

# Asymmetric Cryptography

---

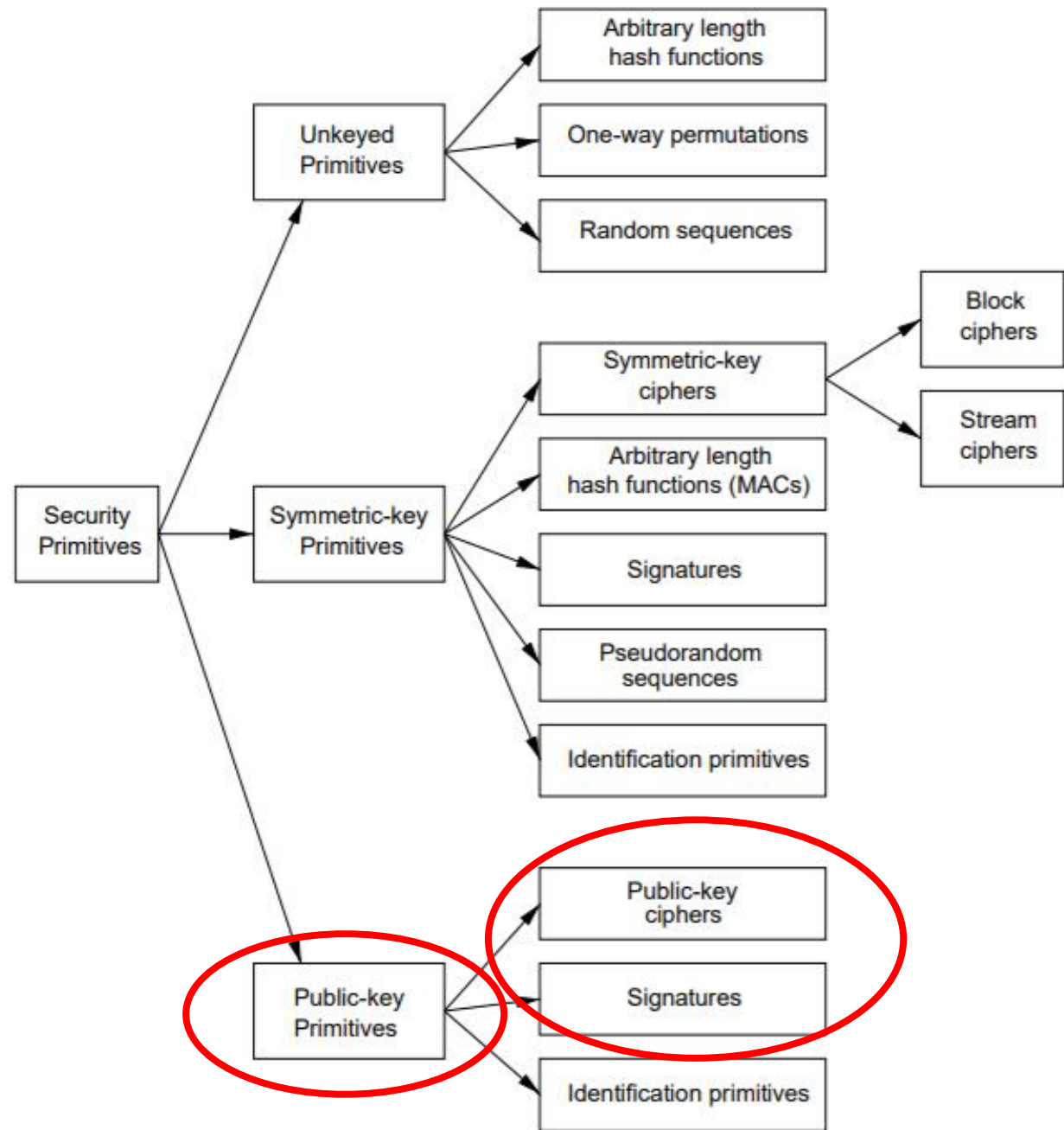
UT CS361S

SPRING 2021

LECTURE NOTES



# Technology Review



**Figure 1.1:** A taxonomy of cryptographic primitives.

# Asymmetric Cryptography

---

Keys come in pairs

Public key can be shared

Private key **MUST** be kept secret

# Uses of Asymmetric Crypto

---

Unlike symmetric, what you can **DO** with asymmetric depends greatly on the algorithm

- RSA – encryption (crypto dropbox), signatures
- ECDSA/DSA – signatures
- Diffie Hellman – key agreement

**For today's class, we will focus on RSA encryption and RSA signature, as well as Diffie Hellman key agreement.**

# RSA Encryption

---

Encrypt SHORT MESSAGES with public key, decrypt with private key

Encrypted Communication between A and B

- A and B have each other's public key
- A encrypts a message for B under B's public key
- B responds by sending A a response under A's public key

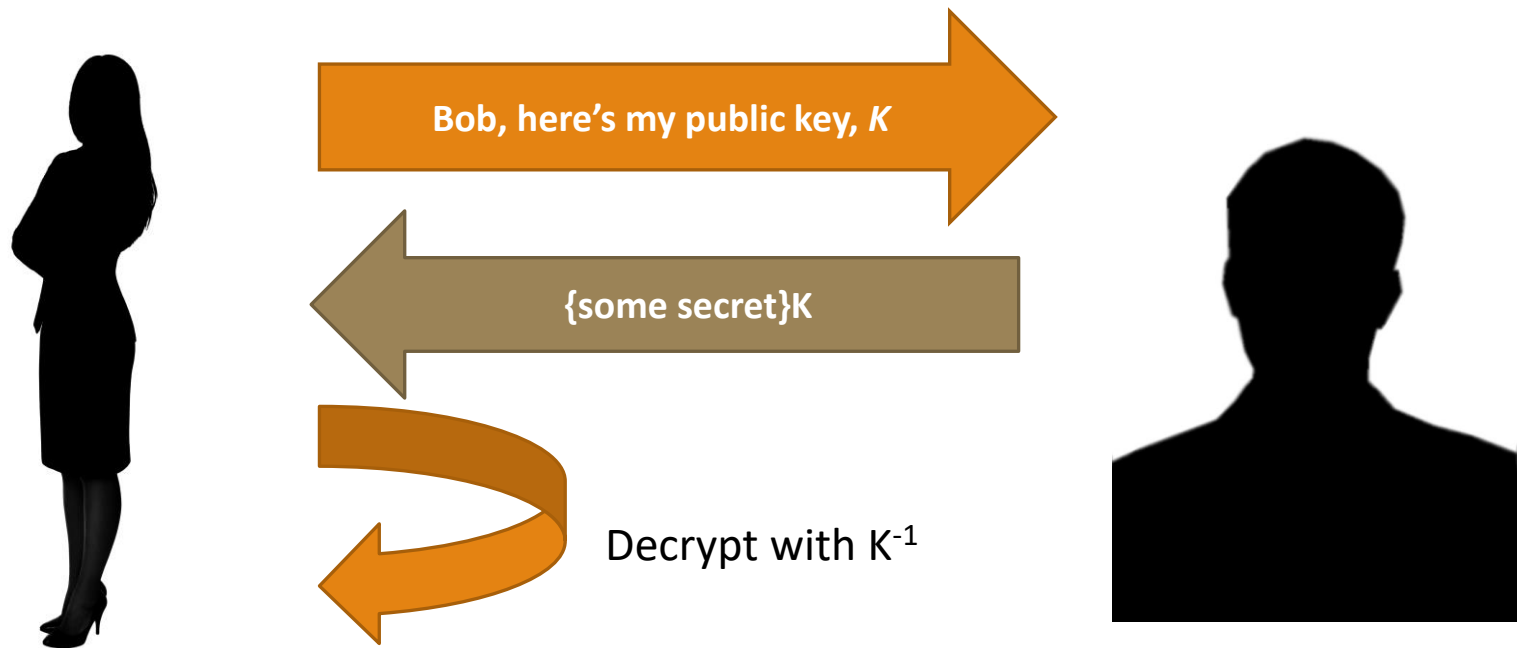
Works fine but...

- It is very slow (asymmetric encryption/decryption is expensive)

***Used almost exclusively for Key Transfer*** (sending a symmetric session key)

# RSA Encrypt Visualization

---



# RSA Signatures

---

RSA encrypts with the PRIVATE KEY for a signature

Step 1: Publisher produces a message  $M$

Step 2: Publisher takes the hash of  $M$   $h(M)$

Step 3: Publisher encrypts the hash with the private key  $\{h(M)\}_{k^{-1}}$

Step 4: Publisher transmits Message  $M$  and  $\{h(M)\}_{k^{-1}}$  as the signature

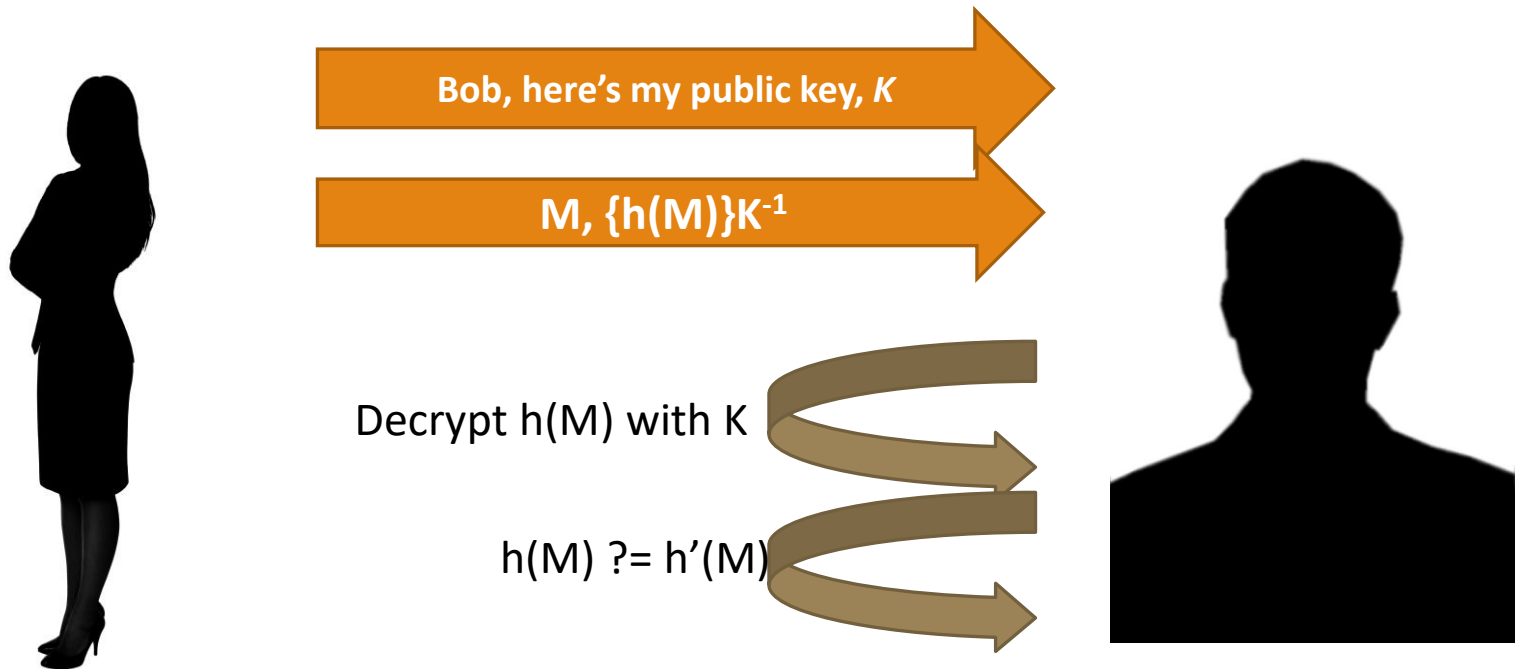
Step 5: A verifier decrypts  $h(M)$  with Publisher's public key

Step 6. A verifier computes their own hash of  $M$   $h'(M)$

Step 7: A verifier determines the signature is valid if  $h'(M) = h(M)$

# RSA Signature

---





# Key Exchange

---

Asymmetric crypto is not good for “bulk data” encryption

RSA can only encrypt small messages SLOWLY.

Other asymmetric algorithms CANT ENCRYPT AT ALL

So, asymmetric is used to authenticate ***KEY EXCHANGE***

There are two forms:

- Key Transfer
- Key Agreement

# Key Transfer

---

Requires asymmetric encryption (e.g., RSA)

Create a session key

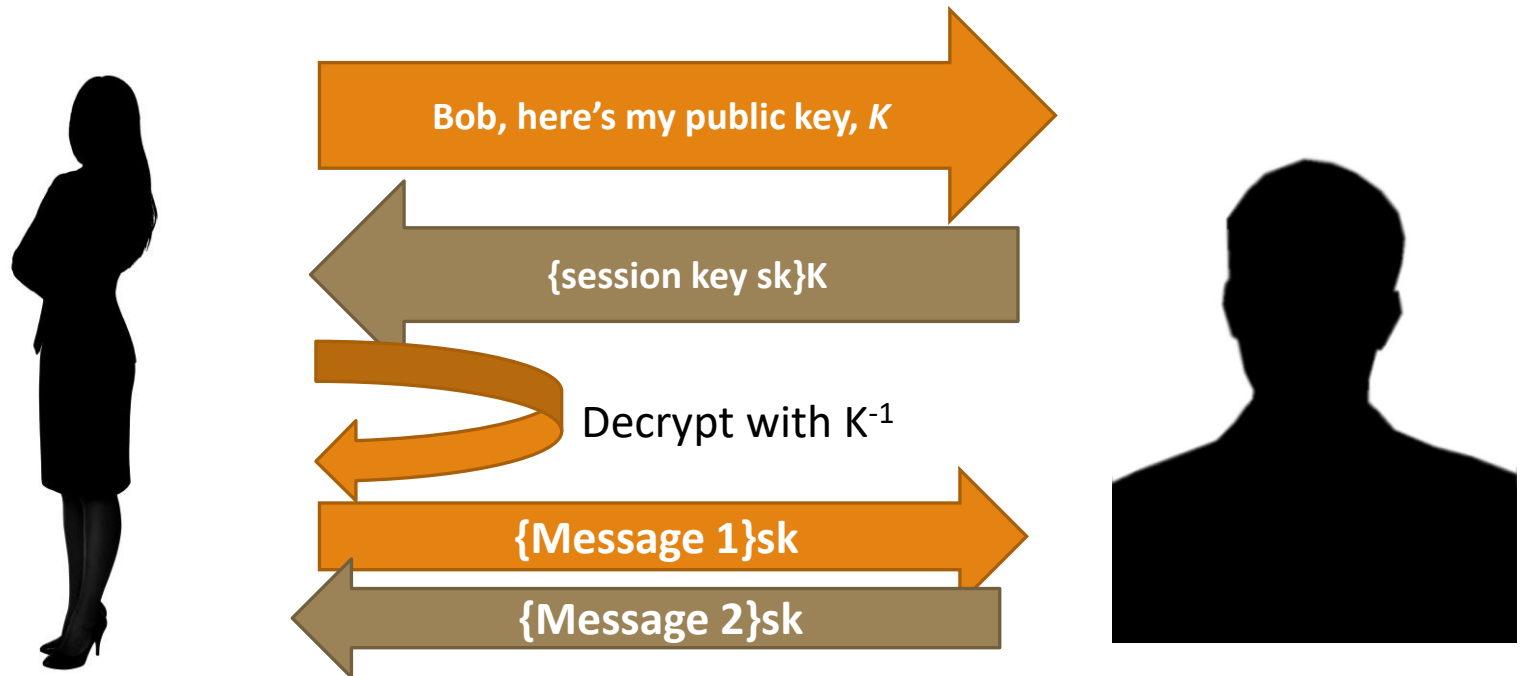
Send session key encrypted with public key

Only party possessing the private key can decrypt it

(Automatically authenticated)

# RSA Key Transport

---



# RSA Weaknesses

---

## *Insecure when NO PADDING IS USED*

Encryption padding schemes

- PKCS 1.5 (**BROKEN!**)
- OAEP

Signature padding schemes

- PKCS 1.5 (**BROKEN!**)
- PSS

Even though there are non-broken versions, RSA is being phased out

Also, key transfer does not have “forward secrecy”

# Catastrophic Loss of RSA Key

---

Assume A and B want to communicate, E is eavesdropping

A and B use RSA key transfer to exchange session keys

E records thousands of sessions between A and B

After 5 years, A disposes her computer and buys a new one

E steals her computer from the junkyard, finds the private key

ALL PREVIOUSLY RECORDED MESSAGES ARE EXPOSED!

# Diffie Hellman Key Exchange

---

The math version has to do with ***commutative properties***.

Using modulo computations over  $p$  which is a prime with certain properties:

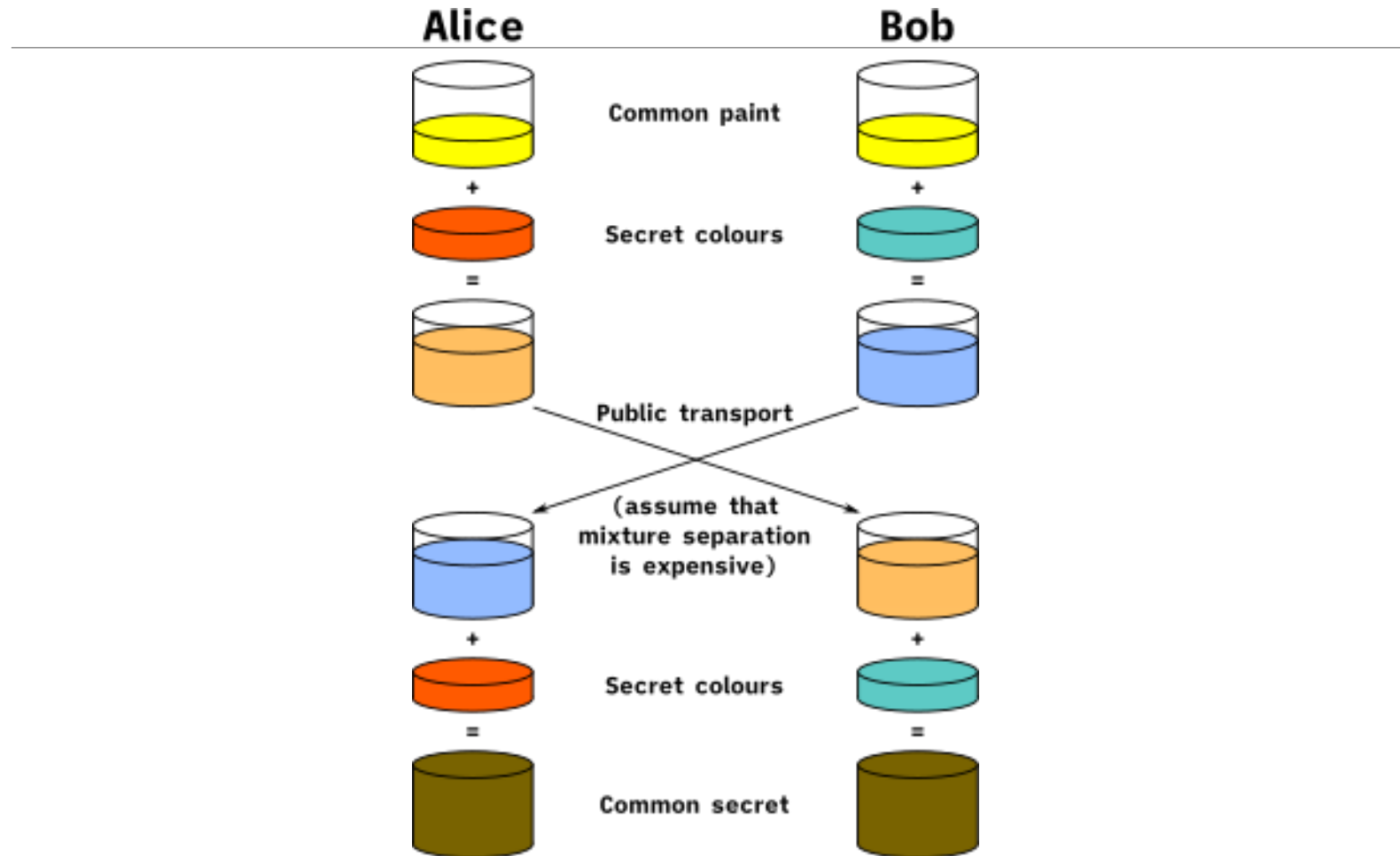
- $A \rightarrow B : g^{RA} \pmod{p}$
- $B \rightarrow A : g^{RB} \pmod{p}$
- $A \rightarrow B : \{M\}g^{RAB}$

A and B are the DH private keys

- Can't be extracted from  $g^{RA} \pmod{p}$
- But, because commutative, can be combined by either side into  $g^{RAB}$

In short, to create a key, exchange DH public keys + parameters

# Wikipedia Visualization



# DHE and Forward Secrecy

---

Diffie Hellman Ephemeral (DHE)

New DH Private Key used for EACH KEY AGREEMENT (session)

RSA key is used to SIGN the DH private key

DHE private key never stored outside of RAM

Now if E steals A's computer, no messages exposed

Compromising a single key exposes only that session

This is "Forward Secrecy"



# No DHE Authentication

---

Next class: how to prove authenticity of a public key

But, spoiler alert!, it HAS to be a long-term key

So, with DHE, you can create keys on the fly (“out of thin air”)

***BUT, you have no idea who they’re coming from!!!***

# Two Asymmetric Steps

---

You caught that there were TWO asymmetric steps for DHE?

First, the DHE is used for key generation

Second, RSA is used to sign (authenticate) the DH public key

There are two asymmetric steps, algorithms, and public keys

# Why not RSA Ephemeral?

---

Why not have a long-term RSA key for signing

And an ephemeral RSA key for each key transfer?

You could create a new RSA key pair each session, just like DH

The problem is that RSA is slow; DH keys are quickly generated

# Other Asymmetric Algorithms

---

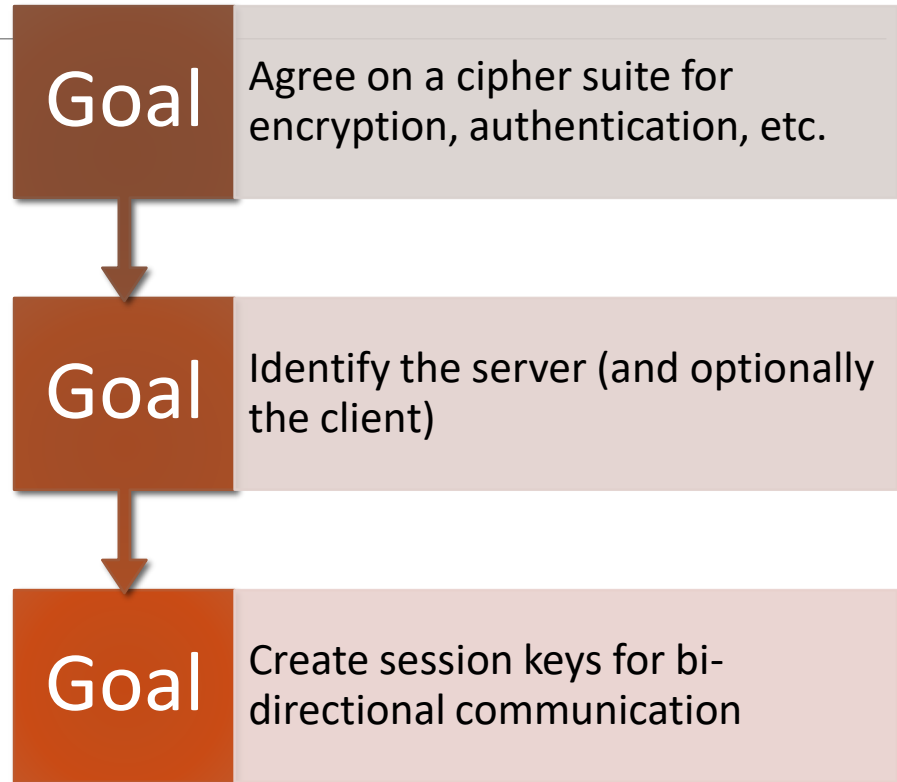
DSA – Just used for signing

ECDH – Elliptic Curve Diffie Hellman (just like DH)

ECDSA – Elliptic Curve DSA (just like DSA)































RSA, DH, DSA, ECDH, ECDSA are the most common I've seen

# TLS 1.2 Handshake



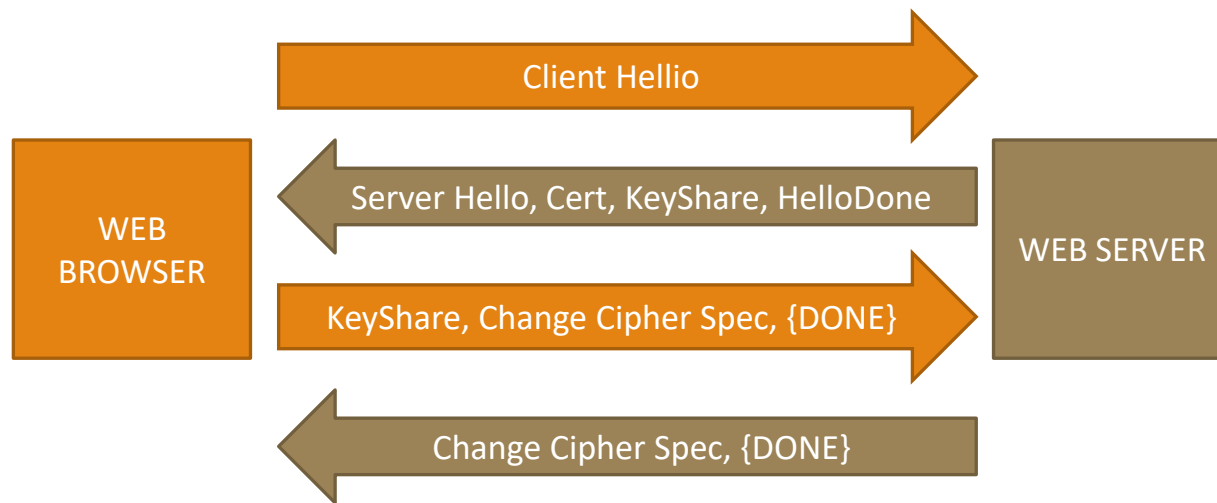
# TLS 1.2 Handshake Review

---

Step	Client	Direction	Message	Direction	Server
1			Client Hello		
2			Server Hello		
3			Certificate		
4			Server Key Exchange		
5			Server Hello Done		
6			Client Key Exchange		
7			Change Cipher Spec		
8			Finished		
9			Change Cipher Spec		
10			Finished		

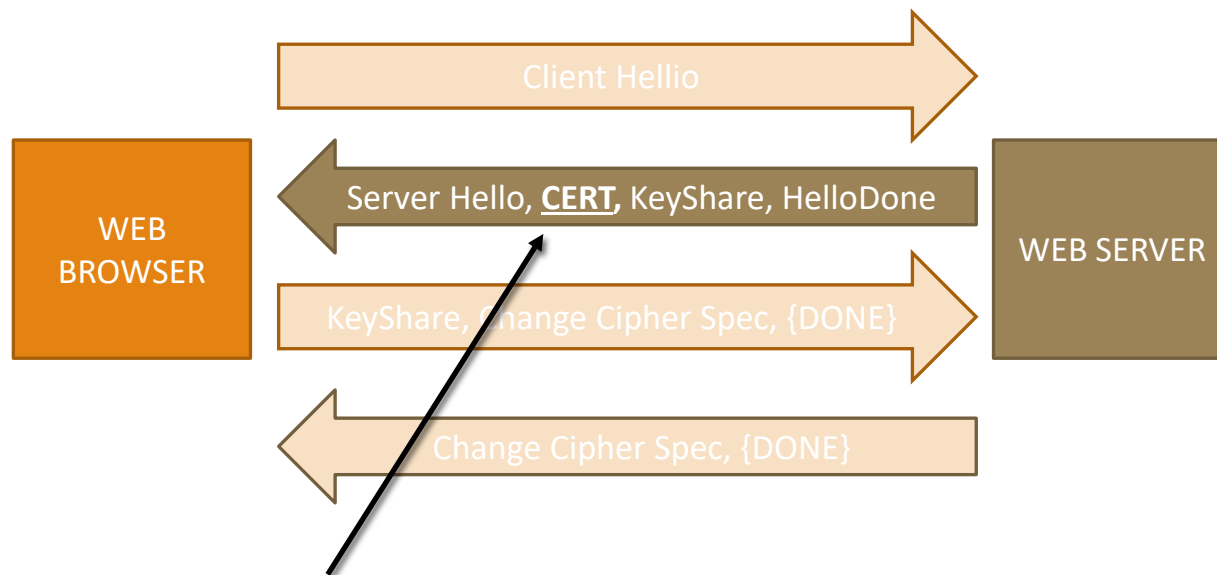
# End-to-End Handshake Visualization #2

---



# Authentication

---



The "Certificate" message includes ONE OR MORE certificates.

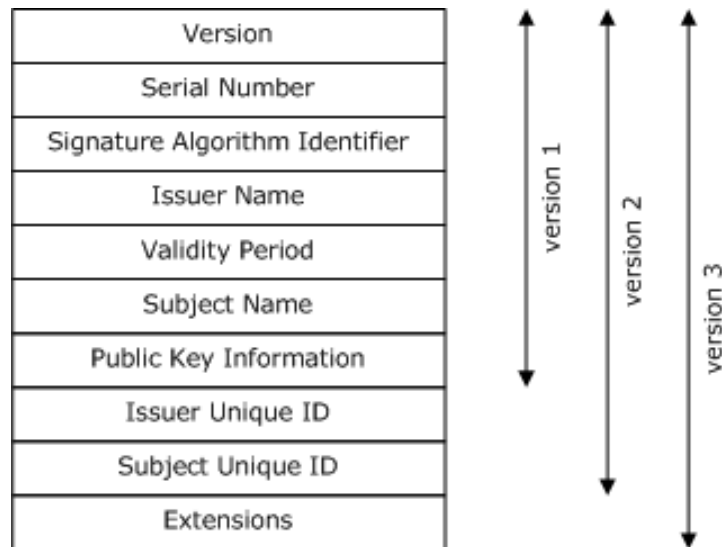


# What is a Certificate?

---

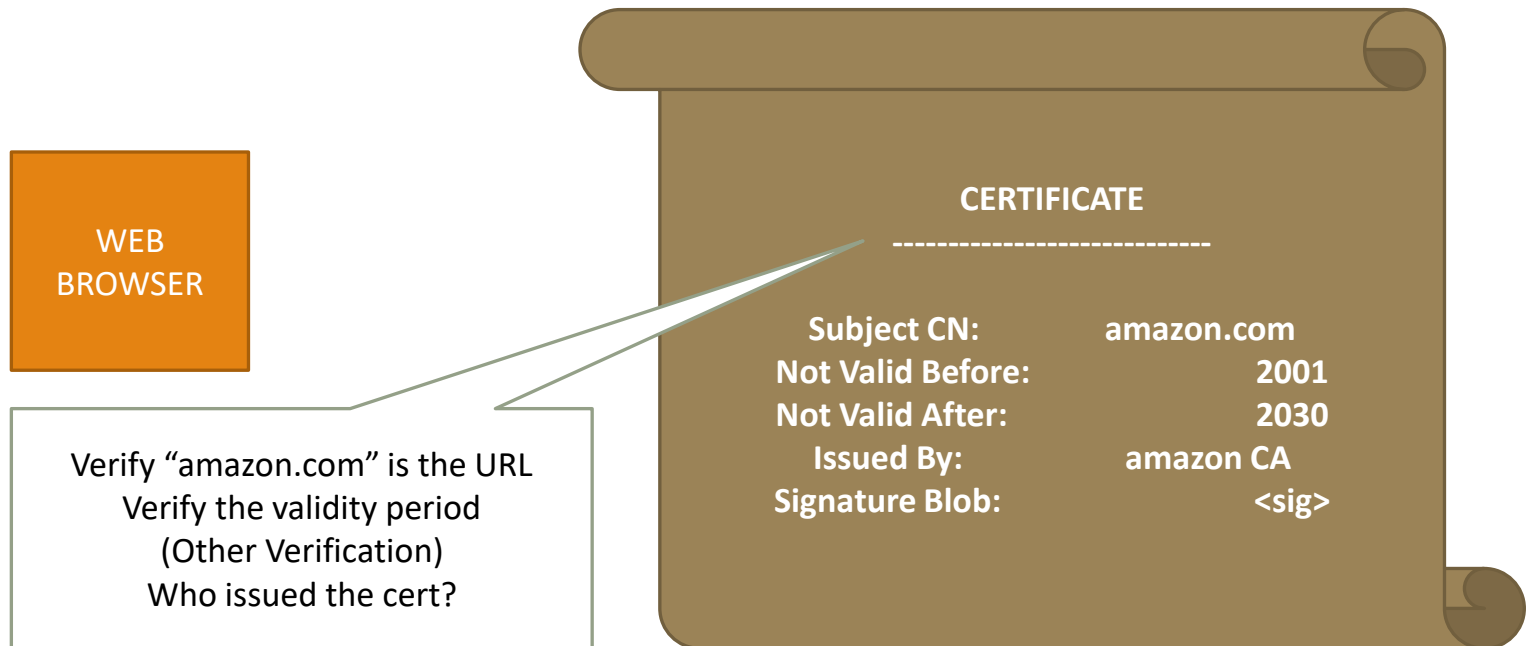
TLS specification (RFC) doesn't specify cert or cert verification

The most common is X 509



# Certificate Verification

---



# Public Key Private Key




PRIVATE KEY

# Certificate Chains

---


The certificate for the Host may be signed by an INTERMEDIATE Certificate Authority

Because the web browser probably doesn't have this intermediate cert, the TLS handshake includes both certificates.



Subject CN: amazon CA  
...  
Issued By: GlobalSign  
Signature Blob: <sig>

This is a stylized representation of a certificate, shown as a brown scroll with rounded corners and a small tab on the right side.

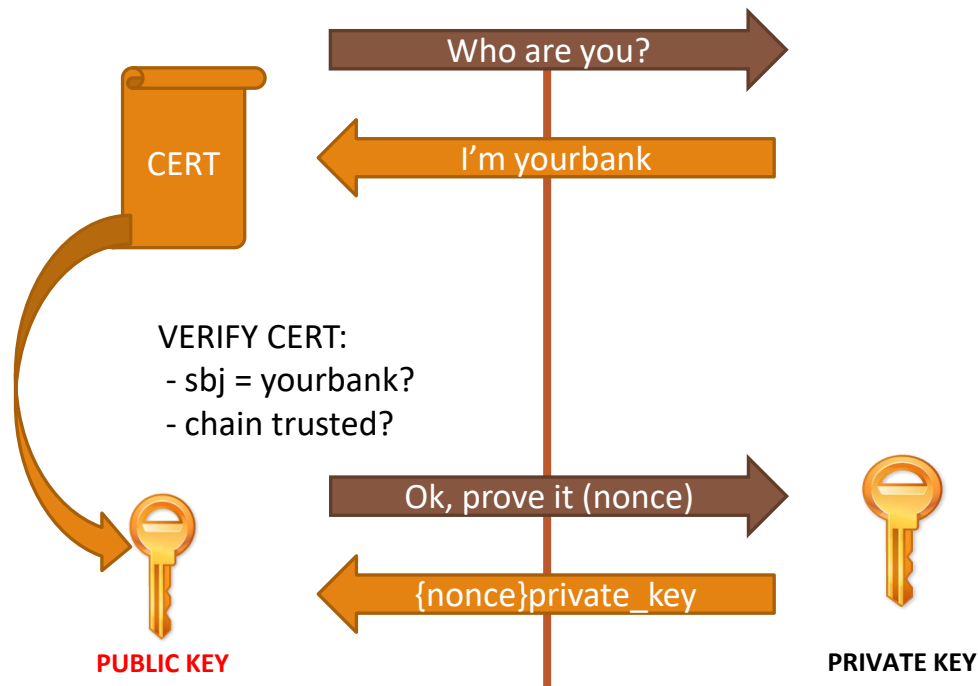


Subject CN: amazon.com  
...  
Issued By: amazon CA  
Signature Blob: <sig>

This is a stylized representation of a certificate, shown as a brown scroll with rounded corners and a small tab on the right side.

# Proving Identity

---



In TLS, the “nonce” is just wrapped up with the other data, such as the client hello, which is all included in the final hash in the finished message.

# Root CA Certificate S

Certificate chains MUST have a  
ROOT

A Root Certificate is SELF  
SIGNED

Browsers trust a set of root  
certificates AXIOMATICALLY

Certificate chains must have a  
trust chain to one of these  
roots.

---

# Trusting Diffie Hellman

Recall that DH keys are EPHEMERAL



The Server's cert includes a long-term public key



The Server's DH key is signed by this key pair



IF the client trusts the cert, THEN it can validate the DH key



---

# TLS Bulk Transport

Both Client  
and Server  
derive keys

Encryption  
keys AND  
MAC keys

MAC's ensure  
continuous  
authentication



When a TLS message is  
Received:

The sender is “proved” by  
the MAC

The MAC is “proved” via MAC  
key derived from DH

Server’s DH key “proved”  
authentic by cert signature

Certificate “proved” authentic  
by chain to trusted root

---

# It All Depends on the Cert

IF a browser trusts MY  
certificate to be  
Amazon's certificate

- THEN the browser  
will trust my DH  
public key

IF the browser trusts  
my DH public key

- THEN the browser  
will derive the same  
MAC key I do

IF the browser derives  
the same MAC key I  
do

- THEN the browser  
will believe my  
messages are from  
Amazon