CS 578: Cyber-security

Part I: Basic crypto for network/internet sec.

Sanghyun Hong

sanghyun.hong@oregonstate.edu

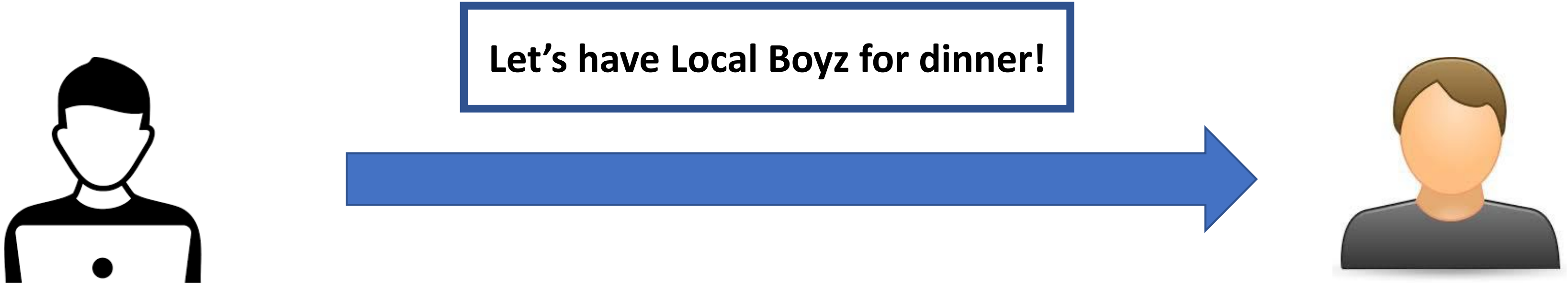
Oregon State University

SAIL
Secure AI Systems Lab

HOW CAN WE DO SECURE COMMUNICATION?

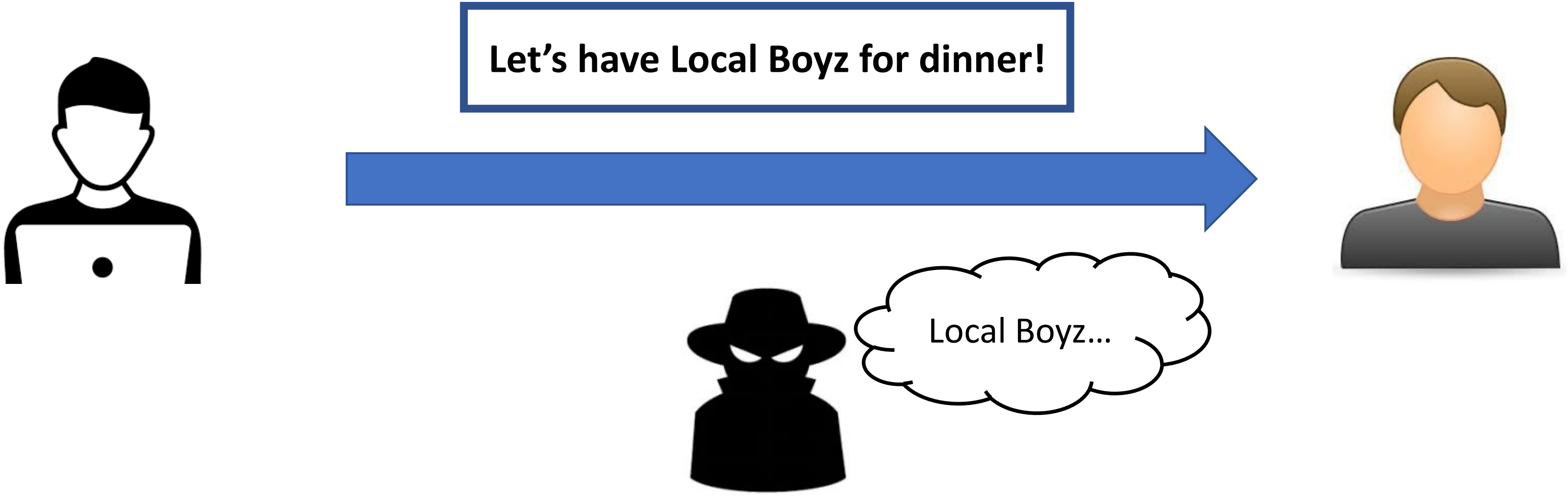
CRYPTO!

- Confidentiality
  - We want to communicate with others securely (and privately)



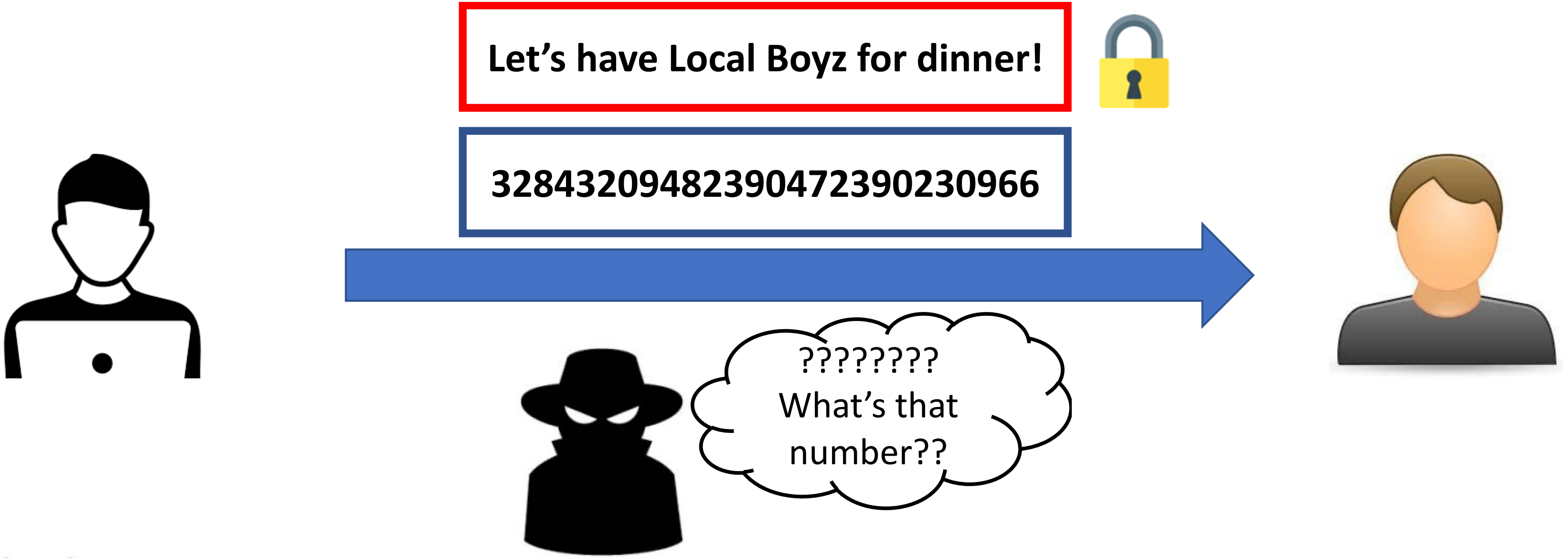
CRYPTO!

- Confidentiality
  - We want to communicate with others securely (and privately)
  - Plaintext communication can be eavesdropped by an adversary



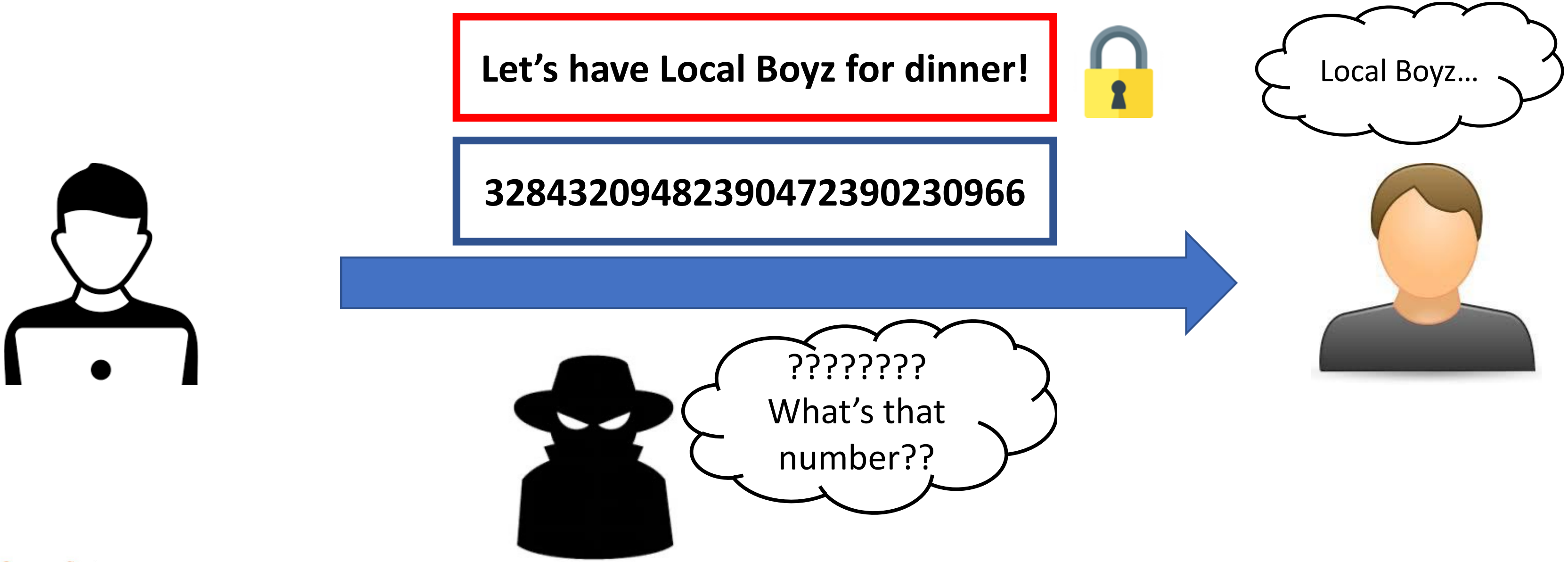
CRYPTO!

- Confidentiality
  - We want to communicate with others securely (and privately)
  - Plaintext communication can be eavesdropped by an adversary
  - Cryptography enables secure (and private) communication



BASIC TERMINOLOGY

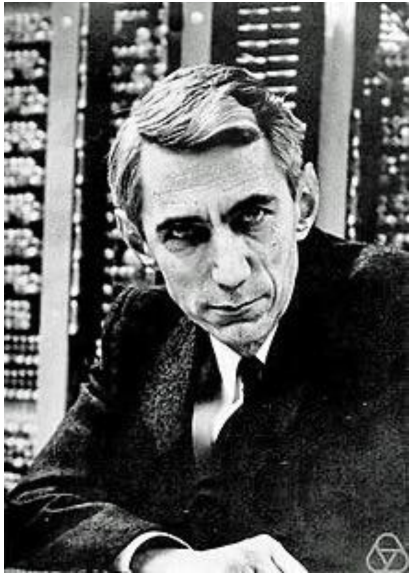
- Terms
  - Plaintext: readable text, before getting encrypted
  - Ciphertext: encrypted text, transformed plaintext using an encryption algorithm
  - Encryption/decryption: the act of encrypting (or decrypting)



WHAT DOES IT MEAN BY PERFECTLY SECURE IN COMMUNICATION?

Perfect security

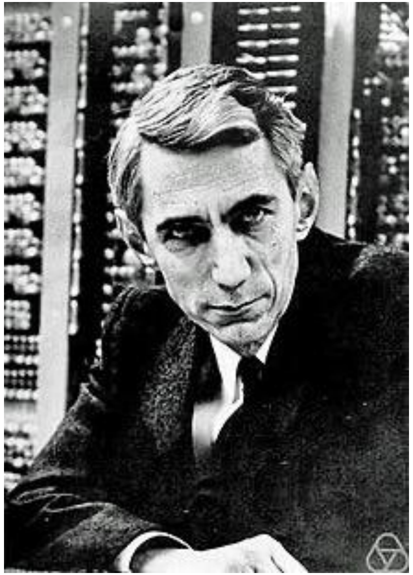
- Shannon's intuition
  - An adversary should not distinguish a message M from a random text R



Claude Shannon (1916 ~ 2001)
A Father of Information Theory and Modern Cryptography

Perfect Security

- Shannon's intuition
  - An adversary should not distinguish a message M from a random text R
  - Formally:
    - $\Pr[M = m | C = c] = \Pr[M = m]$
    - where
      - m is a message (from a set M)
      - c is a ciphertext (from a set of all ciphertexts C)
    - $\Pr[C = c | M = m] = \Pr[C = c]$
  - It means:
    - Ciphertext provides no additional information
    - Observing c does not help with guessing M = m
    - c is independent of the message m



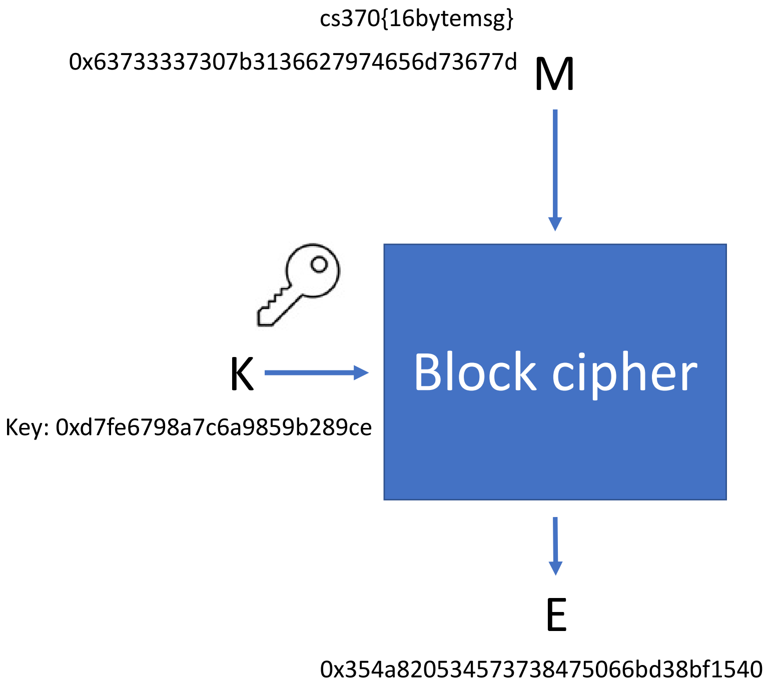
Claude Shannon (1916 ~ 2001)
A Father of Information Theory and Modern Cryptography

HOW CAN WE DO PERFECTLY SECURE IN COMMUNICATION?

BLOCK CIPHER: ENCRYPTION

- Block cipher
  - Cryptographic algorithm that work only with fixed-length set of bits

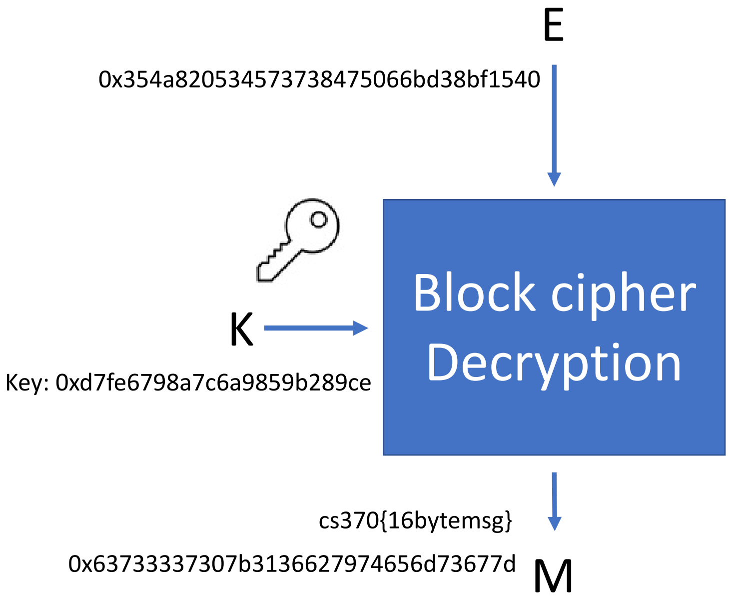
- Terminology
  - **Block:** a fixed size message *M*
  - **Key:** a secret we use for encryption
    - Shared between a sender and a receiver
  - **Encryption:** use *K* to convert *M* into *E*



BLOCK CIPHER: DECRYPTION

- Block cipher
  - Cryptographic algorithm that work only with fixed-length set of bits

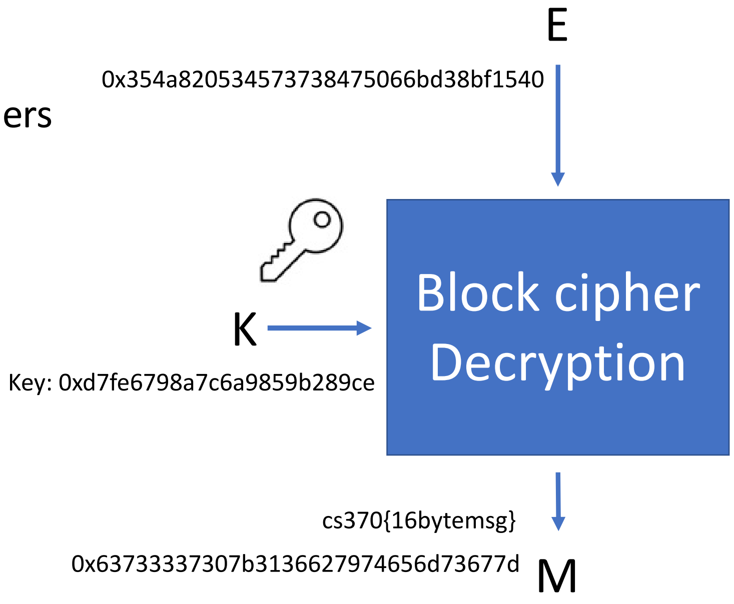
- Terminology
  - **Block:** a fixed size message *M*
  - **Key:** a secret we use for encryption
    - Shared between a sender and a receiver
  - **Encryption:** use *K* to convert *M* into *E*
  - **Decryption:** use *K* to convert *E* into *M*



BLOCK CIPHER

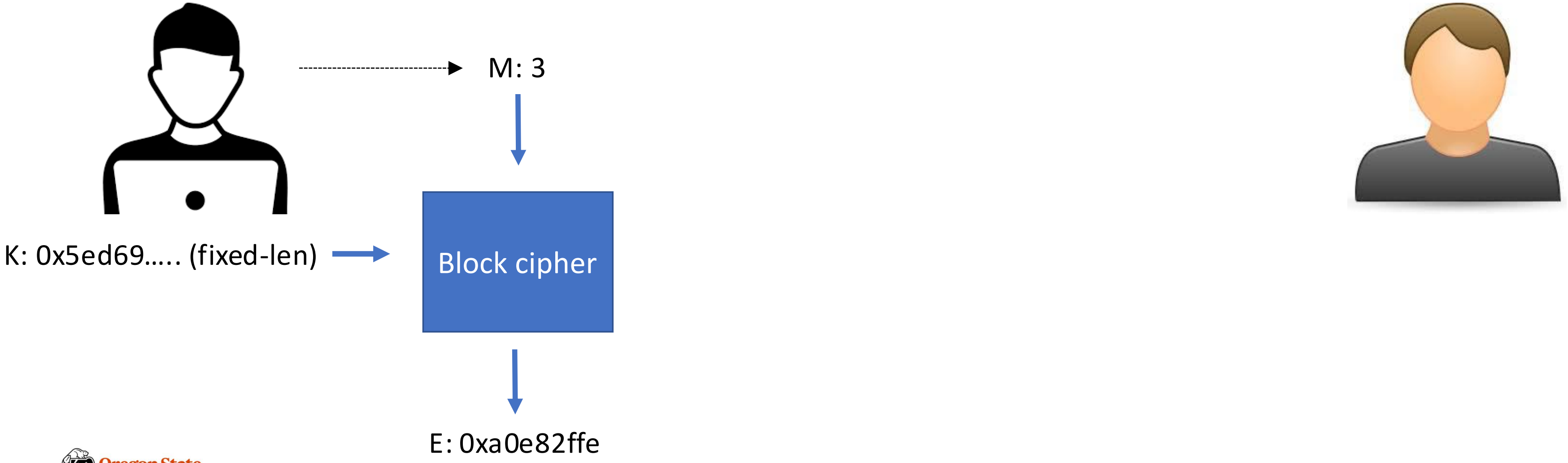
- Formally
  - You can see encryption and decryption as
  - Generating a permutation of numbers:
    - $\{0,1\}^n \rightarrow \{0,1\}^n$
    - Mappings should be 1-to-1
  - The key determines how to permute the numbers

M	Ciphertext
0	0xaf531b0e1
1	0x14a986e7a
2	0xad738009d
3	0x5ed6985c5
4	0xf3b8aa2e8
5	0xad04ec00e
…	0x59fd94c21



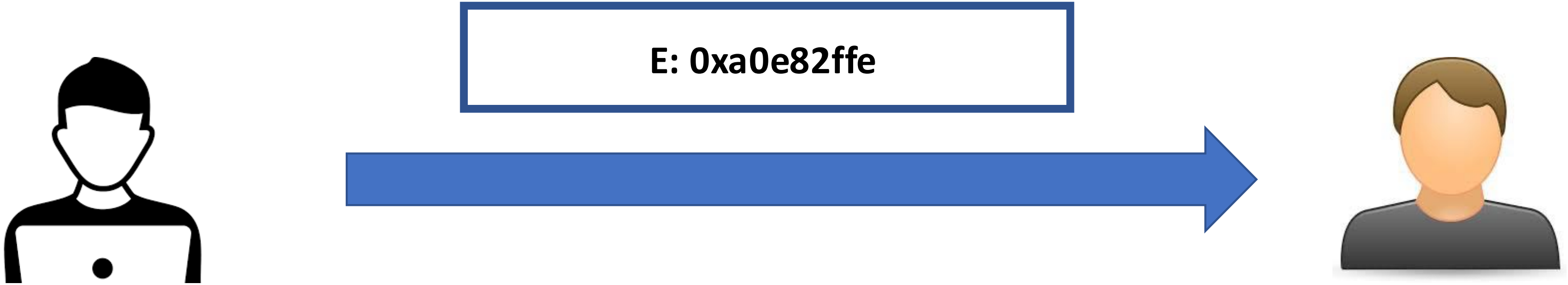
BLOCK CIPHER: IN OPERATION

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm



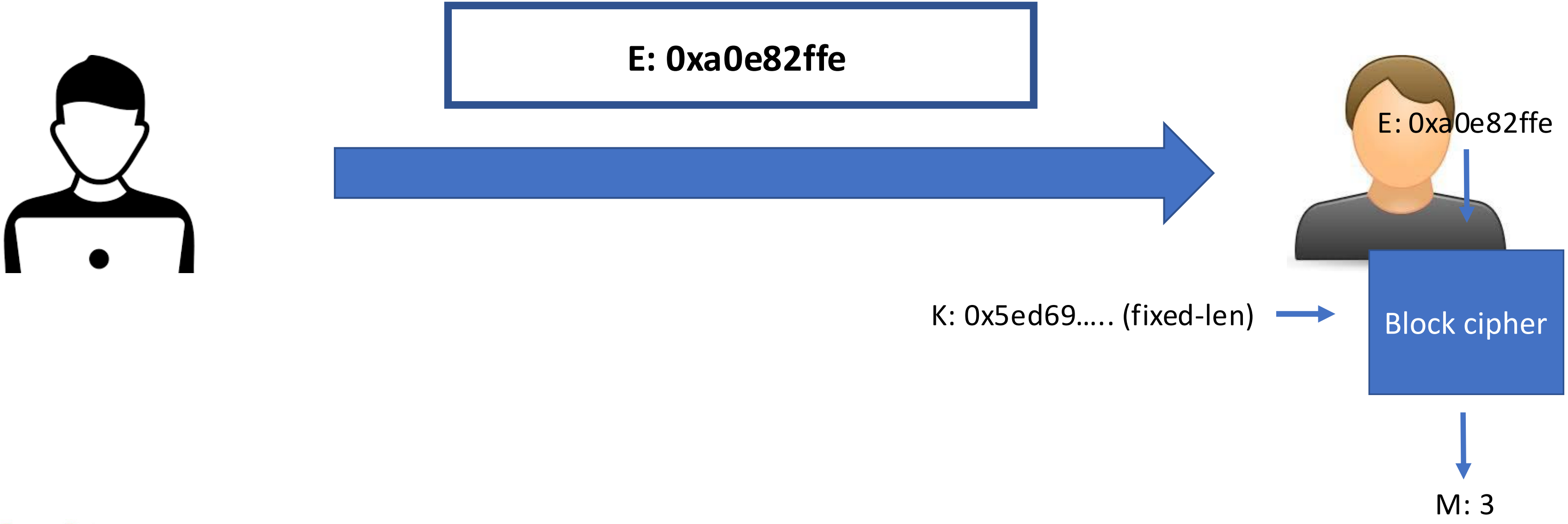
BLOCK CIPHER: IN OPERATION

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm



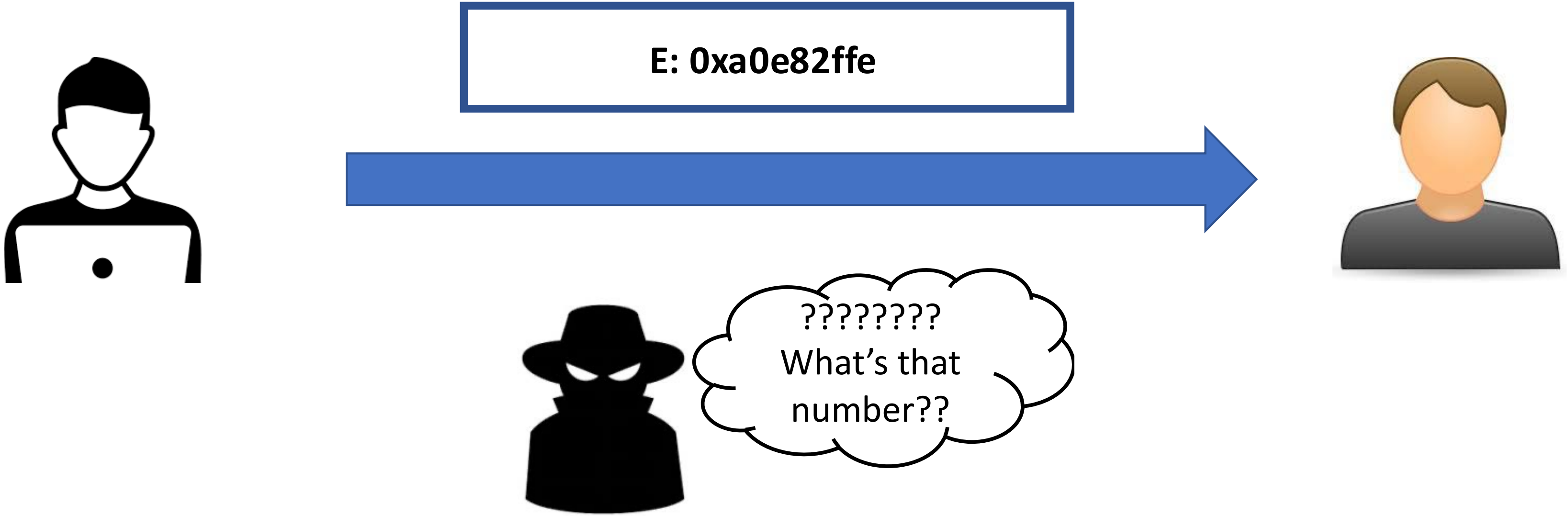
BLOCK CIPHER: IN OPERATION

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm



BLOCK CIPHER: IN OPERATION

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm



WHAT IS THE PROBLEM?

SYMMETRIC KEY CRYPTOGRAPHY

- Problems
  - How can we securely share the key between two parties?
  - How can we manage communications from/to multiple parties (100+)?

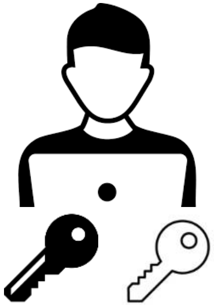
SYMMETRIC KEY CRYPTOGRAPHY

- Problems
  - How can we securely share the key between two parties?
  - How can we manage communications from/to multiple parties (100+)?

- Solutions
  - What if I have two keys?
    - Key A that only can encrypt a message (but can't decrypt)
    - Key B that can encrypt and decrypt a message
  - How can I leverage the two keys?
    - Share Key A to others
    - Do not share; keep Key B private

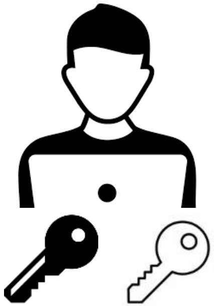
PUBLIC KEY CRYPTOGRAPHY

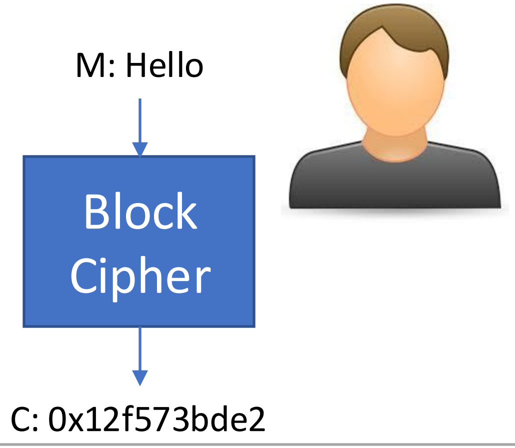
- The key idea
  - Asymmetric key cryptography
  - Use two different keys for encryption and decryption
    - Public key: share to others, only can encrypt a message
    - Private key: do not share, can encrypt and decrypt
  - What is possible?



PUBLIC KEY CRYPTOGRAPHY

- The key idea
  - Asymmetric key cryptography
  - Use two different keys for encryption and decryption
    - Public key: share to others, only can encrypt a message
    - Private key: do not share, can encrypt and decrypt
  - What is possible?





PUBLIC KEY CRYPTOGRAPHY

- The key idea
  - Asymmetric key cryptography
  - Use two different keys for encryption and decryption
    - Public key: share to others, only can encrypt a message
    - Private key: do not share, can encrypt and decrypt
  - What is possible?

PUBLIC KEY CRYPTOGRAPHY

- The key idea
  - Asymmetric key cryptography
  - Use two different keys for encryption and decryption
    - Public key: share to others, only can encrypt a message
    - Private key: do not share, can encrypt and decrypt
  - What is possible?
    - No one can decrypt a ciphertext unless they have the private key
    - We do not need to share the private key to anyone else
    - We share public key that can only encrypt the message

PUBLIC KEY CRYPTOGRAPHY: ADVANTAGE

- Key exchange complexity
  - Each person shares their public key to everybody
  - But they do not share their private key
  - We need O(N) keys
- Benefit: it scales!
  - Suppose we have a crypto conference with 400 folks
  - Symmetric key crypto: we need 400 x 399 / 2 keys for secure comm.
  - Asymmetric key crypto: we only need 400 public-private key pairs

WHAT ARE THE PUBLIC-KEY CRYPTO WE USE IN PRACTICE?

PUBLIC KEY CRYPTOGRAPHY: RSA

- RSA (Rivest, Shamir, Adleman)
  - A popular public key cryptography algorithm
  - It exploits the difficulty of prime factorization
    - To break RSA, an adversary solves the prime factorization of a large number
  - It is used for digital signature (we will revisit this later)

RSA

- Asymmetric key cryptography
  - Public key: e and N
  - Private key: d

- Key selection:
  - Choose two large prime number, p and q
    - Public key:
      - Set $N = pq$
      - Choose e as a coprime of $\phi = (p-1)(q-1)$
    - Private key:
      - Find d that satisfies $de == 1 \pmod{\phi}$

RSA

- Key selection:
  - Choose two large prime number, p and q
    - Public key:
      - Set $N = pq$
      - Choose e (e.g., 65537) as a coprime of $\phi = (p-1)(q-1)$
    - Private key:
      - Find d that satisfies $de == 1 \pmod{\phi}$
- Security
  - Concern: can an adversary guess the private key from the public key?
  - To do such an attack, the attacker needs to find ϕ
  - But we choose p and q as a large prime number; thus, it is difficult

RSA ENCRYPTION

- Suppose we have
  - Public key: e, N
  - Message: M
  - Ciphertext: $M^e \bmod N$

RSA DECRYPTION

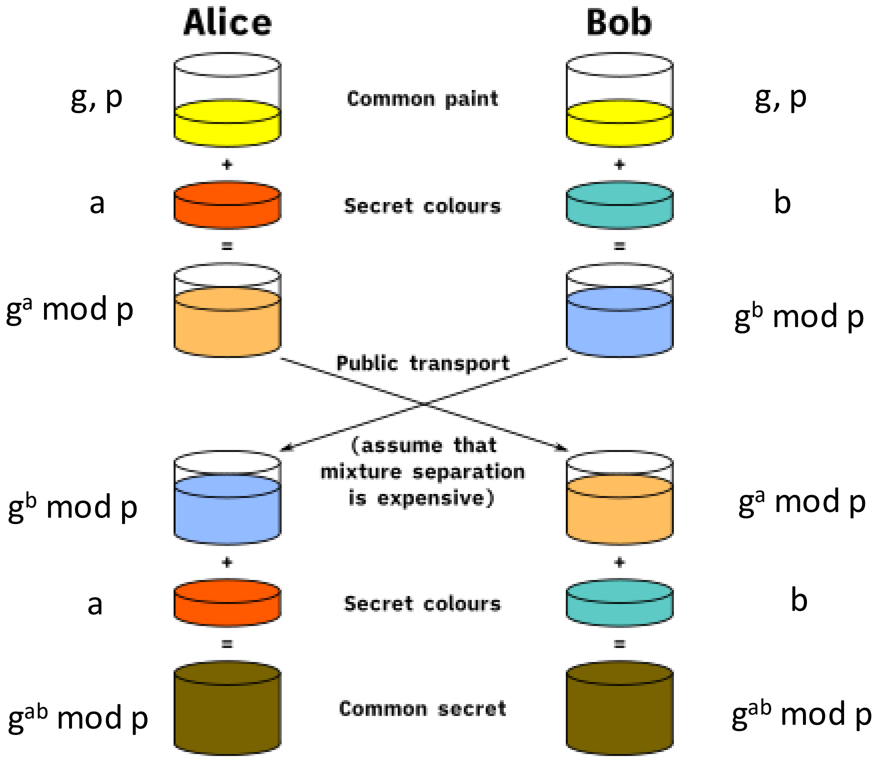
- We have
  - Public key: e, N
  - Message: M
  - Ciphertext: M^e mod N
- Suppose we also have
  - Public key: e N
  - Private key: d (that satisfies ed = 1)
  - Ciphertext: C = M^e
  - Plaintext: C^d mod N
    - = $(M^e)^d$ mod N
    - = M^{ed} mod N
    - = M mod N (N is a really large prime, so mostly it's N)

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

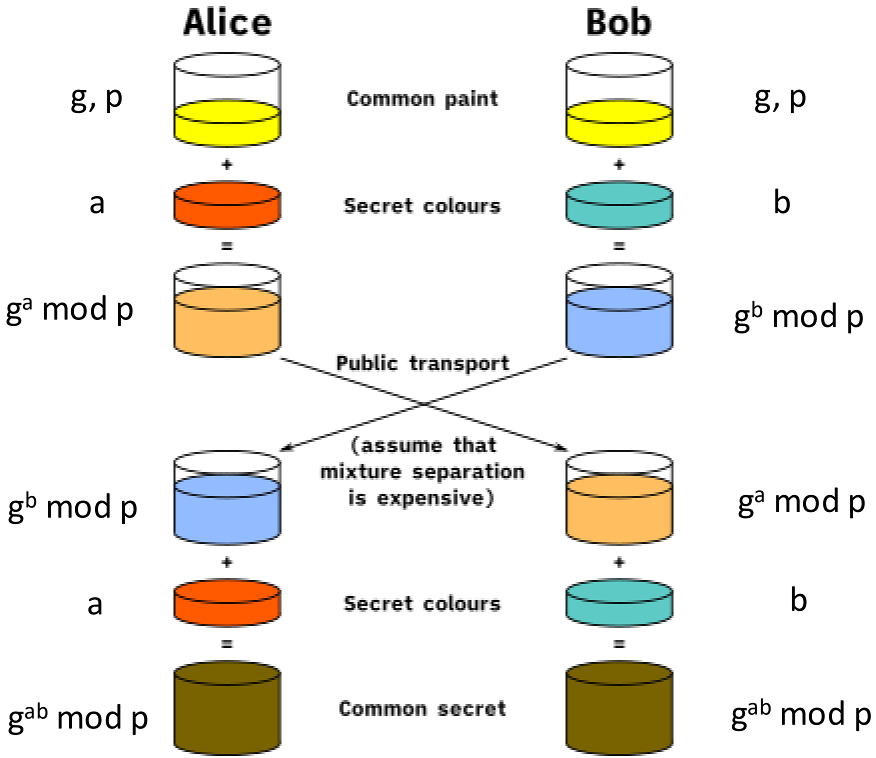
DIFFIE-HELLMAN KEY EXCHANGE

- 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 mod 23 = 625 mod 23 = 4
- B chooses b = 3
  - B = 5^3 mod 23 = 125 mod 23 = 10

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

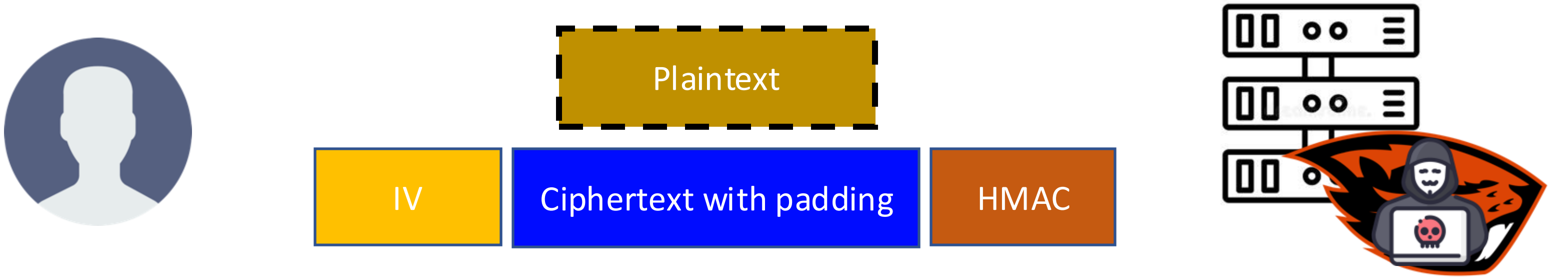
- By sharing g^a mod P and g^b mod p
  - Both shares g^{ab} mod P without leaking a nor b

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

DIGITAL CERTIFICATE AND ITS ECOSYSTEM

DIGITAL CERTIFICATE: MOTIVATION

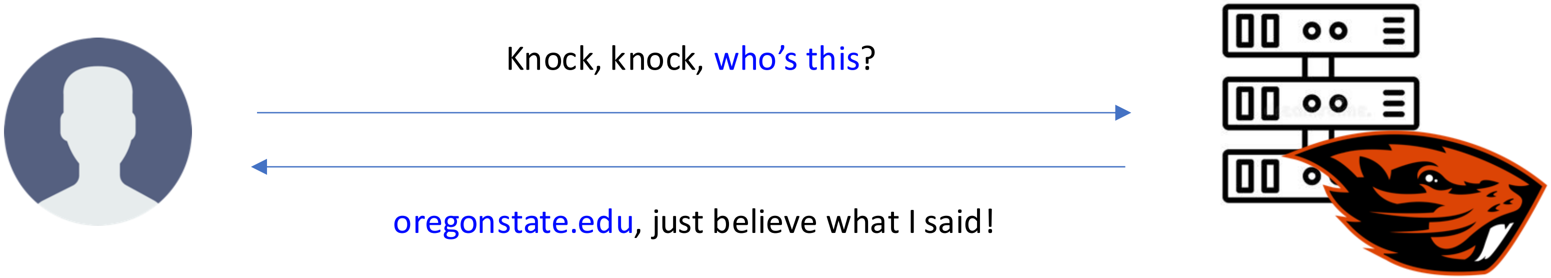
- An example scenario:
  - Suppose the oregonstate.edu server has the public/private key
  - You want to connect to the website securely



  - 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?

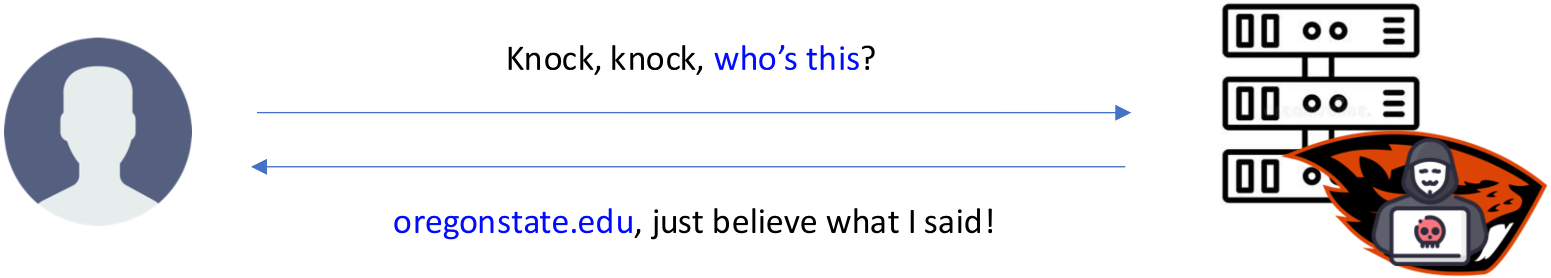


We Need Some Ways to Check If They Are OSU (Authenticity)!

HOW CAN WE CHECK THE AUTHENTICITY?

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

Knock, knock, who's this?



oregonstate.edu, just believe what I said!

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?

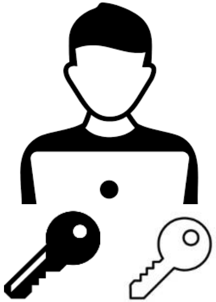
- Intuition
  - Need an identification mechanism
  - Need information that we can use to verify the sender

- 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

DIGITAL SIGNATURE AND RSA

- 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: $M = C^e \bmod N$



DIGITAL SIGNATURE AND RSA

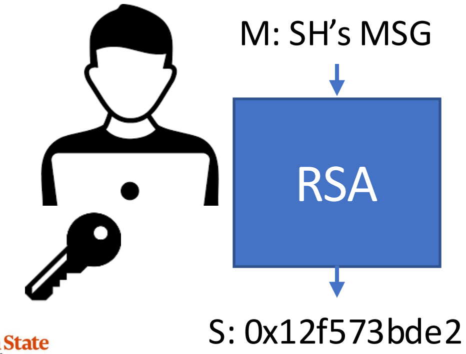
- Digital signature
  - A mathematical scheme for verifying the authenticity of digital messages
  - RSA can be used for "signing"

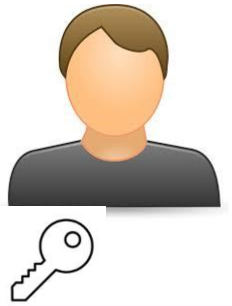
- Encryption and decryption for "signing"
  - Encryption is applying the private exponent to a plaintext: $C = M^d \bmod N$
  - Decryption is applying the public exponent to a ciphertext: $M = C^e \bmod N$

DIGITAL SIGNATURE AND RSA

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

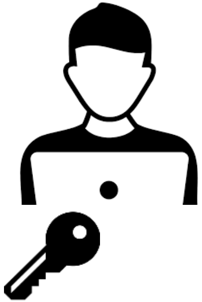
- Encryption and decryption for "signing"
  - Encryption is applying the private exponent to a plaintext: $C = M^d \bmod N$
  - Decryption is applying the public exponent to a ciphertext: $M = C^e \bmod N$

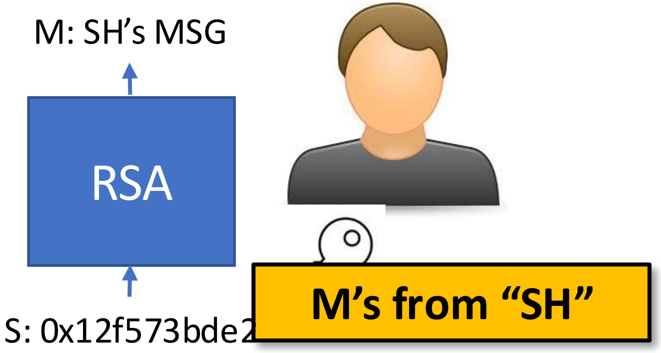




DIGITAL SIGNATURE AND RSA

- 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: $M = C^e \bmod N$





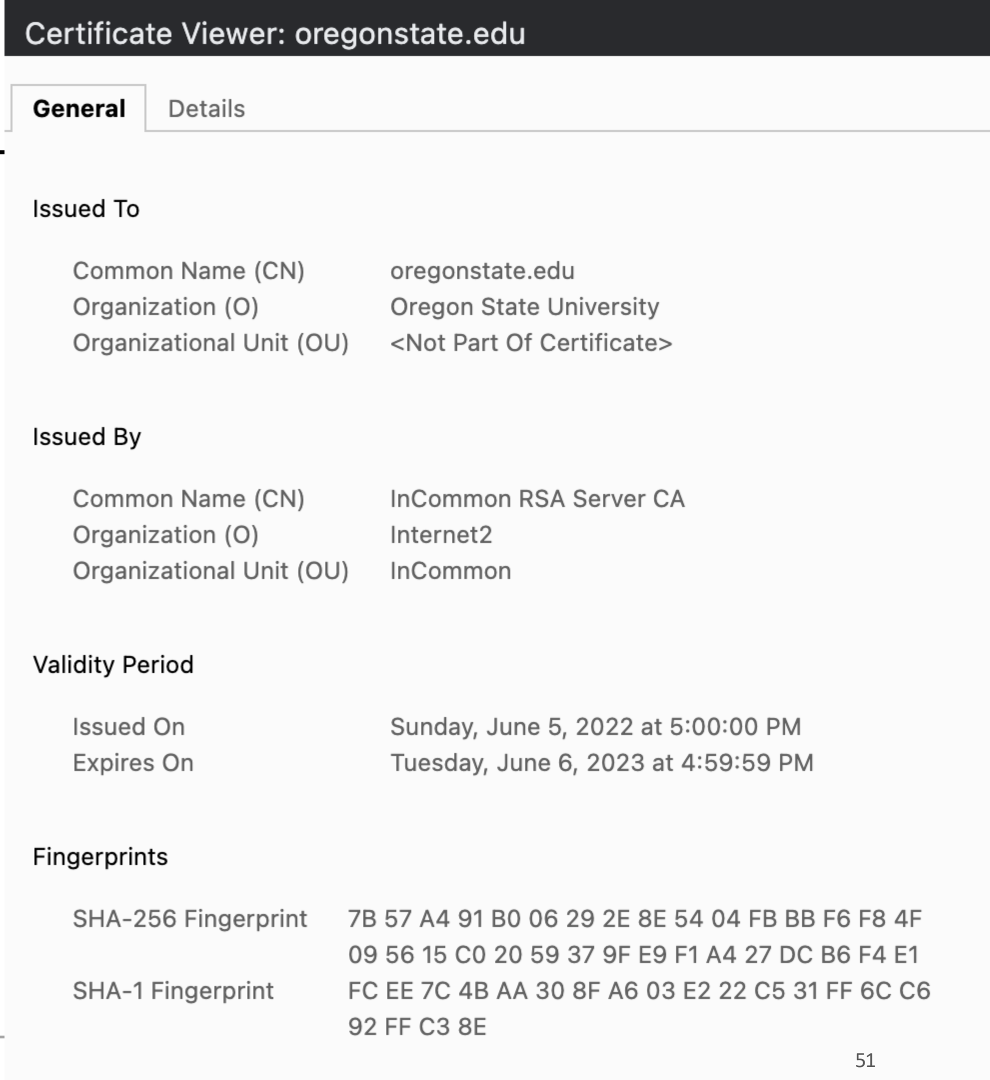
HOW CAN WE DO THIS FOR ONLINE COMMUNICATION?

- 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)

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)

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

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

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
    - 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

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: 0x2233445566778899 0011aabbccddeeff
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

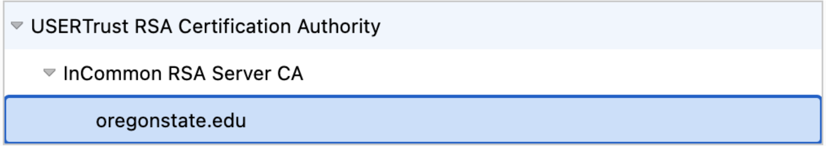
- OSU owns “oregonstate.edu”
  - Verified by InCommon RSA

- 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

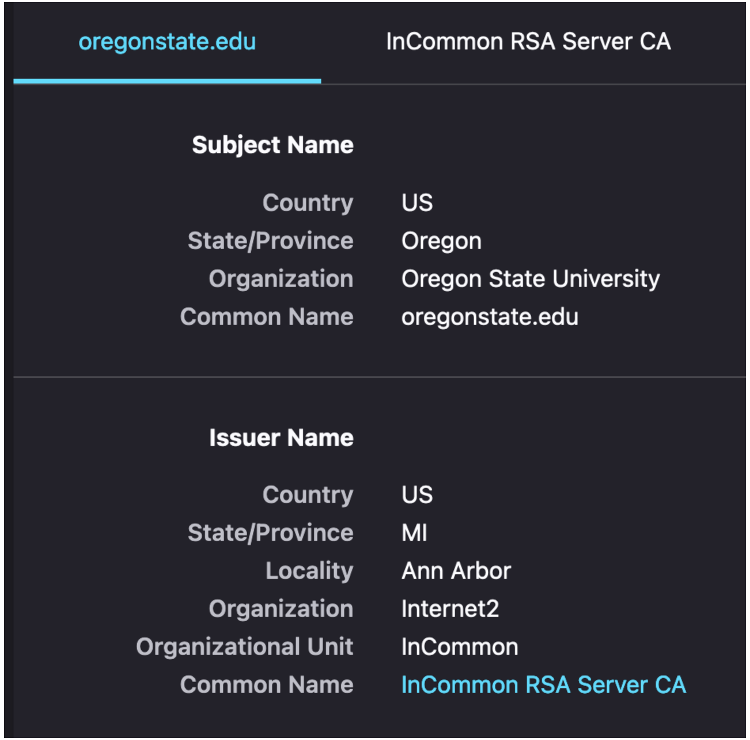
- "oregonstate.edu"
  - Verified by InCommon RSA Server CA



- InCommon RSA Server CA
  - Verified by USERTrust RSA Certificate Authority
- USERTrust RSA CA
  - Verified by self

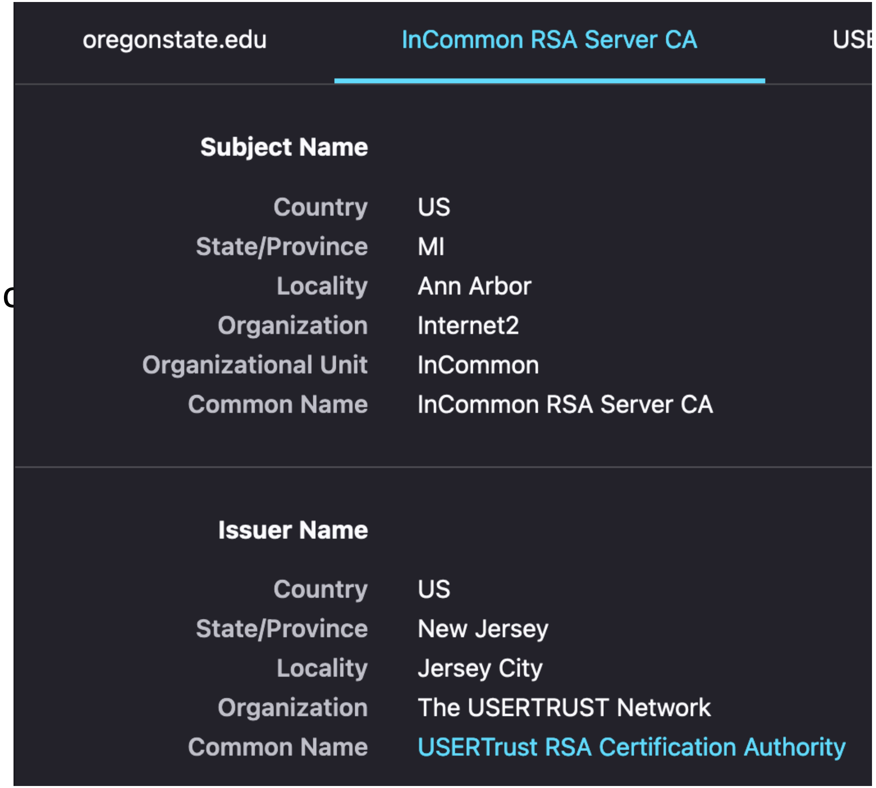
TRUST CHAIN

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



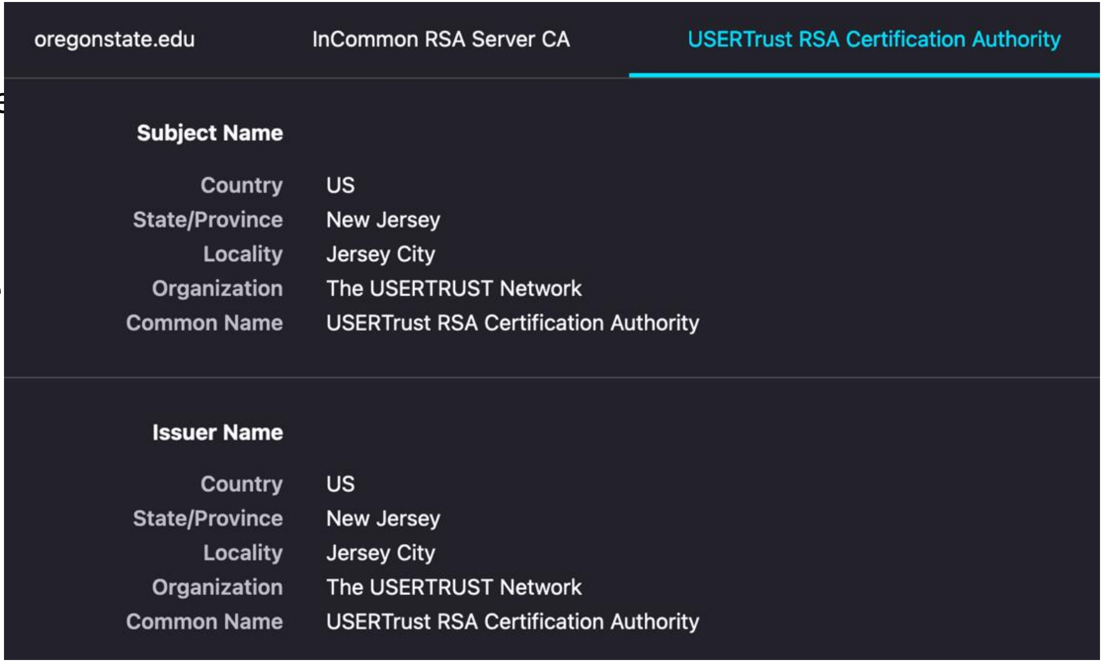
TRUST CHAIN – CONT'D

- "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

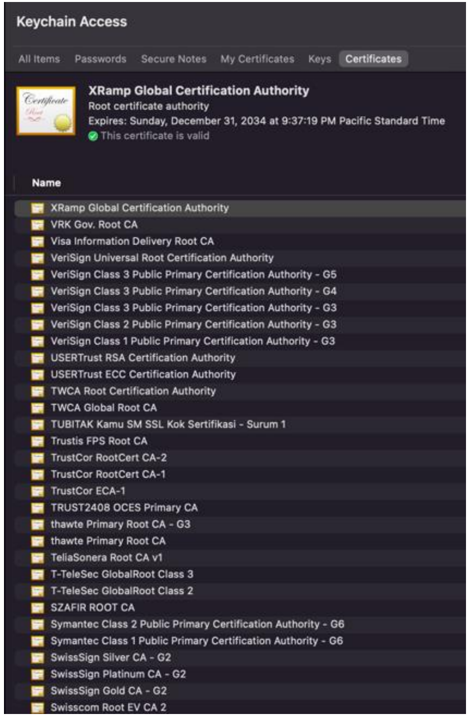
- An example:
  - Student
  - Oregon resident
  - U.S. Citizen

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..

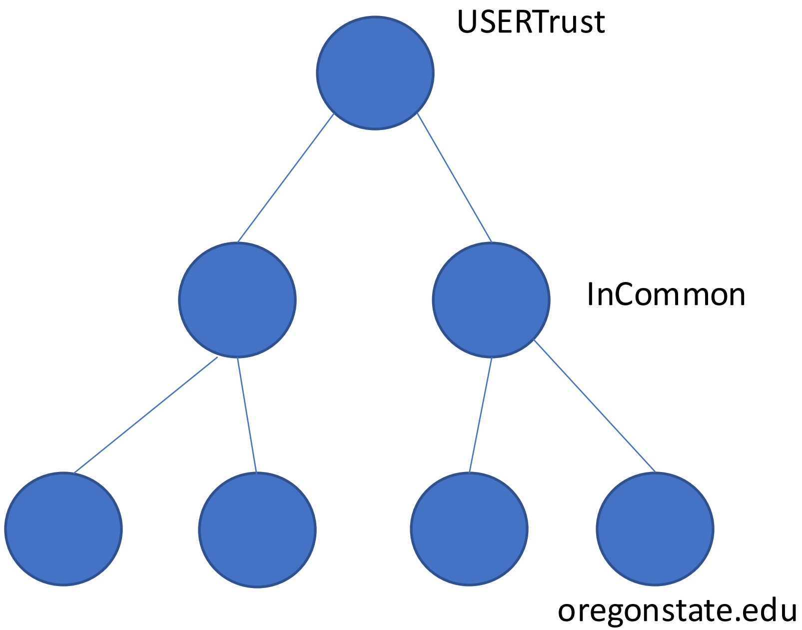
Root certificate authority (Root CA ≈ US in prev. example)

- 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



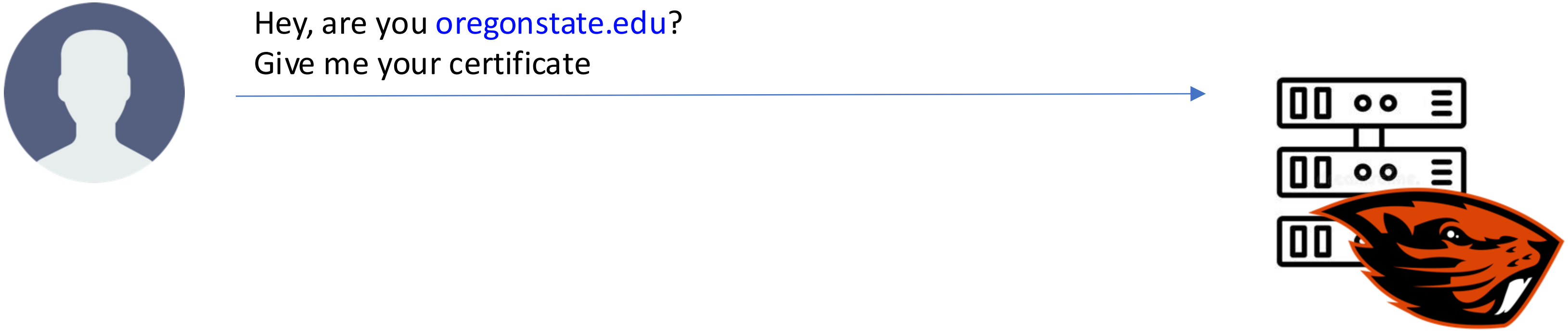
PUBLIC KEY INFRASTRUCTURE (PKI)

- 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!

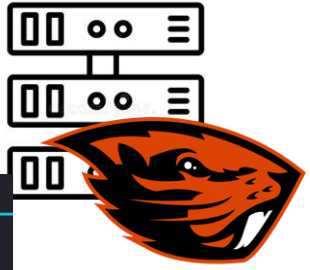


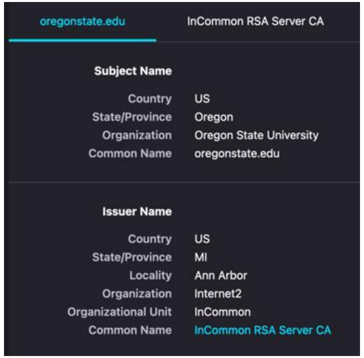
LET'S VERIFY OREGONSTATE.EDU

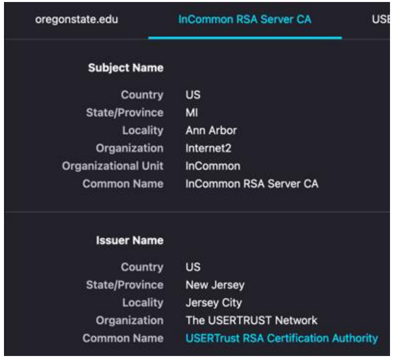
- Using the digital certificate!

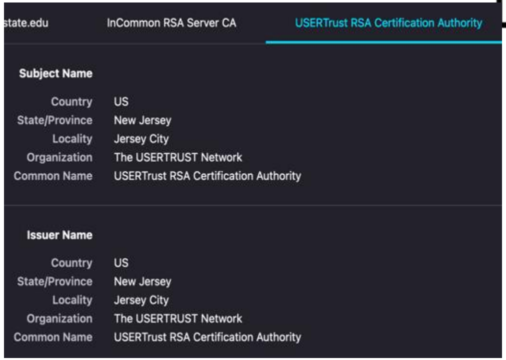
Hey, are you oregonstate.edu?
Give me your certificate

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







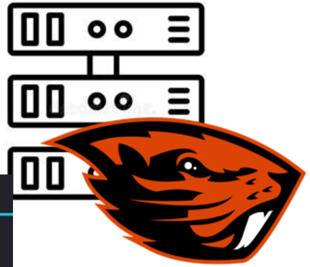


LET'S VERIFY OREGONSTATE.EDU

- 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

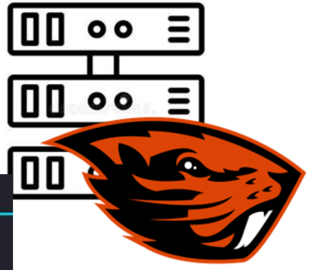
Oregonstate verified by InCommon RSA

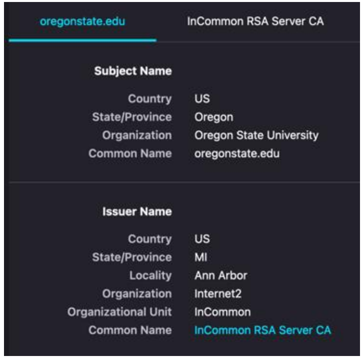
LET'S VERIFY OREGONSTATE.EDU

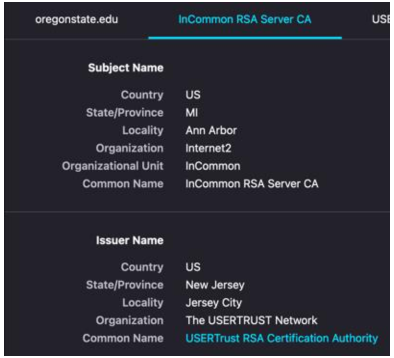
- Using the digital certificate!

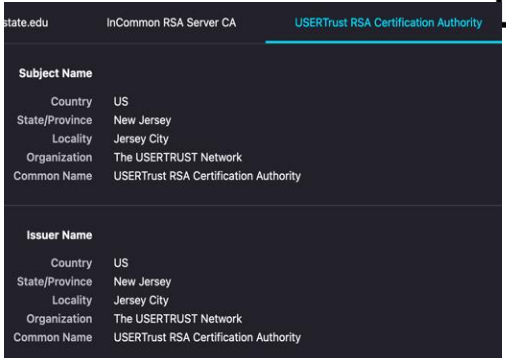
Hey, are you oregonstate.edu?
Give me your certificate

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









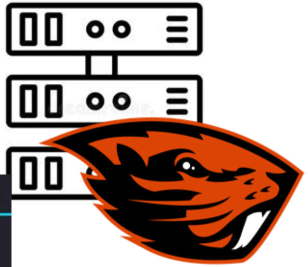
InCommon RSA verified by USERTrust RSA

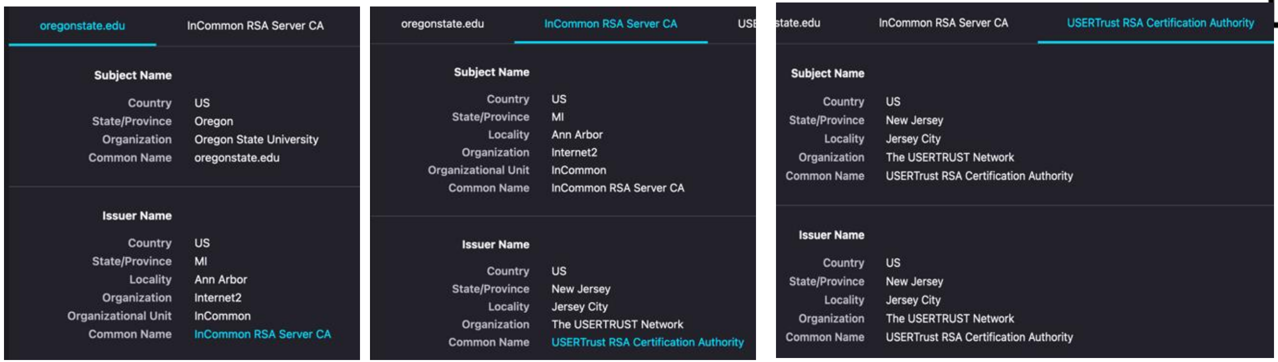
LET'S VERIFY OREGONSTATE.EDU

- Using the digital certificate!

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

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





USERTrust RSA is self-verified (ROOT CA)

Thank You!

Sanghyun Hong
https://secure-ai.systems/courses/Sec-Grad/current

Oregon State University

SAIL
Secure AI Systems Lab