so far

building programs — Makefiles for automation, dynamic libraries

hardware support for *processes*

- kernel mode, exceptions, context switches
- virtual memory: let OS choose where program's memory goes

accounts and OS-enforced isolation

networking — layered implementation

- simulating streams of data / routing

secure communication

last time (1)

confidentiality / authenticity

need for secret information

working with shared secrets

- symmetric encryption (confidentiality)
- message authentication code (authenticity, kind-of)

asymmetric schemes

- public/private keypairs
- asymmetric encryption
- digital signatures

last time (2)

replay and machine-in-the-middle attacks

need for secure initial communication

partial workaround 1: public keys (broadcast)

partial workaround 2: certificates (forwarding keys)

anonymous feedback (1)

“The next time you teach this class you should release the working code for each part after it’s due. This assignment is literally just a way to make students who had something come up anytime in the past 3 weeks fail this class. Trying to make a multilevel page table work when my code for LEVELS =1 barely works is so horrible. ”

getting LEVELS = 1 work + README/Makefile/etc. should be enough partial credit that ‘fail this class’ isn’t a likely direct result

(third submission is worth more, but this is mostly deferred grading of stuff that should’ve been done on early submissions)

a lot of the assignment is about organizing your code/etc. — doesn’t work so well when we give code

I’m not sure spending time understanding our LEVELS = 1 solution would’ve saved students much time overall

anonymous feedback (2)

"Hi! I was wondering if it would be possible at the end of the lectures to take like 3-5 minutes to just review everything that you covered in the lecture just because a lot happens in the 75 minute period and sometimes it can be helpful to be like, okay these were the topics that were covered, these are the ones that I understand, these are the ones that are confusing and I need to work on. I know you go over what we learned last class at the beginning of the period but I think it would be much more helpful to have that check in when the information is fresh in our heads."

> not being certain where I'm going to end lecture makes this tricky on a lecture-by-lecture basis
>
> probably better to do topic-by-topic summaries? (which I haven't been...)

anonymous feedback (3)

“I just wanted to say that I’m a huge fan of your use of the anonymous feedback tool, and am extremely appreciative of your willingness to continue improving as a lecturer for the benefit of the students. I will say, though, that at times it might seem like you take the perspective of a single student a bit too seriously, so if possible, it would be nice if there was some sort of anonymous ”upvote/downvote” feature, so that the overall class population could validate specific concerns. I recognize though, that the backend implementation might be difficult, so perhaps it’s just a suggestion for future semesters...”

anonymous feedback (4)

"I think the weekly quizzes are way too confusing. It takes me hours to do one quiz because I have to rewatch the lectures to help me answer the questions. After watching the lectures and reading the readings, I'm still very unsure about what the correct answer is. Is this supposed to be the case? I feel like the readings and the lectures are too vague and general to be helpful in answering some of the quiz questions. Also some of the answer choices are worded so confusingly I spend most of the time trying to understand what it is saying. I wish the quizzes were more straightforward. If the current expectation of the quizzes is to gauge understanding, then I am never understanding what is going on - even though I feel like I understand the content in the lectures."

- I do expect that review of material will be needed
- do hope questions about things actually too vague in lecture, etc.
- question clarity — obviously, I try but sometimes unintended second/third interpretations
  - hope comments field mitigates that somewhat

anonymous feedback (5)

“Bejoy and Andrew were super helpful in OH and did a good job of managing the students and the queue.”

quiz Q1

UDP sockets

write: FIRST; then SECOND; then THIRD

read: ?, ?, ?

UDP sends messages (datagrams), not stream of bytes

will read whole messages only

not parts of messages, multiple messages at once

can read in any order, can lose messages on network (after sent)

quiz Q3

each ISP's DNS server caches IP address, so...

at most 1 query from each ISP's server every 10000 seconds

10000 seconds = 2.8 hours

so 2 queries per server of 5 hours = 20 queries

quiz Q4

A + B commmunicating with each other + others using public key encryption + digital signatures

they need: their own private keys + each other's public keys

should not have each other's private keys — would let A read messages from third-parties for B

don't need other things — not useful if using these public keys

(yes, could use shared secret for symmetric encryption, but that wasn't the plan...)

(yes, could have message signed by B containing B's public key, but not really useful since we need that key to verify the signature anyways)

quiz Q5

$S \to U$: N

$U \to S$: MAC(key, N + password), command

did not require thing passed to MAC contained command

so attacker can manipulate while on network

yes, that would be a good idea, but our specification didn't say to do it

did require that N is one-time

so attacker can't reuse MAC(key, N + password) later

did 'encode' password with MAC(key, N + password), but...

MAC should not reveal information about N + password without key

attackers won't have key

getting public keys?

browser talking to websites
needs public keys of every single website?

not really feasible, but…

certificate idea

let's say A has B's public key already.

if C wants B's public key and knows A's already:

A can send C:

- "B's public key is XXX" AND
- Sign(A's private key, "B's public key is XXX")

if C trusts A, now C has B's public key

- if C does not trust A, well, can't trust this either

certificate authorities

instead, have public keys of trusted *certificate authorities*

only 10s of them, probably

websites go to certificates authorities with their public key

certificate authorities sign messages like:
"The public key for foo.com is XXX."

these signed messages called "certificates"

example web certificate (1)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            81:13:c9:49:90:8c:81:bf:94:35:22:cf:e0:25:20:33
        Signature Algorithm: sha256WithRSAEncryption
        Issuer:
            commonName                = InCommon RSA Server CA
            organizationalUnitName    = InCommon
            organizationName          = Internet2
            localityName              = Ann Arbor
            stateOrProvinceName       = MI
            countryName               = US
        Validity
            Not Before: Feb 28 00:00:00 2022 GMT
            Not After : Feb 28 23:59:59 2023 GMT
        Subject:
            commonName                = collab.its.virginia.edu
            organizationalUnitName    = Information Technology and Communication
            organizationName          = University of Virginia
            stateOrProvinceName       = Virginia
            countryName               = US
.....
```

example web certificate (1)

```
Certificate:
    Data:
....
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:a2:fb:5a:fb:2d:d2:a7:75:7e:eb:f4:e4:d4:6c:
                    94:be:91:a8:6a:21:43:b2:d5:9a:48:b0:64:d9:f7:
                    f1:88:fa:50:cf:d0:f3:3d:8b:cc:95:f6:46:4b:42:
....
        X509v3 extensions:
....
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
....
            X509v3 Subject Alternative Name:
                DNS:collab.its.virginia.edu
                DNS:collab-prod.its.virginia.edu
                DNS:collab.itc.virginia.edu
    Signature Algorithm: sha256WithRSAEncryption
         39:70:70:77:2d:4d:0d:0a:6d:d5:d1:f5:0e:4c:e3:56:4e:31:
....
```

certificate chains

That certificate signed by “InCommon RSA Server CA”

CA = certificate authority

so their public key, comes with my OS/browser?

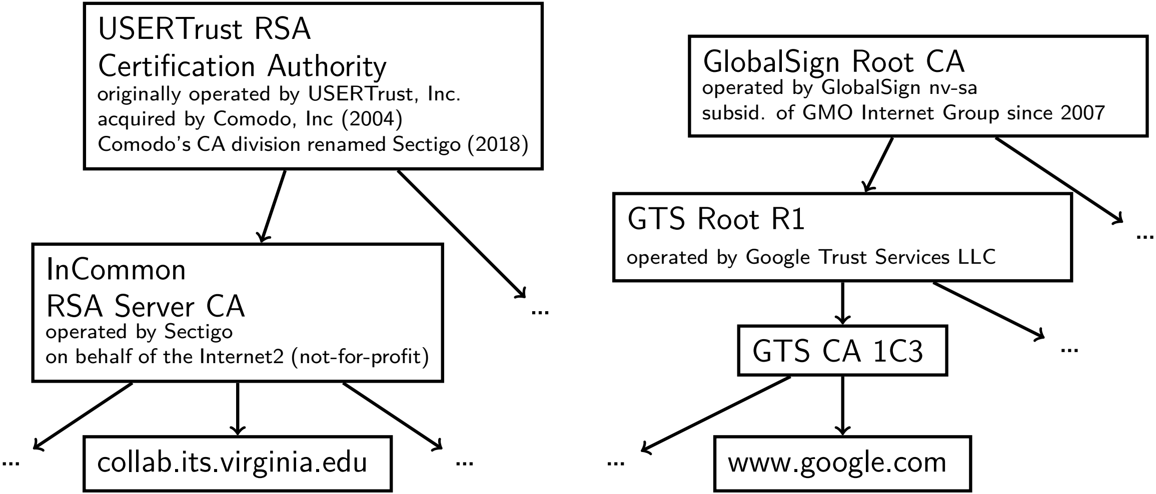
not exactly...

they have their own certificate signed by “USERTrust RSA Certification Authority”

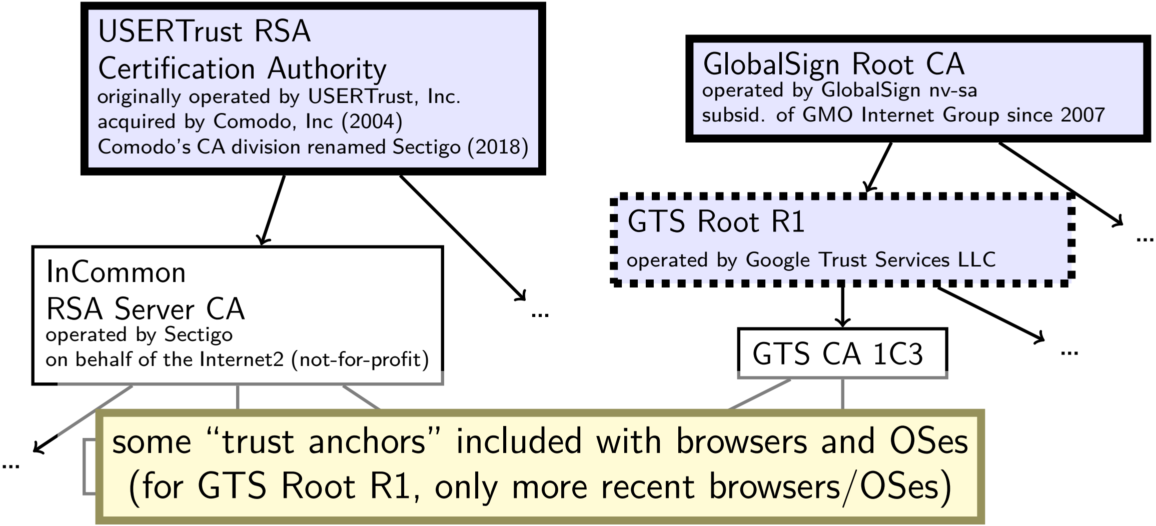
and their public key comes with your OS/browser?

(but both CAs now operated by UK-based Sectigo)

certificate hierarchy



certificate hierarchy



how many trust anchors?

Mozilla Firefox (as of 27 Feb 2023)

155 trust anchors
operated by 55 distinct entities

Microsoft Windows (as of 27 Feb 2023)

237 trust anchors
operated by 86 distinct entities

public-key infrastructure

ecosystem with certificate authorities
and certificates for everyone

called "public-key infrastructure"

several of these:

- for verifying identity of websites
- for verifying origin of domain name records (kind-of)
- for verifying origin of applications in some OSes/app stores/etc.
- for encrypted email in some organizations
- ...

exercise

exercise: how should website certificates verify identity?

how do certificate authorities verify

for web sites, set by CA/Browser Forum

organization of:

- everyone who ships code with list of valid certificate authorities
  - Apple, Google, Microsoft, Mozilla, Opera, Cisco, Qihoo 360, Brave, …
- certificate authorities

decide on rules (“baseline requirements”) for what CAs do

BR domain name identity validation

options involve CA choosing random value and:

sending it to domain contact (with domain registrar) and receive response with it, or

observing it placed in DNS or website or sent from server in other specific way

exercise: problems this doesn't deal with?

some other things public CAs do

keep their private keys in tamper-resistant hardware

maintain publicly-accessible database of *revoked* certificates

- some browsers check these, sometimes

certificate transparency

- public logs of every certificate issued
- some browsers reject non-logged certificates
- so you can tell if bad certificate exists for your website

'CAA' records in the domain name system

- can indicate which CAs are allowed to issue certificates in DNS
- (but CAs apparently not required to use DNSSEC (certificate infrastructure for signing domain name records) when looking this up)

some other things public CAs do

keep their private keys in tamper-resistant hardware

maintain publicly-accessible database of *revoked* certificates

- some browsers check these, sometimes

certificate transparency

- public logs of every certificate issued
- some browsers reject non-logged certificates
- so you can tell if bad certificate exists for your website

'CAA' records in the domain name system

- can indicate which CAs are allowed to issue certificates in DNS
- (but CAs apparently not required to use DNSSEC (certificate infrastructure for signing domain name records) when looking this up)

some other things public CAs do

keep their private keys in tamper-resistant hardware

maintain publicly-accessible database of *revoked* certificates

- some browsers check these, sometimes

certificate transparency

- public logs of every certificate issued
- some browsers reject non-logged certificates
- so you can tell if bad certificate exists for your website

'CAA' records in the domain name system

- can indicate which CAs are allowed to issue certificates in DNS
- (but CAs apparently not required to use DNSSEC (certificate infrastructure for signing domain name records) when looking this up)

some other things public CAs do

keep their private keys in tamper-resistant hardware

maintain publicly-accessible database of *revoked* certificates

- some browsers check these, sometimes

certificate transparency

- public logs of every certificate issued
- some browsers reject non-logged certificates
- so you can tell if bad certificate exists for your website

'CAA' records in the domain name system

- can indicate which CAs are allowed to issue certificates in DNS
- (but CAs apparently not required to use DNSSEC (certificate infrastructure for signing domain name records) when looking this up)

other cryptographic tools

motivation: summary for signature

mentioned that asymmetric encryption has size limit

same problem for digital signatures

solution: sign "summary" of message

how to get summary?

hash function, but...

cryptographic hash

$\text{hash}(M) = X$

given X:

hard to find message other than by guessing

given X, M:

hard to find second message so that $\text{hash}(\text{second message}) = H$

cryptographic hash uses

find shorter 'summary' to substitute for data

- what hashtables use them for, but…
- we care that adversaries can't cause collisions!

cryptographic hash uses

find shorter 'summary' to substitute for data

- what hashtables use them for, but...
- we care that adversaries can't cause collisions!

deal with message limits in signatures/etc.

password hashing — but be careful! [next slide]

constructing message authentication codes

- hash message + secret info (+ some other details)

password hashing

cryptographic hash functions are good at requiring guesses to 'reverse'

problem: guessing passwords is very fast

solution: slow/resource-intensive cryptographic hash functions

- Argon2i
- scrypt
- PBKDF2

just asymmetric?

given public-key encryption + digital signatures...

why bother with the symmetric stuff?

symmetric stuff much faster

symmetric stuff much better at supporting larger messages

key agreement

problem: A has B's public encryption key
wants to choose shared secret

some ideas:

- A chooses a key, sends it encrypted to B
- A sends a public key encrypted B, B chooses a key and sends it back

key agreement

problem: A has B's public encryption key
wants to choose shared secret

some ideas:

- A chooses a key, sends it encrypted to B
- A sends a public key encrypted B, B chooses a key and sends it back

alternate model:

- both sides generate random values
- derive public-key like "key shares" from values
- use math to combine "key shares"
- kinda like A + B both sending each other public encryption keys

Diffie-Hellman key agreement (2)

A and B want to agree on shared secret

A chooses random value Y

A sends public value derived from Y (“key share”)

B chooses random value Z

B sends public value derived from Z (“key share”)

A combines Y with public value from B to get number

B combines Z with public value from A to get number

and b/c of math chosen, both get same number

Diffie-Hellman key agreement (1)

math requirement:

some f, so $f(f(X,Y),Z) = f(f(X,Z),Y)$
(that's hard to invert, etc.)

choose X in advance and:

A randomly chooses Y	B randomly chooses Z
A sends $f(X,Y)$ to B	B sends $f(X,Z)$ to A
A computes $f(f(X,Z),Y)$	B computes $f(f(X,Y),Z)$

key agreement and asym. encryption

can construct public-key encryption from key agreeement

private key: generated random value Y

public key: key share generated from that Y

key agreement and asym. encryption

can construct public-key encryption from key agreeement

private key: generated random value Y

public key: key share generated from that Y

PE(public key, message) =

generate random value Z
combine with public key to get shared secret
use symmetric encryption + MAC using shared secret as keys
output: (key share generated from Z) (sym. encrypted data) (mac tag)

key agreement and asym. encryption

can construct public-key encryption from key agreeement

private key: generated random value Y

public key: key share generated from that Y

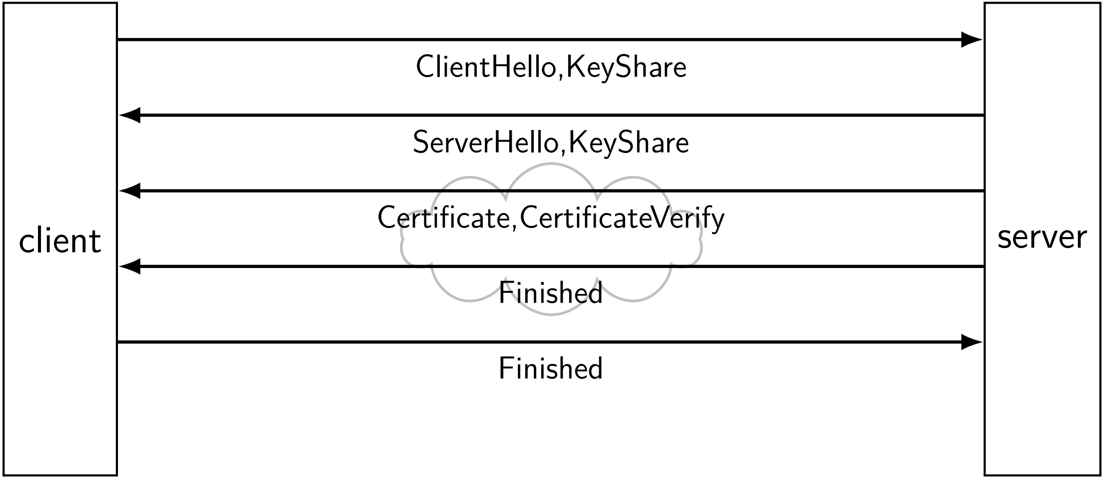
PE(public key, message) =

generate random value Z
combine with public key to get shared secret
use symmetric encryption + MAC using shared secret as keys
output: (key share generated from Z) (sym. encrypted data) (mac tag)

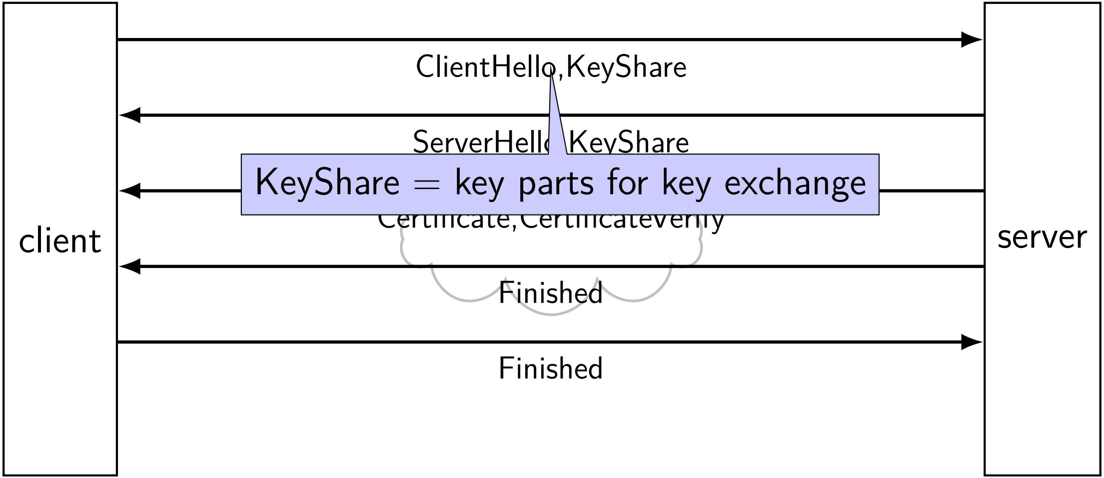
PD(private key, message) =

extract (key share generated from Z)
combine with private key to get shared secret, …

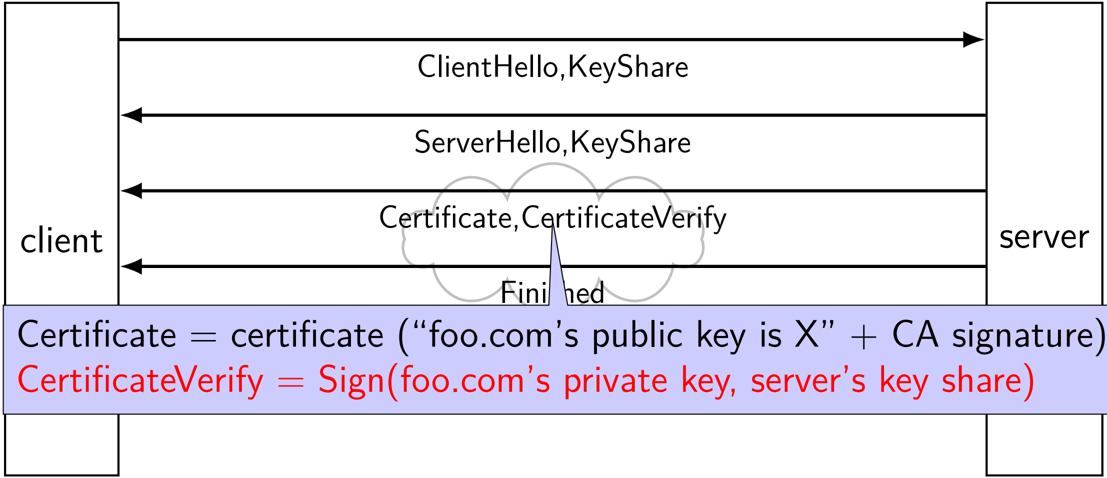
typical TLS handshake



typical TLS handshake



typical TLS handshake

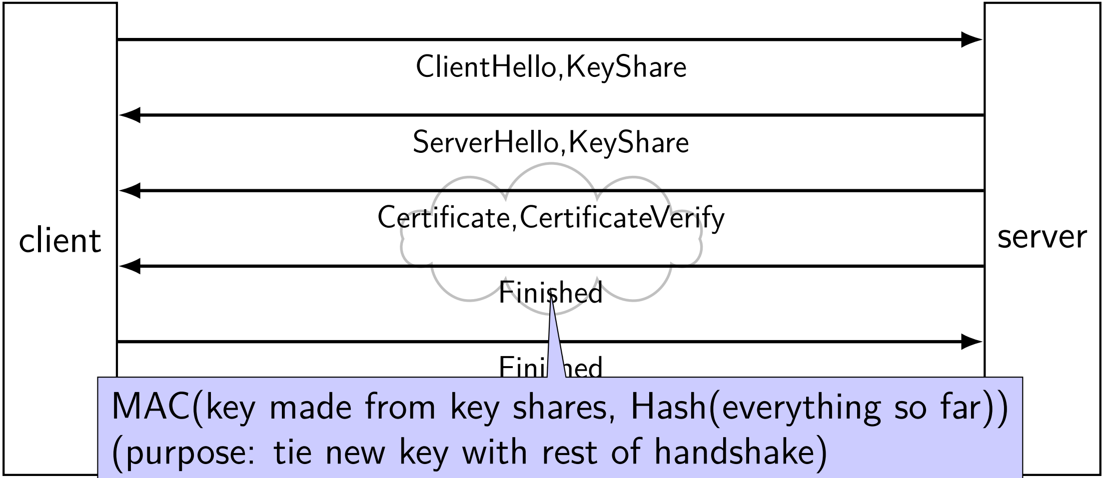


Certificate = certificate (“foo.com’s public key is X” + CA signature)
CertificateVerify = Sign(foo.com’s private key, server’s key share)

typical TLS handshake

MAC(key made from key shares, Hash(everything so far))
(purpose: tie new key with rest of handshake)

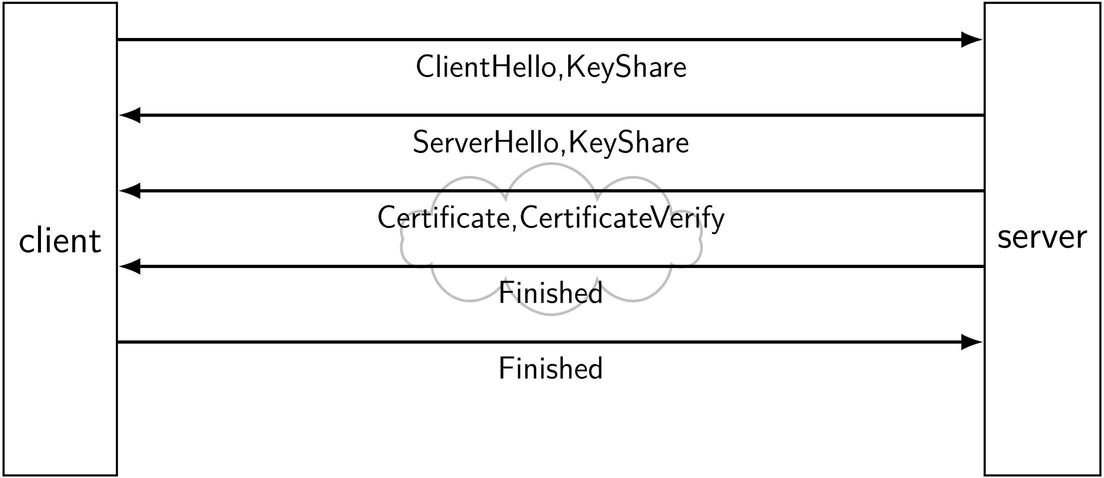
typical TLS handshake



MAC(key made from key shares, Hash(everything so far))
(purpose: tie new key with rest of handshake)

typical TLS handshake

typical TLS handshake



TLS: after handshake

use key shares results to get **several** keys

  take hash(something + shared secret) to derive each key

separate keys for each direction (server → client and vice-versa)

often separate keys for encryption and MAC

later messages use encryption + MAC + nonces

things modern TLS usually does

(not all these properties provided by all TLS versions and modes)

confidentiality/authenticity

- server = one ID'd by certificate
- client = same throughout whole connection

forward secrecy

- can't decrypt old conversations (data for KeyShares is temporary)

fast

- most communication done with more efficient symmetric ciphers
- 1 set of messages back and forth to setup connection

denial of service (1)

so far: worried about network attacker disrupting confidentiality/authenticity

what if we're just worried about just breaking things

well, if they control network, nothing we can do...

but often worried about less

denial of service (2)

if you just want to inconvenience...

attacker just sends lots of stuff to my server

my server becomes overloaded?

my network becomes overloaded?

but: doesn't this require a lot of work for attacker?

exercise: why is this often not a big obstacle

denial of service: asymmetry

work for attacker > work for defender

how much computation per message?

- complex search query?
- something that needs tons of memory?
- something that needs to read tons from disk?

how much sent back per message?

resources for attacker > resources of defender

how many machines can attacker use?

denial of service: reflection/amplification

instead of sending messages directly…attacker can send messages "from" you to third-party

third-party sends back replies that overwhelm network

example: short DNS query with lots of things in response

"amplification" =

third-party inadvertantly turns small attack into big one

firewalls

don't want to expose network service to everyone?

solutions:

- service picky about who it accepts connections from
- filters in OS on machine with services
- filters on router

later two called "firewalls"

firewall rules examples?

ALLOW tcp port 443 (https) FROM everyone

ALLOW tcp port 22 (ssh) FROM my desktop's IP address

BLOCK tcp port 22 (ssh) FROM everyone else

ALLOW from address X to address Y

...

network security summary (1)

communicating securely with math

- secret value (shared key, public key) that attacker can't have
- symmetric: shared keys used for ed/encryption + auth/verify; fast
- asymmetric: public key used by any for encrypt + verify; slower
- asymmetric: private key used by holder for decrypt + sign; slower

protocol attacks — repurposing encrypt/signed/etc. messages

certificates — verifiable forwarded public keys

key agreement — for generated shared-secret "in public"

- publish key shares from private data
- combine private data with key share for shared secret

network security summary (2)

TLS: combine all cryptography stuff to make “secure channel”

denial-of-service — attacker just disrupts/overloads (not subtle)

firewalls

backup slides