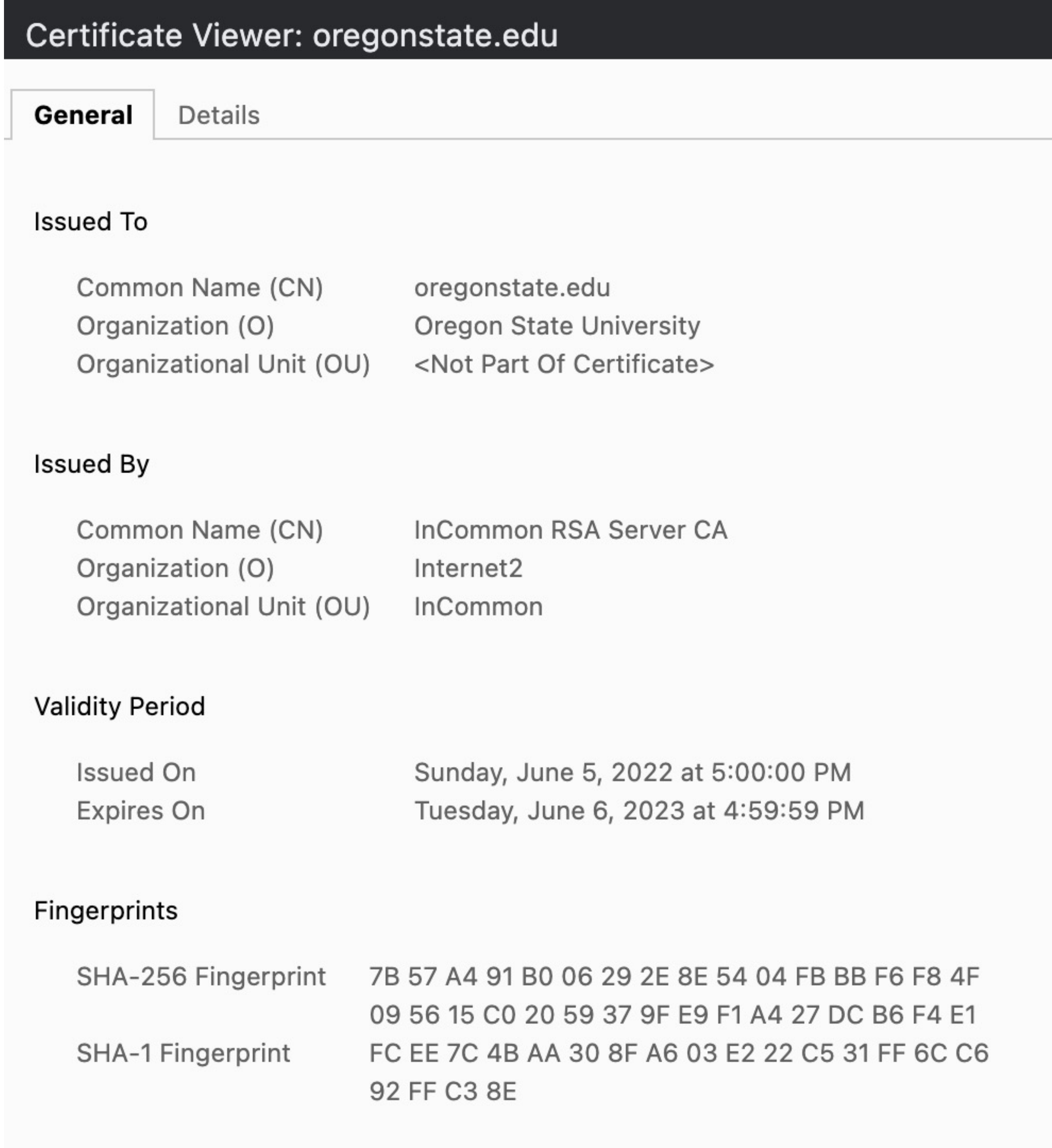
Public Key: 0x112233445566778899aabbccddeeff….
        (beaver's public key)

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

Issued Certificate

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





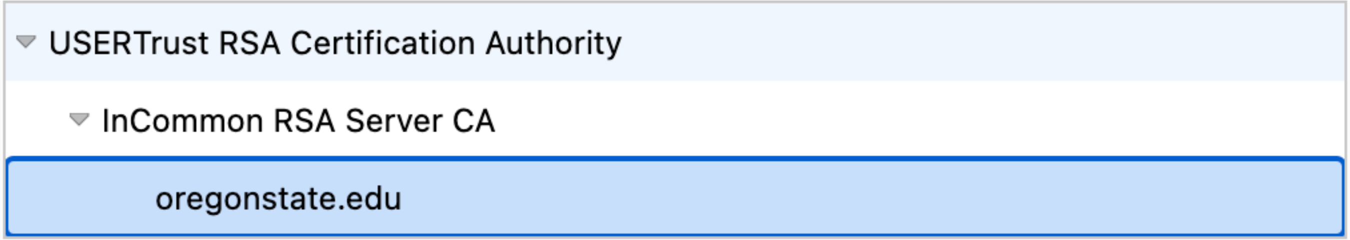
Trust Chain

- oregonstate.edu is owned by Oregon State University
  - Verified by InCommon RSA

- We can verify the certificate using InCommon RSA’s public key
  - Where is it? It is written in InCommon RSA’s certificate

- InCommon RSA, who will verify their identity?
  - oregonstate.edu was verified by InCommon RSA
  - Who will verify InCommon RSA?

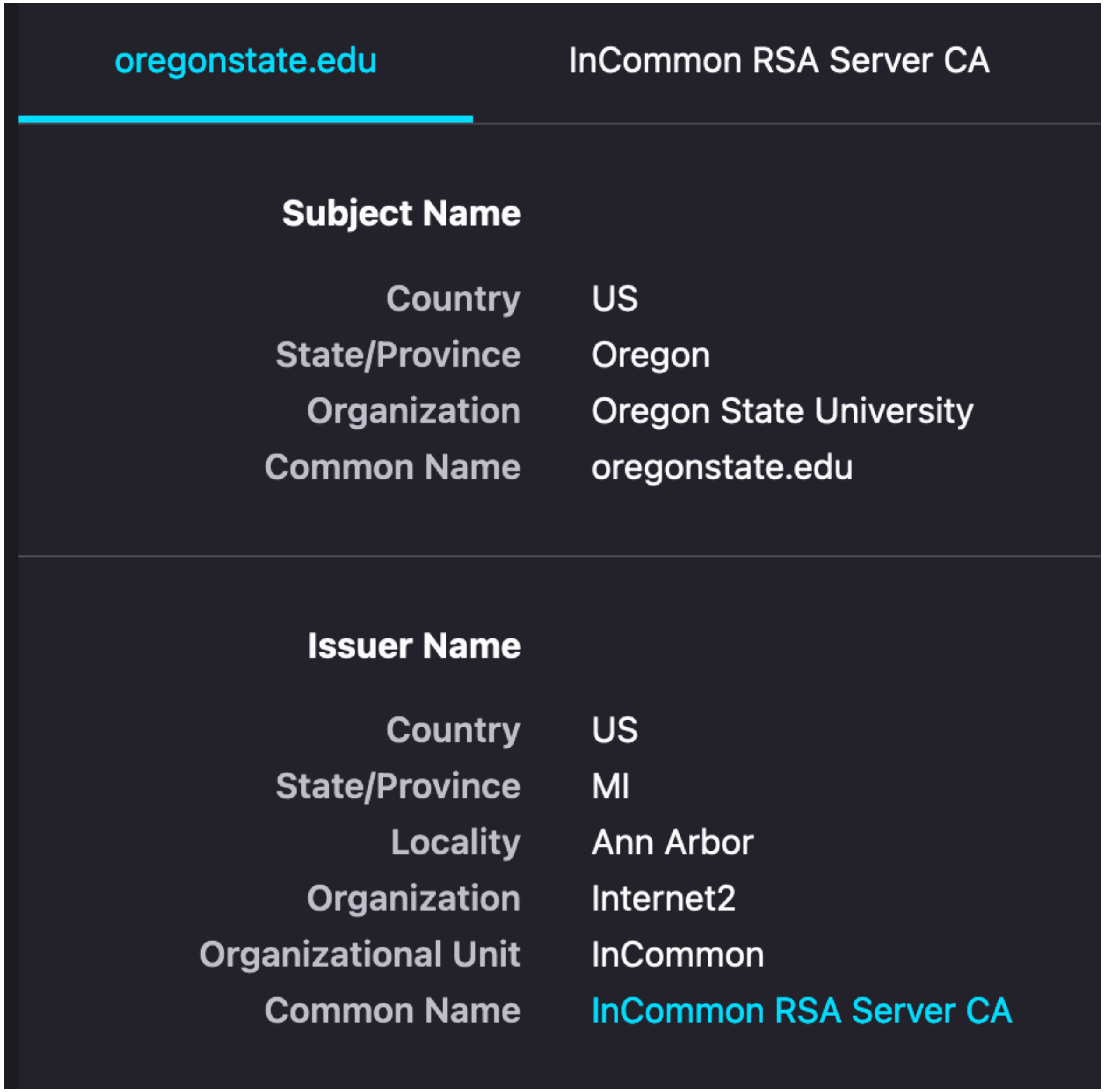
Trust Chain



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

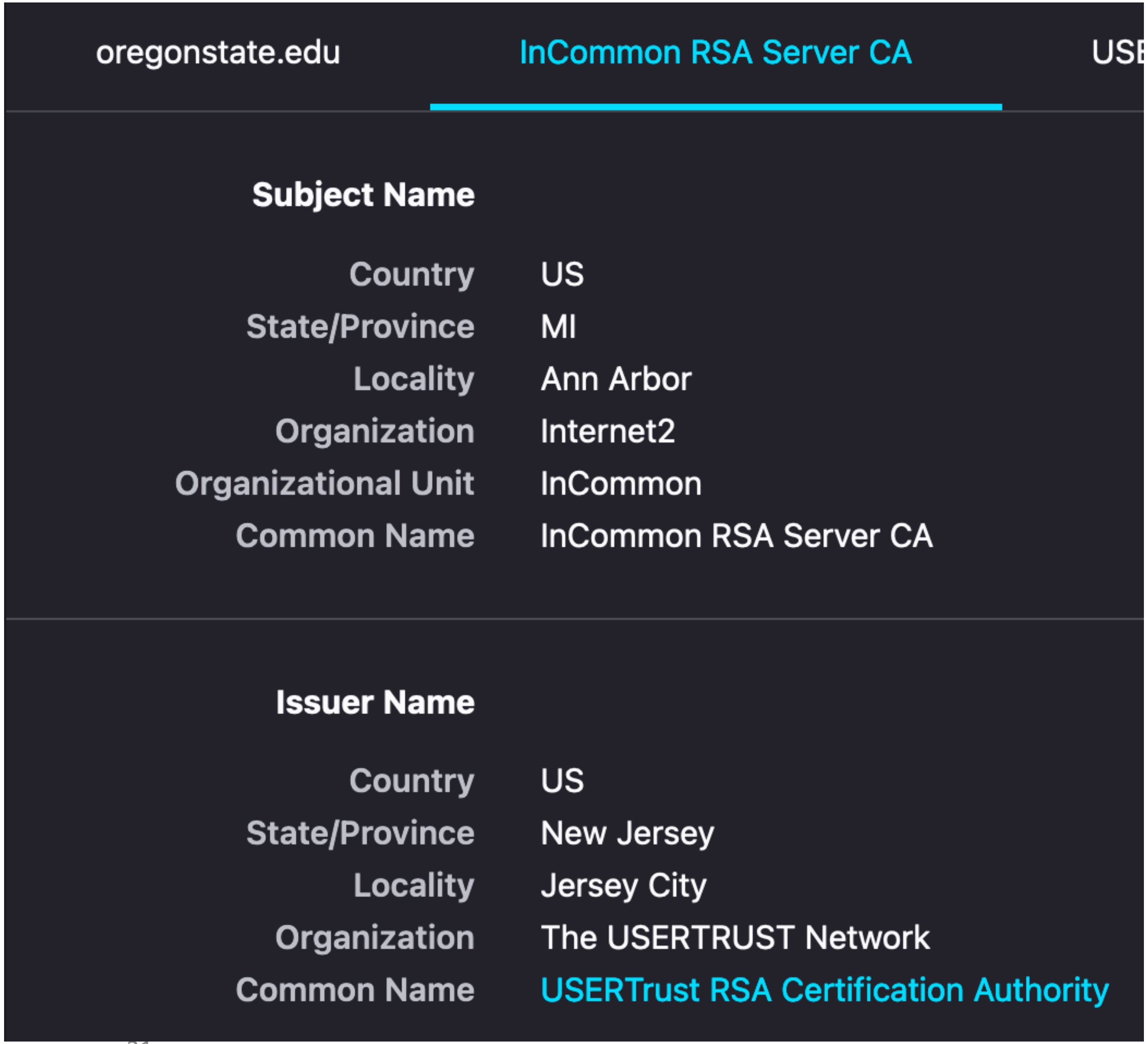
Trust Chain

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



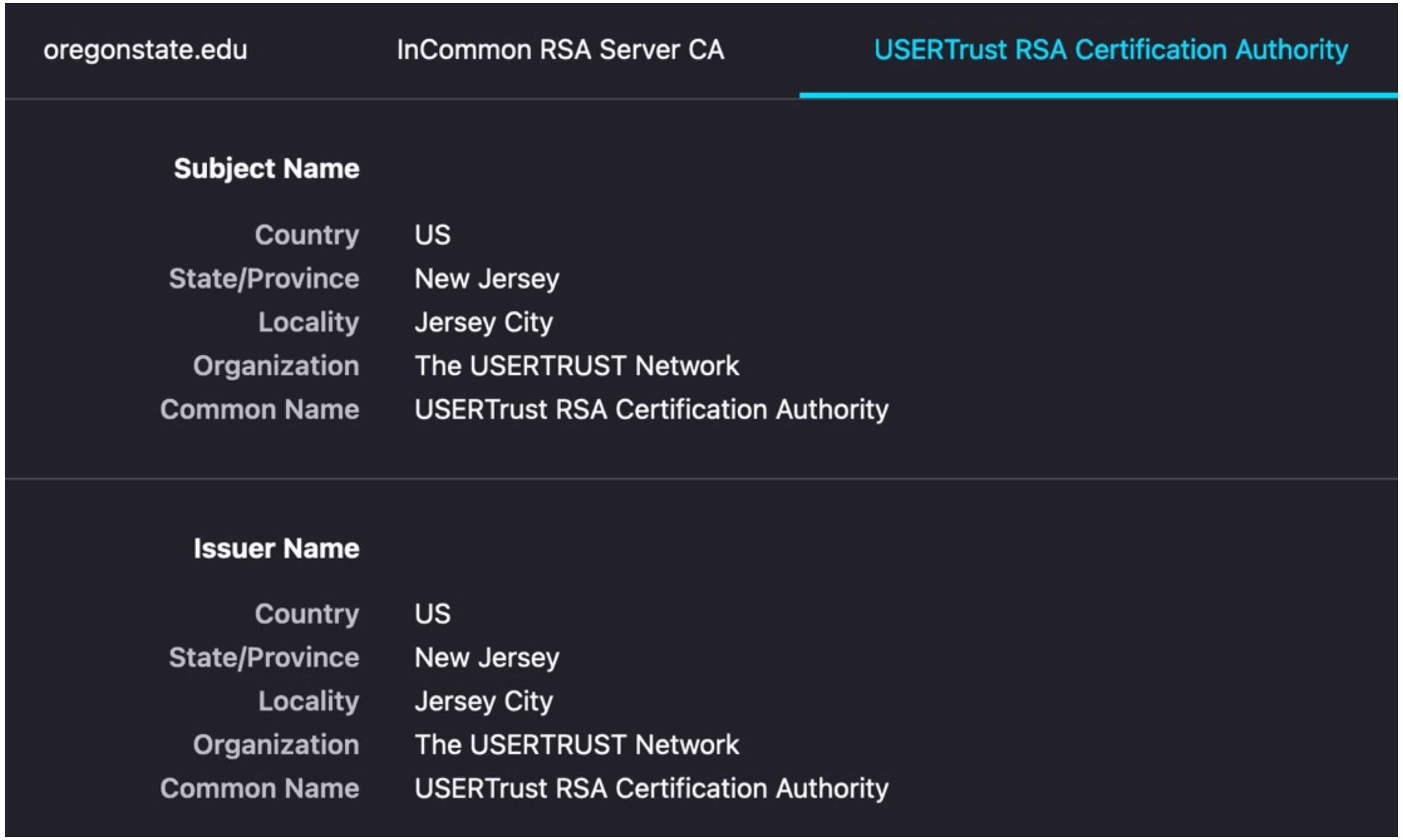
Trust Chain

- 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

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



How Do We Verify Our Identity?

- You as a
  - Student
  - Oregon resident
  - U.S. Citizen
- When issuing the student ID
  - We verify your Oregon ID…

How Do We Verify Our Identity?

- You as a
  - Student
  - Oregon resident
  - U.S. Citizen

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

How Do We Verify Our Identity?

- You as a
  - Student
  - Oregon resident
  - U.S. Citizen

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

How Do We Verify Our Identity?

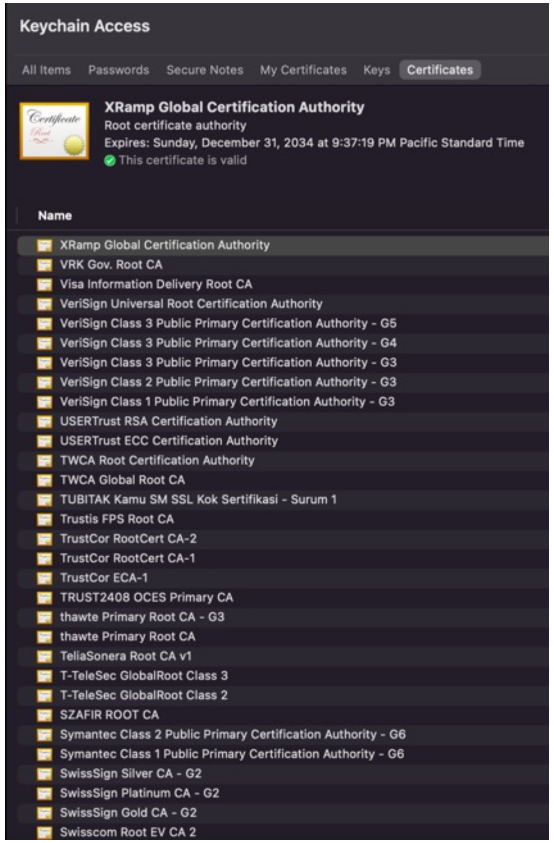
- You as a
  - Student
  - Oregon resident
  - U.S. Citizen

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

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

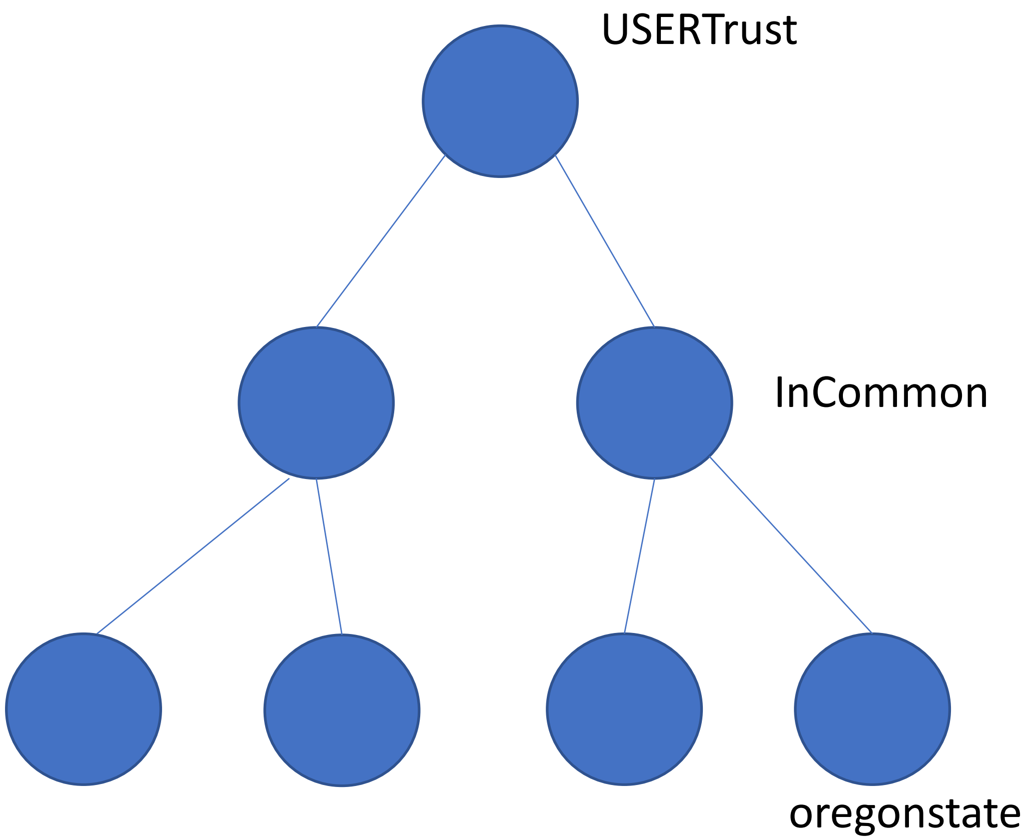
Root Certificate Authority (Root CA)

- We 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..
- We 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
  - We set a small set of entities use self-signed cert
- Verify the certificate chain!
  - We must verify the entire chain



Trust Chain

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

Trust Chain

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

Trust Chain

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

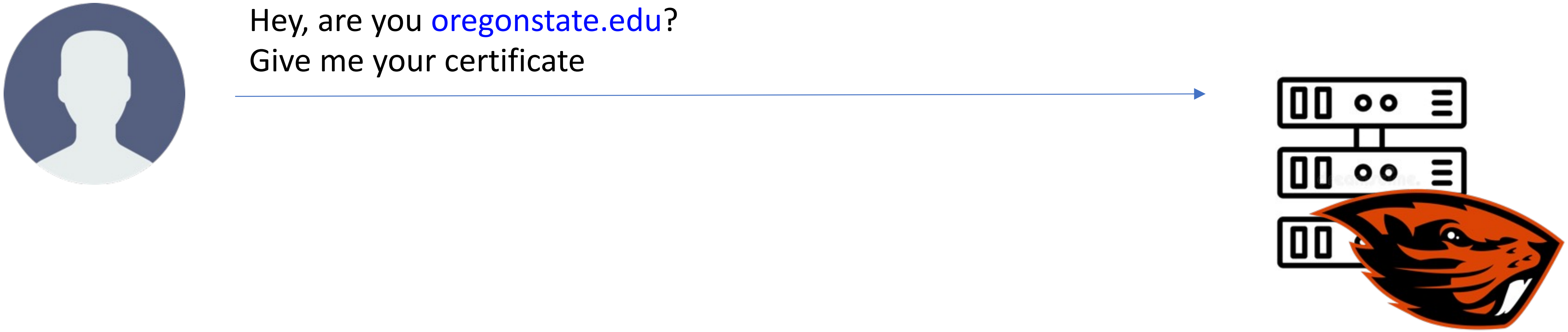
Subject Name	
Country	US
State/Province	MI
Locality	Ann Arbor
Organization	Internet2
Organizational Unit	InCommon
Common Name	InCommon RSA Server CA
Issuer Name	
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

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

How To Verify oregonstate.edu?

- Using the digital certificate!

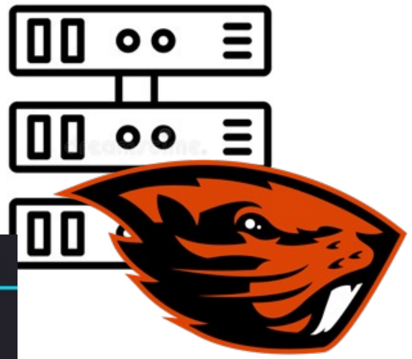


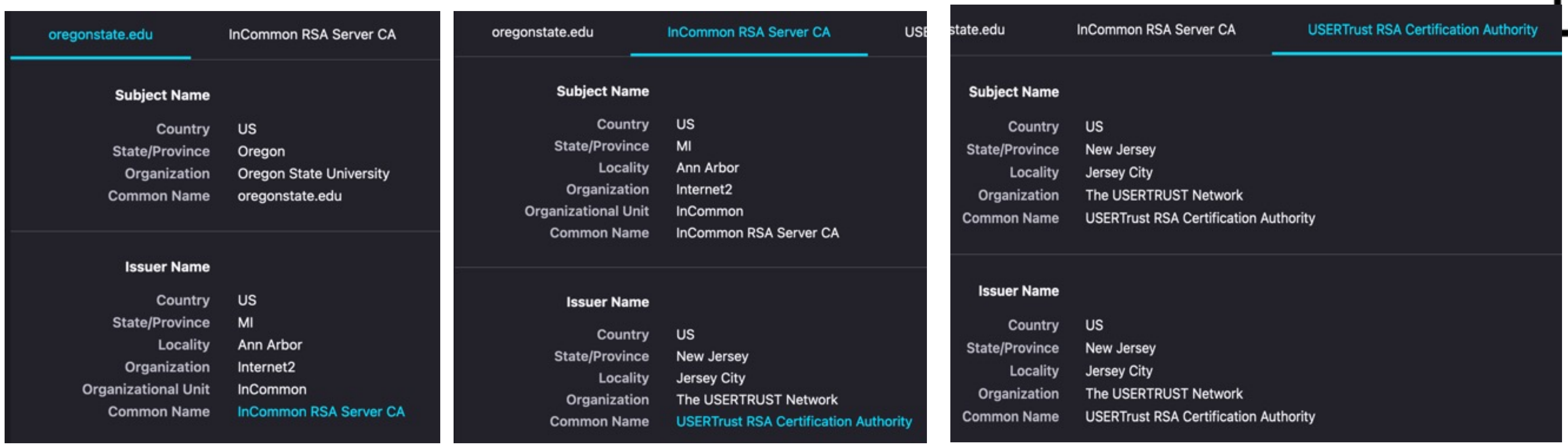
How To Verify oregonstate.edu?

- 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

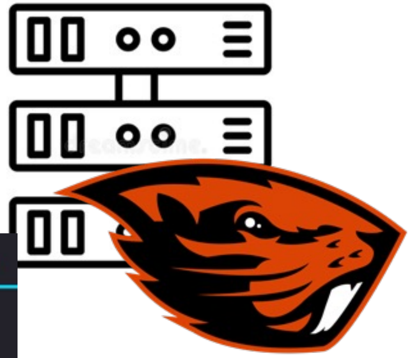
Country	US
State/Province	New Jersey
Locality	Jersey City
Organization	The USERTRUST Network
Common Name	USERTrust RSA Certification Authority

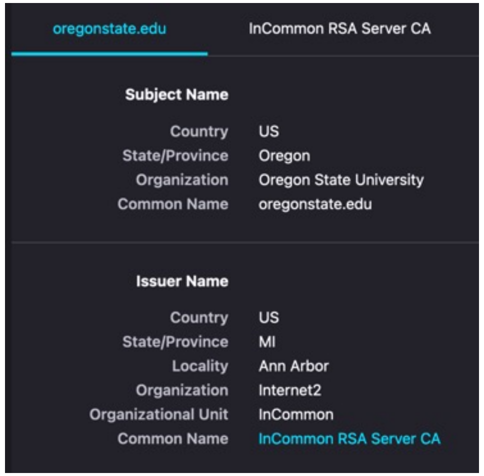
How To Verify oregonstate.edu?

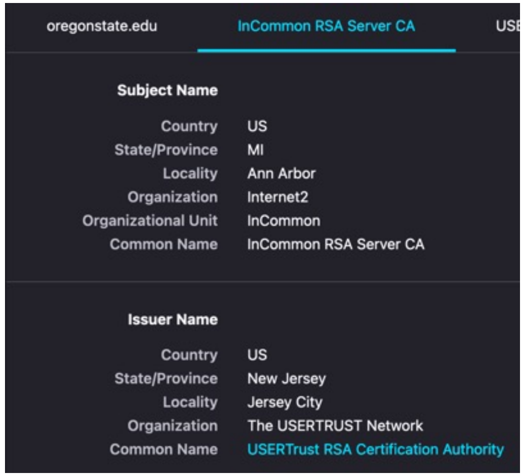
- Using the digital certificate!

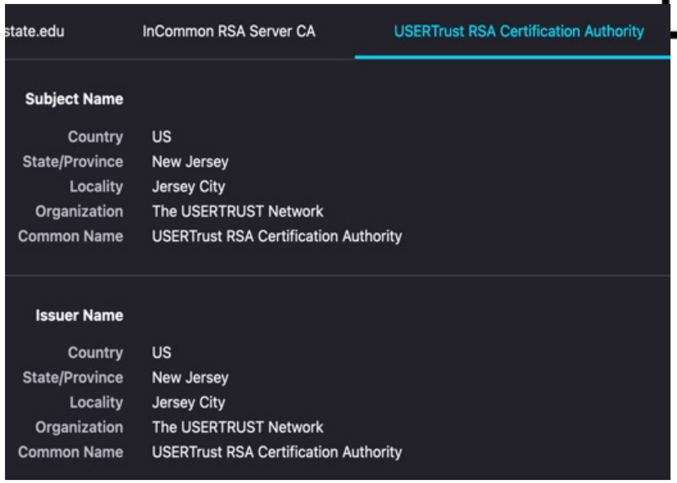
Hey, are you oregonstate.edu?
Give me your certificate

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









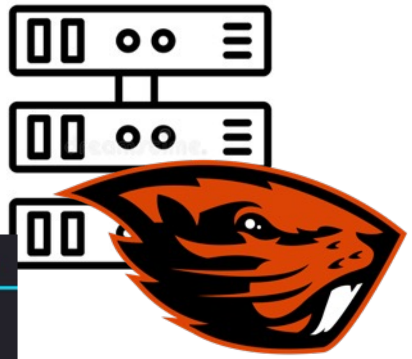
Oregonstate verified by InCommon RSA

How To 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

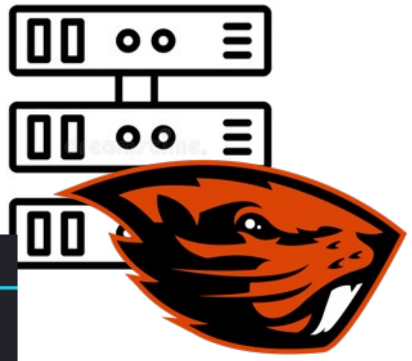
InCommon RSA verified by USERTrust RSA

How To 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)

Summary

- RSA encryption with private key can be used as digital signature
  - Only the private key holder can generate the message
  - Anyone with public key can verify this!
- We use digital certificates to share public key information
  - Entity name, address, other information with
  - Public key!
- Certificates are signed by other trustful entities
  - Verify the entity info and the public key, and then, sign the certificate!

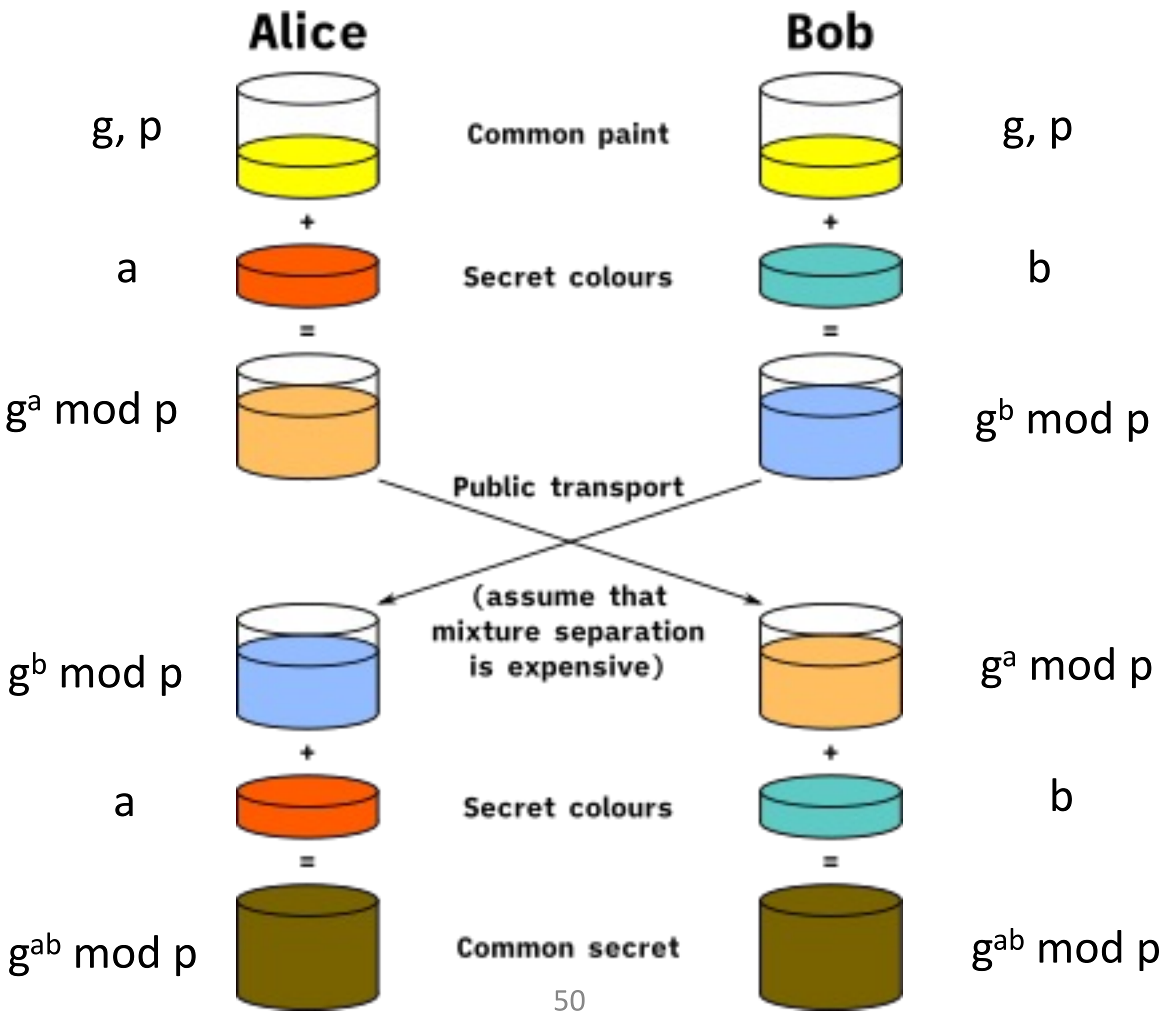
Summary

- A certificate need to be verified by other entity
  - That other entity is also need to be verified by…
- Root CA is the list of trusted Certificate Authority
  - We accept their self-signed certificate
- We must verify the entire certificate trust chain
  - oregonstate.edu -> InCommon RSA -> USERTrust RSA …

Diffie-Hellman Key Exchange

- A method of securely exchanging cryptographic keys over
  - Public channel!
- Based on the difficulty of mathematical problem of
  - Discrete logarithm
  - E.g., For given g, a, A, B where
  - g^a mod p = A
  - g^b mod p = B
  - can you compute g^{ab} mod p?
- Getting ab is difficult..

DH in Graphics



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

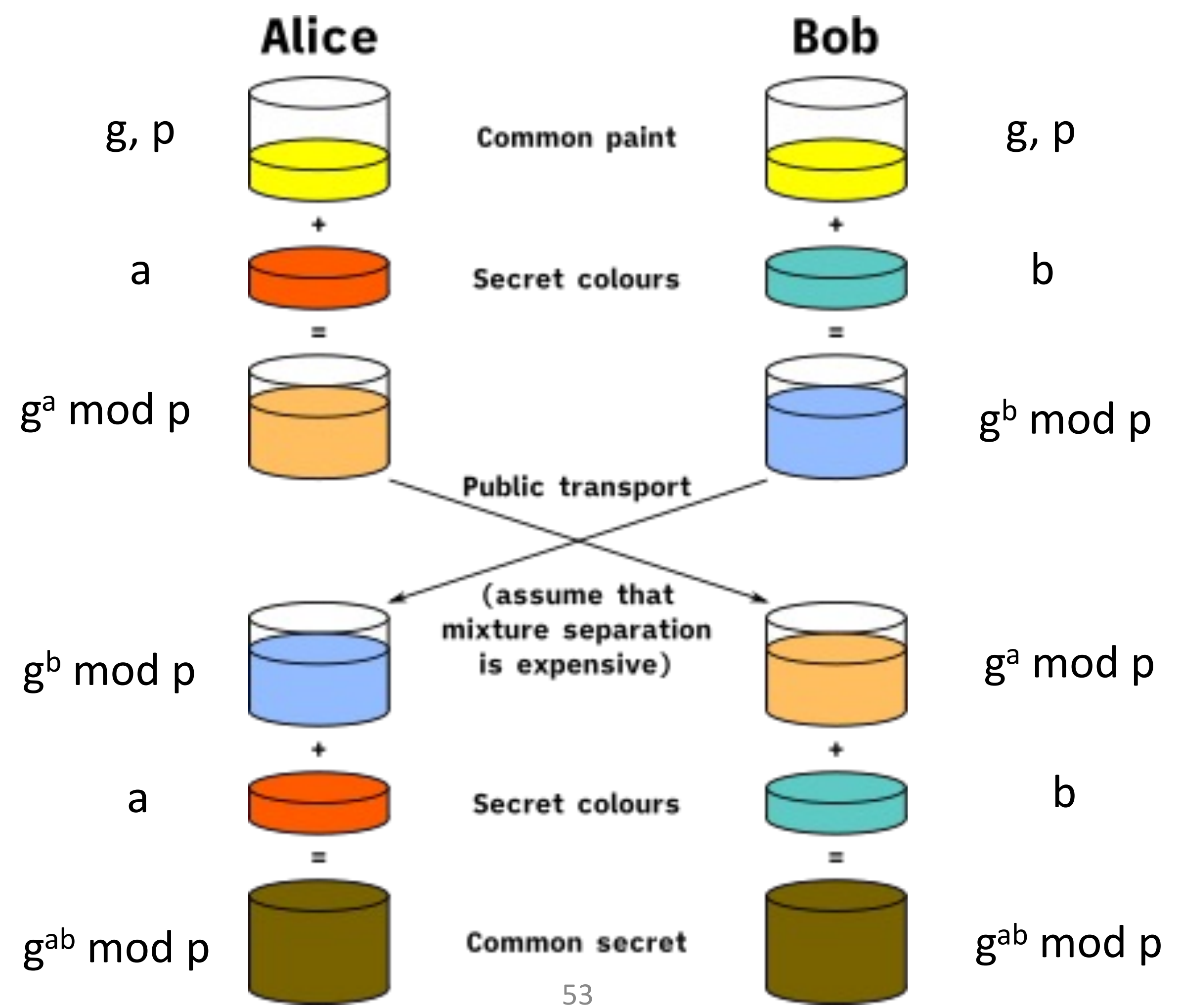
Diffie-Hellman Key Exchange

- g^{ab} mod p is our secret

- 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
- Hard to compute g^{ab} from those values
  - Discrete logarithm; Can you get a from $A = g^a$ mod p

DH in Graphics



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 Implications

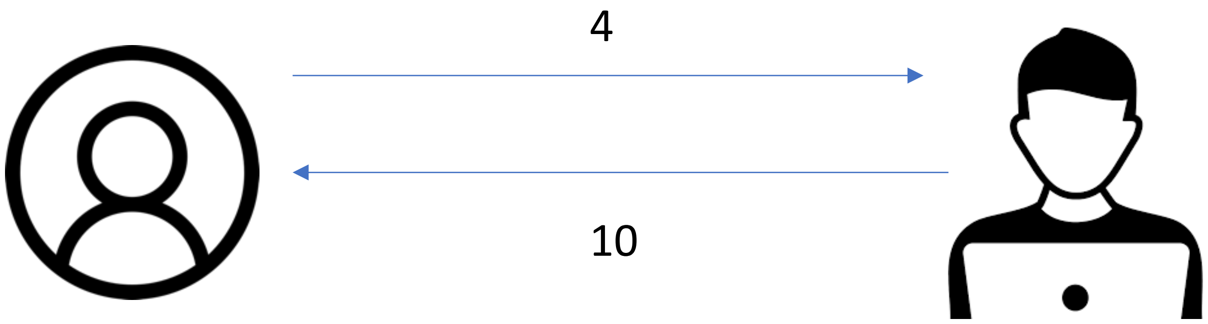
- Users are agreeing on two prime numbers
  - g, p
- User A chooses any integer a, nobody knows it
- User B chooses any integer b, nobody knows it
- By sharing g^a mod P and g^b mod p
  - Both shares g^{ab} mod P without leaking a nor b

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

Diffie-Hellman Weakness: Man-in-the-middle Attack

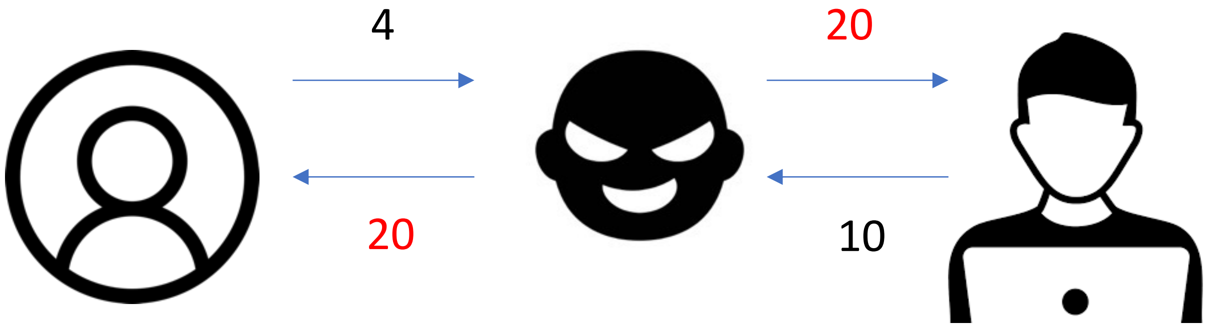
- Suppose A and B wants to share a secret
- g = 5, p = 23

- 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 Attack

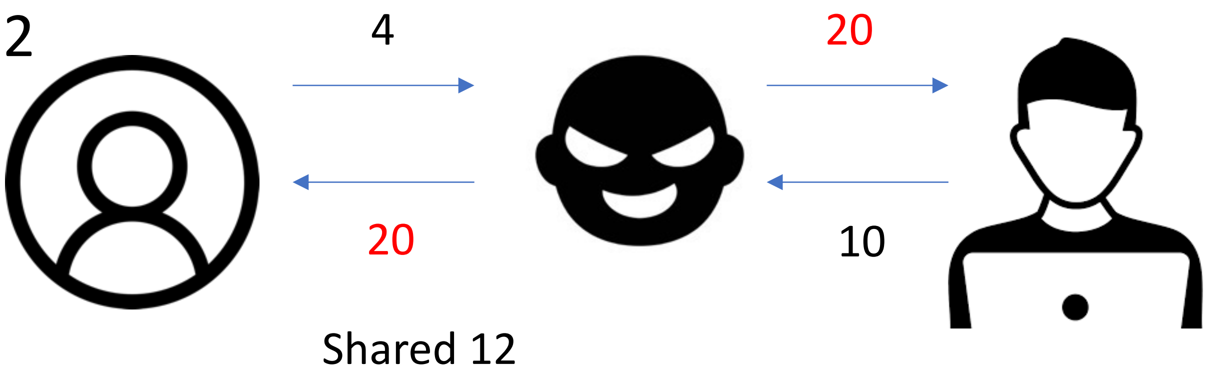
- Suppose C intercepts communication between A and B
- A chooses a = 4
  - A = 5^4 mod 23 = 625 mod 23 = 4
- B chooses b = 3
  - B = 5^3 mod 23 = 125 mod 23 = 10
- C chooses c = 5
  - C = 55 mod 23 = 3125 mod 23 = 20



- C sends 20 to both A and B

Diffie-Hellman Weakness: Man-in-the-middle Attack

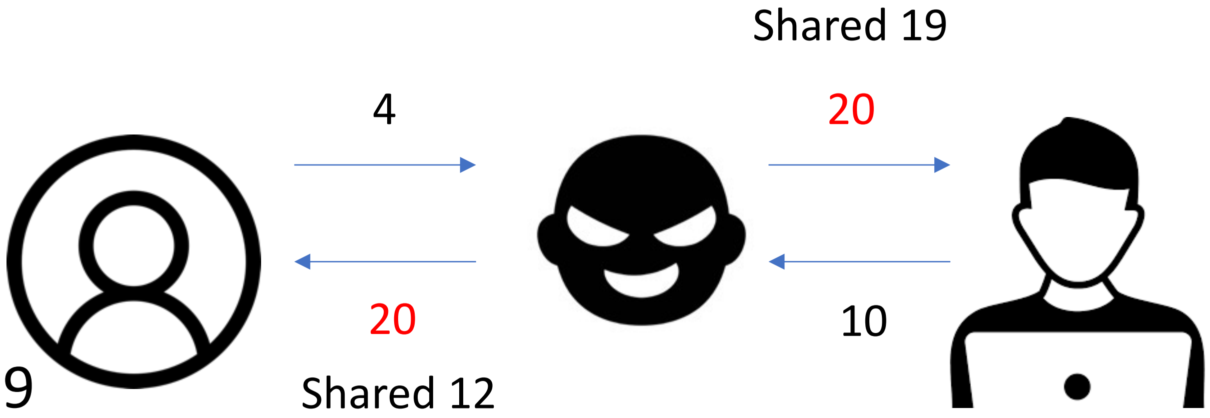
- A chooses a = 4
  - $A = 5^4 \bmod 23 = 625 \bmod 23 = 4$
  - $C^a = 20^4 \bmod 23 = 160000 \bmod 23 = 12$
- C chooses c = 5
  - $C = 5^5 \bmod 23 = 3125 \bmod 23 = 20$
  - $A^c = 4^5 \bmod 23 = 1024 \bmod 23 = 12$



- C shares a secret of 12 with A

Diffie-Hellman Weakness: Man-in-the-middle Attack

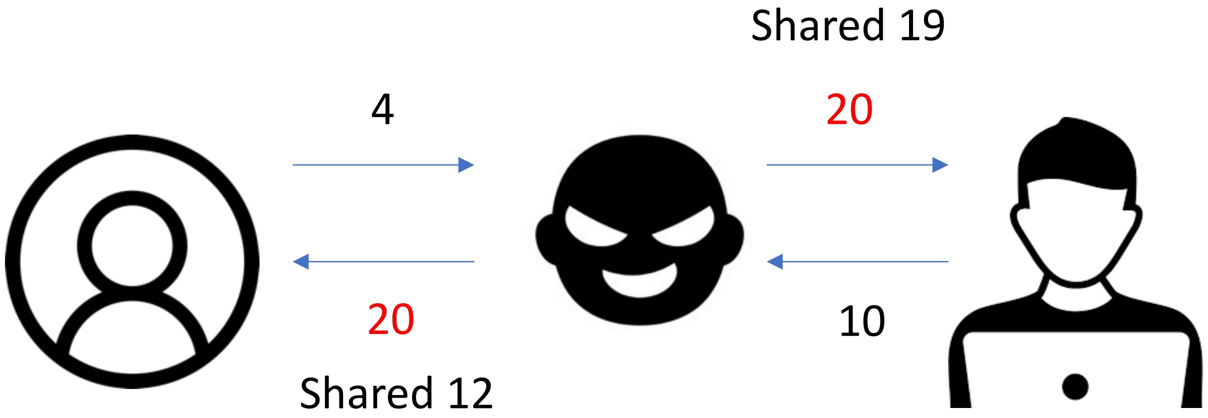
- B chooses b = 3
  - $B = 5^3 \bmod 23 = 125 \bmod 23 = 10$
  - $C^b = 20^3 \bmod 23 = 8000 \bmod 23 = 19$
- C chooses c = 5
  - $C = 5^5 \bmod 23 = 3125 \bmod 23 = 20$
  - $B^c = 10^5 \bmod 23 = 100000 \bmod 23 = 19$



- C shares a secret of 19 with B

Diffie-Hellman Weakness: Man-in-the-middle Attack

- 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 attack!

How Can We Secure the Internet Communication?

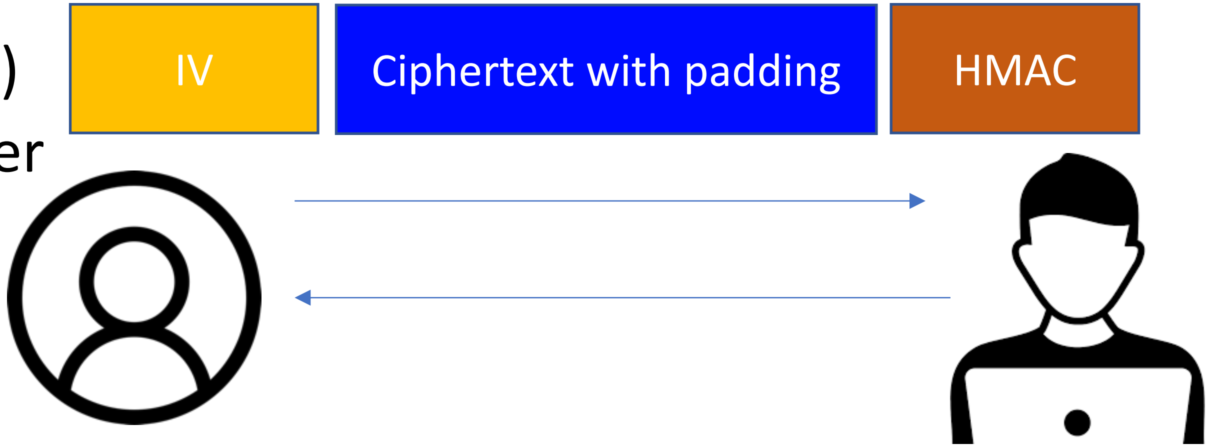
- Authentication
  - Get the certificate of each entity
  - Verify their public key
  - Using certificate trust chain!

- 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

Killed
Man-in-the-middle attack!

How Can We Secure the 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
Integrity
has been established!

Tutorials

- raw-rsa
- raw-dh