STACIE FARMER @ OPENWEST 2019

HTTPS ALL THE THINGS!

WHO AM I?

- Stacie Farmer
- Owner of Mayet Security
  - Helping everyday people learn cybersecurity fundamentals
- Programmer, former community leader, endless learner
- NOT a cryptography expert

WHAT IS HTTP?

- HTTP - default way websites are served
- ANYTHING can be viewed and/or modified

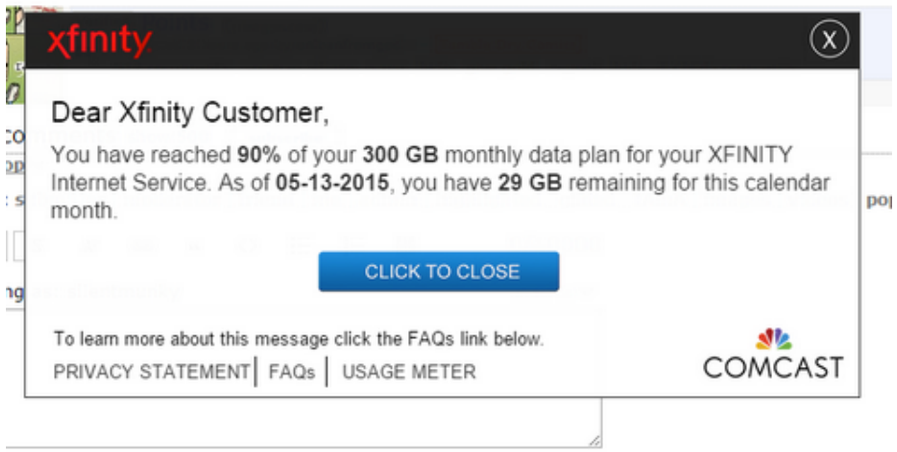
```
GET /load.php?cb=7840027840012&debug=false&lang=en&modules=site&only=styles&skin=oasis&* HTTP/1.1
Host: toontown.wikia.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/css,*/*;q=0.1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://toontown.wikia.com/wiki/Gardening
Cookie: wikia_beacon_id=KsKCk2bcG5; wikia_session_id=9pi1KyiyZv;
Geo={%22region%22:%22WA%22%2C%22country%22:%22US%22%2C%22continent%22:%22NA%22}
Connection: close
```

What is HTTP?

- How we deliver websites to users
- Clear, plaintext protocol
- Can be viewed and modified by MITM

HTTP (UNENCRYPTED) TRAFFIC

- ANYTHING in the traffic can be viewed/modified



An example of MITM modifying web traffic

- ISP is constant MITM
- Only modifies HTTP traffic

HTTP POST REQUEST

```
POST /ut/v3/prebid HTTP/1.1
Host: ib.adnxs.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://toontown.wikia.com/wiki/Gardening
Content-Type: text/plain
Content-Length: 1316
Origin: http://toontown.wikia.com
Cookie: uuid2=4437431820410632482;
anj=dTM7k!M40meTF>+ghqdmU(7TPXfGZ.#tB.2l[vNEDWC1oHG`q4?nJZGBllt]^21=Hd)U`Hv80Zkp*x9JDO(Unv%$K<R[W/7CKJk=_4CZ?vqIla-!3#]!]
x'B%84`XDlWtu(]WE]V^hr$Z5=h.VD2Zl^#AiNT04[f[n@lLB'4FzZ4H*<E]bM5l_gJI:YH-P20u@jAC'0$*bpRz*bd$'6F1!A;
icu=ChgIxIA9EAoYASABKAEwtNqG4AU4AUABSAEQtNqG4AUYAA..
Connection: close

{"tags":[{"sizes":[{"width":728,"height":90},{"width":970,"height":250}],"primary_size":{"width":728,"height":90},"ad_typ
es":["banner"],"uuid":"25c8a0bc24591bf","id":11977073,"allow_smaller_sizes":false,"use_pmt_rule":false,"prebid":true,"dis
able_psa":true},{"sizes":[{"width":300,"height":250},{"width":300,"height":600}],"primary_size":{"width":300,"height":250
},"ad_types":["banner"],"uuid":"266a88b294f843a","id":11977073,"allow_smaller_sizes":false,"use_pmt_rule":false,"prebid":
true,"disable_psa":true},{"sizes":[{"width":728,"height":90},{"width":970,"height":250}],"primary_size":{"width":728,"hei
ght":90},"ad_types":["banner"],"uuid":"27bd69bd58dddcd","id":11977096,"allow_smaller_sizes":false,"use_pmt_rule":false,"p
rebid":true,"disable_psa":true},{"sizes":[{"width":160,"height":600},{"width":300,"height":600},{"width":300,"height":250
}],"primary_size":{"width":160,"height":600},"ad_types":["banner"],"uuid":"287f02e10323189","id":11977016,"allow_smaller_
sizes":false,"use_pmt_rule":false,"prebid":true,"disable_psa":true}],"sdk":{"source":"pbjs","version":"1.31.0"},"gdpr_con
sent":{"consent_string":null,"consent_required":false},"referrer_detection":{"rd_ref":"http%3A%2F%2Ftoontown.wikia.com%2F
wiki%2FGardening","rd_top":true,"rd_ifs":0,"rd_stk":"http%3A%2F%2Ftoontown.wikia.com%2Fwiki%2FGardening"}}
```

Nothing sent over HTTP is private

- POST data can still be viewed
- Anything sent over HTTP is sent in plaintext

WHY DOES IT MATTER?

- Susceptible to MITM attacks:
  - Injecting malicious scripts, links, files, etc
  - Modifying or sniffing data
  - All around general spying

MITM Attacks

- Anyone sitting between user and server can read, modify, and inject into the HTTP requests
- Malware can be injected
- Sensitive data can be stolen
- All kinds of fun stuff

WHAT IS HTTPS?

- HTTPS = HTTP over SSL/TLS
- Not a part of HTTP, but can be ADDED to it
- Changing something you can see to something you can't

```
POST /GTSGIAG3 HTTP/1.1
Host: ocsp.pki.goog
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Length: 75
Content-Type: application/ocsp-request
Connection: close

0I0G0E0C0A0        ▒▒▒▒+▒▒▒▒
```

NOT EVERYTHING IS ENCRYPTED

- Changing something you can see to something you can't
- Can still see some things

```
POST /GTSGIAG3 HTTP/1.1
Host: ocsp.pki.goog
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Length: 75
Content-Type: application/ocsp-request
Connection: close
```

```
0I0G0E0C0A0        ...+...k...k...  ...  ..cb2..5..u..  ...l.  ...  ..w.P.gvv.-.....K  ...  ..m..s2.
```

Some metadata will still be visible

- Everything in red box
- Source & destination IPs; data packets need to know where they're going and where they came from.
- Some info can still be gleaned, just not nearly as much

HTTPS

- Secure COMMUNICATION
- NOT safe traffic

Confusion about what HTTPS really does

- Green lock & assume the website they’re visiting is “safe”
- Only enables secure COMMUNICATION; doesn’t guarantee data being sent is “clean” or “safe”
- Malicious websites use HTTPS to fool and prevent malware being detected
- No one can watch you download malware
- Safe example - don’t know what’s inside; money or a deadly virus - protected from being viewed or stolen while in transit. Once opened, all bets are off.

SYMMETRIC KEY ENCRYPTION

- We both have the same key
- We can both encrypt & decrypt with our key
- Keys must be kept private
- Ultimate goal of TLS Handshake

2 Types of Key Encryption - Symmetric & Asymmetric

- We'll be focusing on symmetric key encryption in this talk, but will briefly discuss asymmetric as well
- Symmetric is best & hardest to do. Goal of TLS handshake
- Keys must be kept private - never passed across the wire; Depending on algorithm, they are computed independently by the browser & server but include values passed across the wire.
- It works because of math. We want our keys to be mathematically impossible to guess by anyone observing these values being passed.
- Can read more about Diffie-Hellman if you want
- Once symmetric keys are created by both parties, secure communication tunnel exists

ASYMMETRIC KEY ENCRYPTION

- I share *public* key to **encrypt**
- I have *private* key to **decrypt**
- Used during TLS Handshake

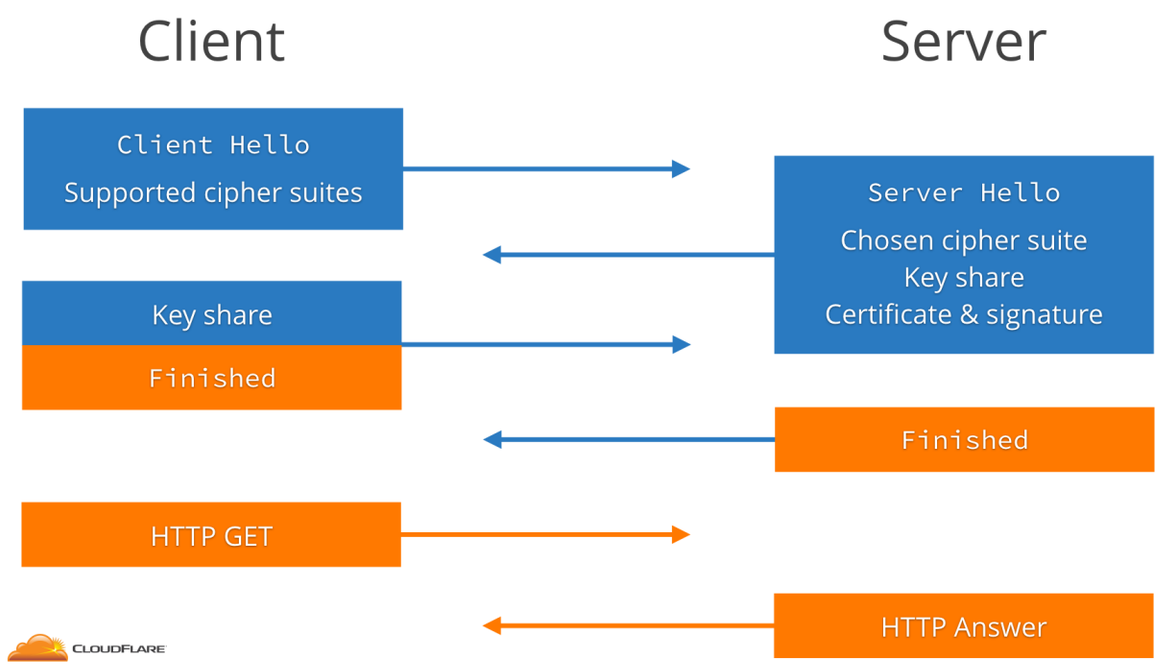
My Public Key - shared with anyone

My private key - kept super secret & safe

Briefly discuss asymmetric since it used by RSA for TLS handshake

- Allows the browser to send the server a secret value over the wire after its verified the server's digital certificate to prove its identity
- Server shares it public key (can share with anyone)
- Browser encrypts secret value with public key & sends it
- Can only be decrypted by the server with its private key
- Private keys must be kept private

TLS 1.2

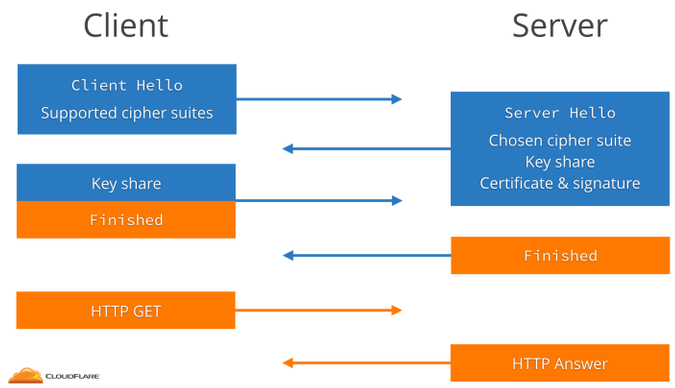


Discuss TLS 1.2 first.

- Most widely used
- Talk about TLS 1.3 after
- Go through steps briefly
- Blue steps sent unencrypted
- Orange steps encrypted using symmetric keys

TLS 1.2: CLIENT HELLO

- Random number
- Supported cipher suites
- Supported TLS versions



Client sends a hello message to the server

TLS 1.2: CLIENT HELLO

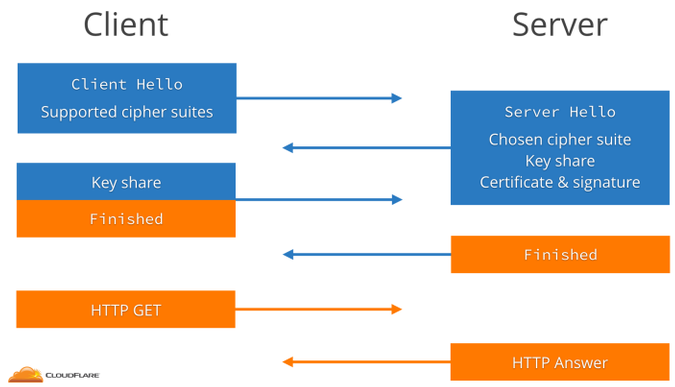
TLSv1.2 Record Layer: Handshake Protocol: Client Hello
- Content Type: Handshake (22)
- Version: TLS 1.0 (0x0301)
- Length: 512
- Handshake Protocol: Client Hello
  - Handshake Type: Client Hello (1)
  - Length: 508
  - Version: TLS 1.2 (0x0303)
  - **Random: d97ede776765111ee28e6625e302e20d073884b9d786015f...**
  - Session ID Length: 32
  - Session ID: 33b18f066dd2e301e371d7fcdfe1c3d0d70f934c23a9aa09...
  - Cipher Suites Length: 28
  - **Cipher Suites (14 suites)**
    - **Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)**
    - **Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)**
    - **Cipher Suite: TLS_AES_256_GCM_SHA384 (0X1302)**
    - **Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)**
    - **Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)**
    - **Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)**

TLS 1.2: CLIENT HELLO (CONTINUED . . .)

Extension: key_share (len_107)
Type: key_share (51)
Length: 107
Key Share extension
Client Key Share Length: 105
Key Share Entry: Group: secp256r1, Key Exchange length: 65
Group: secp256r1 (23)
Key Exchange Length: 65
Key Exchange: 0470ffcbf99bd4ca1df1c12701cb3435413c2e3e7ea2e6cb...
Extension: supported_versions (len=9)
Type: supported_versions (43)
Length: 9
Supported Versions length: 8
Supported Version: TLS 1.3 (0x0304)
Supported Version: TLS 1.2 (0x0303)
Supported Version: TLS 1.1 (0x0302)
Supported Version: TLS 1.0 (0x0301)
Extension: signature_algorithms (len=24)
Extension: psk_key_exchange_modes (len=2)

TLS 1.2: SERVER HELLO

- Chosen TLS version
- Random number
- Chosen cipher suite

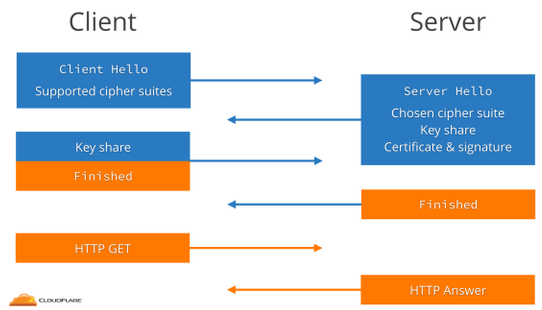


TLS 1.2: SERVER HELLO

TLSv1.2 Record Layer: Handshake Protocol: Server Hello
- Content Type: Handshake (22)
- Version: TLS 1.2 (0x0303)
- Length: 89
- Handshake Protocol: Server Hello
  - Handshake Type: Server Hello (2)
  - Length: 85
  - **Version: TLS 1.2 (0x0303)**
  - **Random: 0b78dd81c14e82e7a0fc09015ab95103c3c3a6f08f6665ac...**
  - Session ID Length: 32
  - Session ID: 30c588aafa634dfa601bcd8f7647fddae62a8ae5bd20f6c7...
  - Cipher Suites Length: 28
  - **Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)**
  - Compression Method: null (0)
  - Extensions Length: 13
  - Extension: renegotiation_info (len=1)
  - Extension: ec_point_formats (len=4)

TLS 1.2: SERVER CERTIFICATE

- Domain name of the server (*common name*)
- Public key
- Owner of the certificate (*subject*)
- Issuer of the certificate (*certificate authority*)
- Expiration date
- Serial number
- Signature



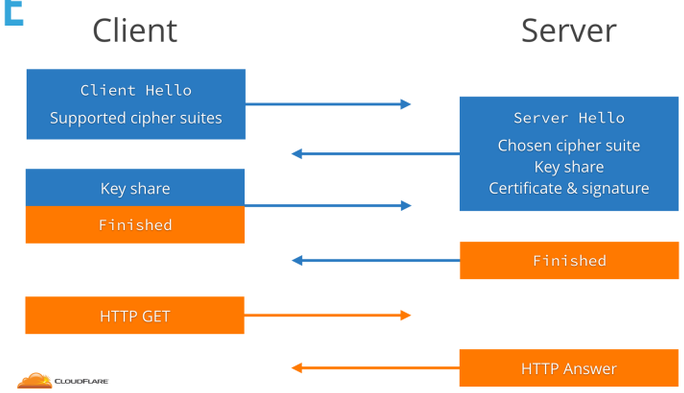
TLS 1.2: SERVER CERTIFICATE

TLSv1.2 Record Layer: Handshake Protocol: Certificate
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 2568
Handshake Protocol: Certificate
Handshake Type: Certificate (11)
Certificate: 3082055f30820447a0038201020210017e45a31aa50bc350…
signedCertificate
serialNumber: 0x017e45a31aa50bc35053bc50f9b69bad
Signature (sha256WithRSAEncryption)
Algorithm ID: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
issuer:
id-at-commonName=DigiCert SHA2 Secure Server CA
validity
notBefore: utcTime: 17-10-03 00:00:00 (UTC)
notAfter: utcTime: 20-01-08 12:00:00 (UTC)
subject:
id-at-organizationName=Mozilla Corporation
id-at-commonName=*.services.mozilla.com
subjectPublicKey: 3082010a0282010100c1282a88266c85d9a4ee1e2ded6e42…

TLS 1.2: SERVER KEY EXCHANGE (ECDHE)

- Public Key
- Signature

TLS 1.2: SERVER HELLO DONE



TLS 1.2: SERVER KEY EXCHANGE (ECDHE)

TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
- Content Type: Handshake (22)
- Version: TLS 1.2 (0x0303)
- Length: 333
- Handshake Protocol: Server Key Exchange
  - Handshake Type: Server Key Exchange (12)
  - Length: 329
  - EC Diffie-Hellman Server Params
    - Curve Type: named_curve (0x03)
    - Named Curve: secp256r1 (0x0017)
    - Pubkey Length: 65
    - **Pubkey: 04c8c3b639e8f4cc992a9ceea395f5b21b39c1b6cbe979df...**
    - Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
    - Signature Length: 256
    - **Signature: 67b69a99fac93f45e0a013f5fa806805aa05667596890fad...**

TLS 1.2: SERVER HELLO DONE

TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 4
    Handshake Protocol: Server Hello Done
        Handshake Type: Server Hello Done (14)
        Length: 0

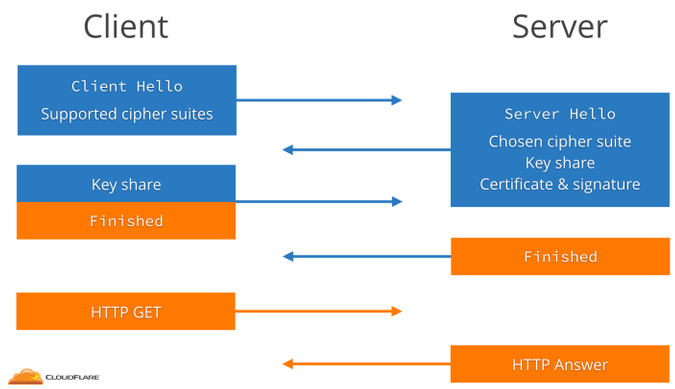
TLS 1.2: CLIENT KEY EXCHANGE (ECDHE)

- Public Key

TLS 1.2: CLIENT CHANGE CIPHER SPEC

- To use symmetric keys

TLS 1.2: ENCRYPTED – CLIENT FINISHED



TLS 1.2: CLIENT KEY EXCHANGE (ECDHE)

TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
- Content Type: Handshake (22)
- Version: TLS 1.2 (0x0303)
- Length: 70
- Handshake Protocol: Client Key Exchange
  - Handshake Type: Client Key Exchange (16)
  - Length: 66
  - EC Diffie-Hellman Server Params
    - Pubkey Length: 65
    - **Pubkey: 0492f91b91d25332b3d0c57f5b85187cdda60cbb06cb6fb3...**

TLS 1.2: CLIENT CHANGE CIPHER SPEC

TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
Content Type: Change Cipher Spec (20)
Version: TLS 1.2 (0x0303)
Length: 1
Change Cipher Spec Message

TLS 1.2: ENCRYPTED – CLIENT FINISH

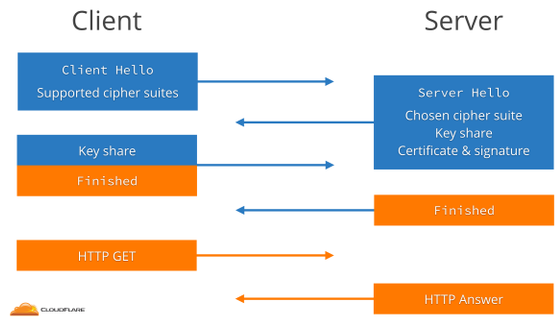
TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 40
Handshake Protocol: Encrypted Handshake Message

TLS 1.2: SERVER CHANGE CIPHER SPEC

- To use symmetric keys

TLS 1.2: ENCRYPTED – SERVER FINISHED

- Application data can now begin flowing between client and server - encrypted



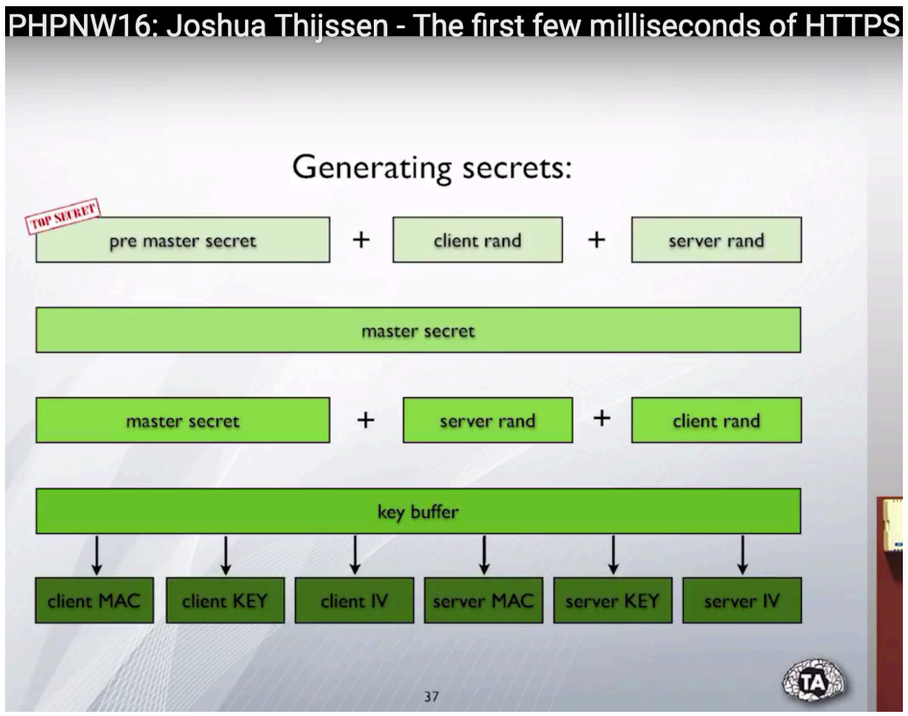
TLS 1.2: SERVER CHANGE CIPHER SPEC

TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
Content Type: Change Cipher Spec (20)
Version: TLS 1.2 (0x0303)
Length: 1
Change Cipher Spec Message

TLS 1.2: ENCRYPTED – SERVER FINISHED

TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 40
Handshake Protocol: Encrypted Handshake Message

TLS HOW SYMMETRIC KEYS ARE DERIVED (ROUGHLY)...

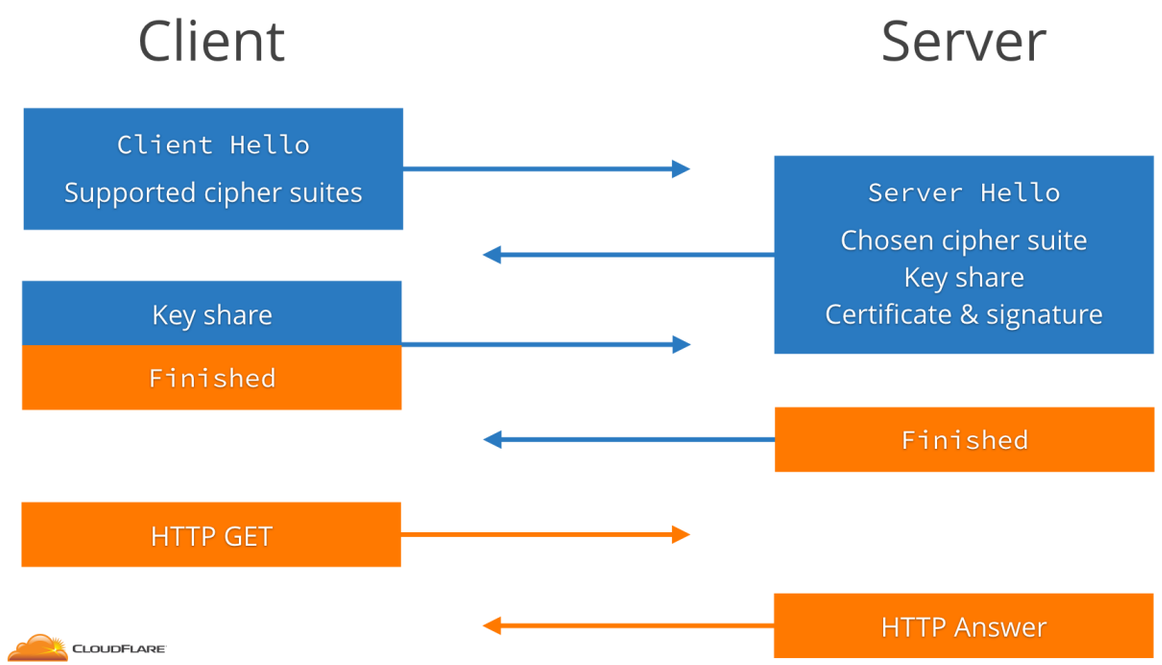


https://www.youtube.com/watch?v=4Jg2ALtioMA

To derive our symmetric keys, we have many steps.

- First create a pre-master secret; derived from server key share & the client key share. {jump to inner slide}
- Combine pre-master secret with client & server random values
- Creates master secret
- Combine that with client & server random values again
- Creates key buffer where symmetric keys come from
- Technically 2 symmetric keys are created. One for data flowing from browser to server & one for data flowing from server to browser.
- Great video at that link if you want to look into this TLS handshake a little more

TLS 1.2



Key share happens in 2nd & 3rd steps of the TLS 1.2 handshake

- in RSA, Client's key share is encrypted with server's public key (using asymmetric key encryption) so it isn't sent plaintext on the wire
- In ECDHE (Elliptical-Curve Diffie-Hellman Ephemeral), which we saw in our example, neither key share needs to be encrypted so everything is sent in plaintext over the wire
- Ultimately, the client's key share is combined with the server's key share
- both used to calculate the pre-master secret. {back up one slide}

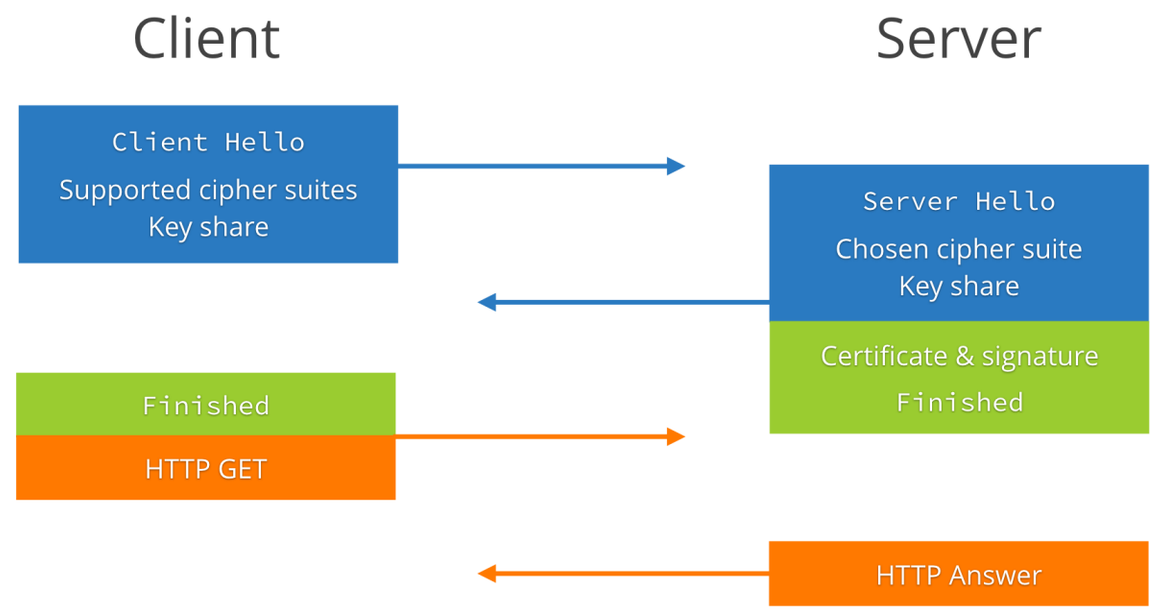
TLS 1.2 VULNERABILITIES

- Protocol downgrade attacks by MITM
  - Only vulnerable if lower version of TLS/SSL and/or weak cipher suites are allowed
  - Once downgraded, other vulnerabilities are exploited to decrypt communication like lack of forward secrecy
- Just slower than 1.3

A few vulnerabilities with TLS 1.2

- MITM can do a protocol downgrade attack if lower versions of TLS/SSL are allowed and/or weak cipher suites can be used. This is usually due to a configuration on the server side.
- Protocol downgrade attack can allow attacker to capture data and later decrypt it
- Cipher suites with forward secrecy make this harder; New symmetric keys are calculated for each packet transmission; MITM has to decrypt one packet transmission at a time, instead of doing one & getting them all
- TLS 1.3 is much faster; less roundtrips and quicker to reestablish a session
- Let’s see how that works

FASTER AND MORE SECURE – TLS 1.3

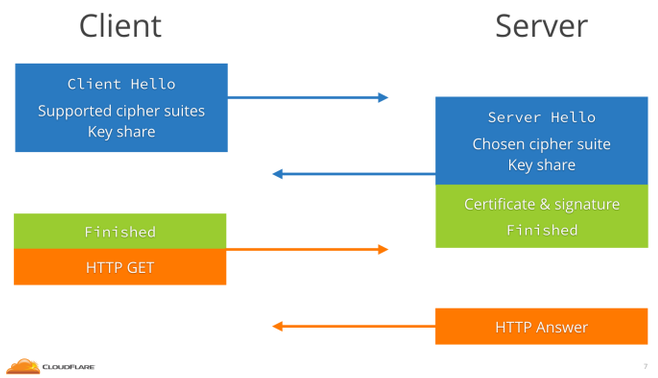


Client guesses 2 or more cipher suites the server will pick

- calculates a key share for each of those cipher suites and sends them over
- Server picks one of those cipher suites & now has everything to create symmetric keys
- Sends blue data unencrypted
- Sends green data encrypted with symmetric keys
- Client calculates symmetric keys
- Sends finished message and can immediately send HTTP requests

TLS 1.3: CLIENT HELLO

- Random number
- Supported cipher suites
- Calculated key shares for cipher suites likely to be chosen
- Supported TLS versions



TLS 1.3: CLIENT HELLO

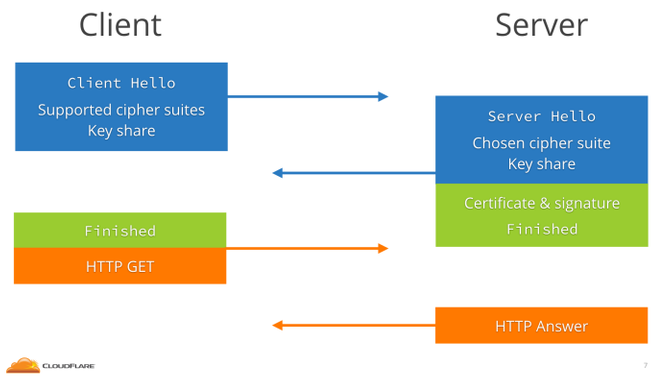
TLSv1.3 Record Layer: Handshake Protocol: Client Hello
- Content Type: Handshake (22)
- Version: TLS 1.0 (0x0301)
- Length: 512
- Handshake Protocol: Client Hello
  - Handshake Type: Client Hello (1)
  - Length: 508
  - Version: TLS 1.2 (0x0303)
  - **Random: eff9f856dafa6c01909e0d00565aab421d5204494233093b...**
  - Session ID Length: 32
  - Session ID: 49423a906a64d0d161ff0c0c62bc67a9c6181c65a7dae31e...
  - Cipher Suites Length: 28
  - **Cipher Suites (14 suites)**
    - **Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)**
    - **Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)**
    - **Cipher Suite: TLS_AES_256_GCM_SHA384 (0X1302)**
    - **Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)**
    - **Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)**
    - **Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)**

TLS 1.3: CLIENT HELLO (CONTINUED . . .)

Extension: key_share (len_107)
Key Share extension
Client Key Share Length: 105
Key Share Entry: Group: x25519, Key Exchange length: 32
Group: x25519 (29)
Key Exchange Length: 32
Key Exchange: 252ca5a47155e89ebc138f5e01e155f534dadf242c296a35...
Key Share Entry: Group: secp256r1, Key Exchange length: 65
Group: secp256r1 (23)
Key Exchange Length: 65
Key Exchange: 04c3d526016edc735a861fbb3a64d65a87084798f0d7af1a...
Extension: supported_versions (len=9)
Type: supported_versions (43)
Length: 9
Supported Versions length: 8
Supported Version: TLS 1.3 (0x0304)
Supported Version: TLS 1.2 (0x0303)
Supported Version: TLS 1.1 (0x0302)
Supported Version: TLS 1.0 (0x0301)

TLS 1.3: SERVER HELLO

- Chosen TLS version
- Random number
- Chosen cipher suite
- Calculated key share



TLS 1.3: SERVER HELLO

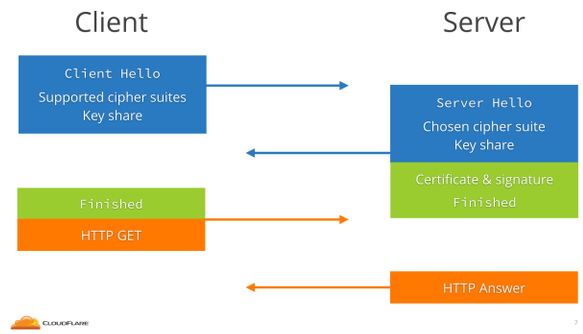
TLSv1.3 Record Layer: Handshake Protocol: Server Hello
Content Type: Handshake (22)
Length: 122
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 118
Random: c405943892dde6a87cb91a6a57740dbdfcaa1a3b87565d59...
Session ID Length: 32
Session ID: 49423a906a64d0d161ff0c0c62bc67a9c6181c65a7dae31e...
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Compression Method: null (0)
Extensions Length: 13
Extension: supported_versions (len=2)
Supported Version: TLS 1.3 (0x0304)
Extension: key_share (len=36)
Key Share extension
Key Share Entry: Group: x25519, Key Exchange length: 32
Key Exchange: b754b50cfc3f1e6031d5ba9e05c3b6095c57f43127d...

TLS 1.3: SERVER CHANGE CIPHER SPEC

- Encrypt everything using symmetric key

TLS 1.3: ENCRYPTED – SERVER CERTIFICATE AND FINISHED

- All the certificate info as before, except encrypted



TLS 1.3: SERVER CHANGE CIPHER SPEC

TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
Content Type: Change Cipher Spec (20)
Version: TLS 1.2 (0x0303)
Length: 1
Change Cipher Spec Message

TLS 1.3: ENCRYPTED – SERVER CERTIFICATE AND FINISHED

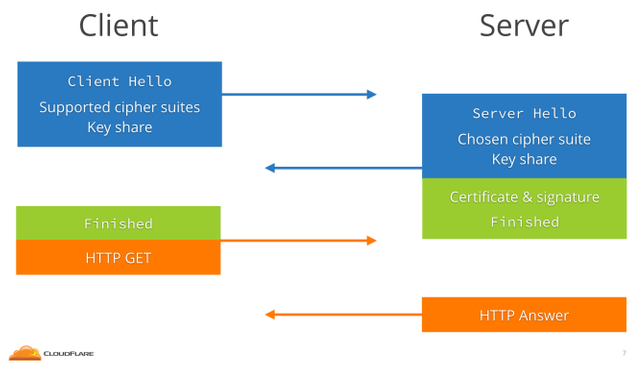
TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
Opaque Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 32
Encrypted Application Data: 1e5507ebe81fab6eff7f3df93f5deaef4b0375a879a9...

TLS 1.3: CLIENT CHANGE CIPHER SPEC

- To use symmetric keys

TLS 1.3: ENCRYPTED – CLIENT FINISHED

- Application data can now begin flowing between client and server - encrypted



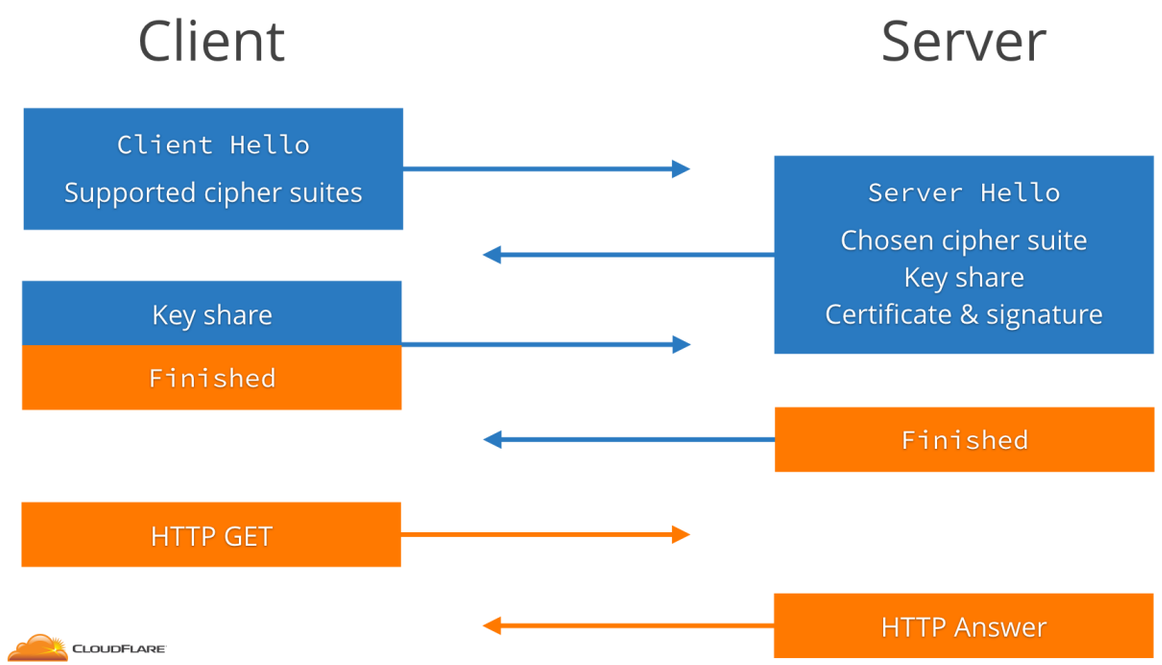
TLS 1.3: CLIENT CHANGE CIPHER SPEC

TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message

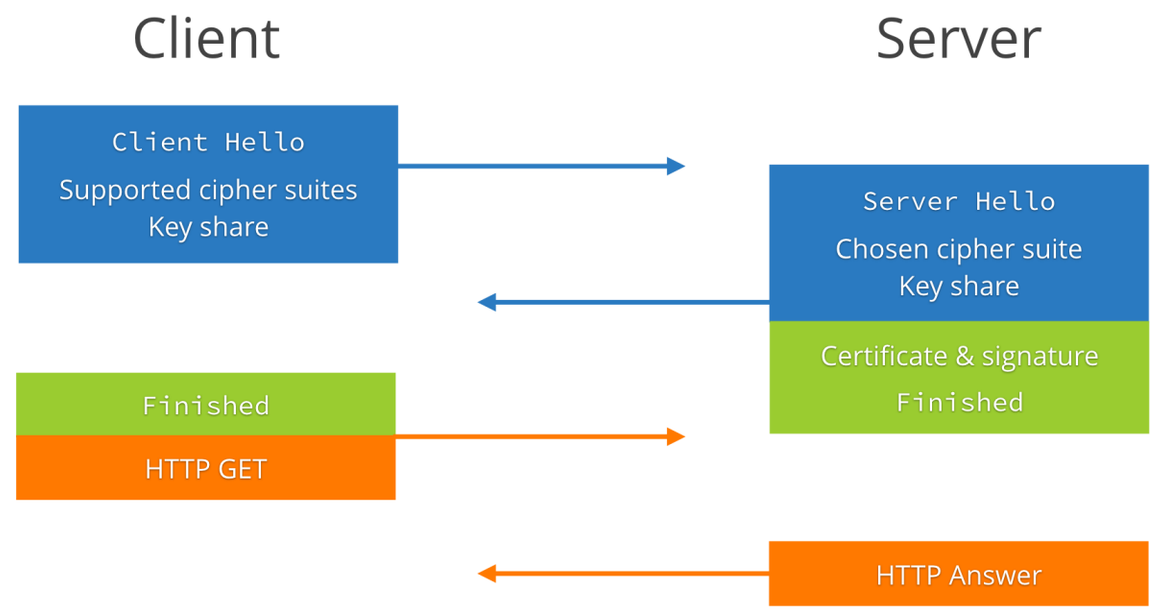
TLS 1.3: ENCRYPTED – CLIENT FINISHED

TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 69
    Encrypted Application Data: 52502d5efc2fc17f51c7096dd21258df6c15220e62b...

TLS 1.2


Client
Server
Client Hello
Supported cipher suites
Server Hello
Chosen cipher suite
Key share
Certificate & signature
Key share
Finished
Finished
HTTP GET
HTTP Answer
CLOUDFLARE

FASTER AND MORE SECURE – TLS 1.3



ADVANTAGES OF 1.3

- Shorter handshake = faster
- Old, weaker cipher suites removed
- No TLS version negotiation (downgrade attacks)
- Perfect Forward Secrecy ciphers required
  - New symmetric keys are created for each transaction

TLS 1.2 & 1.3 ONLY WORK PROPERLY IF . . .

- Trust in Certificate Authorities
- Certificate details haven't been compromised
  - Pointing DNS records to a different, malicious IP
- Private keys stay private
- Server configured correctly
  - MITM can resubmit client packet (like money transfer). These are called ***replay attacks***.

IF CONFIGURED PROPERLY, HTTPS PREVENTS...

- View or modification of web traffic (MITM attacks)

IF...

- ALL connections use and force HTTPS
  - ***Attacks***: SSL Stripping, Insecure images or CSS, etc
- CDNs, advertisers, external resources use HTTPS
- All links use HTTPS
  - ***Common Example***: Insecure links in email trackers

HTTPS DOESN'T PREVENT OR PROTECT FROM...

- Tracking of metadata
- Malware distribution from your site or malicious site
  - Malicious websites use SSL certificates too
- Bad/insecure code
- Man-in-the-browser/device attacks
  - Malicious browser extensions or screen capture malware

HTTPS DOESN'T PREVENT OR PROTECT FROM...

- DNS Records compromise
- Content Scanning/SSL Inspection products

WHAT CAN YOU DO?

- Use HTTPS **EVERYWHERE** on your sites
- Learn how to install & configure it correctly
- Use TLS 1.3 where you can and at least use TLS 1.2
- Know where you're still vulnerable, and set up other protections
  - Like 2-factor authentication, code review/QA, and good code security practices
- Keep on learning

KEEP ON LEARNING

- SSL Labs - https://www.ssllabs.com/
- The first few milliseconds of HTTPS by Joshua Thijssen at PHPNW16 - https://www.youtube.com/watch?v=4Jg2ALtioMA
- Why HTTPS For Everything? https://https.cio.gov/everything/
- Understanding the Limitations of HTTPS https://textslashplain.com/2018/02/14/understanding-the-limitations-of-https/
- An Overview of TLS 1.3 and Q&A https://blog.cloudflare.com/tls-1-3-overview-and-q-and-a/