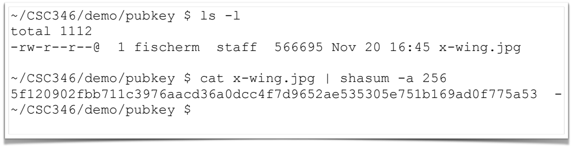
Public Key Cryptography

Some crypto is actually really useful!

Hashing for Fun and Security

Hash Functions

- A **hash function** is a mathematical function that turns a very large input into smaller integer. It must be fast to compute. (most of the time, see passwords)
  - Example: SHA256 turns any file (of any size) to a 256-bit number



```
~/CSC346/demo/pubkey $ ls -l
total 1112
-rw-r--r--@  1 fischerm  staff  566695 Nov 20 16:45 x-wing.jpg

~/CSC346/demo/pubkey $ cat x-wing.jpg | shasum -a 256
5f120902fbb711c3976aacd36a0dcc4f7d9652ae535305e751b169ad0f775a53  -
~/CSC346/demo/pubkey $
```

Hashing for Fun and Security

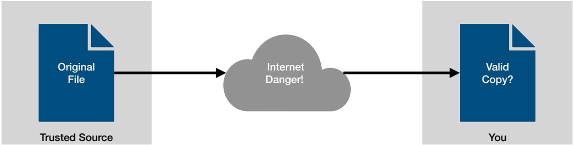
Hash Functions

- The best hash functions are irreversible, meaning that there is no known way to find a file, if you are given the hash
- Always possible to do a brute-force search, but if the output is very large, it is impossible to complete in time
  - 256 bits = 2^{256} possible hash
    = more than the atoms in the universe

Hashing for Fun and Security

Verification

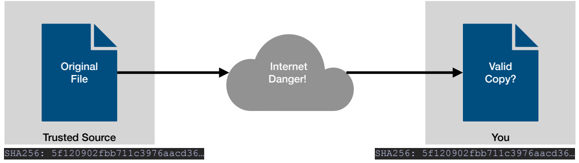
- Suppose you have a file, which you hope is the right version, but you worry that an attacker has corrupted it.



Hashing for Fun and Security

Verification

- If you can get the hash of the file (in a trusted way), then you can use that to confirm you have the right file.



```
SHA256: 5f120902fbb711c3976aacd36…
```

```
SHA256: 5f120902fbb711c3976aacd36…
```

Hashing for Fun and Security

Verification

- Using hashes for verification is very common
  - Distribute a file through an untrusted source (shared distro site, BitTorrent, etc.)
  - Display the hash on a trusted website
- Remember Password Authentication?
  - Compare login password hash to stored hash value
  - Having “slow” hashes for passwords is actually a security feature! Helps mitigate brute force attacks

Symmetric Cryptography

- A **symmetric cipher** is an encryption algorithm where the encoder and decoder use the same key
  - Often fast
  - Only option until a few decades ago
- Examples:
  - Substitution ciphers
  - WW2 “Enigma” machines

Symmetric Cryptography

- Since the endpoints have to share keys, there are several problems:
  - Have to communicate the key beforehand, safely (hard to do on the Internet!)
  - Have to create different keys for each (a,b) pair
    - Every computer would have to store 1000s of keys (one for each partner)

Public Key Infrastructure to the Rescue

Big Brains

- Public key encryption algorithms require **two different keys** to encrypt & decrypt a message.
  - If you attempt to decrypt using the same key as you encrypted, you get garbage
- These algorithms work in both directions:
  - A will decrypt data encoded by B, and
  - B will decrypt data encoded by A

Public Key Infrastructure

Public & Private Keys

- In any **key pair**, we designate one as public, and one as private.
- We assume that the public key is accessible to **anyone, anywhere**
  - “Even my worst enemy”
- The private key must be **absolutely private**, not shared with anyone

Public Key Infrastructure

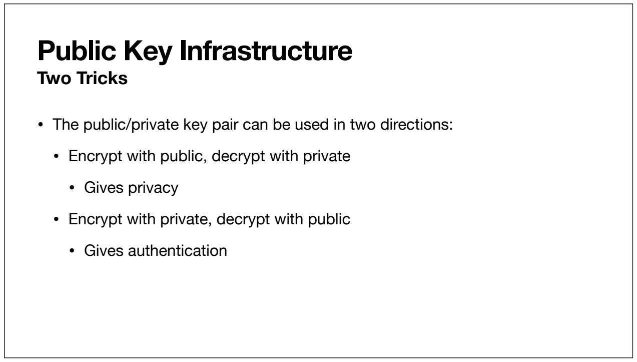
Public & Private Keys

- It’s easy to generate new key pairs. You probably have one for:
  - Each web browser you use (each browser on your computer will have a different key pair)
  - Each library that supports https
  - Connecting to servers via ssh
  - Each https server you create
  - and many more…

Public Key Infrastructure

Public & Private Keys

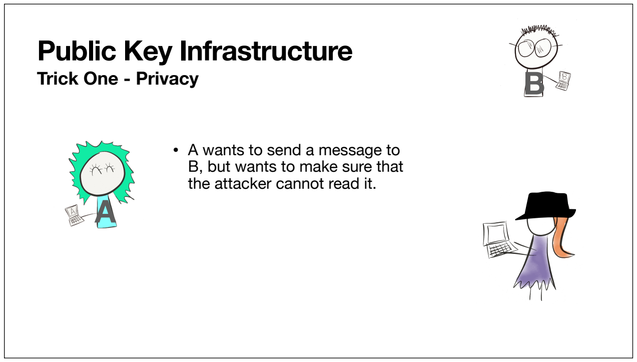
- However, we assume that each endpoint in a communication has the same key pair, so long as the communication is ongoing.
- Not normally necessary to re-generate key pairs unless they’re stolen; you can use the same one for a very long time.
  - SSH Key-Pairs are usually very long lived
  - TLS *certificates* expire in a year these days, but the *underlying Key-Pair* can be used again to generate a new certificate.



Public Key Infrastructure

Two Tricks

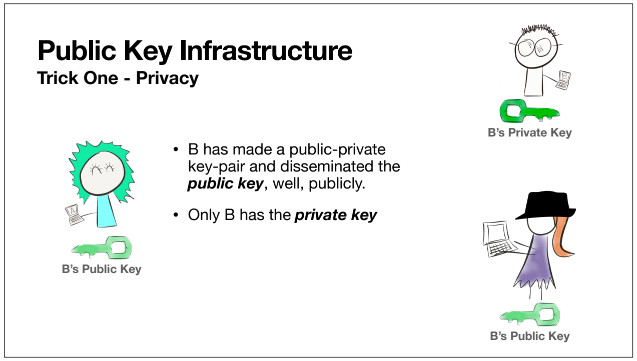
- The public/private key pair can be used in two directions:
  - Encrypt with public, decrypt with private
    - Gives privacy
  - Encrypt with private, decrypt with public
    - Gives authentication



Public Key Infrastructure

Trick One - Privacy

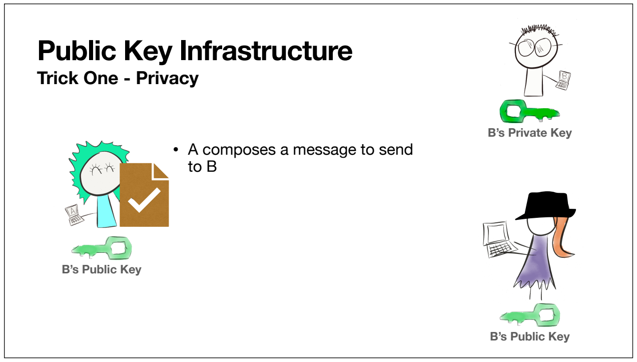
- A wants to send a message to B, but wants to make sure that the attacker cannot read it.



Public Key Infrastructure

Trick One - Privacy

- B has made a public-private key-pair and disseminated the ***public key***, well, publicly.
- Only B has the ***private key***



Public Key Infrastructure

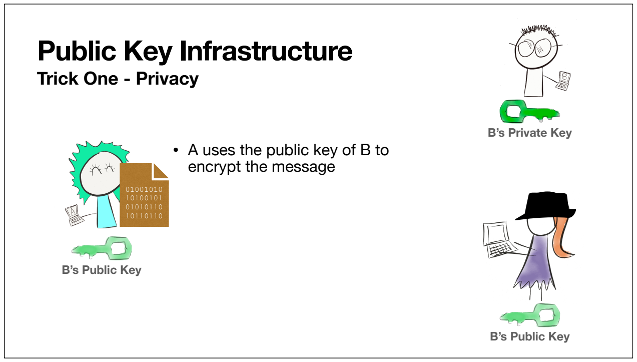
Trick One - Privacy

- A composes a message to send to B

B's Private Key

B's Public Key

B's Public Key



Public Key Infrastructure

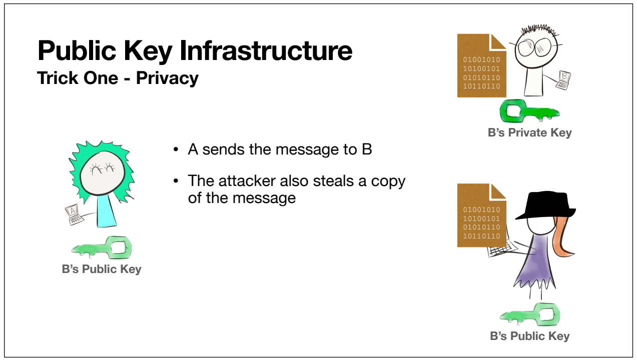
Trick One - Privacy

- A uses the public key of B to encrypt the message

B's Private Key

B's Public Key

B's Public Key



Public Key Infrastructure

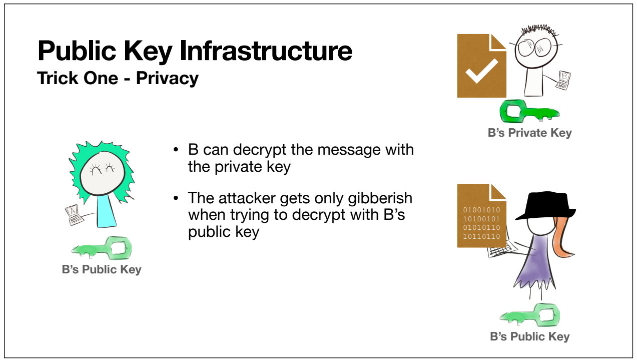
Trick One - Privacy

- A sends the message to B
- The attacker also steals a copy of the message

B's Private Key

B's Public Key

B's Public Key



Public Key Infrastructure

Trick One - Privacy

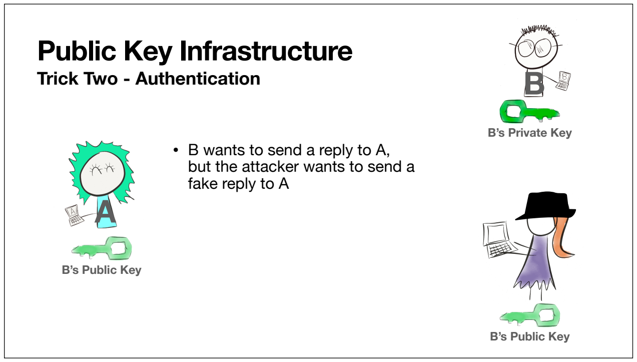
- B can decrypt the message with the private key
- The attacker gets only gibberish when trying to decrypt with B's public key

B's Private Key

01001010
10100101
01010110
10110110

B's Public Key

B's Public Key



Public Key Infrastructure

Trick Two - Authentication

- B wants to send a reply to A, but the attacker wants to send a fake reply to A

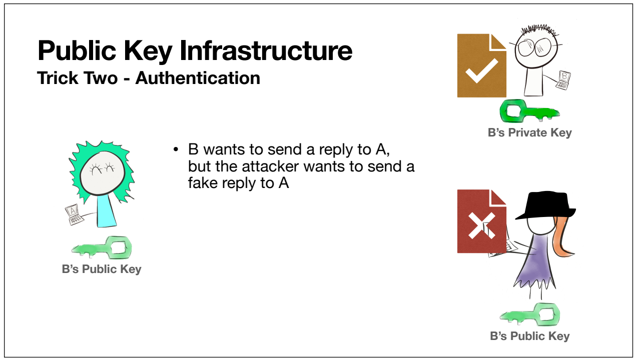
B

B's Private Key

A

B's Public Key

B's Public Key



Public Key Infrastructure

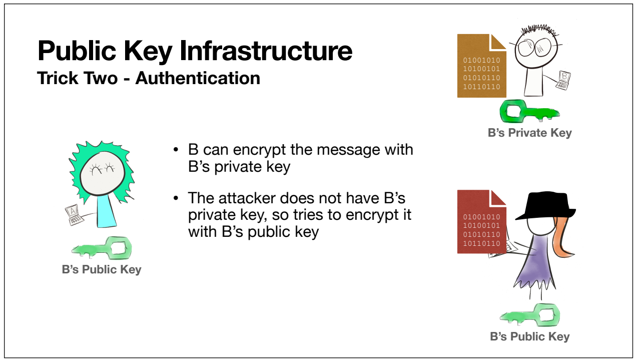
Trick Two - Authentication

- B wants to send a reply to A, but the attacker wants to send a fake reply to A

B's Private Key

B's Public Key

B's Public Key

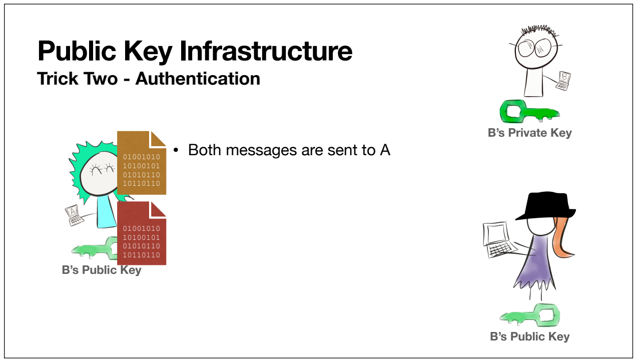


Public Key Infrastructure

Trick Two - Authentication

- B can encrypt the message with B's private key
- The attacker does not have B's private key, so tries to encrypt it with B's public key

01001010
10100101
01010110
10110110
B's Private Key
B's Public Key
01001010
10100101
01010110
10110110
B's Public Key

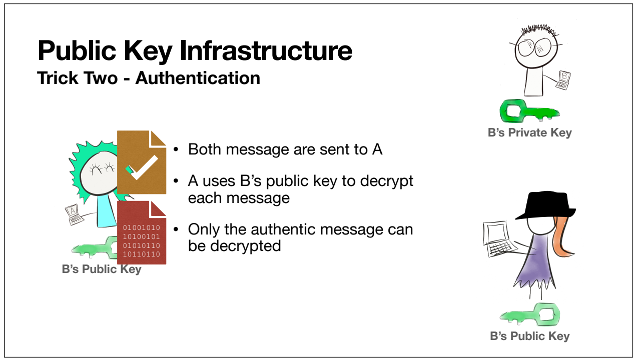


Public Key Infrastructure

Trick Two - Authentication

- Both messages are sent to A

B's Private Key
01001010
10100101
01010110
10110110
01001010
10100101
01010110
10110110
B's Public Key
B's Public Key

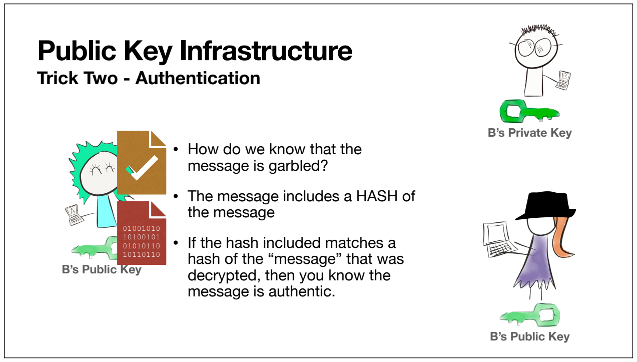


Public Key Infrastructure

Trick Two - Authentication

- Both message are sent to A
- A uses B's public key to decrypt each message
- Only the authentic message can be decrypted

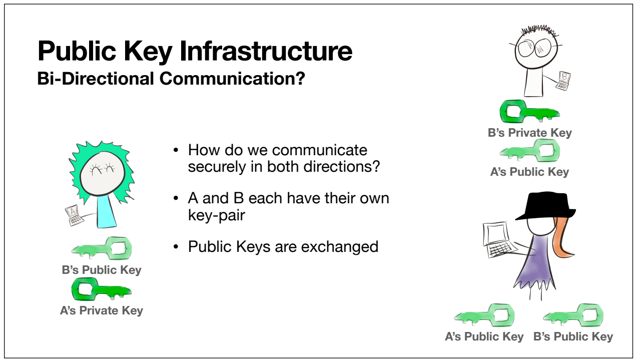
B's Private Key
01001010
10100101
01010110
10110110
B's Public Key
B's Public Key



Public Key Infrastructure

Trick Two - Authentication

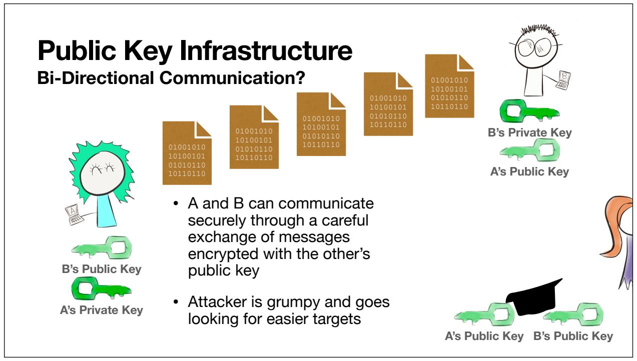
- How do we know that the message is garbled?
- The message includes a HASH of the message
- If the hash included matches a hash of the "message" that was decrypted, then you know the message is authentic.



Public Key Infrastructure

Bi-Directional Communication?

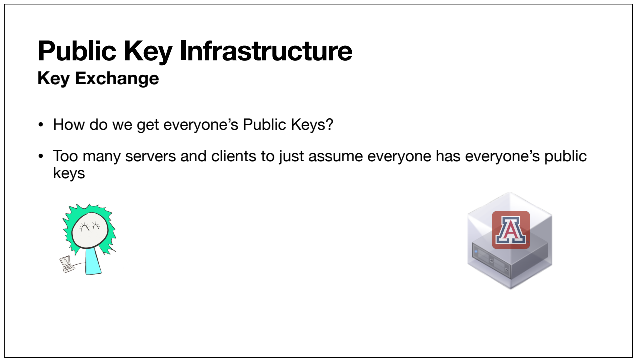
- How do we communicate securely in both directions?
- A and B each have their own key-pair
- Public Keys are exchanged



Public Key Infrastructure

Bi-Directional Communication?

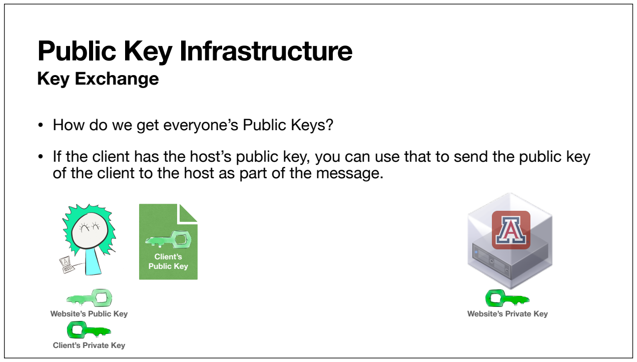
- A and B can communicate securely through a careful exchange of messages encrypted with the other's public key
- Attacker is grumpy and goes looking for easier targets



Public Key Infrastructure

Key Exchange

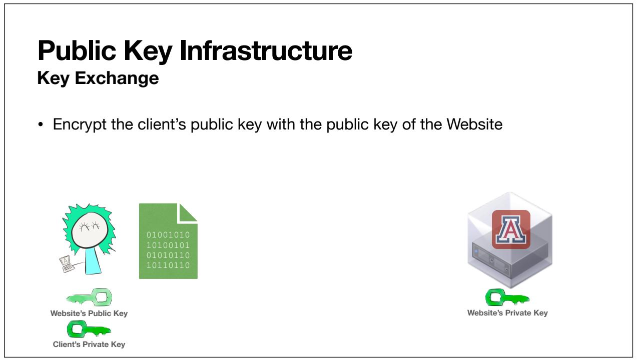
- How do we get everyone’s Public Keys?
- Too many servers and clients to just assume everyone has everyone’s public keys



Public Key Infrastructure

Key Exchange

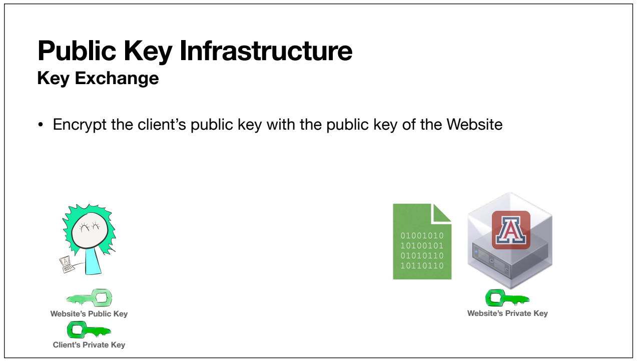
- How do we get everyone’s Public Keys?
- If the client has the host’s public key, you can use that to send the public key of the client to the host as part of the message.



Public Key Infrastructure

Key Exchange

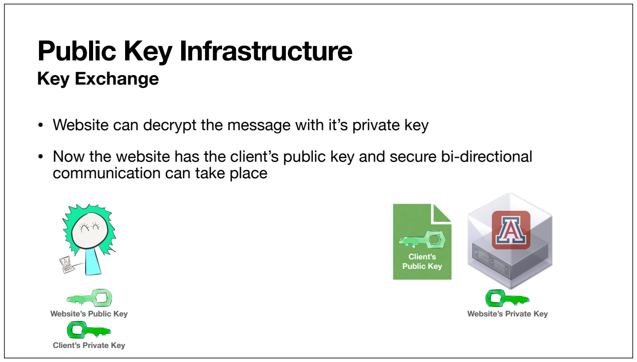
- Encrypt the client’s public key with the public key of the Website



Public Key Infrastructure

Key Exchange

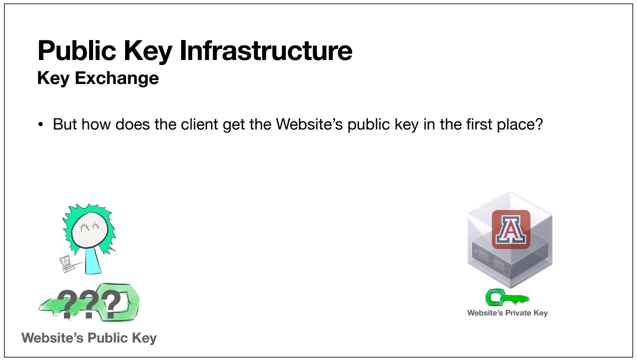
- Encrypt the client’s public key with the public key of the Website



Public Key Infrastructure

Key Exchange

- Website can decrypt the message with it’s private key
- Now the website has the client’s public key and secure bi-directional communication can take place



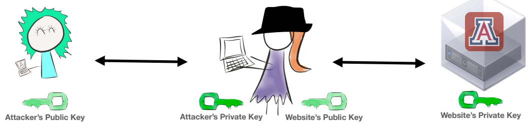
Public Key Infrastructure

Key Exchange

- But how does the client get the Website’s public key in the first place?

Public Key Infrastructure
Key Exchange

- We can't just ask for it. How could we trust that it's really the website giving us the public key?
- Attacker could impersonate the website and we may not know.

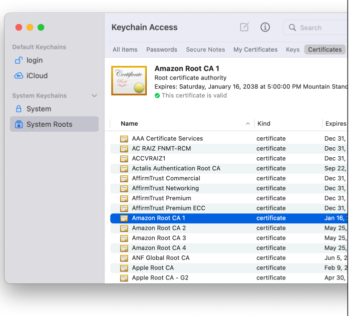


Certificates
Certificate Authority

- A **certificate** is a document, signed by a trusted third party – a "**certificate authority**" or CA – that tells us what the public key of a server is.
- We use certificates, instead of talking to the CA directly, because the load on the CA would be immense.

Certificates
Certificate Authority

- While it is not possible for a client to innately know the public key for every conceivable service out there, it is possible for the client to have innate knowledge or trust for a smaller fixed set of keys.
- Your Operating System ships with an initial set of trusted keys in the form of root or intermediary certificates.



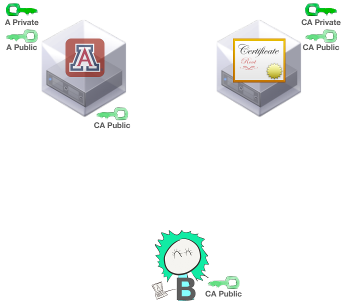
Amazon Root CA 1
Root certificate authority
Expires: Saturday, January 16, 2038 at 5:00:00 PM Mountain Stand
This certificate is valid

Name	Kind	Expires
AAA Certificate Services	certificate	Dec 31,
AC RAIZ FNMT-RCM	certificate	Dec 31,
ACCVRAIZ1	certificate	Dec 31,
Actalis Authentication Root CA	certificate	Sep 22,
AffirmTrust Commercial	certificate	Dec 31,
AffirmTrust Networking	certificate	Dec 31,
AffirmTrust Premium	certificate	Dec 31,
AffirmTrust Premium ECC	certificate	Dec 31,
Amazon Root CA 1	certificate	Jan 16,
Amazon Root CA 2	certificate	May 25,
Amazon Root CA 3	certificate	May 25,
Amazon Root CA 4	certificate	May 25,
ANF Global Root CA	certificate	Jun 5, 2
Apple Root CA	certificate	Feb 9, 2
Apple Root CA - G2	certificate	Apr 30,

Certificates

Certificate Authority

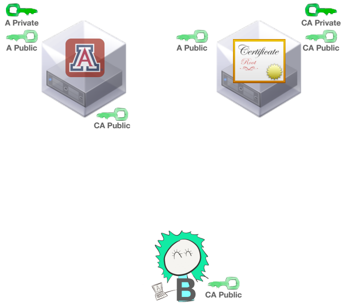
- A and B both trust C, a certain Certificate Authority.
- When we begin, both A and B know C's public key, but they do not know each other's.
- We want B to come to know A's public key – after which we can communicate (as shown earlier).



Certificates

Certificate Authority

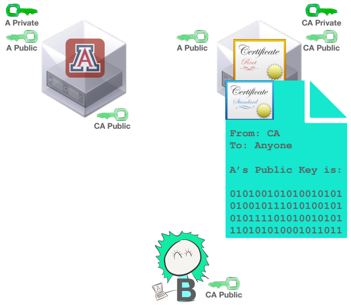
- A first demonstrates to the CA that it is who it says it is
- This is usually done through DNS records
- A can then send it's Public Key to the CA



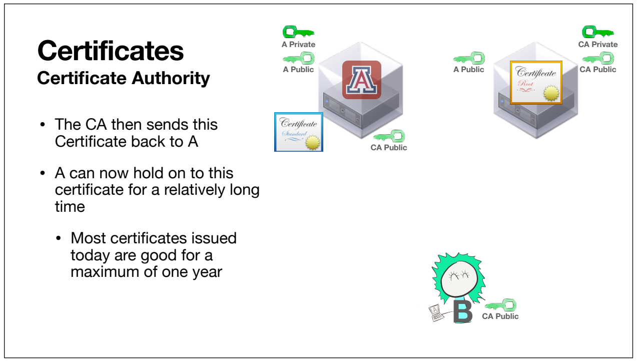
Certificates

Certificate Authority

- Once the CA knows A's Public Key, the CA creates a message that basically says "As the CA, you can trust that this is indeed A's Public Key"
- The CA includes a Hash of the message in a Certificate that is signed with the CA's Private Key



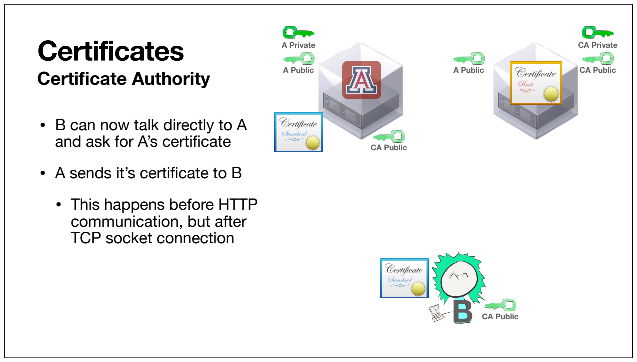
```
From: CA
To: Anyone

A's Public Key is:

0101001010101001010 1
01001011101010100101
01011110101001010 1
11010101000101 1011
```



Certificates

Certificate Authority

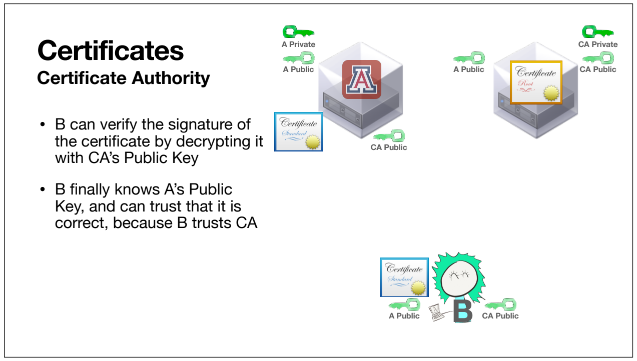
- The CA then sends this Certificate back to A
- A can now hold on to this certificate for a relatively long time
  - Most certificates issued today are good for a maximum of one year



Certificates

Certificate Authority

- B can now talk directly to A and ask for A's certificate
- A sends it's certificate to B
  - This happens before HTTP communication, but after TCP socket connection



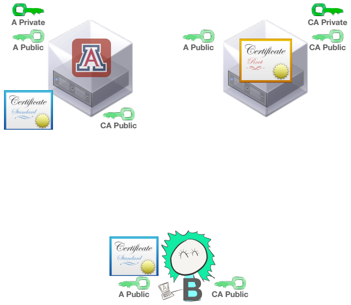
Certificates

Certificate Authority

- B can verify the signature of the certificate by decrypting it with CA's Public Key
- B finally knows A's Public Key, and can trust that it is correct, because B trusts CA

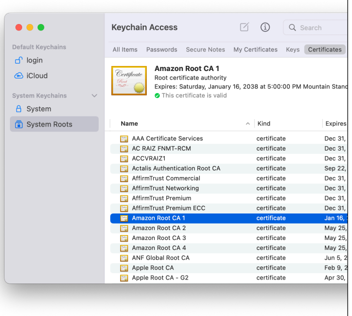
Certificates
Chain of Trust

- This is a very basic example of the Chain of Trust
- “I can trust this new thing, because it is signed by something I already trust”



Certificates
Chain of Trust

- Compared to the number of hosts on the internet, there are only a very small number of Root Certificate Authorities
- My Laptop knows and trusts 149 Root Certificates in 2024
- In 2022 it was a 163 Root Certificates. OS vendors take on the responsibility to maintain this initial trust list.
- You can add your own if needed.

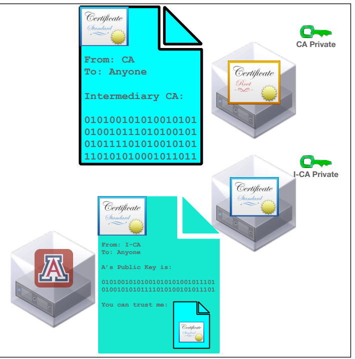


Amazon Root CA 1
Root certificate authority
Expires: Saturday, January 16, 2038 at 5:00:00 PM Mountain Stanc
✓ This certificate is valid

Name	Kind	Expires
AAA Certificate Services	certificate	Dec 31,
AC RAIZ FNMT-RCM	certificate	Dec 31,
ACCVRAIZ1	certificate	Dec 31,
Actalis Authentication Root CA	certificate	Sep 22,
AffirmTrust Commercial	certificate	Dec 31,
AffirmTrust Networking	certificate	Dec 31,
AffirmTrust Premium	certificate	Dec 31,
AffirmTrust Premium ECC	certificate	Dec 31,
Amazon Root CA 1	certificate	Jan 16,
Amazon Root CA 2	certificate	May 25,
Amazon Root CA 3	certificate	May 25,
Amazon Root CA 4	certificate	May 25,
ANF Global Root CA	certificate	Jun 5, 2
Apple Root CA	certificate	Feb 9, 2
Apple Root CA - G2	certificate	Apr 30,

Certificates
Chain of Trust

- A few hundred Root CAs cannot deal with Billions of requests for new certificates
- Therefor there are Intermediary Certificate Authorities
- Organizations that the Root CAs trust to sign certificates on their behalf



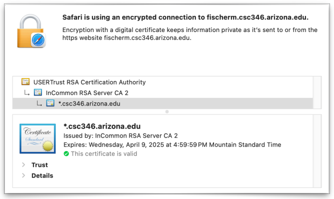
```
From: CA
To: Anyone

Intermediary CA:

01010010101010010101
01001011101010100101
01011110101010010101
11010101010001011011
```

```
From: I-CA
To: Anyone

A’s Public Key is:

0101001010100101010101001011101
0100101010111101010100101011101

You can trust me:
```

Certificates

Chain of Trust

- The Certificate for our class was signed by the InCommon RSA Intermediary CA
- The InCommon RSA CA was in turn signed by the USERTrust RSA CA
- The USERTrust RSA CA is one of the initial Root CAs that come with most OS installations
- So our browser trusts the certificate



Safari is using an encrypted connection to fischerm.csc346.arizona.edu.

Encryption with a digital certificate keeps information private as it's sent to or from the https website fischerm.csc346.arizona.edu.

USERTrust RSA Certification Authority
└ InCommon RSA Server CA 2
　└ *.csc346.arizona.edu

***.csc346.arizona.edu**

Issued by: InCommon RSA Server CA 2

Expires: Wednesday, April 9, 2025 at 4:59:59 PM Mountain Standard Time

✓ This certificate is valid