Lecture 17: TLS

MIT - 6.893

Fall 2020

Henry Corrigan-Gibbs

Plan

- Recap: Certs & CT
- Privacy issues and TLS
- TLS 1.3

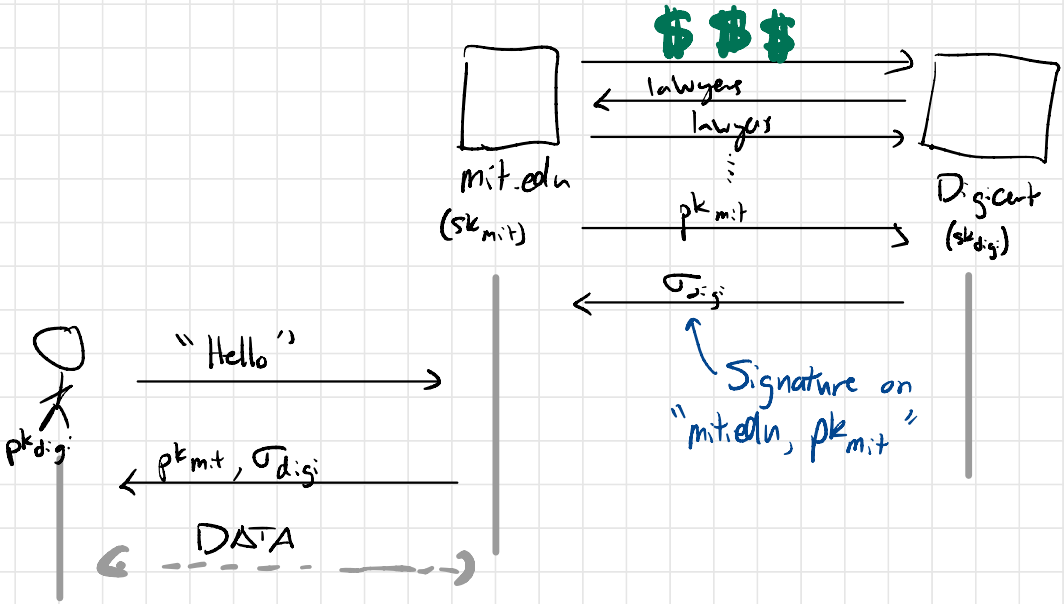
Logistics

- No class next week! Have a great holiday!
- HW6 due on 12/4 at 5pm via Gradescope. NO LATE DAYS...
- Feel free to come to OH or email me to ask about opps for research or other post-class OS.

Recap: Public-Key Infrastructure & CT

To encrypt messages to mit.edu's server, I need pk_{MIT}, but where do I get this?

A: Public-key infrastructure



Client runs - $Verify(pk_{dig}, "mit.edu, pk_{mit}", \sigma_{dig})$

- Runs other checks
- Accepts if all pass.

(This is simplified! Many layers of CAs)

Problem: Many CAs, Many Single Points of Failure

pk_{digi}, pk_{Turkey}, $pk_{Hongkong}$, $pk_{Germany}$, ...

↖ If you look at the list of CAs that your browser trusts, you'll be surprised.

In the old days, compromising any one of those CAs would have given you ability to mint certs for any domain.

↳ Middlebox vendors abused this

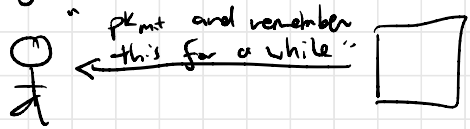
Recap: Two Ideas

1. Public-key Pinning

Static

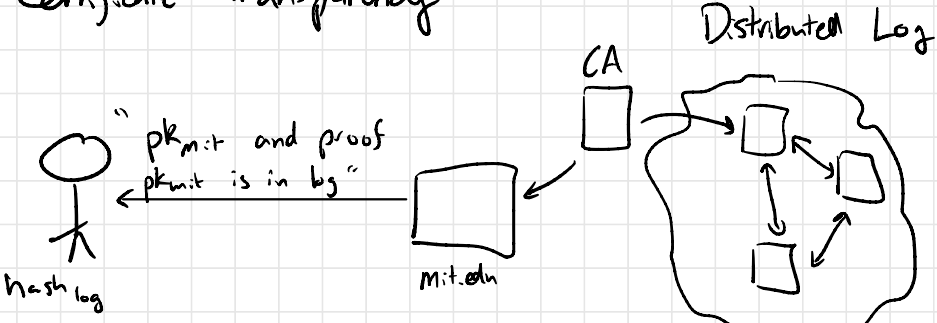
$pk_{Twitter}$
pk_{Tor}
pk_{Google}
pk_{FB}

Dynamic



→ Fallen out of favor. Brittle, hard to use.

2. Certificate Transparency



Goal: If CA "goes rogue", eventually someone should notice.

Challenges: Who runs the logs? Do you "fail open"?

Privacy and TLS

* PKI is all about getting the public key for your counterparty... then what?

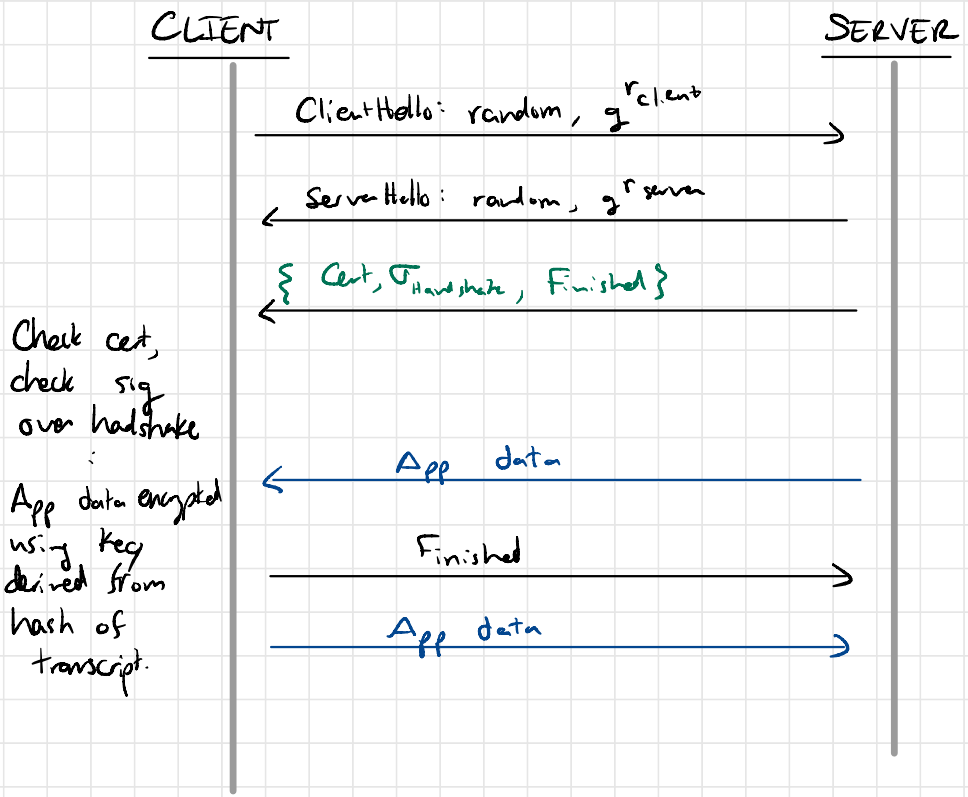
* TLS... the most common encryption layer (HTTPS, SMTPS, ...)
  ...others: IPSec, SSH, ...

* Also, one of the most powerful tools for protecting against network surveillance.
  ↳ Makes it difficult* to read data on wire.

* TLS 1.3 addresses large classes of prior vulnerabilities
  ↳ We hope that there will be fewer protocol bugs this time around.

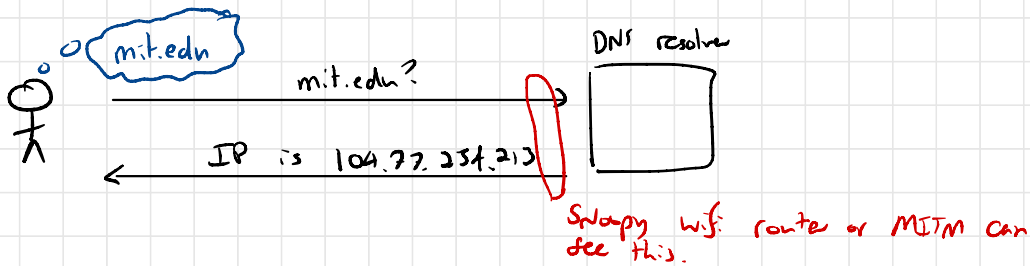
TLS 1.3 [Based on Eric Rescorla's notes]

SIMPLIFIED?

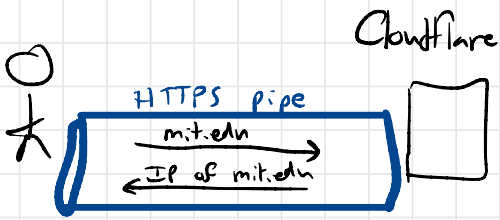


Caveats to TLS Privacy

- DNS leaks which sites you're visiting to various DNS servers ... all unencrypted



Recent innovation: DNS over HTTPS ("DoH") (enabled by default in FF and Chrome)



Cloudflare still sees your DNS queries, but no one else does.

→ Lots of controversy! DNS was a convenient place for govts and companies to filter/monitor web browsing.

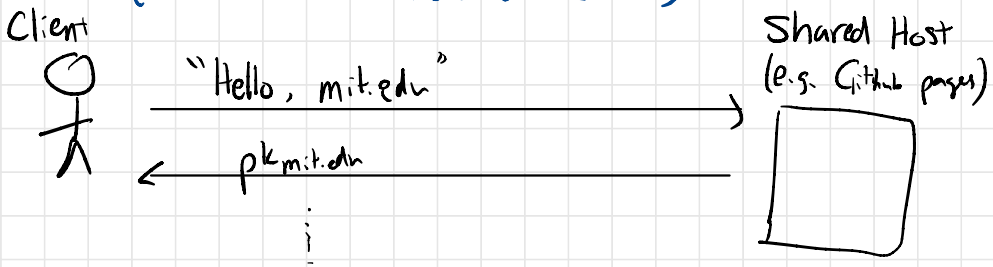
↳ see: UK

STILL: Worries about leakage to DoH resolver.

Suppose we fix the DNS problem...

- TLS still does not hide which site you're connecting to... even if many sites behind same IP

Why? If a single machine/IP host, 100 websites the server needs to know which certificate to send client
("Server name indication" ... SNI)



⇒ If you're in an internet cafe, the owner of the wifi router can see hostnames — not only IP addresses.
(Also, pk_{mit} sent unencrypted pre-1.3)

TLS 1.3: - Server cert is sent encrypted (protects against eavesdropper)
- Option to encrypt SNI ... more complicated

Problems with prior versions of TLS (pre 1.3)

+ SSL/TLS enabled online commerce.
− Lots of security issues
  ↳ One meta-problem: Protocol designed first and analyzed second... fixed w/ TLS 1.3

Examples of security problems...

Support for old ciphers

Modern cipher modes of operation provide authenticity & confidentiality
↳ See Boneh-Shoup book "AEAD modes"

Old versions of TLS supported non-AEAD modes.
Also old versions of RSA signatures
Also ciphers w/ 64-bit blocks

Brittle!
"Padding-oracle attacks"

TLS 1.3: Only AEAD, many fewer ciphersuites, eliminates many esoteric options (e.g. custom DH groups)

(Bleichenbacher, ...)

Digression: Example Padding-Oracle Attacks

RSA encryption w/ PKCS#1 v1.5 padding

public key $N = p \cdot q$, exponent $e = 3$

$Enc(N, m \in \{0,1\}^{384}) \{$

$\mu \leftarrow$ | 00 | 02 | random non-zero stuff | 00 | m | $\in \mathbb{Z}_N$

output $\mu^3 \in \mathbb{Z}_N$

$\}$

$Dec(p, q, ct \in \mathbb{Z}_N) \{$

$\mu' \leftarrow ct^{1/3} \in \mathbb{Z}_N$

if $\mu' \neq \{0x00, 0x02, \ldots\}$: FAIL

else output low-order bytes

$\}$

$\Rightarrow$ By choosing random $r \xleftarrow{R} \mathbb{Z}_N$ and asking for decryptions of $ct' = r^3 ct \in \mathbb{Z}_N$, the attacker can recover all of m.

$\rightarrow$ Attack often works even if server tries to hide whether failure occurred (timing side-channels)

Compress then encrypt ("CRIME")

TLS (like most enc schemes) makes no attempt to hide message lengths. What could go wrong?

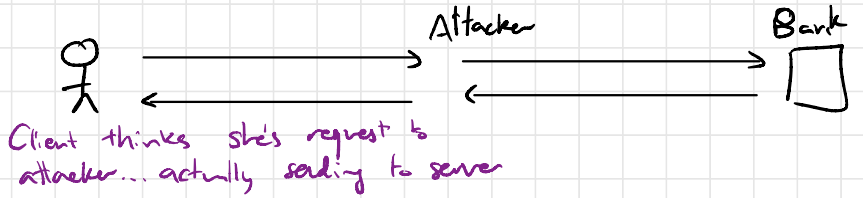
GZIP("secret || 00")
GZIP("secret || 11")
⋮

↳ Can recover sensitive authentication cookies, etc.

TLS 1.3: No more compression, except for certain headers
↳ still possible to get wrong

Renegotiation Attack (Ray & Dispensa)

Allows a MITM attacker to append traffic on encrypted session b/w client & server.



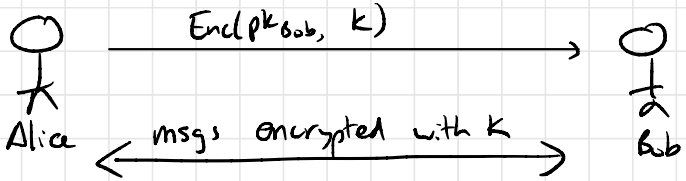
↳ Super-subtle & clever. Many variants

→ Very bad... "Send $ to attacker's userid"

TLS 1.3: Protocol tweaks to try to eliminate this sort of confusion.

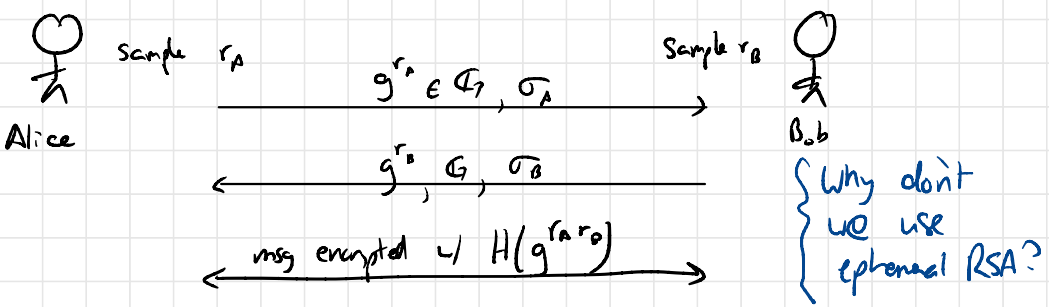
No "forward secrecy"

Old school key exchange



Problem: Attacker who can record all traffic + steal sk_{Bob} can decrypt all past traffic.

Example EDH key exchange (simplified!) ... DDH Group $\mathbb{G}$



Alice & Bob can delete secrets r_A, r_B as soon as key exchange is done. Can rotate keys on every connection.

↳ Breach doesn't affect past connections
↳ Still vulnerable to total break of cryptosystem (ECDlog, etc.)

TLS 1.3: No RSA key exchange.
Only ephemeral DH.
↑ Also controversial!

* TLS is one of the most important privacy-preserving tools we have.

* TLS 1.3 removes many of the problematic features of earlier versions of SSL/TLS

* Still, metadata leakage is a serious privacy threat...

... NEXT TIME!