Let's go

#CiscoLive

Cisco Webex App

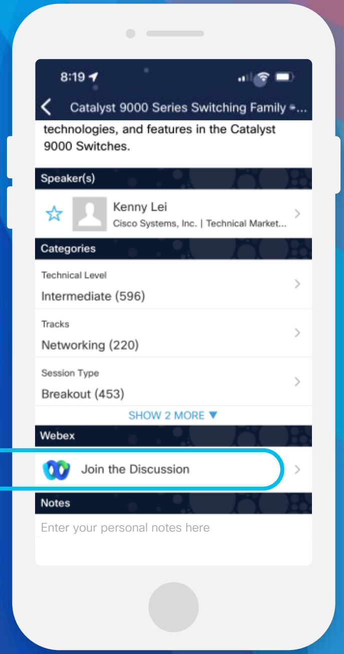
Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 9, 2023.



https://ciscolive.ciscoevents.com/ciscolivebot/#BRKSEC-3320

cisco Live!

The bridge to possible

Demystifying TLS Decryption and Encrypted Visibility Engine on Cisco Secure Firewall Threat Defence

Christopher Grabowski
Technical Marketing Engineer, Security Business Group
&
George M Koikara
Principal Engineer, Security Business Group

BRKSEC-3320

Your TLS Speaker

Christopher Grabowski
Technical Marketing Engineer
CCIE Security #42466

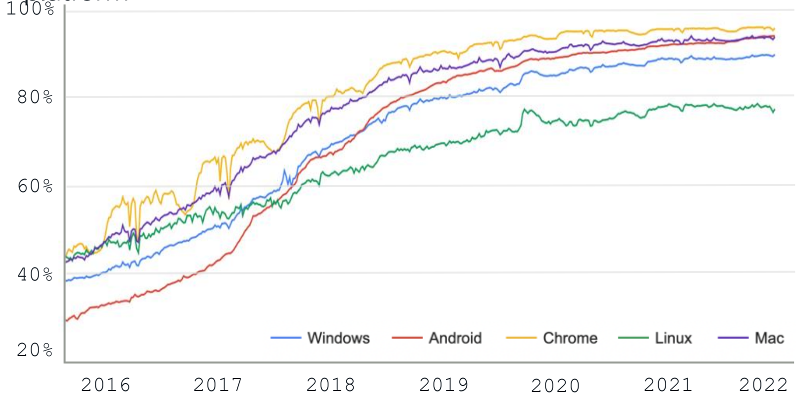
- Based in Warsaw, Poland
- With Cisco since May 2012
- Started with TAC Security, then Advanced Services, now Technical Marketing Engineer
- Focusing on Identity Firewall, SDA/ACI Integration and TLS Decryption
- Enjoys cooking and spending time with the family

What is an average parent tempted to do when seeing this at home?

TLS Encryption is Almost Everywhere...

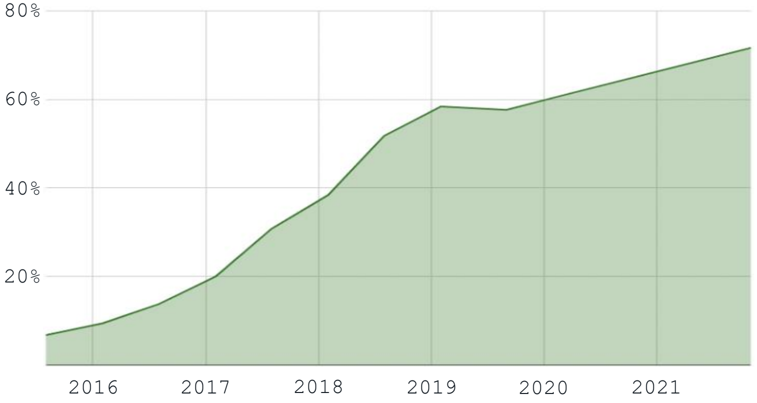
<u>Over 90% of Internet traffic is being encrypted with Transport Layer Security (TLS)</u>

Percentage of pages loaded over HTTPs in Chrome by platform



Source: https://transparencyreport.google.com/https/overview

Percentage of Alexa Top 1M redirecting to HTTPS



Source: https://scotthelme.co.uk/top-1-million-analysis-november-2021

... and So Are the Threats!!!

“76% OF CRITICAL AND HIGH-RISK THREATS detected by Cisco Secure Network Analytics were **discovered in** ENCRYPTED TRAFFIC”

Cisco Encrypted Traffic Analytics White Paper, Cisco Systems

*”If **attackers know that most defenders aren’t scanning encrypted traffic** [...] it doesn’t take high volume for attacks to succeed.”*

Internet Security Report Q3 2022, WatchGuard

*”More than 85% of attacks now use encrypted channels **across various stages of the kill chain** (phishing, malware delivery, C&C activity, and more)”*

The State of Encrypted Attacks 2022, ThreatLabz

*“Of all respondents who were victims of a cyberattack, nearly **half claimed the attack leveraged SSL traffic** to evade detection. Another 15 percent were unsure.”*

Uncovering Hidden Threats within Encrypted Traffic 2018, Ponemon Institute

Well, duh?!

Decrypt and inspect your traffic!!!

The TLS Decryption Paradox

TLS was designed to ensure malicious actors cannot see or alter what you transmit over the open network.

With TLS decryption, we are trying to break into a protocol specifically designed to protect against it...

Top Respondent Obstacles for TLS Decryption

Lack of proper tools – hardware and software

Challenging technically to configure and operate

Concern of degradation of service

Insufficient Resources and capacity

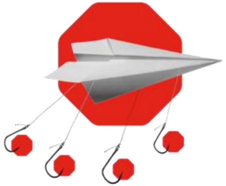
Agenda

- A TLS 1.3 Handshake Walkthrough
- TLS Decryption Under the Hood
- TLS Decryption Challenges
- Introduction to QUIC
- Challenges posed by QUIC
- EVE Overview

A TLS 1.3 Handshake Walkthrough

The Primary Goals of the TLS Handshake

- Negotiate encryption scheme and parameters
- Authenticate the server (and optionally the client)
- Calculate shared keying material



Assume handshake runs over an unsecure channel

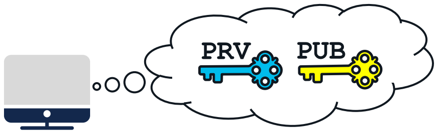
Prevent Man-in-the-Middle and eavesdropping

Understanding a TLS Session Flow – Client Hello

My offer is cool!!! The server will accept for sure...

Why do we consider the TLS 1.3 handshake optimistic?

Understanding a TLS Session Flow – Client Hello



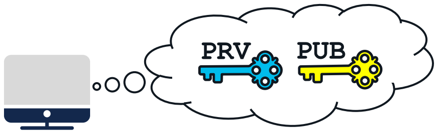
TLS Client

TLS Server

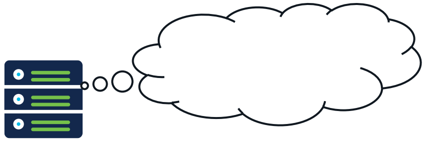
The client **makes a guess** of server's preferences to accelerate the handshake...

... and blindly generates a **public/private key pair**.

Understanding a TLS Session Flow – Client Hello



TLS Client



TLS Server

Knowing, client's public key so early in the handshake, allows TLS 1.3 to send most of the Server's messages encrypted.

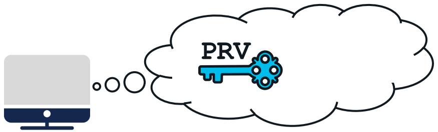
CLIENT HELLO

```
Client Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec…<RANDOM>

Cipher Suite (list) =
  TLS_AES_128_GCM_SHA256
  TLS_AES_256_GCM_SHA384
  TLS_CHACHA20_POLY1305_SHA256
  ...

SNI = www.cisco.com

Extension: Supported Versions =
  TLS 1.2, TLS 1.3

Extension: Key Share =
  Group: x25519
  Length: 32
  Key Exchange:      PUB
```

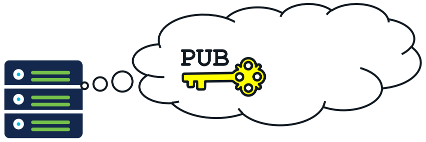
The Client provides an ordered list of supported cipher suites. The higher on the list, the more preferable by the Client.

The client's ECDHE public key is shared with the TLS server in the first handshake message.

Understanding a TLS Session Flow – TLS Version Negotiation



TLS Client



TLS Server

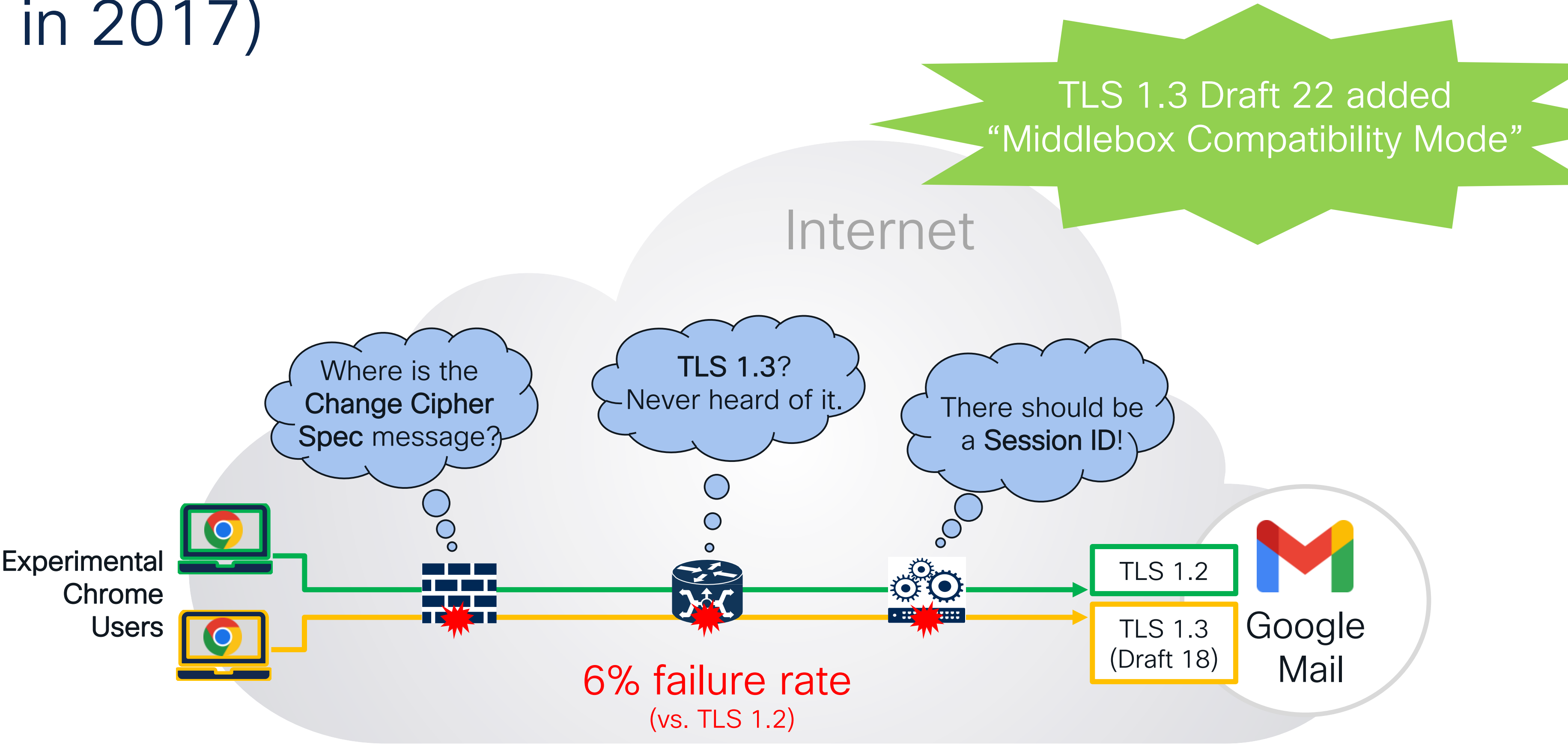
CLIENT HELLO

```
Client Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec…<RANDOM>

Cipher Suite (list) =
  TLS_AES_128_GCM_SHA256
  TLS_AES_256_GCM_SHA384
  TLS_CHACHA20_POLY1305_SHA256
  ...

SNI = www.cisco.com

Extension: Supported Versions =
  TLS 1.2, TLS 1.3

Extension: Key Share =
  Group: x25519
  Length: 32
  Key Exchange:
```

The Client Version is set to TLS 1.2.

TLS 1.2?! …is that correct?

The Experimental Launch of TLS 1.3 Draft (back in 2017)

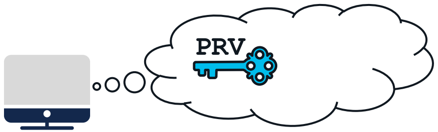


The solution being... a **convincing** disguise

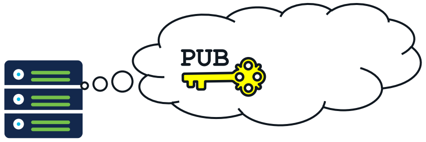
Middlebox Compatibility Mode:

- Make the TLS 1.3 handshake look like TLS 1.2 session resumption
- Include a non-empty Session ID
- Send a dummy ChangeCipherSpec record

Understanding a TLS Session Flow – Middlebox Compatibility



TLS Client



TLS Server

CLIENT HELLO

```
Client Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec…<RANDOM>

Cipher Suite (list) =
  TLS_AES_128_GCM_SHA256
  TLS_AES_256_GCM_SHA384
  TLS_CHACHA20_POLY1305_SHA256
  ...

SNI = www.cisco.com

Extension: Supported Versions =
  TLS 1.2, TLS 1.3

Extension: Key Share =
  Group: x25519
  Length: 32
  Key Exchange:
```

The Client Version and Session ID fields are redundant from TLS 1.3 standpoint. We see them in the handshake to pass through ossified middlebox devices.

The Supported Versions extension is the only indication the client supports TLS 1.3.

Understanding a TLS Session Flow – Server Name Indication (SNI)

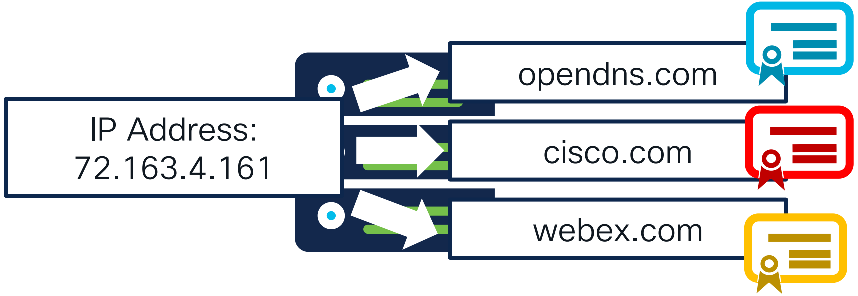
TLS Client

TLS Server

CLIENT HELLO

```
Client Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec…<RANDOM>

Cipher Suite (list) =
  TLS_AES_128_GCM_SHA256
  TLS_AES_256_GCM_SHA384
  TLS_CHACHA20_POLY1305_SHA256
  ...

SNI = www.cisco.com

Extension: Supported Versions =
  TLS 1.2, TLS 1.3

Extension: Key Share =
  Group: x25519
  Length: 32
  Key Exchange:
```

Why do we need the **clear text** Server Name Indication (SNI) extension?



Understanding a TLS Session Flow – Server Name Indication (SNI)

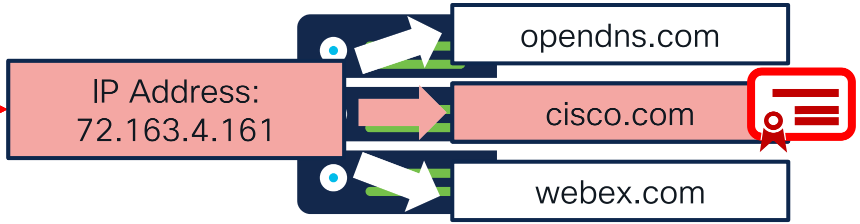
TLS Client

TLS Server

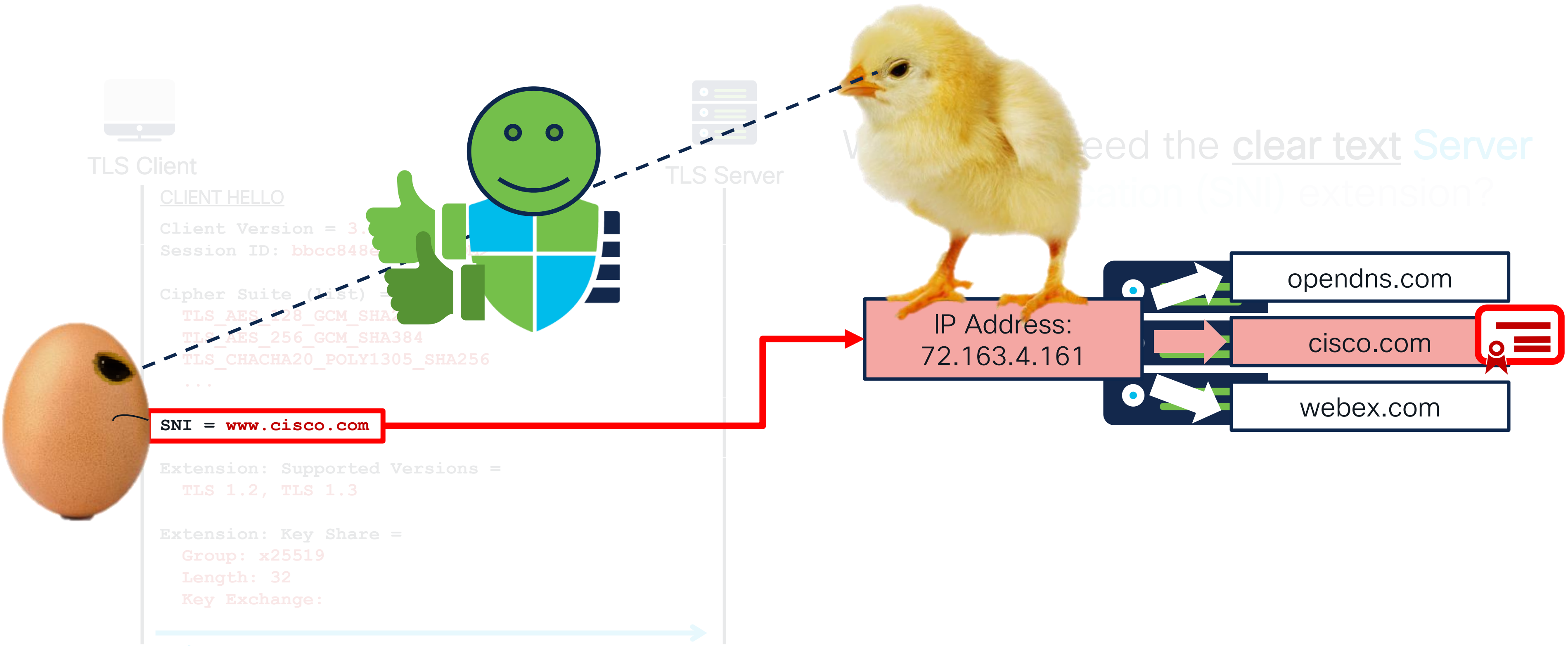
CLIENT HELLO

```
Client Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec…<RANDOM>

Cipher Suite (list) =
  TLS_AES_128_GCM_SHA256
  TLS_AES_256_GCM_SHA384
  TLS_CHACHA20_POLY1305_SHA256
  ...

SNI = www.cisco.com

Extension: Supported Versions =
  TLS 1.2, TLS 1.3

Extension: Key Share =
  Group: x25519
  Length: 32
  Key Exchange:
```

Why do we need the clear text Server Name Indication (SNI) extension?



Understanding a TLS Session Flow – Server Name Indication (SNI)



Understanding a TLS Session Flow – Server Hello

TLS Client

TLS Server

CLIENT HELLO

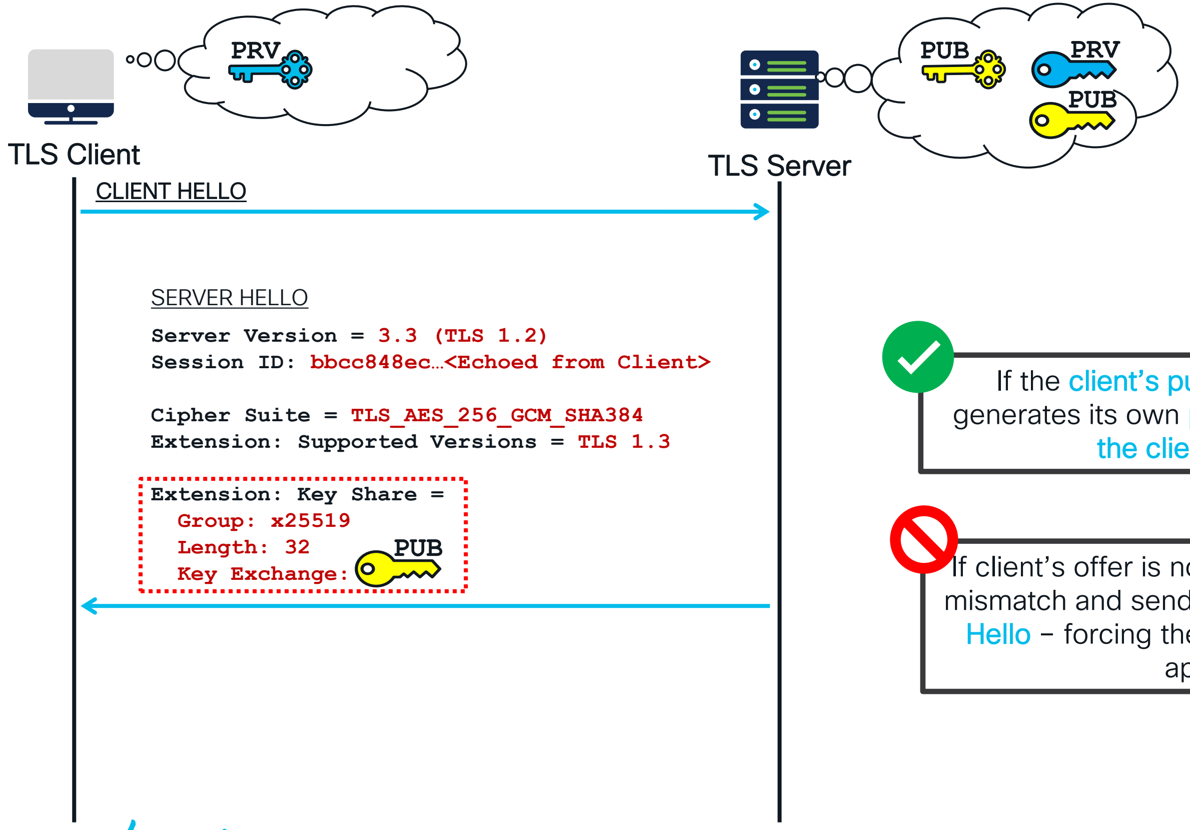
SERVER HELLO

```
Server Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec…<Echoed from Client>

Cipher Suite = TLS_AES_256_GCM_SHA384
Extension: Supported Versions = TLS 1.3

Extension: Key Share =
  Group: x25519
  Length: 32
  Key Exchange:
```

The server continues the Middlebox Compatibility masquerade and echoes TLS 1.2 Client Version and Session ID back to the client.

The server indicates the negotiated TLS 1.3 version along with server selected Cipher Suite.

Understanding a TLS Session Flow – Server Hello

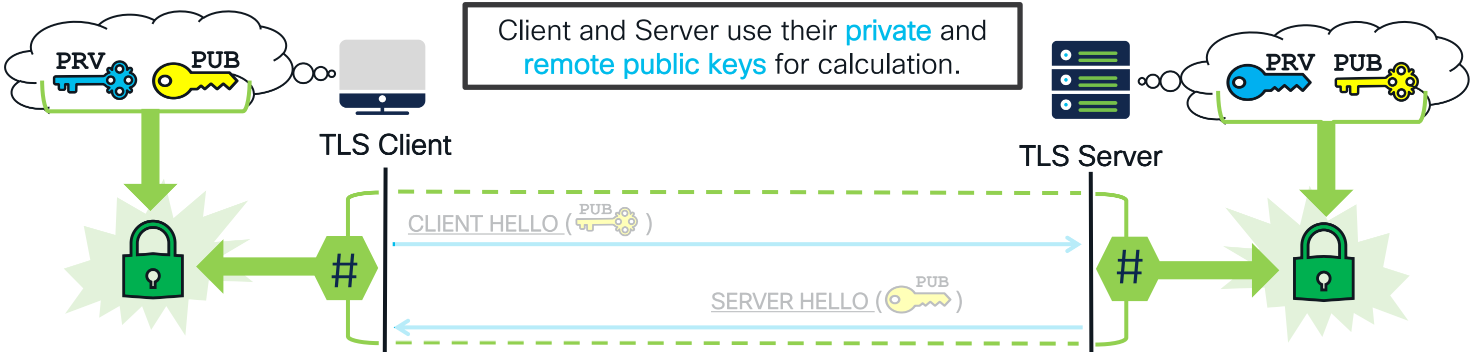


If the client's public key is acceptable, the Server generates its own public/private key pair and sends it to the client in Key Share extension.

If client's offer is not acceptable, the Server corrects the mismatch and sends HelloRetryRequest instead of Server Hello – forcing the client to restart the handshake with appropriate Key Share.

Understanding a TLS Session Flow – Calculating the Shared Keying Material

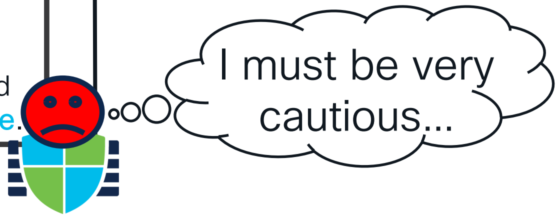
Client and Server use their private and remote public keys for calculation.



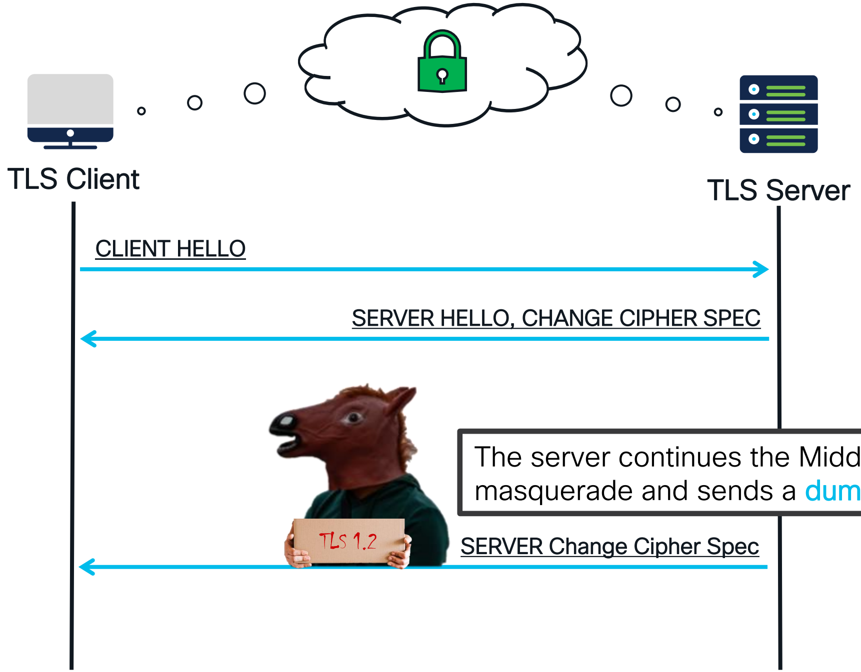
The same keying material ( ) is now available on both the Client and the Server for bulk encryption.

Both ends use a hash of Client Hello and Server Hello messages in shared secret calculation.

Even if a single bit if changed by a MITM device, the calculation results in error and handshake fails in a downstream exchange.

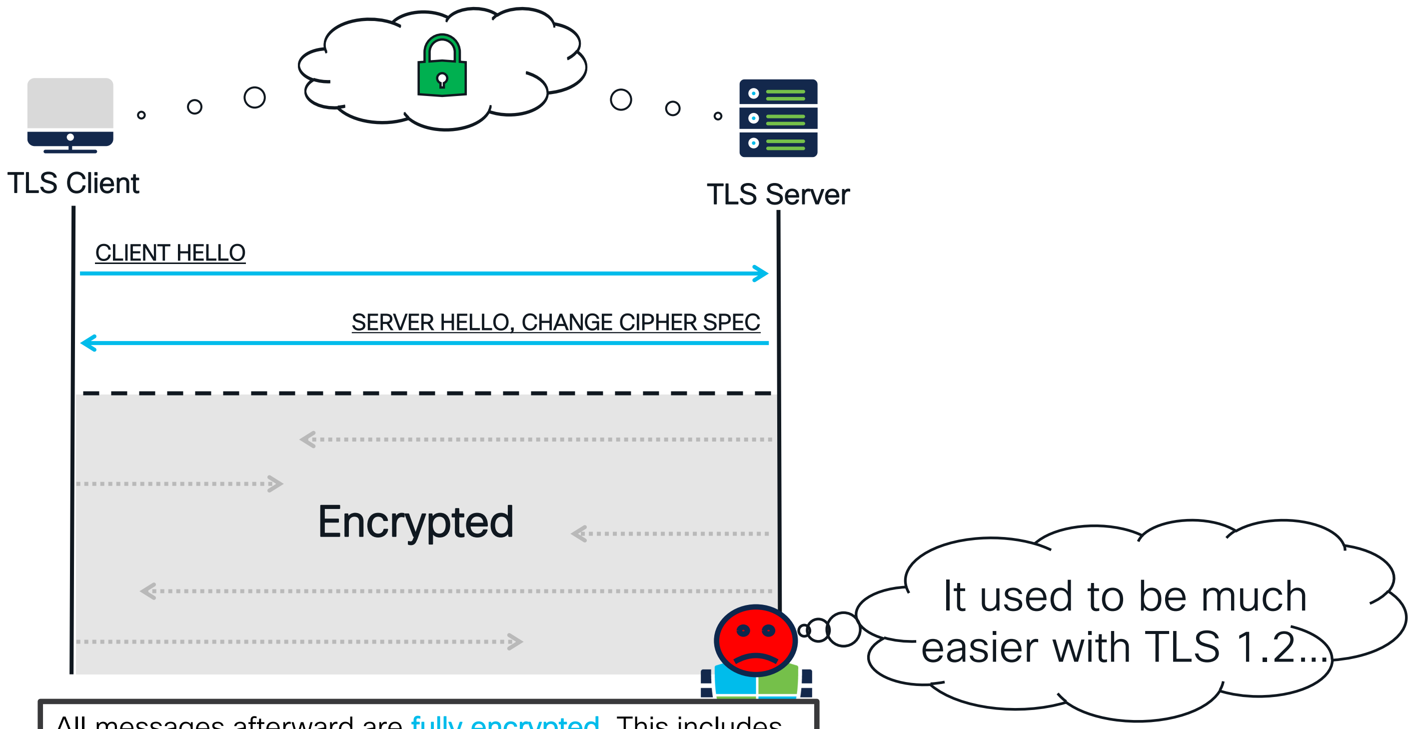


Understanding a TLS Session Flow – Encrypted Handshake



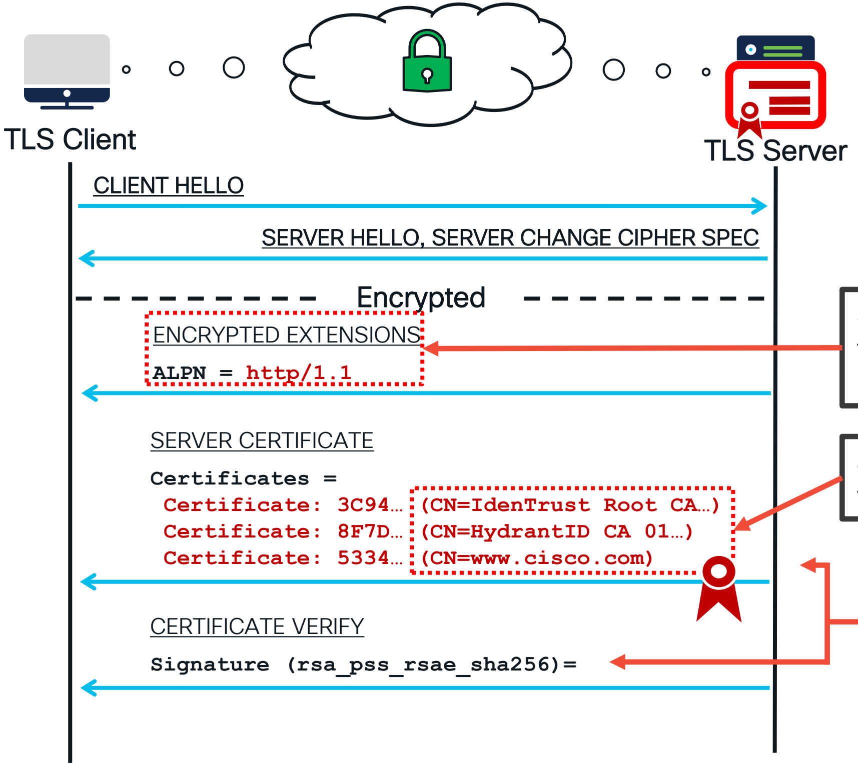
The server continues the Middlebox Compatibility masquerade and sends a dummy Change Cipher Spec.

Understanding a TLS Session Flow – Encrypted Handshake



All messages afterward are fully encrypted. This includes server certificate, ALPN and other extensions.

Understanding a TLS Session Flow – Encrypted Extensions and Certificate

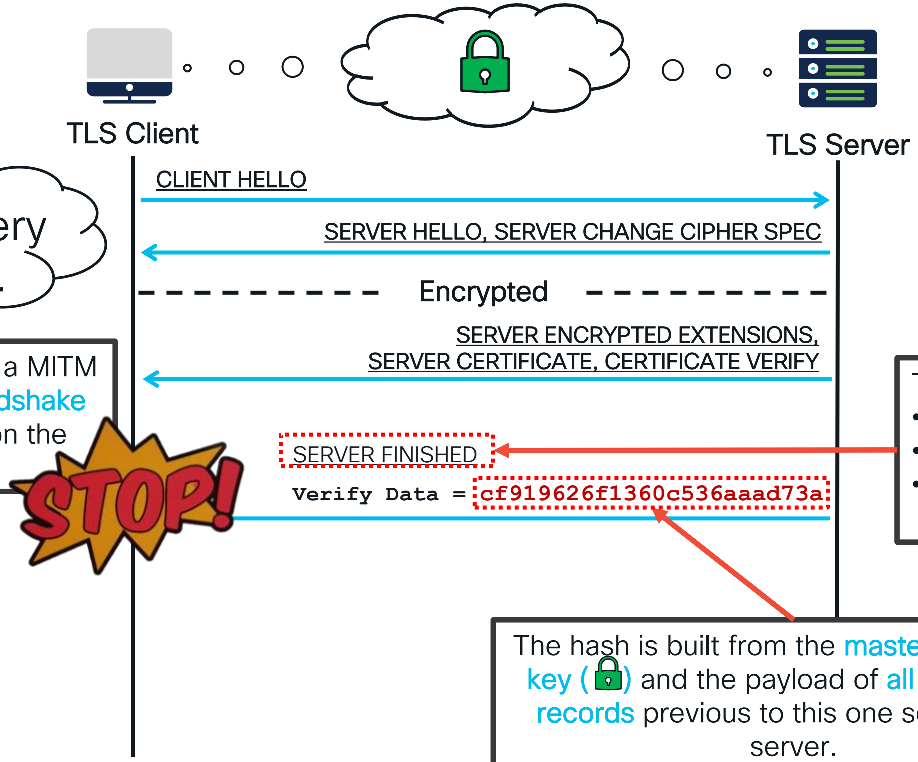


Server may securely share additional extensions with the client. Application-Layer Protocol Negotiation (ALPN) being the most common.

Server provides its own certificate (www.cisco.com) along with all certificates in the trust chain up to the root.

Server has to authenticate itself to client. It uses its certificate private key (www.cisco.com) to sign hash of the handshake messages.

Understanding a TLS Session Flow – Server Finished



I must be very cautious...

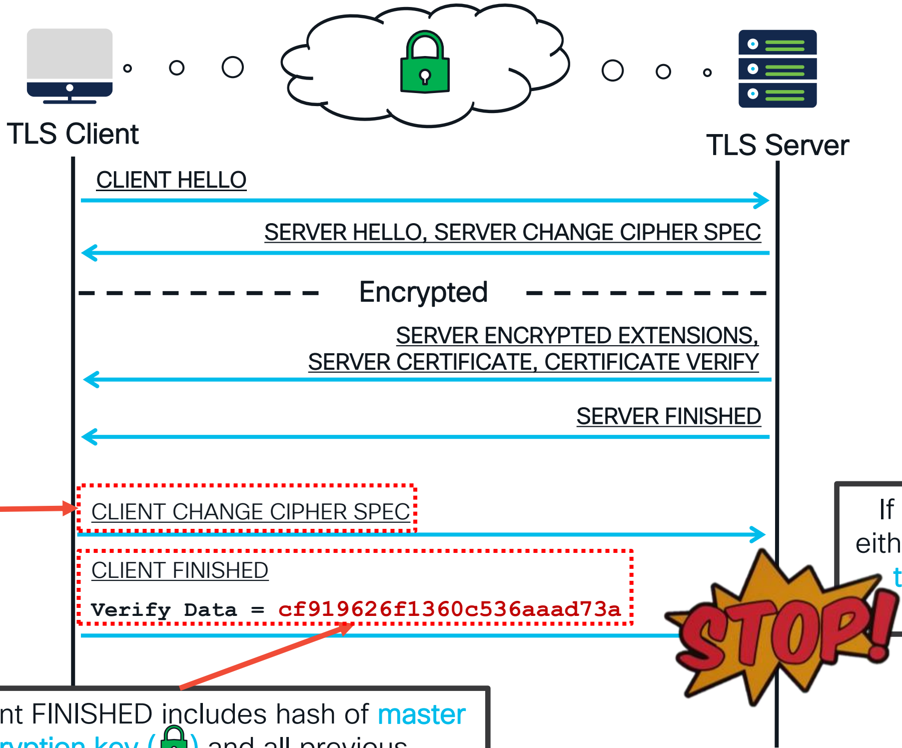
Even if a single bit is changed by a MITM entity in any of the previous handshake messages the hash will differ on the Client side.

The FINISHED message verifies:

- the handshake was successful,
- the keys work (),
- the handshake has not been tampered with.

The hash is built from the master encryption key () and the payload of all handshake records previous to this one seen by the server.

Understanding a TLS Session Flow – Client Side Finish

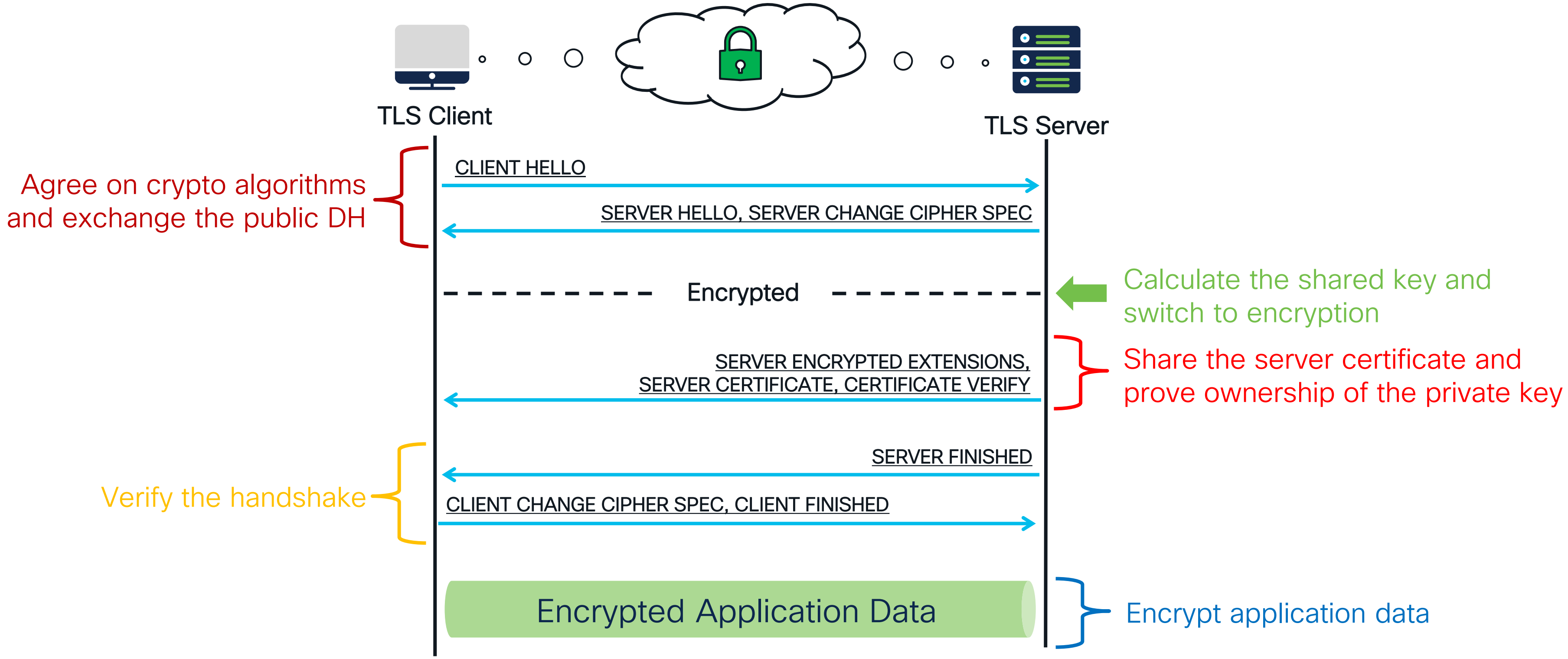


The client side dummy Change Cipher Spec.

If the Verify Data is not valid, on either side, the receiving peer must terminate the handshake with "decrypt_error" alert.

The Client FINISHED includes hash of master encryption key (🔒) and all previous messages (including Server's Handshake Finished)

Understanding a TLS Session Flow – Client Side Finish



TLS 1.3 Decryption Under the Hood

Decryption Policy – Rule Conditions

Before TLS Handshake
FTD has the L2-L4 information about the flow
Interface Zones, Networks, Geolocation, VLAN Tags, User Identity, Protocol and Ports
TLS handshake inspection not required

Low — Performance Impact / Visibility — High

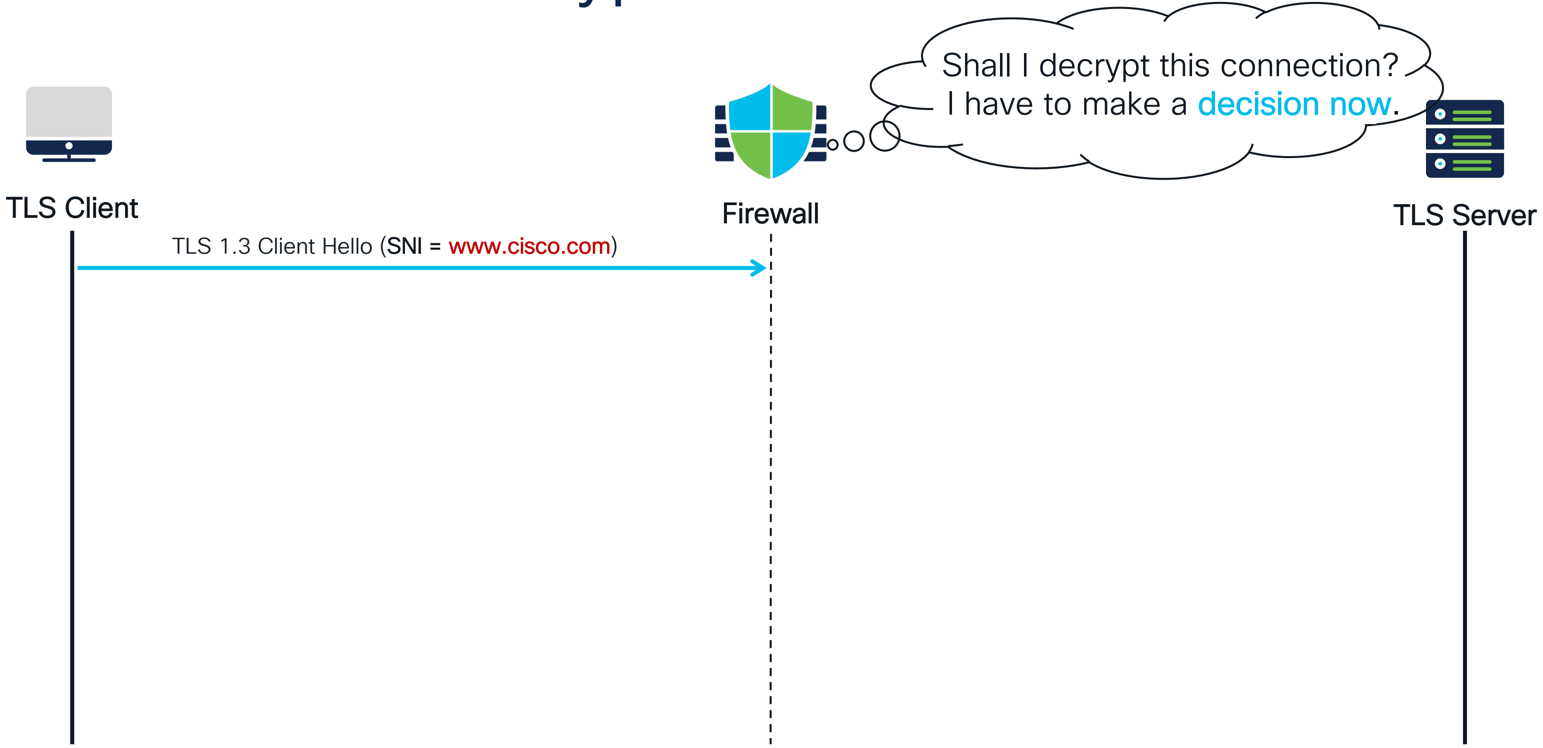
REFERENCE

Decryption Policy – Rule Conditions

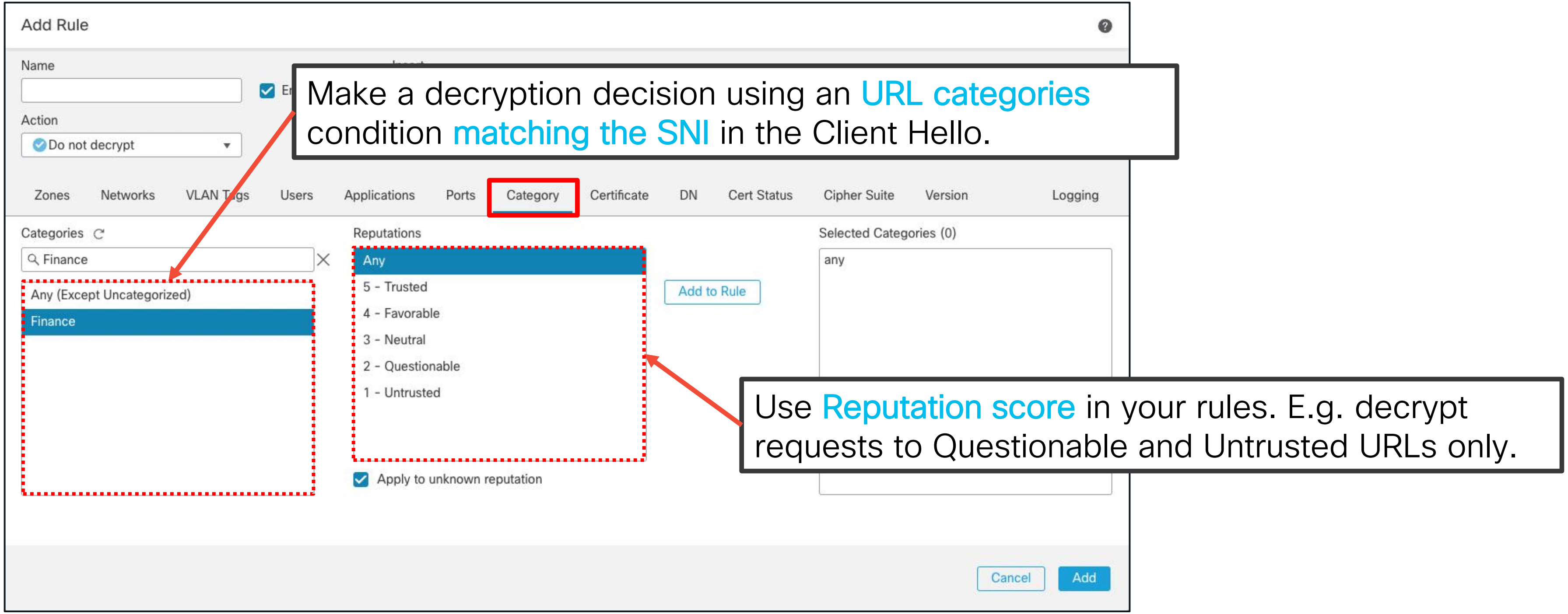
Before TLS Handshake	TLS Client Hello	TLS Server Response
FTD has the L2-L4 information about the flow	FTD has the additional information of SNI (Server Name Identification)	Server Certificate information and Server Hello
Interface Zones, Networks, Geolocation, VLAN Tags, User Identity, Protocol and Ports	**Application, URL Category and Reputation**	**Certificate attributes, Ciphers, Versions, SNI mismatch**
TLS handshake inspection not required	Initial match using SNI	Most reliable information

Low — Performance Impact / Visibility → High

TLS Session Decryption Flow - Client Hello



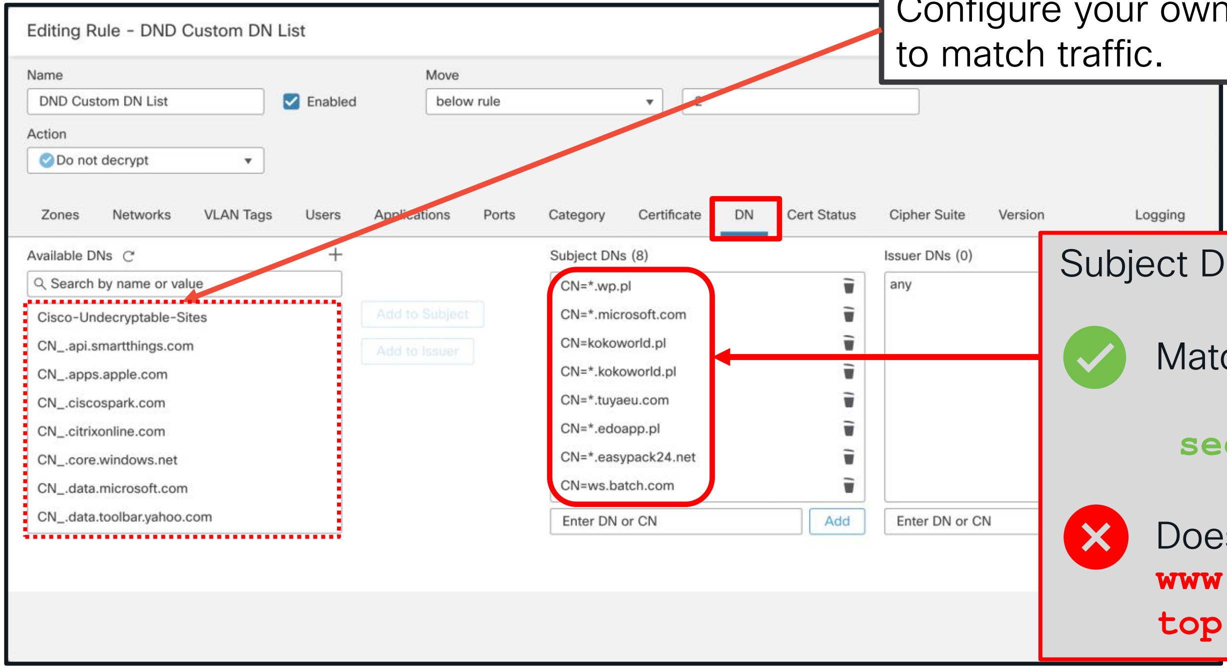
URL Detectors(SNI)



https://www.talosintelligence.com/categories

Subject Distinguished Name Condition

Configure your own Distinguished Names objects to match traffic.



Subject DN: `*.acme.com`

Matches:

`www.acme.com`

`secure.acme.com`

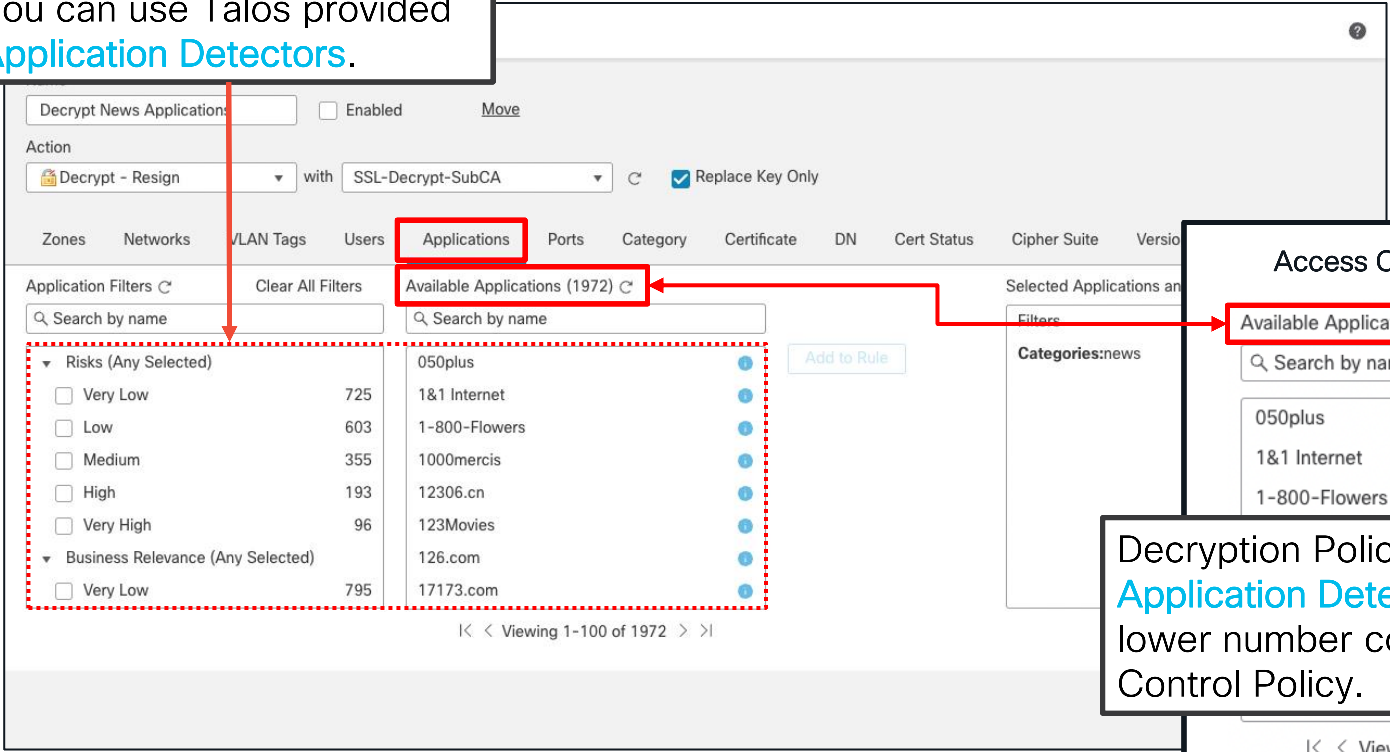
Does not match:

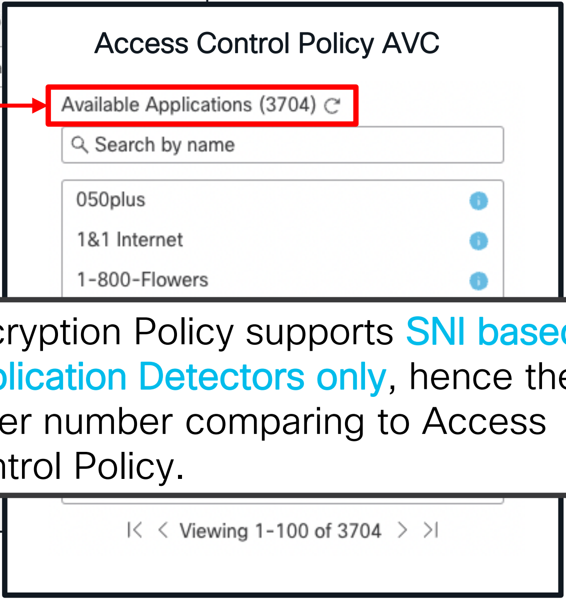
`www.sub.acme.com`

`top.svc.acme.com`

Application Detectors (SNI)

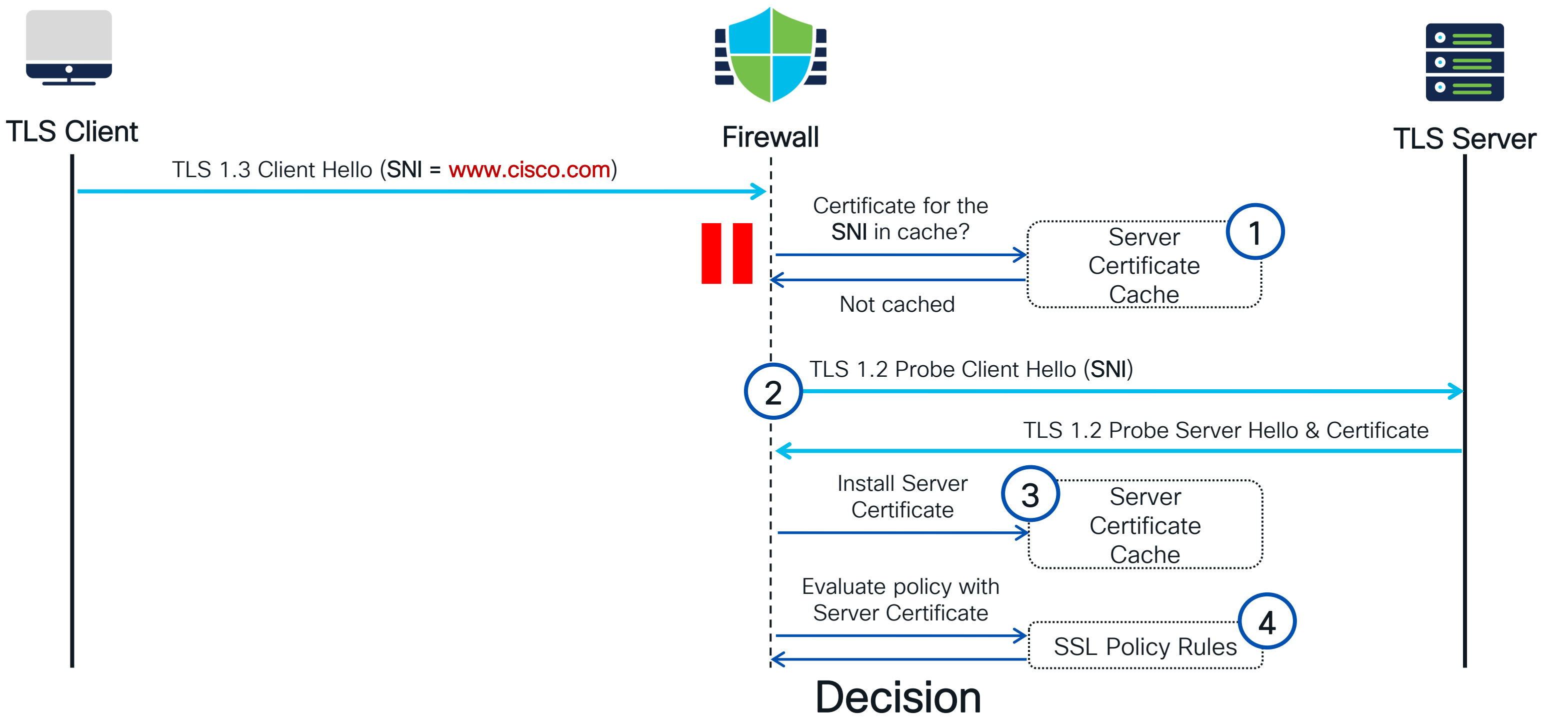
You can use Talos provided **Application Detectors**.



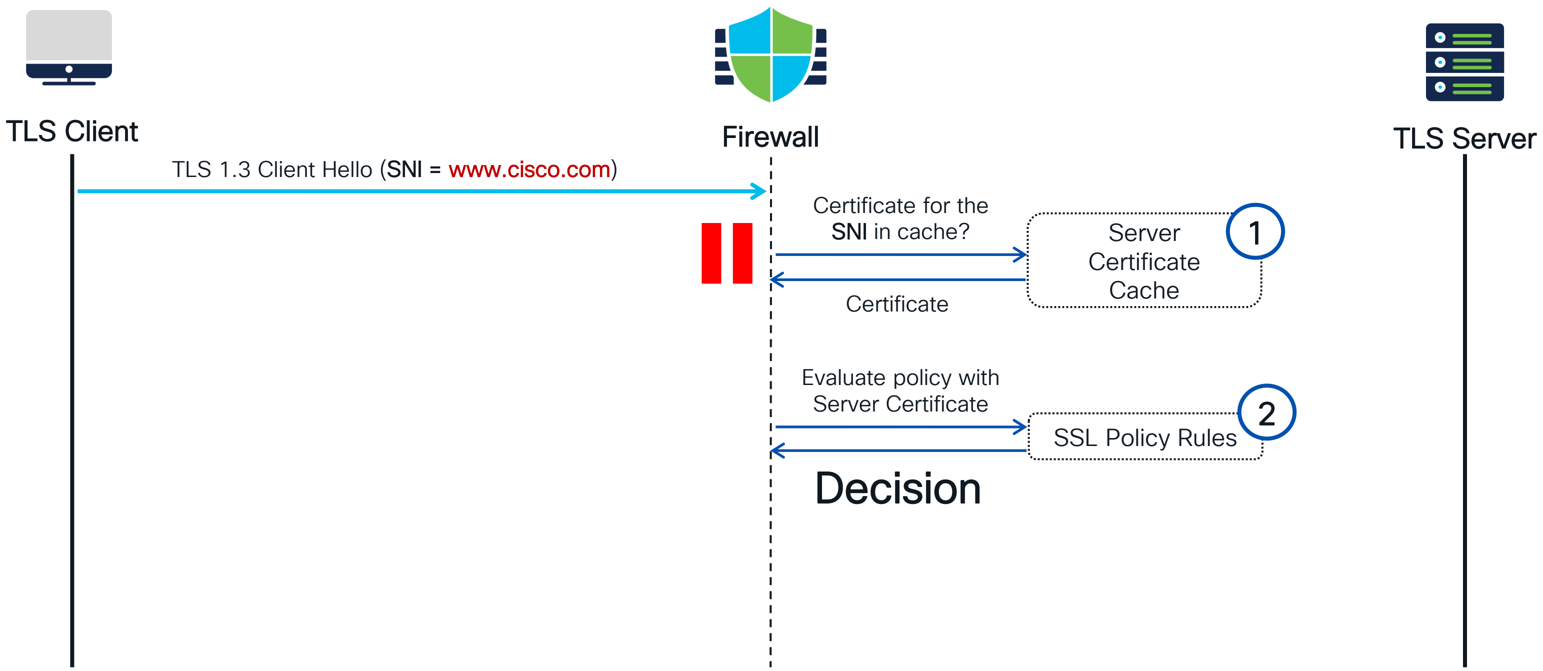


Decryption Policy supports **SNI based Application Detectors only**, hence the lower number comparing to Access Control Policy.

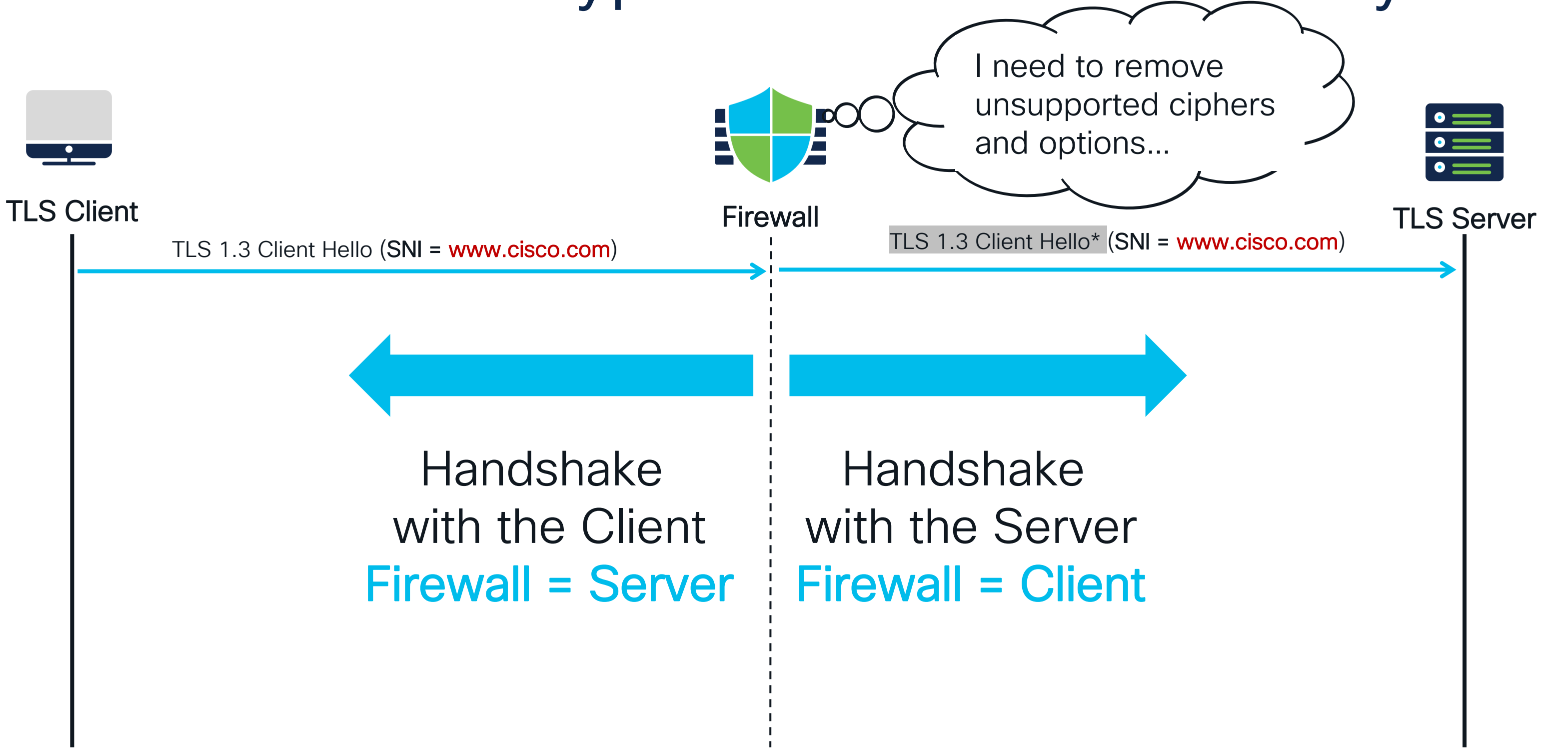
TLS Session Decryption Flow - Client Hello



TLS Session Decryption Flow - Client Hello



TLS Session Decryption Flow – CH Modify



* – modified message

REFERENCE

Under the hood: CH Processing - Modify

version: 3.3
random: e372...676b
session id [32]: 5c3d...9002
cipher_suites len[32]: fafa 0113 0213 0313 2bc0 2fc0 2cc0 30c0 a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500
compression_methods len[1]: 00
---extensions---
grease[19018]: len[0]
server_name[0]: len[18] server name indication: **www.cisco.com**
extended_master_secret[23]: len[0] Extended Master Secret: enabled
renegotiation_info[65281]: len[1]
supported_groups[10]: len[10] 9a9a 001d 0017 0018
ec_point_formats[11]: len[2] 00
session_ticket[35]: len[0] Session ticket is Empty
alpn_extension[16]: len[14] alpn_list_len[12]
ALPN list Entries: **h2 http/1.1**
status_request[5]: len[5]
signature_algorithms[13]: len[18] 0403 0804 0401 0503 0805 0501 0806 0601
signed_cert_timestamp[18]: len[0]
key_share[51]: len[43] groups: grease(39578) x25519(29)
psk_key_exchange_modes[45]: len[2]
supported_versions[43]: len[7] 7a7a 0304 0303
compress_certificate[27]: len[3]
unknown[17513]: len[5]
grease[64250]: len[1]
padding_extension[21]: len[202]

version: 3.3
random: **0292...71f6 (NEW VALUE)**
session id [0]: **<ZEROIZED>**
cipher_suites len[20]: ~~fafa 0113 0213 0313~~ 2bc0 2fc0 2cc0 30c0 ~~a9cc a8cc~~ 13c0 14c0 9c00 9d00 2f00 3500

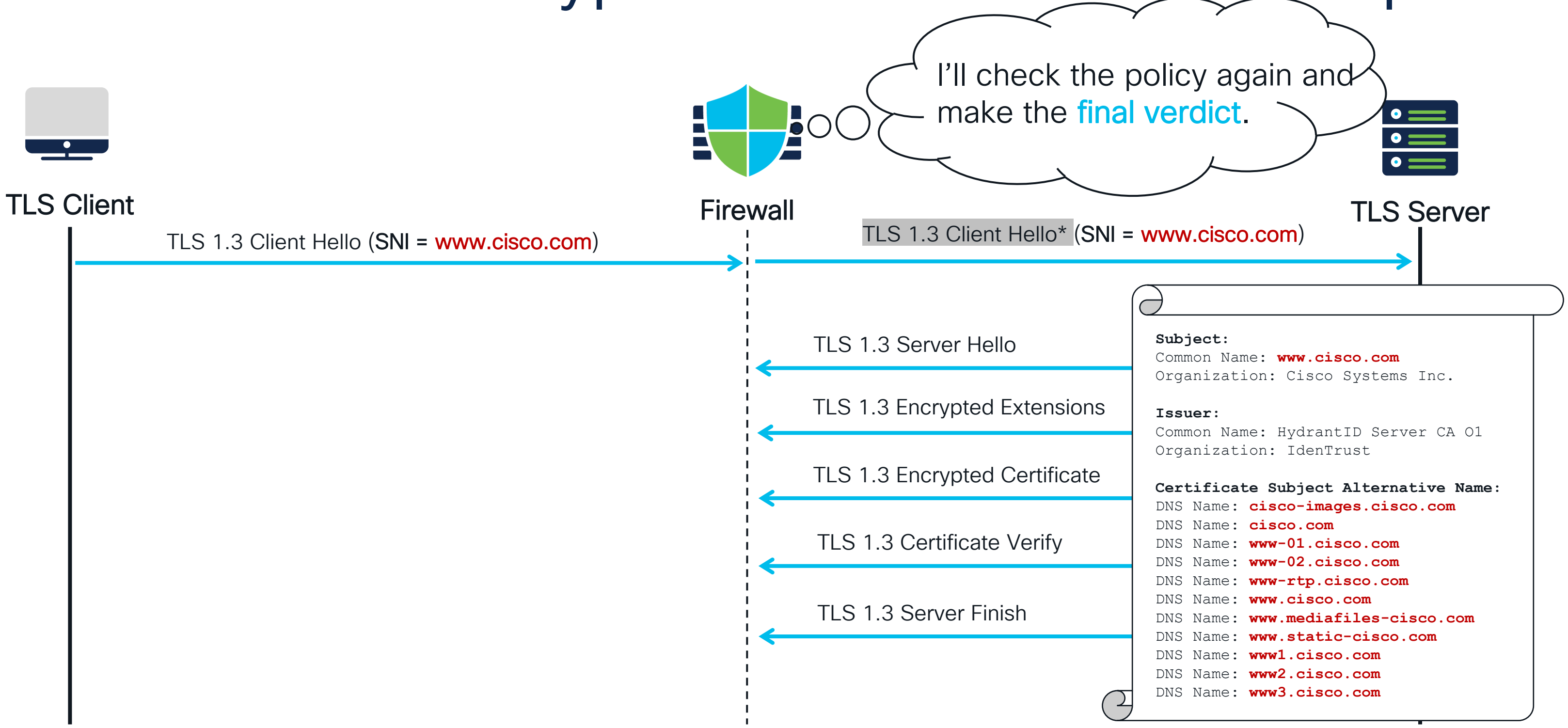
Generate a new Random and zeroize Session ID.

Remove unsupported Cipher Suites.

grease[19018]: len[0]
server_name[0]: len[18] server name indication: **www.cisco.com**
extended_master_secret[23]: len[0] Extended Master Secret: enabled
renegotiation_info[65281]: len[1]
supported_groups[10]: len[8] ~~9a9a~~ 001d 0017 0018
ec_point_formats[11]: len[2] 00
session_ticket[35]: len[0] Session ticket is Empty
alpn_extension[16]: len[14] alpn_list_len[12]
ALPN list Entries: **h2 http/1.1**
status_request[5]: len[5]
signature_algorithms[13]: len[18] 0403 0804 0401 0503 0805 0501 0806 0601
signed_cert_timestamp[18]: len[0]
~~key_share[51]: len[43] groups: grease(39578) x25519(29)~~
~~psk_key_exchange_modes[45]: len[2]~~
~~supported_versions[43]: len[7] 7a7a 0304 0303~~
~~compress_certificate[27]: len[3]~~

Remove TLS 1.3 extensions to downgrade the session to TLS 1.2.

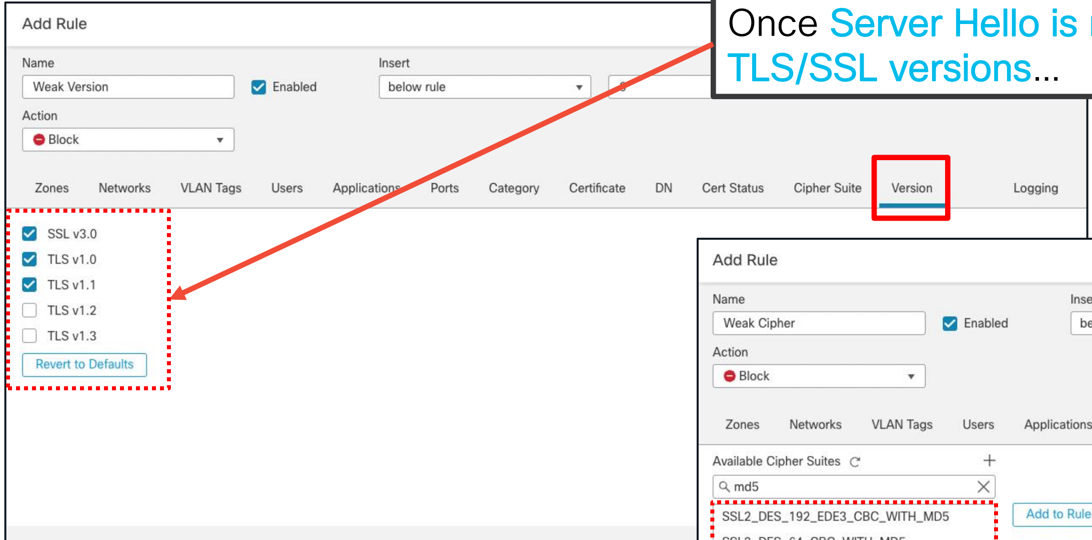
TLS Session Decryption Flow – Server Response



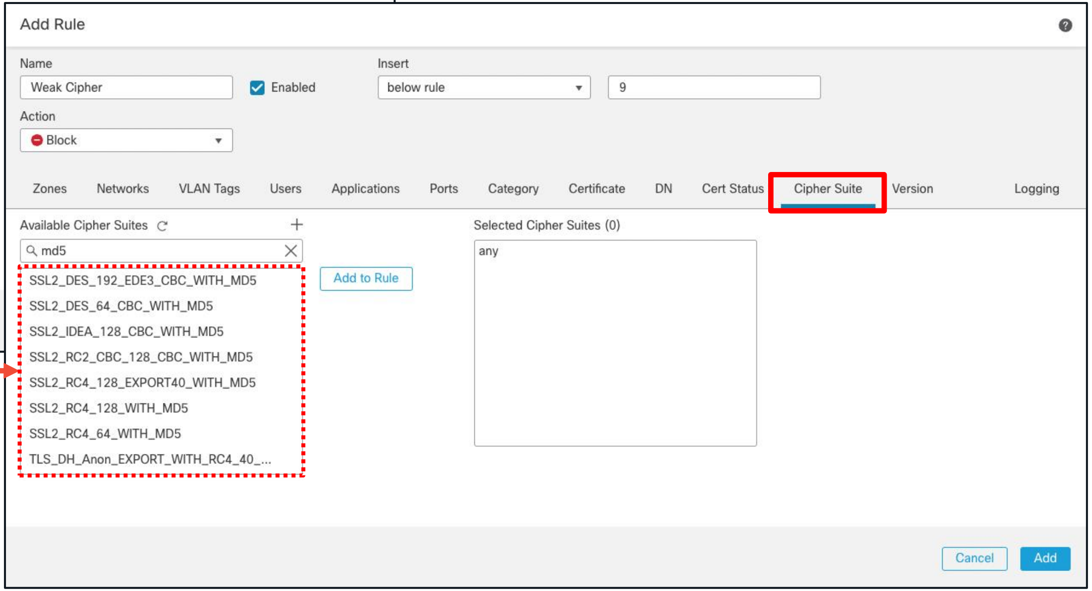
* – modified message

Block Weak Ciphers and TLS/SSL Versions

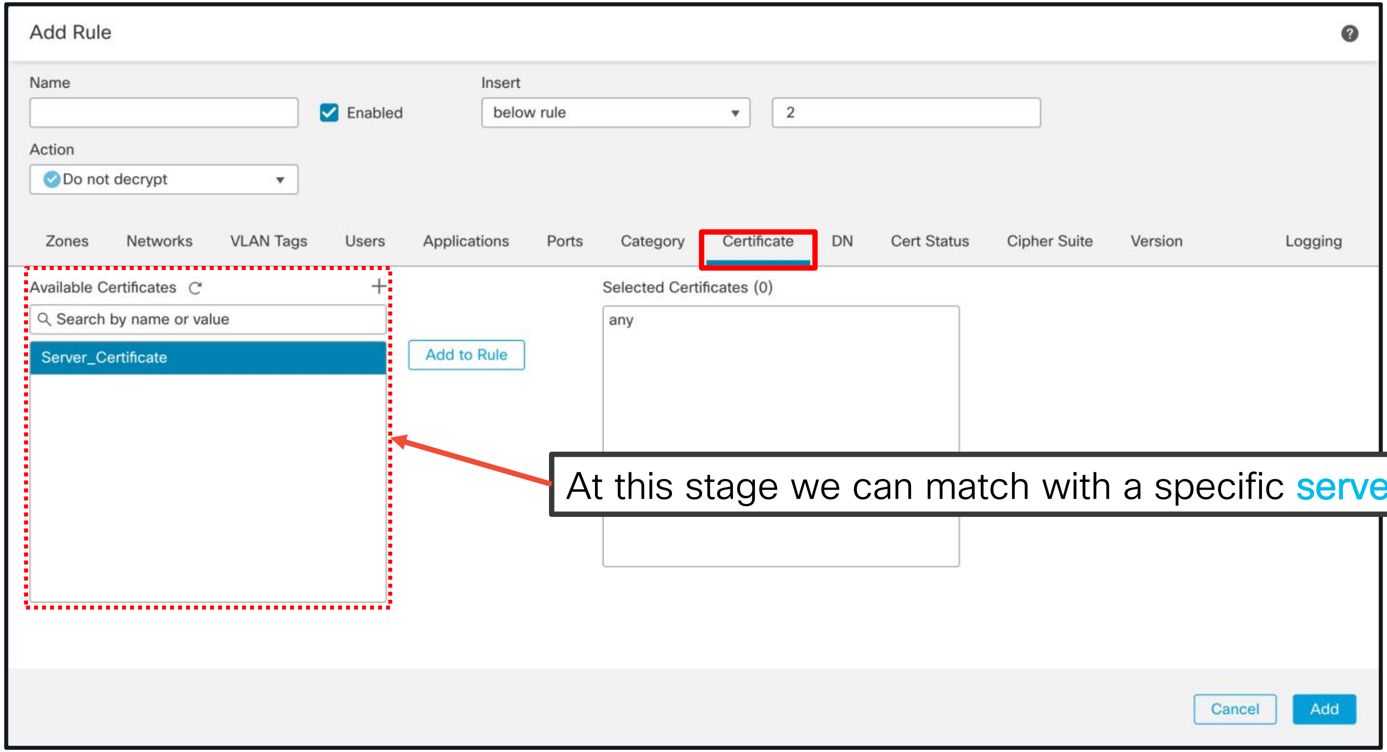
Once Server Hello is received the firewall can match on TLS/SSL versions...



...as well as on Cihper Suites that the server selected.

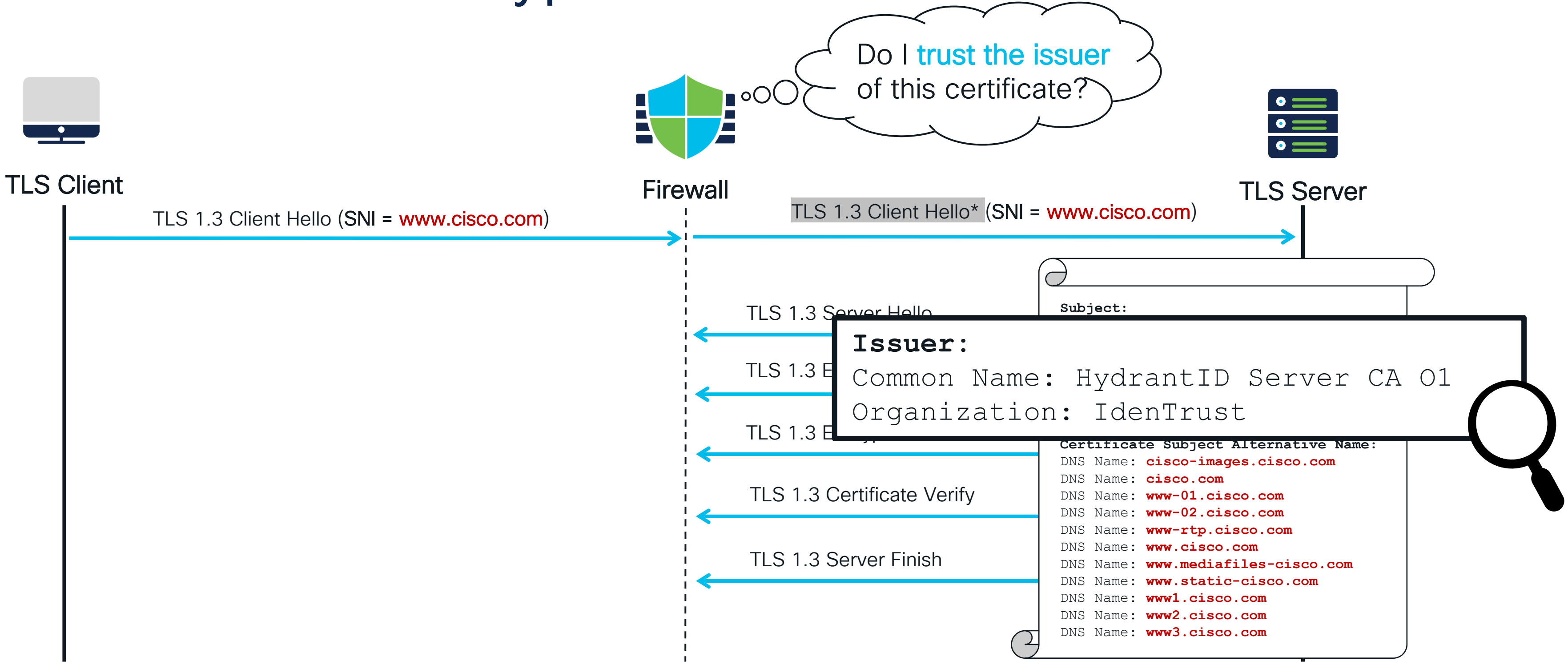


Certificate Conditions



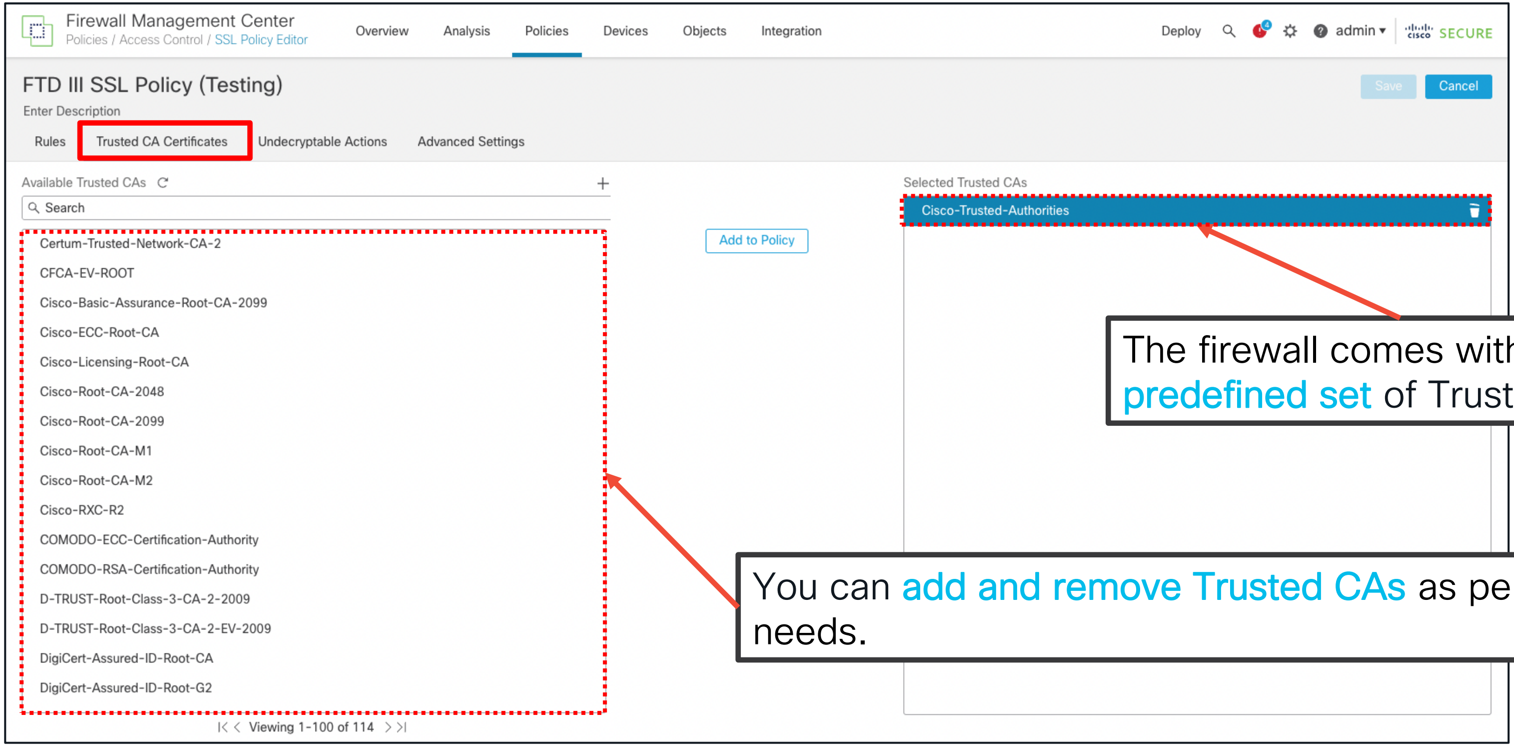
At this stage we can match with a specific server Certificate.

TLS Session Decryption Flow – Certificate Check



* - modified message

Trusted CA Certificates

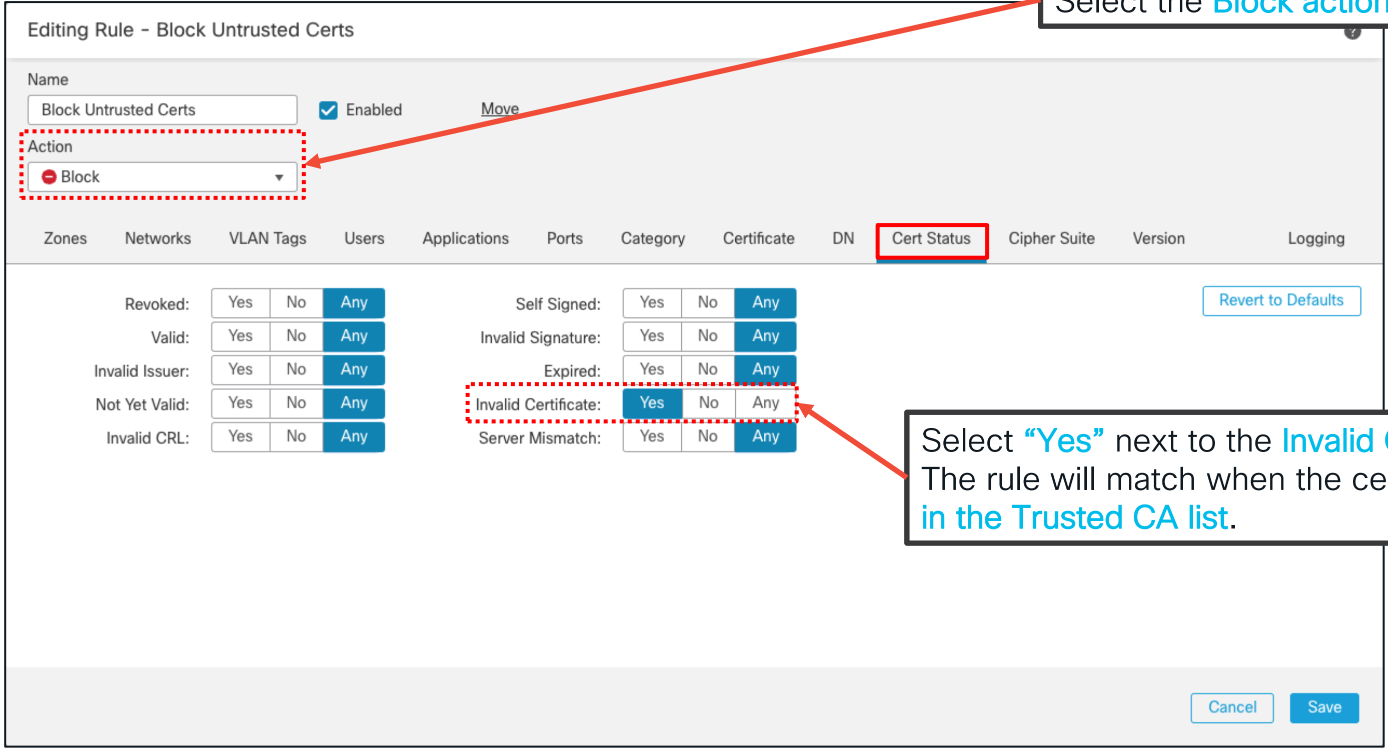


The firewall comes with a **predefined set** of Trusted CAs.

You can **add and remove Trusted CAs** as per your needs.

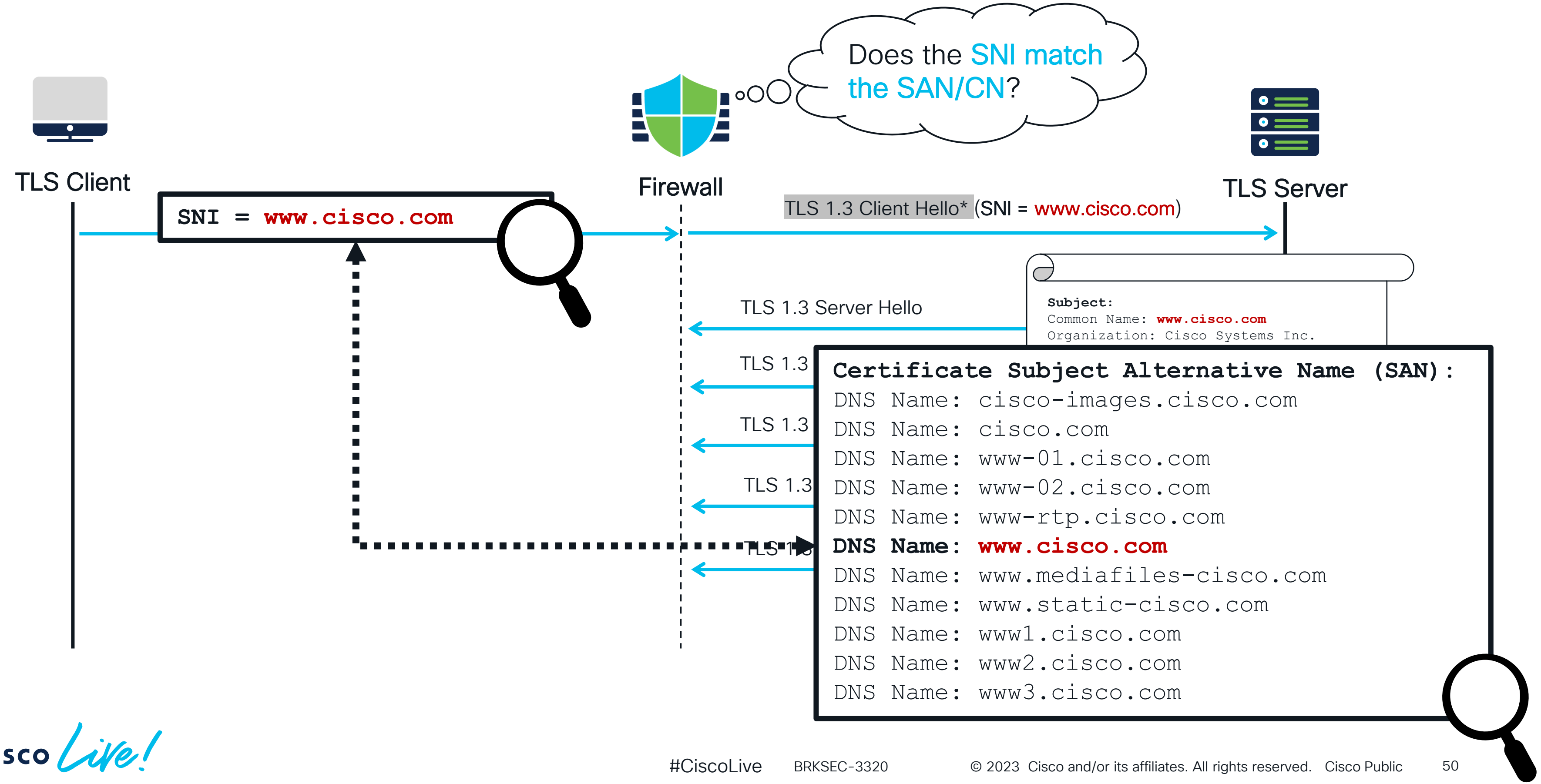
Block Untrusted Certificates

Select the Block action in your SSL Policy Rule.

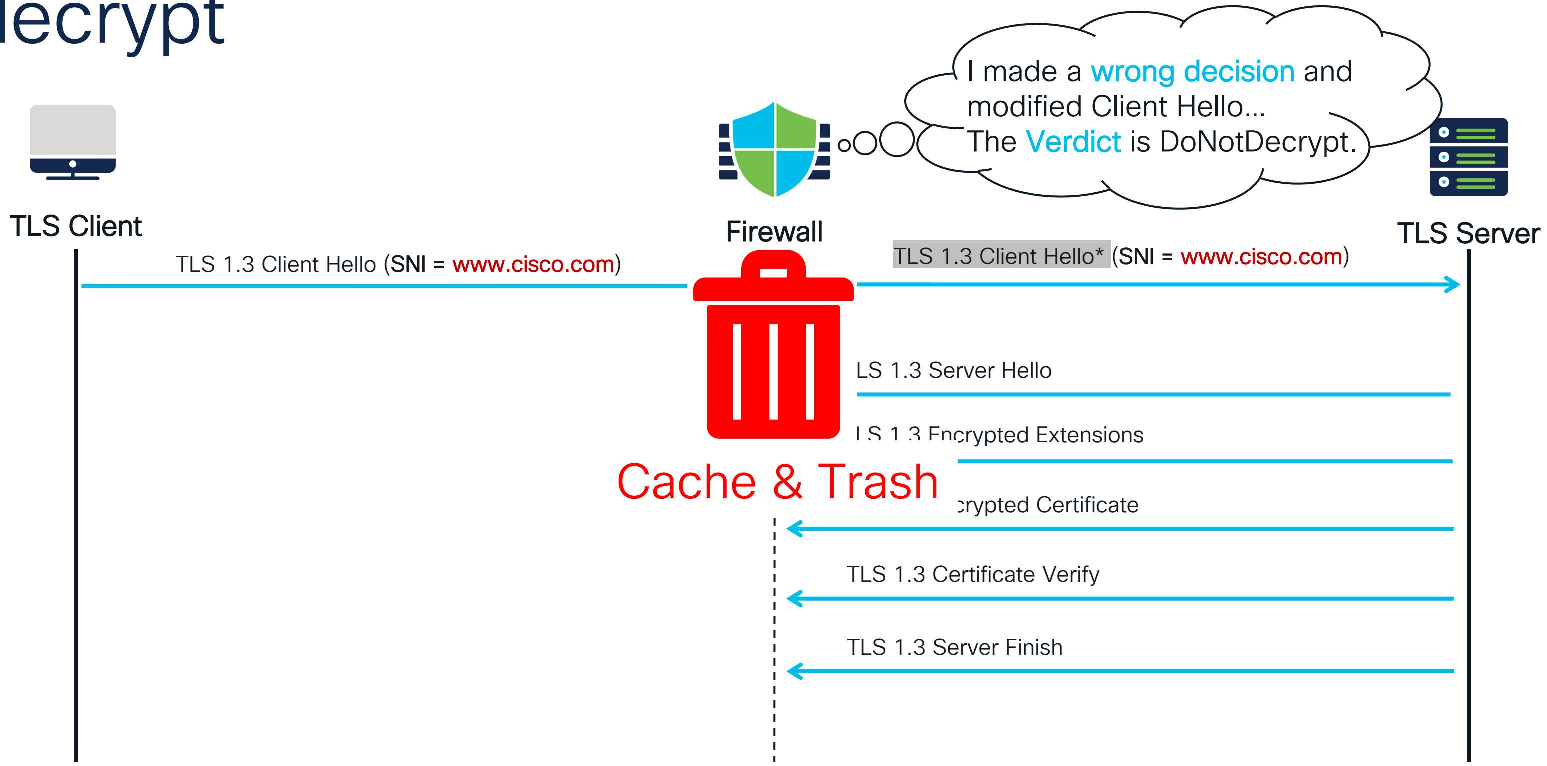


Select "Yes" next to the Invalid Certificate condition. The rule will match when the certificate authority is not in the Trusted CA list.

TLS Session Decryption Flow – CN/SAN and SNI Crosscheck



TLS Session Decryption Flow – Late do not decrypt

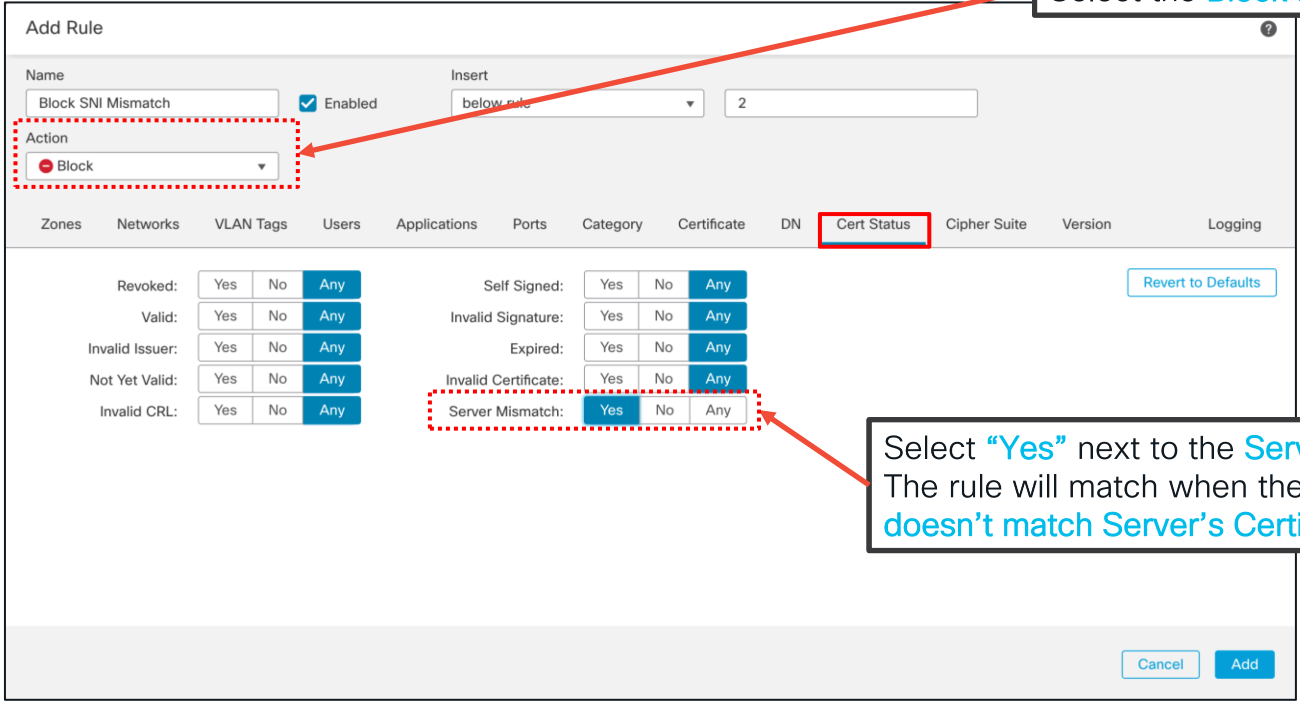


* – modified message

REFERENCE

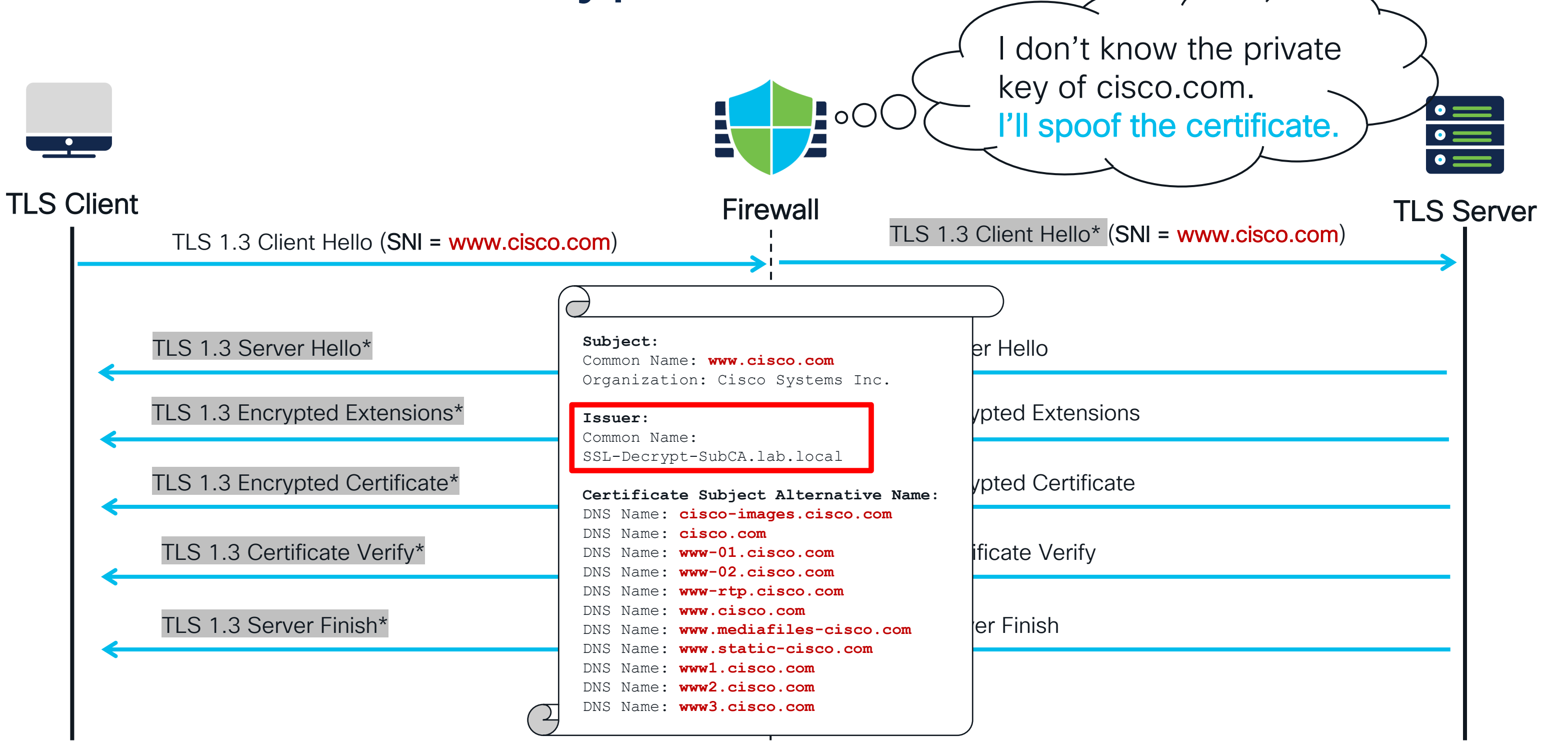
Block Server SNI Mismatch

Select the Block action in your SSL Policy Rule.



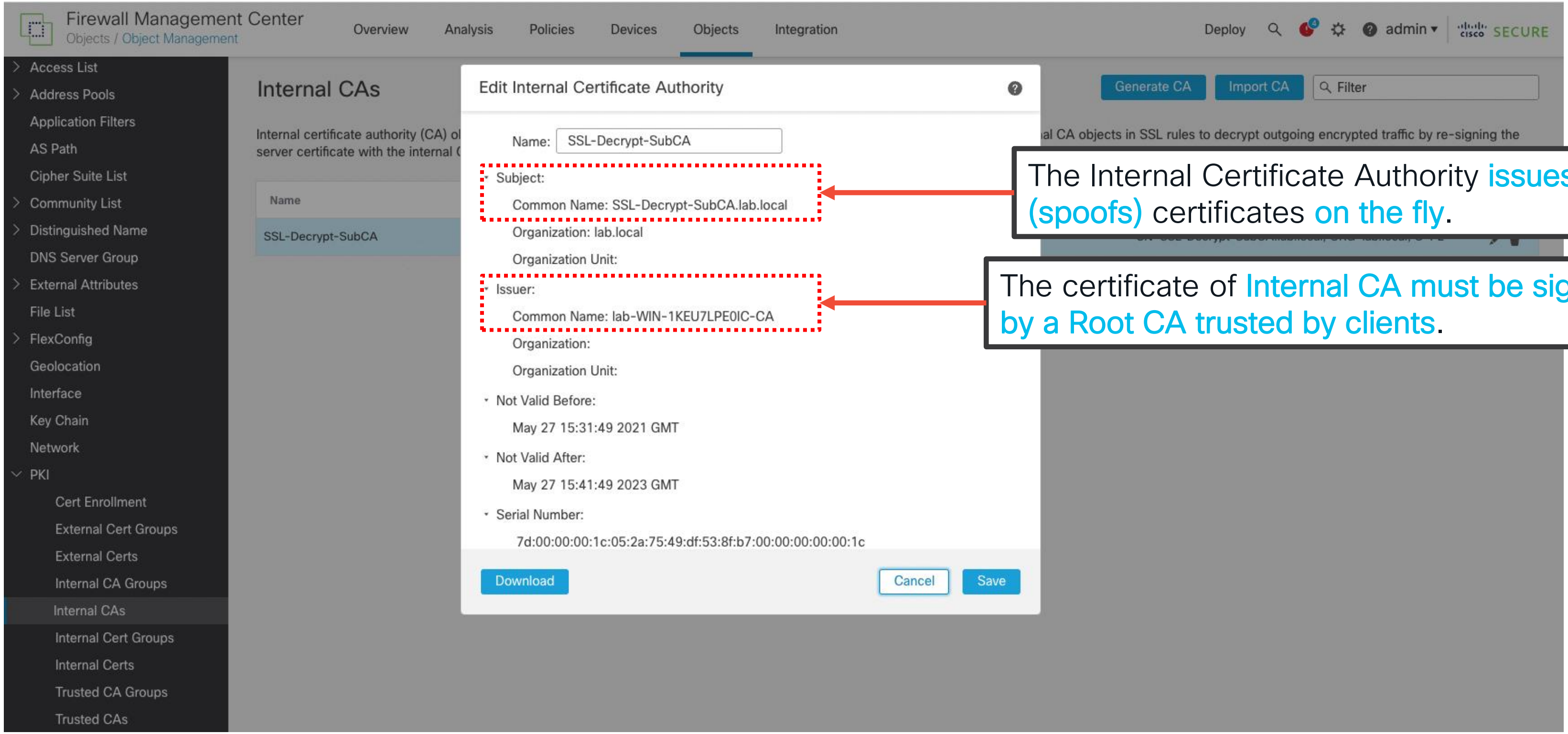
Select "Yes" next to the Server Mismatch condition. The rule will match when the SNI in the Client Hello doesn't match Server's Certificate.

TLS Session Decryption Flow – Certificate Resign



* - modified message

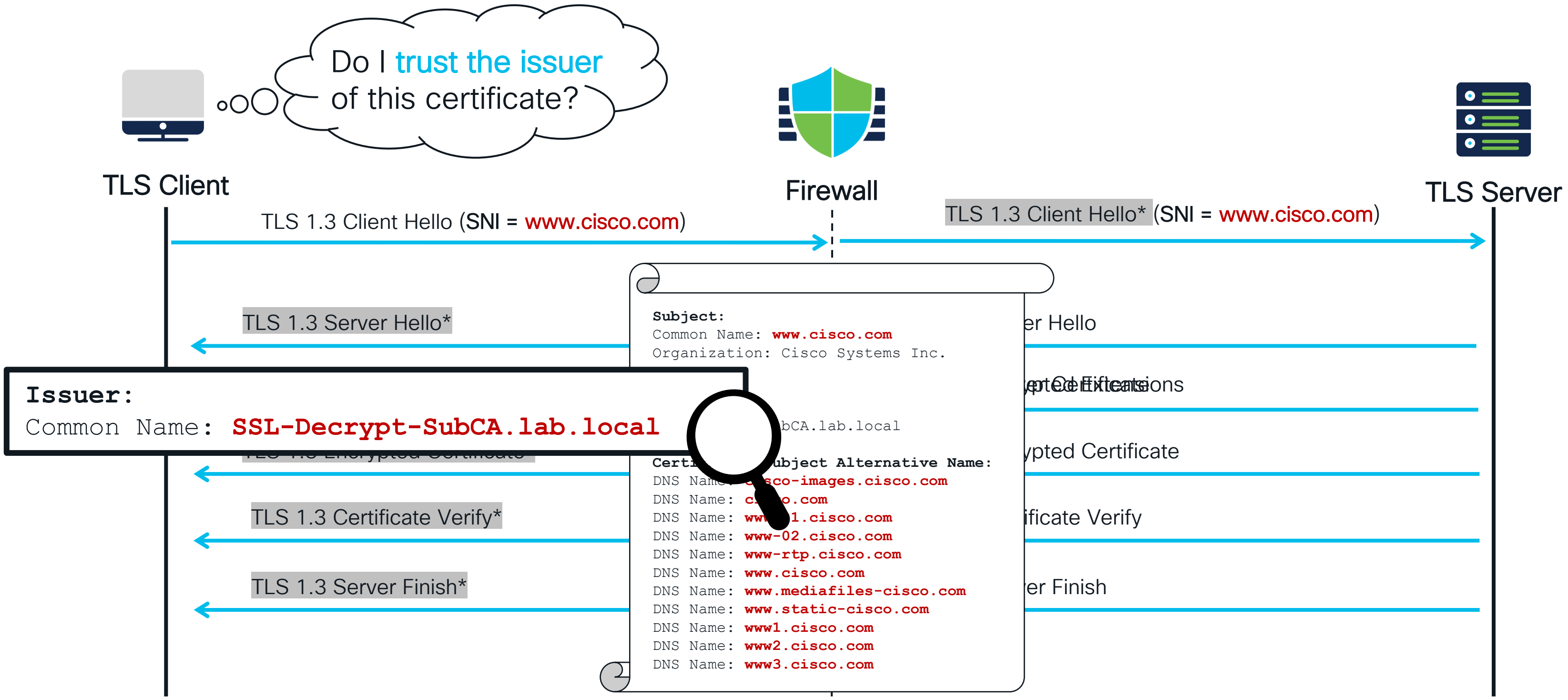
Internal CA screenshot



The Internal Certificate Authority issues (spoofs) certificates on the fly.

The certificate of Internal CA must be signed by a Root CA trusted by clients.

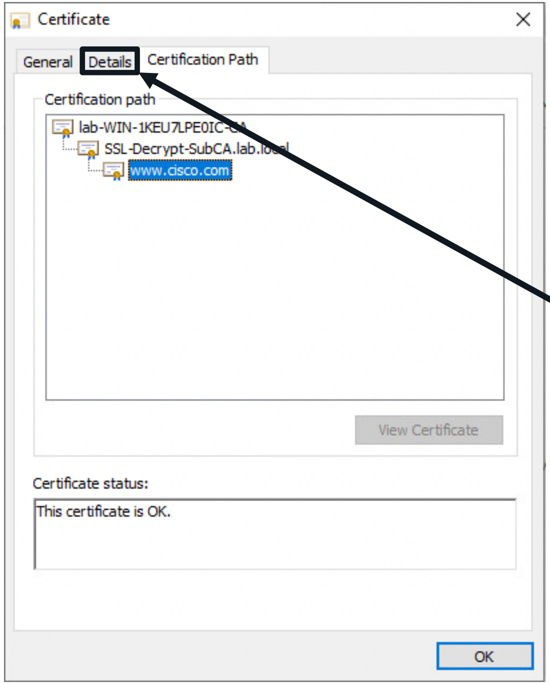
TLS Session Decryption Flow – Client Check



* – modified message

REFERENCE

Screenshot of resigned certificate



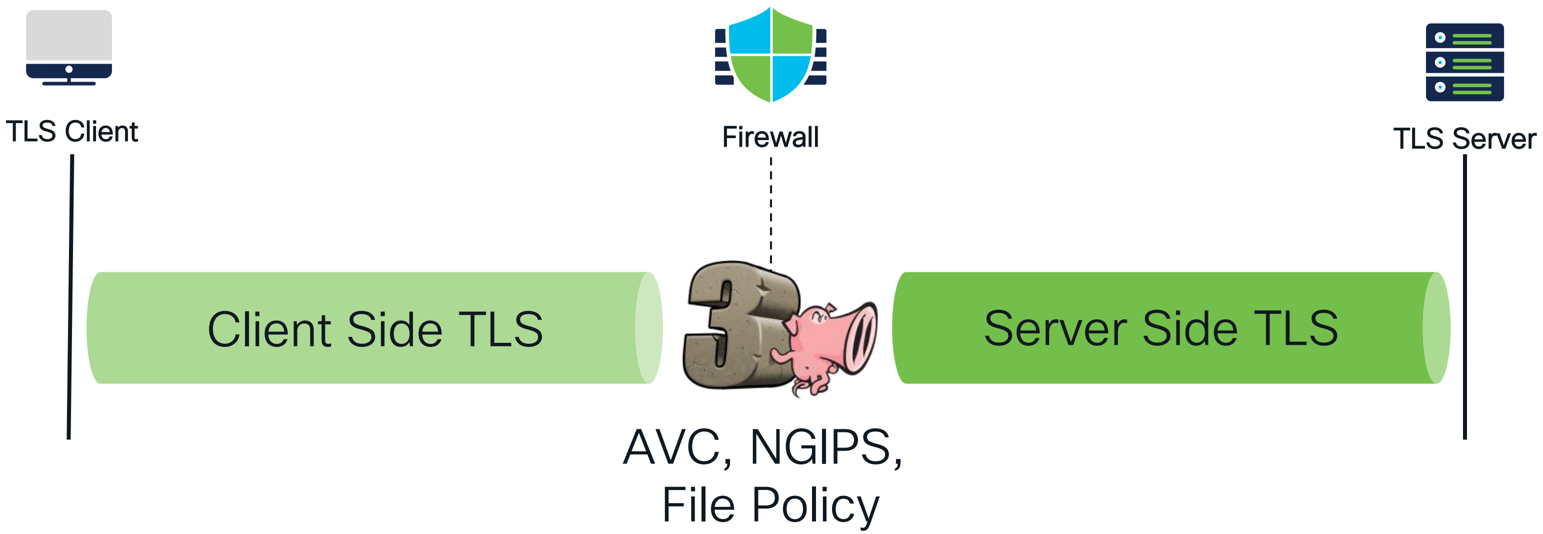
Resigned Certificate

Field	Value
Version	V3
Serial number	17bd3e63bfda13212edf9300
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	SSL-Decrypt-SubCA.lab.local, l...
Valid from	Wednesday, February 16, 20...
Valid to	Thursday, February 16, 2023 ...
Subject	US, California, San Jose, Cisco...
Public key	RSA (2048 Bits)
Public key parameters	05 00
Subject Alternative Name	DNS Name=cisco-images.cisco...
Subject Key Identifier	b18ceccd49a5dfd743e0a60f7...
Enhanced Key Usage	Server Authentication (1.3.6....
SCT List	v1, adf7befa7cff10c88b9d3d9...
Key Usage	Digital Signature, Key Encipher...
Thumbprint	86f51f44a3c93ff68c4b0af1da...

Original Certificate

Field	Value
Version	V3
Serial number	40017f044e5f9214333d982de...
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	HydrantID Server CA O1, Hyd...
Valid from	Wednesday, February 16, 20...
Valid to	Thursday, February 16, 2023 ...
Subject	US, California, San Jose, Cisco...
Public key	RSA (2048 Bits)
Public key parameters	05 00
~~Authority Information Access~~	~~[1]Authority Info Access: Acc...~~
~~Authority Key Identifier~~	~~KeyID=89b89bb69eedfbb0c6...~~
~~Certificate Policies~~	~~[1]Certificate Policy:Policy Ide...~~
~~CRL Distribution Points~~	~~[1]CRL Distribution Point: Distr...~~
Subject Alternative Name	DNS Name=cisco-images.cisco...
Subject Key Identifier	b18ceccd49a5dfd743e0a60f7...
Enhanced Key Usage	Server Authentication (1.3.6....
SCT List	v1, adf7befa7cff10c88b9d3d9...
Key Usage	Digital Signature, Key Encipher...
Thumbprint	0dddb6ce30b00bd75adb198b...

TLS Session Decryption Flow – PIG-in-the-Middle



TLS Decryption Challenges

It Is Not an Easy World for a Man (-in-the-Middle)

TLS 1.3

Certificate Pinning

DNS over HTTPs

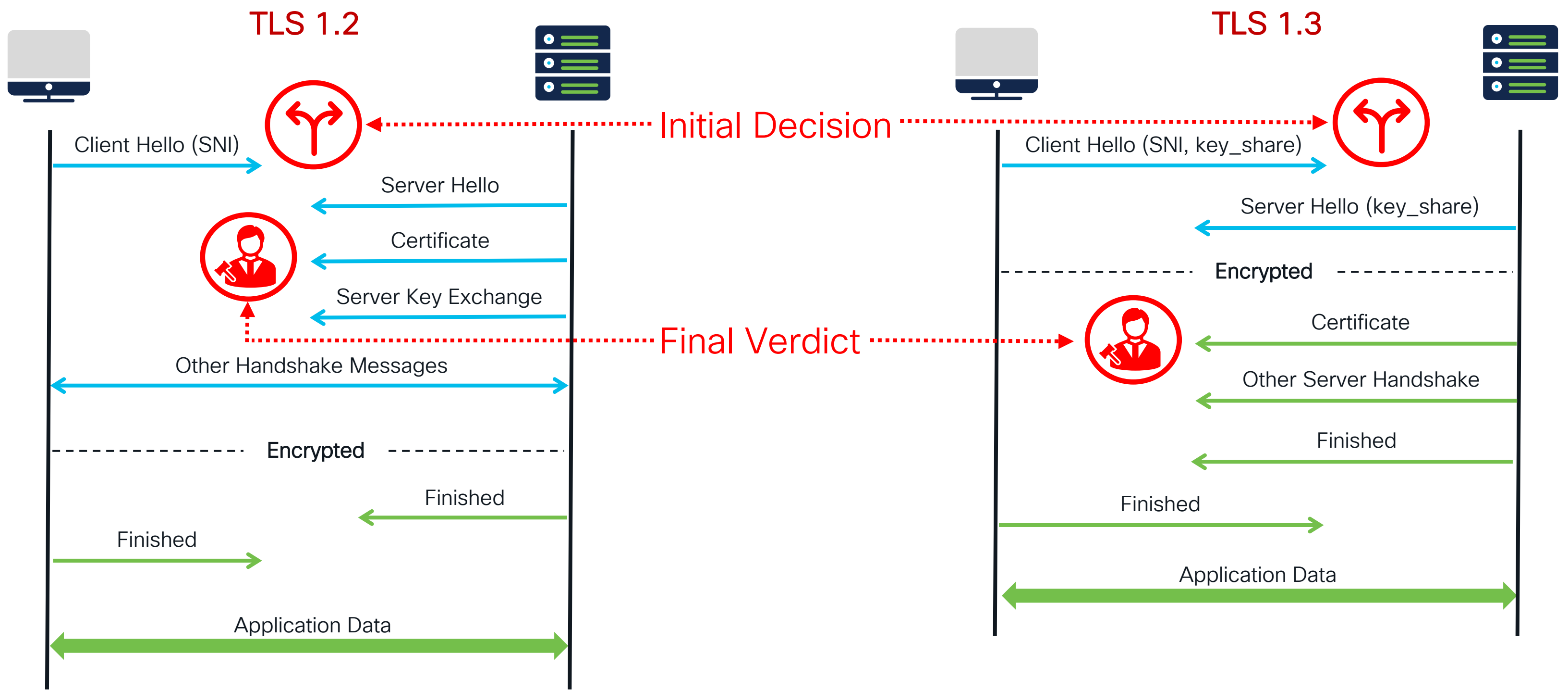
Encrypted SNI

Encrypted Client Hello

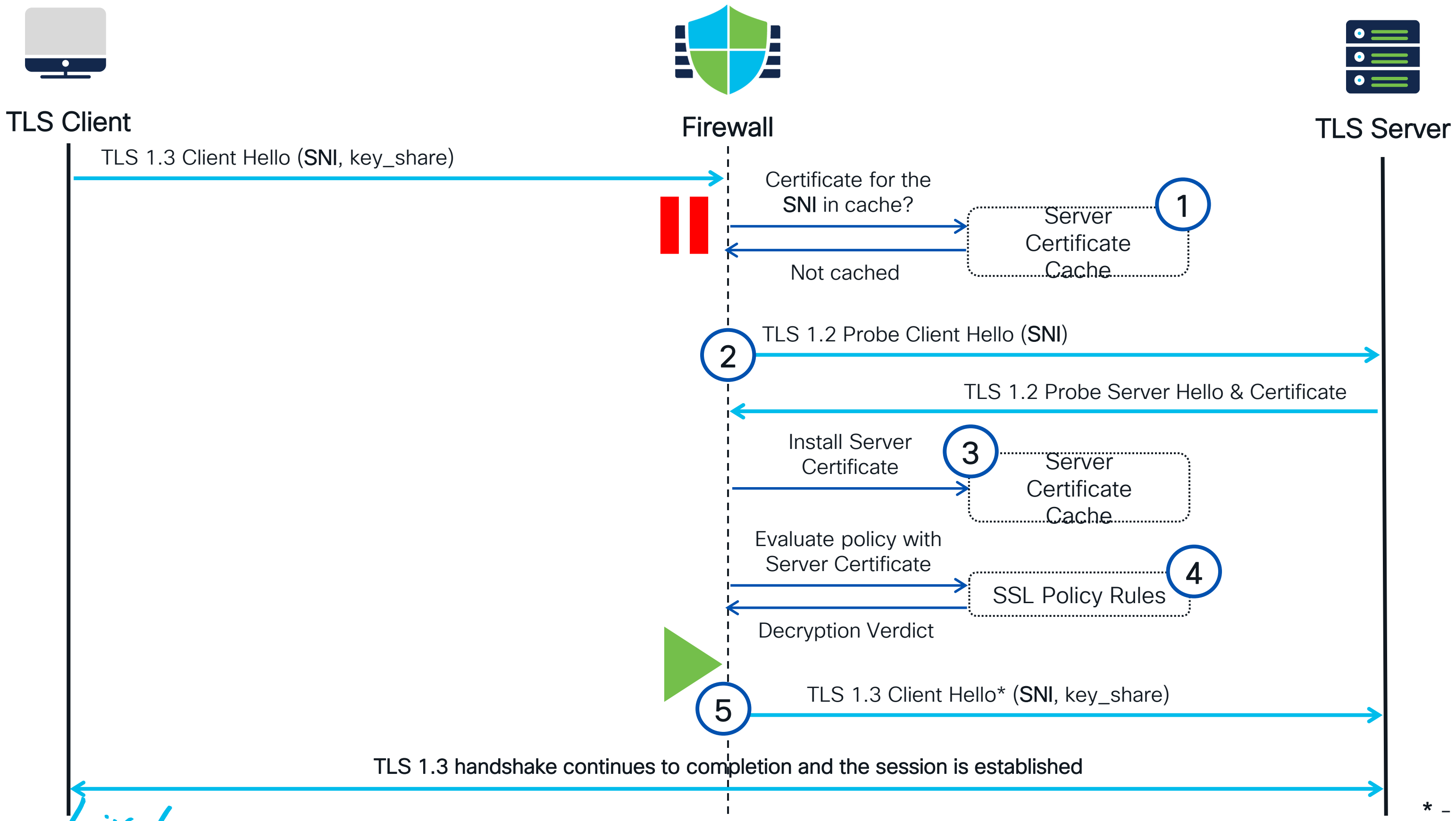
0-RTT

QUIC

TLS 1.2 vs 1.3 Handshake

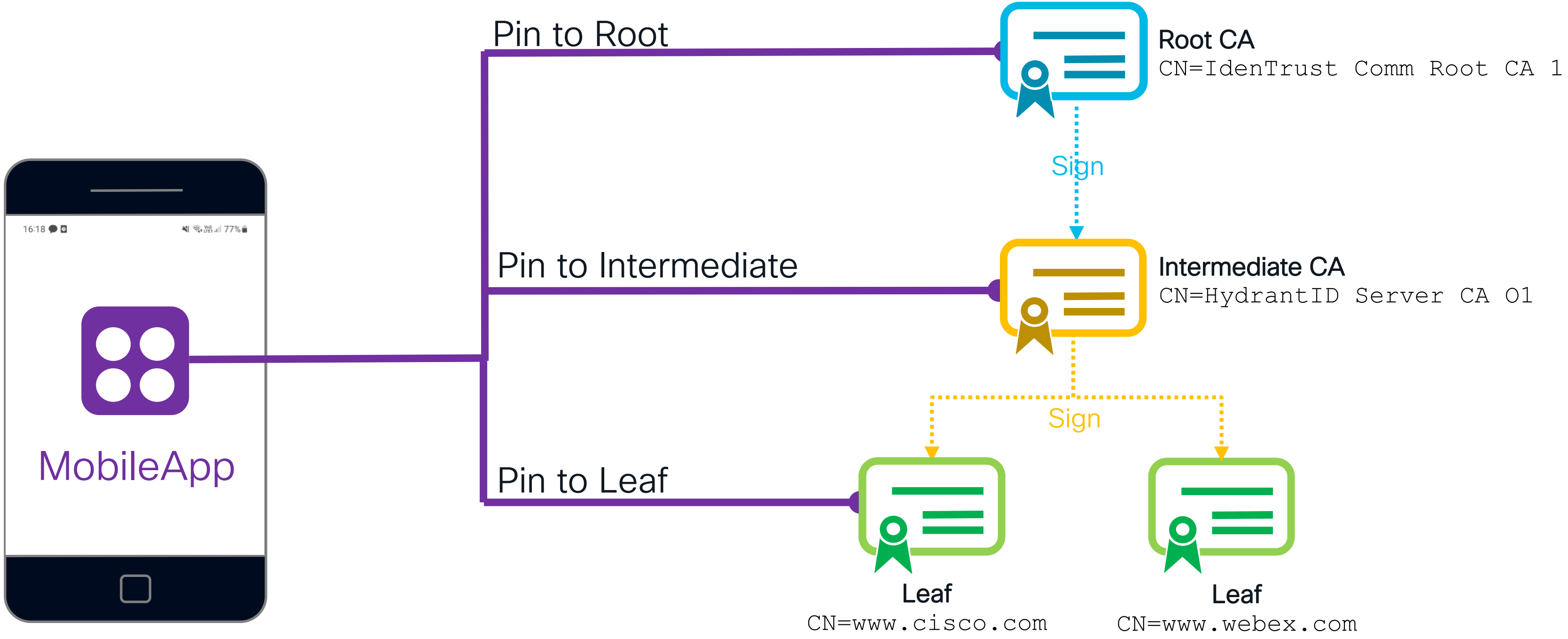


TLS Server Cache & Probing

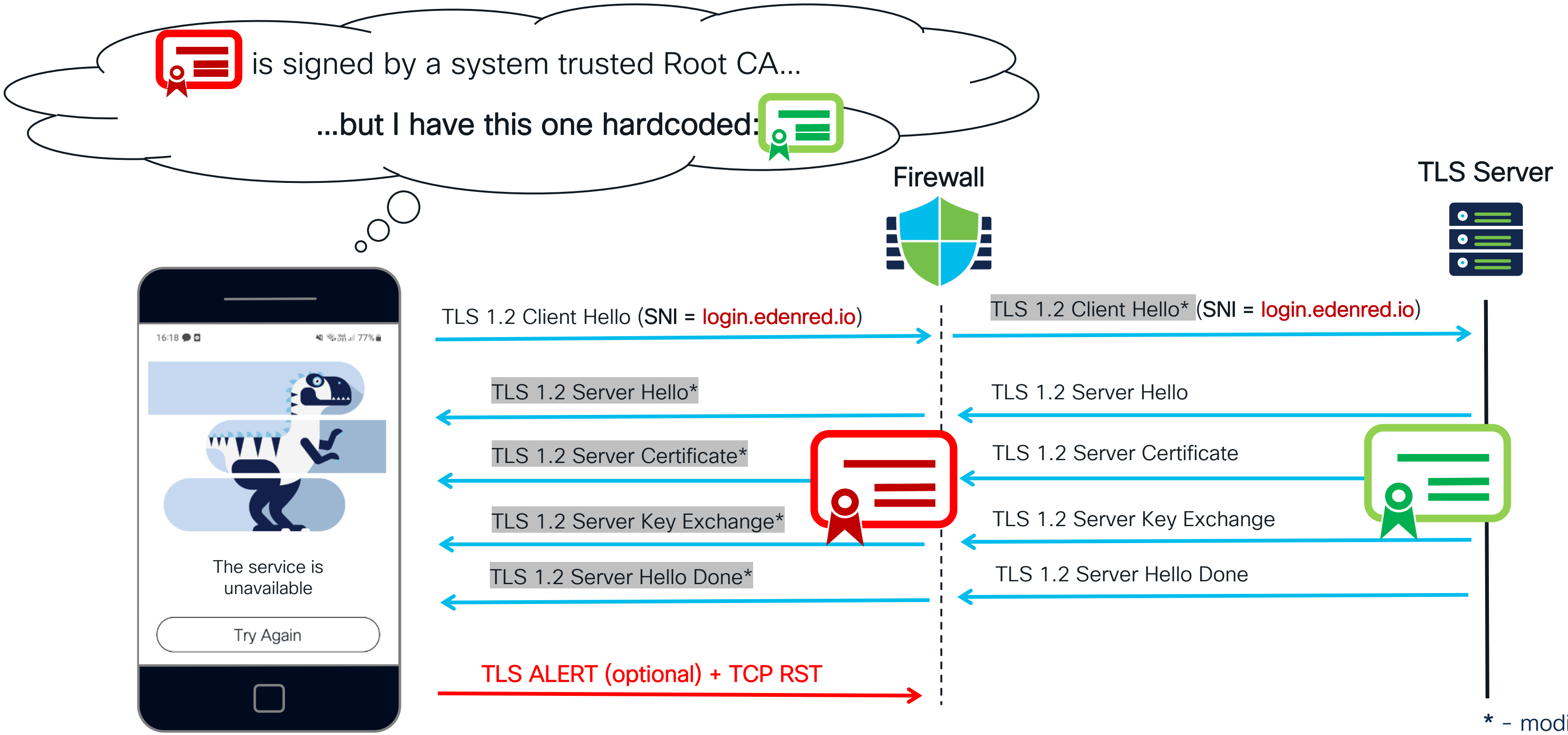


* - modified message

What is Certificate Pinning?



Why is Certificate Pinning a Problem?



* - modified message

Troubleshooting Certificate Pinning - Capture

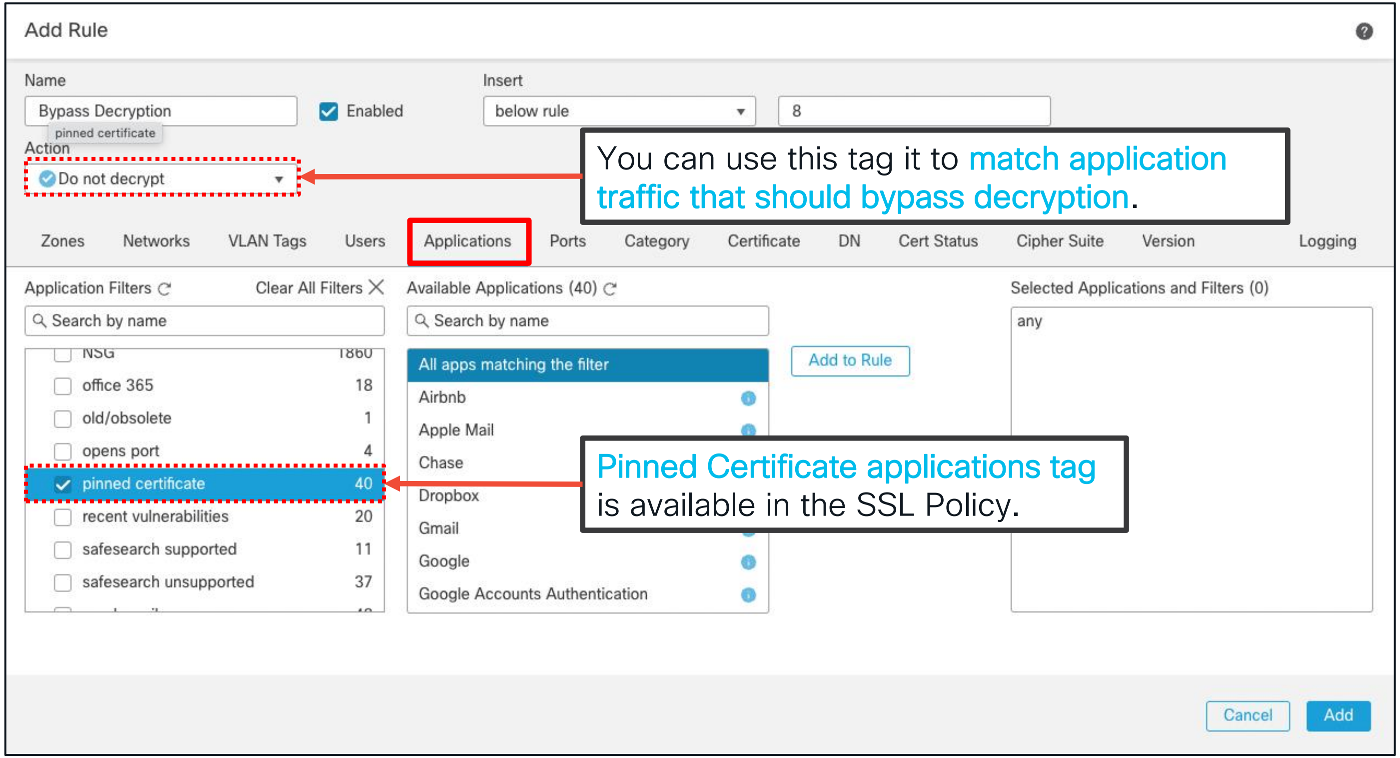
tcp.flags.reset == 1 || tls

No.	Time	Source	Info	SPORT	DPORT	Length	Protocol
4	0.000030	192.1	Client Hello	46616	443	583	TLSv1.2
6	0.028746	3.12	Server Hello	443	46616	164	TLSv1.2
11	0.000016	3.12	Certificate	443	46616	93	TLSv1.2
18	0.001342	3.12	Server Key Exchange	443	46616	404	TLSv1.2
19	0.000016	3.12	Server Hello Done	443	46616	75	TLSv1.2
21	0.002365	192.1	Alert (Level: Fatal, Description: Certificate Unknown)	46616	443	73	TLSv1.2
23	0.000320	192.1	46616 → 443 [RST, ACK] Seq=525 Ack=3464 Win=79872 Len=0…	46616	443	66	TCP
24	0.005142	192.1	46616 → 443 [RST] Seq=518 Win=0 Len=0	46616	443	54	TCP
25	0.000016	192.1	46616 → 443 [RST] Seq=525 Win=0 Len=0	46616	443	54	TCP

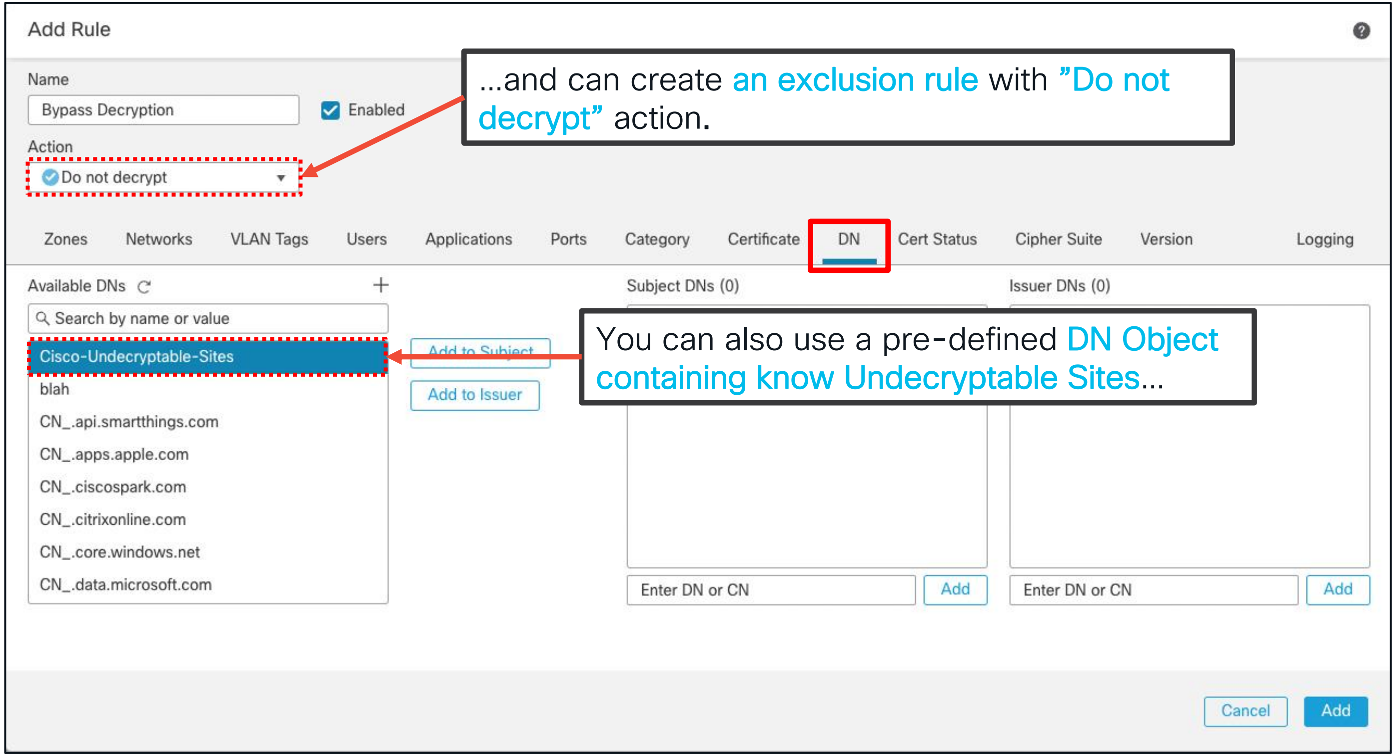

```
> Frame 21: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)
> Ethernet II, Src: 8e:9f:aa:0e:8b:39 (8e:9f:aa:0e:8b:39), Dst: Cisco_f5:28:c9 (08:4f:a9:f5:28:c9)
> Internet Protocol Version 4, Src: 192.168.10.37, Dst: 3.120.204.252
> Transmission Control Protocol, Src Port: 46616, Dst Port: 443, Seq: 518, Ack: 3464, Len: 7
v Transport Layer Security
  v TLSv1.2 Record Layer: Alert (Level: Fatal, Description: Certificate Unknown)
      Content Type: Alert (21)
      Version: TLS 1.2 (0x0303)
      Length: 2
    v Alert Message
        Level: Fatal (2)
        Description: Certificate Unknown (46)
```

The Alert Message indicates the spoofed re-signed certificate was not recognized by the application.

Pinned Applications Tag



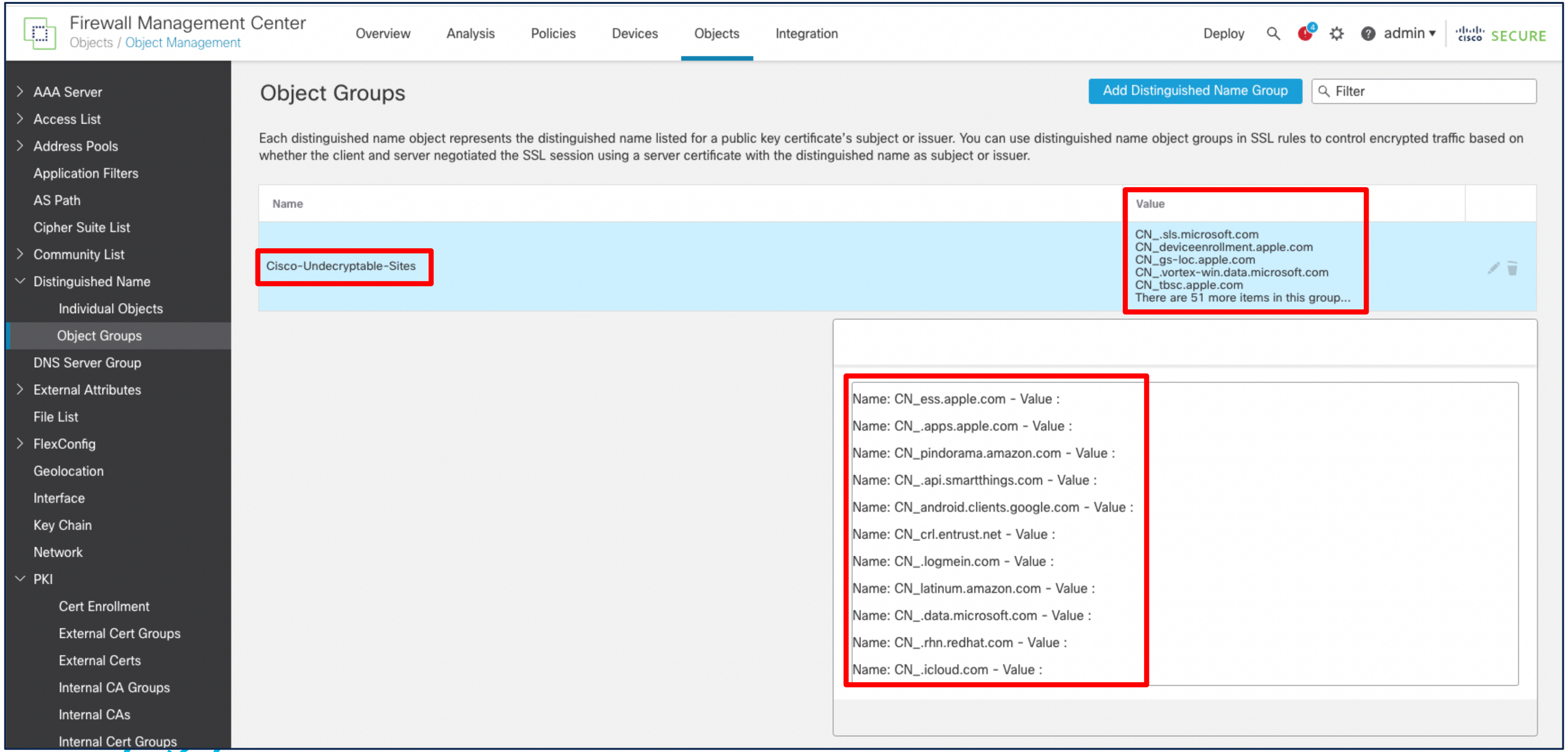
Cisco Provided Undecryptable Sites (1/2)



...and can create an exclusion rule with ”Do not decrypt” action.

You can also use a pre-defined DN Object containing know Undecryptable Sites...

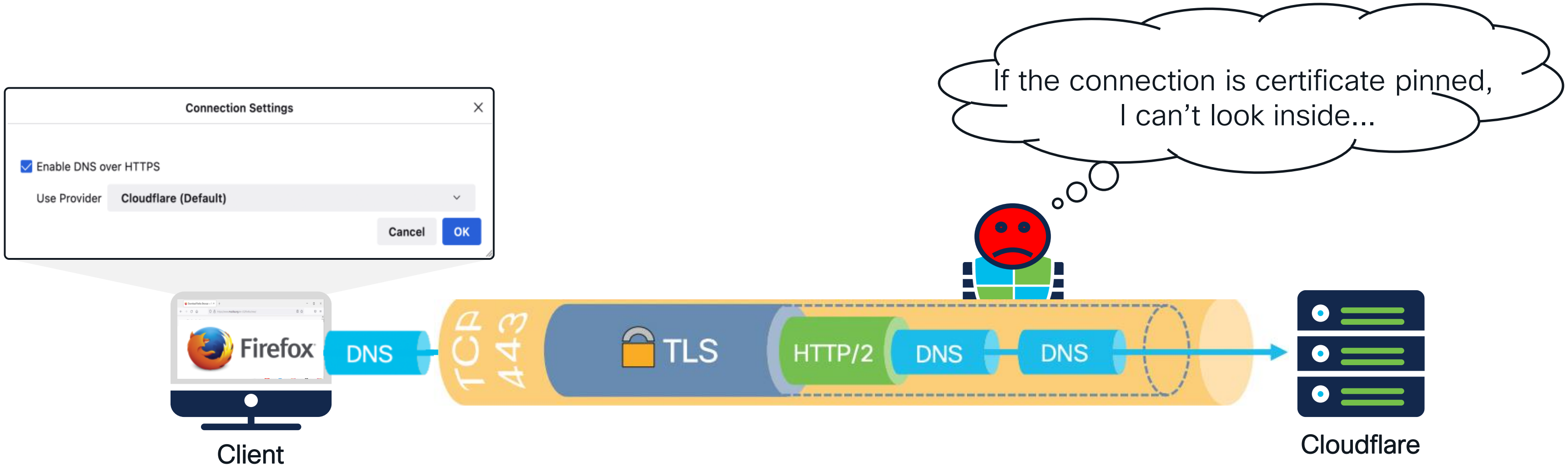
Cisco Provided Undecryptable Sites (2/2)



DNS over HTTPs

IETF standard (RFC8484) proposed to:

- allow web applications to acces DNS information via browser API
- prevent on-path devices from interfering with DNS



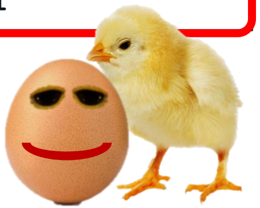
DNS over HTTPs Challenges

- The web browser hijacks OS DNS
- Bypass DNS based security controls and logging
- Delegates DNS control to a content provider (e.g. Cloudflare)
- Difficult to block by firewalls (SNI and/or IP based only)

Cloudflare
DoH Resolver

```
DNS Records:
A, AAAA – IPv4/v6 Addresses
SVCB/HTTPS RR - Encrypted Client Hello
TXT _esni - Encrypted SNI
```

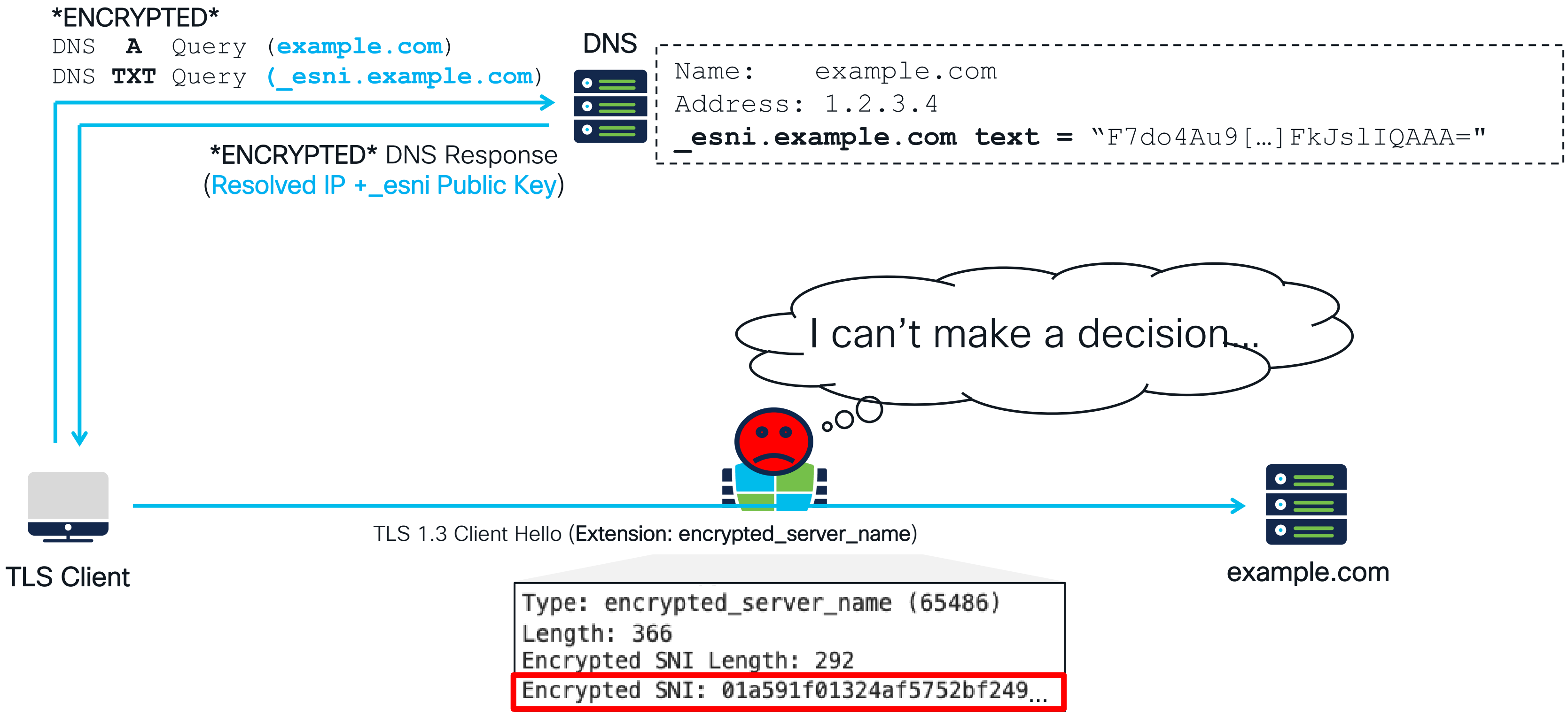
DoH is a very effective distribution method of keying material for SNI obfuscation techniques like **Encrypted SNI** or **Encrypted Client Hello**.



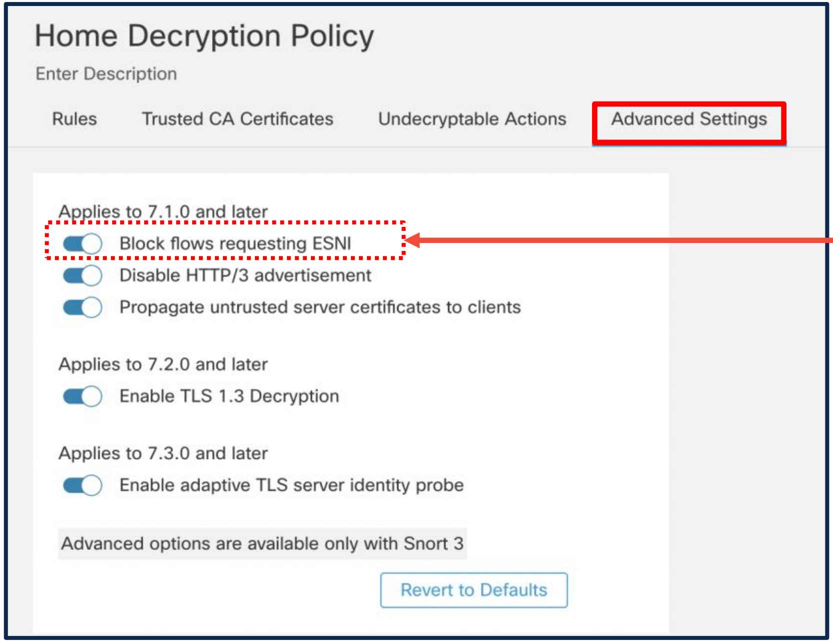
Encrypted SNI (ESNI) – a ”Dodo” Protocol

- An experimental feature available in Firefox up to release 84.0
- Cloudflare used to provide an ESNI test page
- Never reached an RFC Proposed Standard
- Evolved into Encrypted Client Hello (ECH)

Encrypted SNI (ESNI)



Blocking ESNI Requests

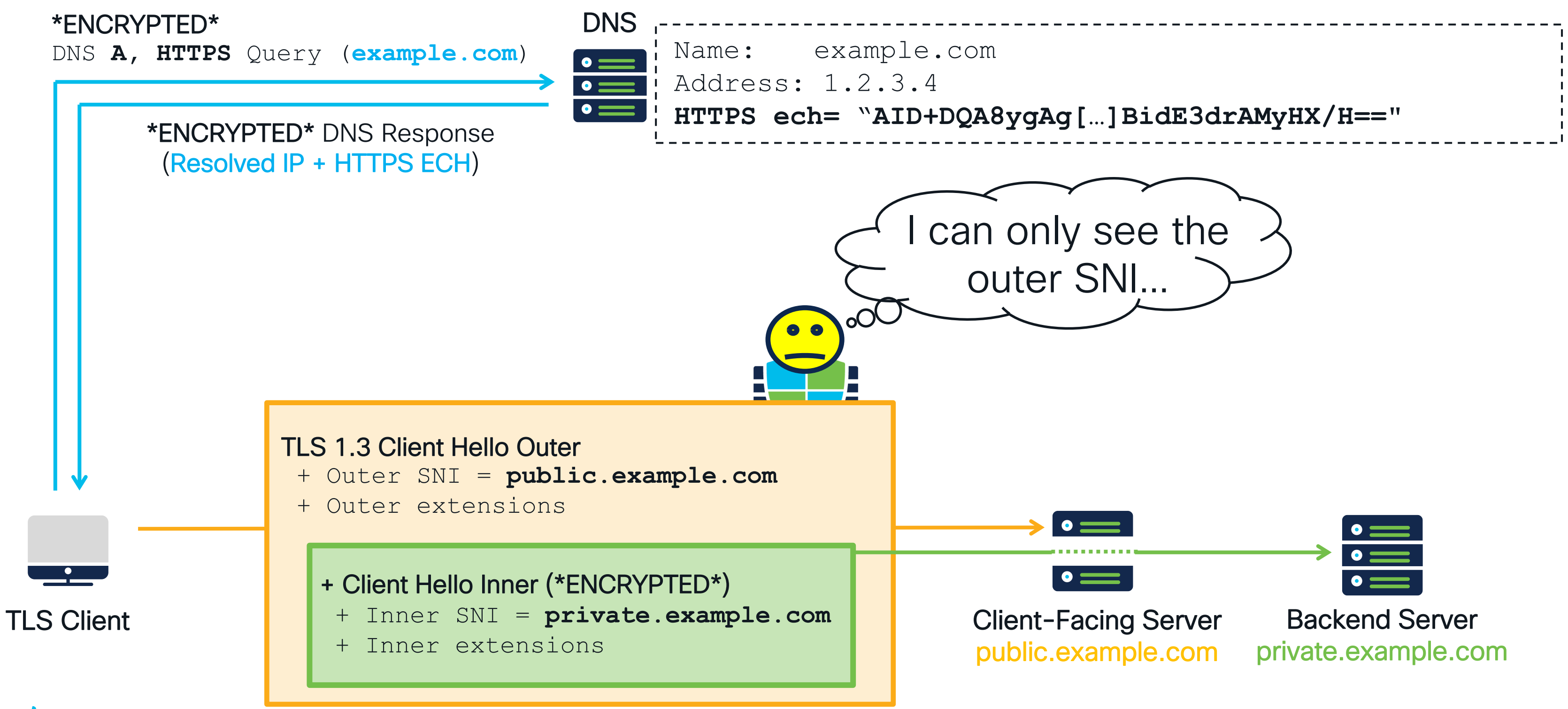


Select this option to block connections with Client Hello containing ENSI extension.

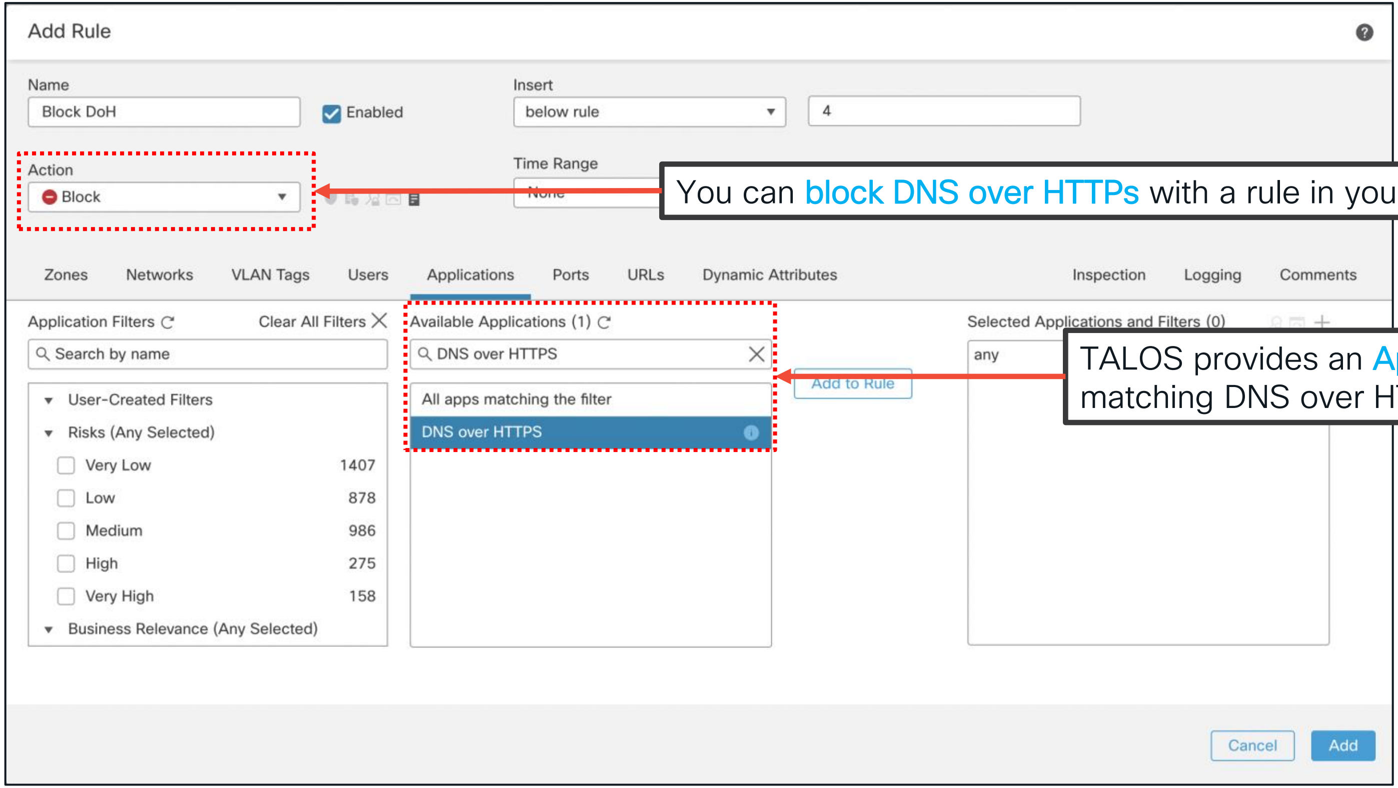
Encrypted Client Hello

- ECH is an IETF Draft version 16 considered fairly stable
- Minor footprint currently – less than 0.2% of flows in Cisco EVE's dataset
- Wider adoption of ECH will make TLS decryption process even more involved
- Today, Cisco Secure Firewall ignores ECH

Encrypted Client Hello (ECH)



DoH Blocking on FirePower



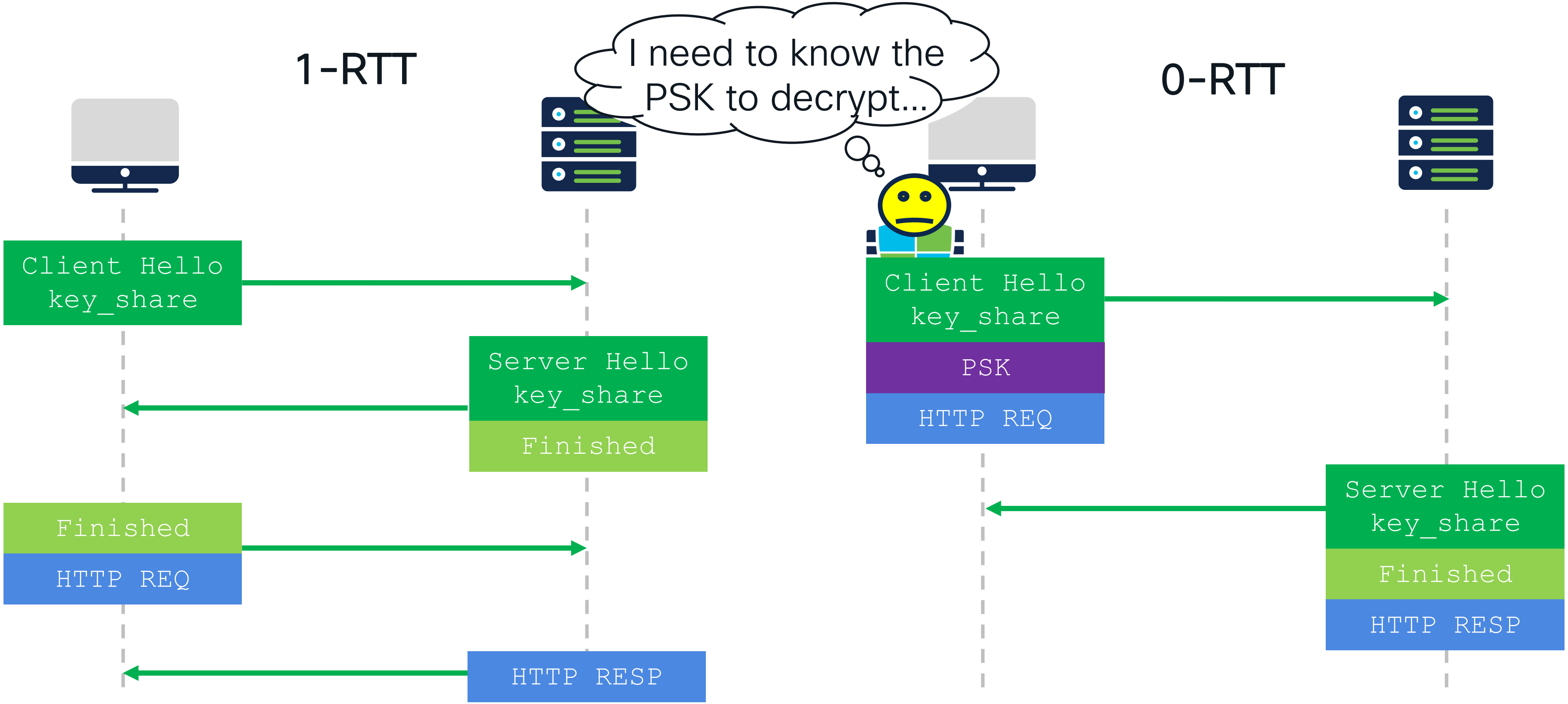
You can block DNS over HTTPs with a rule in your Access Control Policy.

TALOS provides an Application Detector matching DNS over HTTPs traffic.

0-RTT

- 0-RTT is a technique baked into TLS 1.3/QUIC that accelerates application response time
- It requires the client to possess a pre-shared key prior to the connection
- Cisco Secure Firewall strips 0-RTT flag from Client Hello (if decrypting the flow)

0-RTT Saves One Round Trip on Reconnect (TLS 1.3)



Now let’s do a QUIC change of speakers...

Your QUIC Speaker

George Koikara
Principal Engineer

- Based in Bangalore, India
- With Cisco since June 2019
- Architect for QUIC and ZTNA for Secure Firewall
- Hold multiple patents in domain of security

A brief history

- IETF proposed standard (RFCs 9000-9003) – started at Google in 2014
- A new secure transport protocol with baked-in encryption
- Used by ~ 14% of the websites and over 35% of Google’s traffic
- Over ~70% of Facebook traffic is on QUIC
- Provides significant improvements for application and content providers:
  - Youtube Video **rebuffers reduced by up to 18%**
  - Google Search **latency reduction up to 8%**

Pre Poll

slido

Join at

slido.com

#1546 305

Passcode: **clusquic**

Why QUIC ?

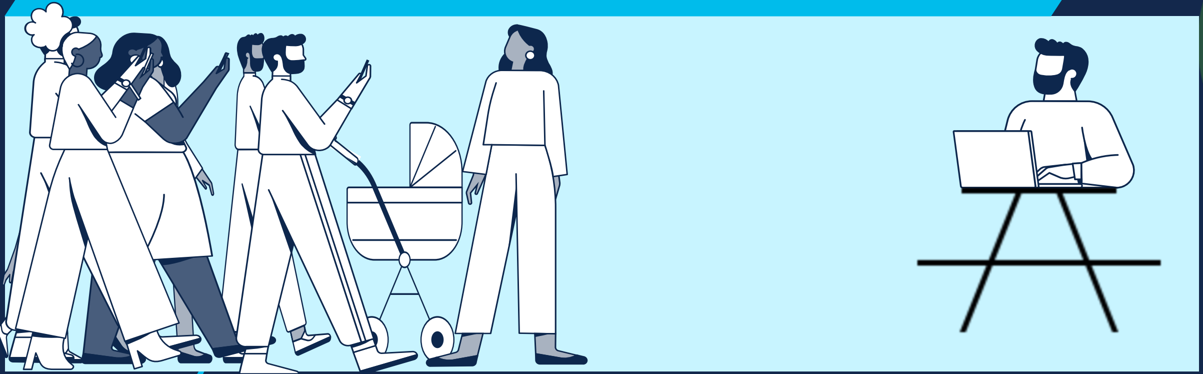
“Change is the essential process of all existence.”

- Mr. Spock, "Let That Be Your Last Battlefield"

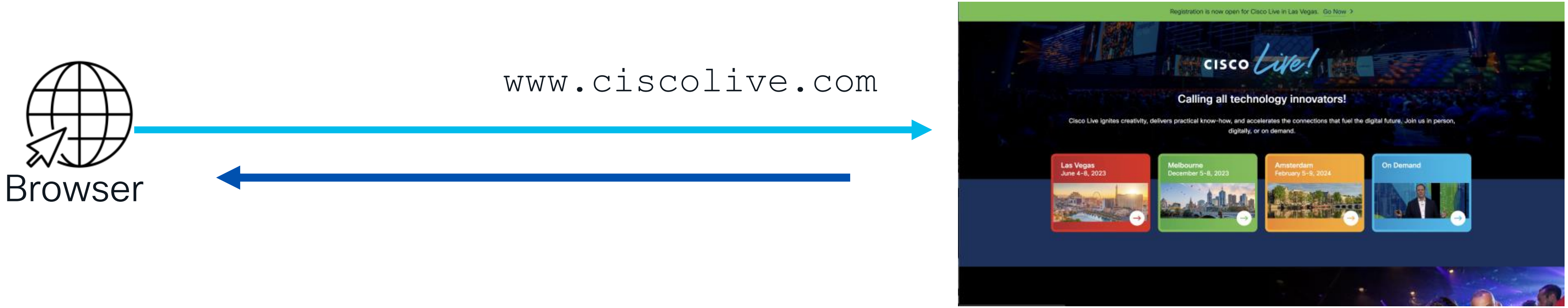
Head of Line Blocking..

NOW PLAYING

PROGNOSIS NEGATIVE



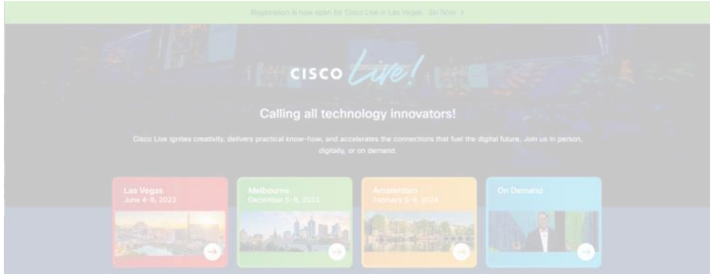
Head of Line Blocking HTTP/1.1



Head of Line Blocking HTTP/1.1

Browser

www.ciscolive.com

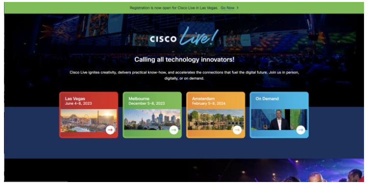


Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.ciscolive.com	/	document	html	14.51 kB	101.80 kB
301	GET	www.ciscolive.com	cdcrSwitch.js	script	js	3.54 kB	10.63 kB
200	GET	www.cisco.com	ctm-core.js	script	js	11.19 kB	35.17 kB
200	GET	www.ciscolive.com	jquery.min.js	script	js	36.32 kB	100.65 kB
200	GET	www.ciscolive.com	utils.min.js	script	js	4.16 kB	8.12 kB
200	GET	www.ciscolive.com	granite.min.js	script	js	2.43 kB	3.68 kB
200	GET	www.ciscolive.com	jquery.min.js	script	js	798 B	16 B
200	GET	www.ciscolive.com	shared.min.js	script	js	6.98 kB	19.98 kB
200	GET	www.ciscolive.com	underscore.min.js	script	js	6.57 kB	16.26 kB
200	GET	www.ciscolive.com	kernel.min.js	script	js	26.83 kB	119.74 kB
200	GET	www.ciscolive.com	modern.min.js	script	js	11.33 kB	29.05 kB

Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js

Browser



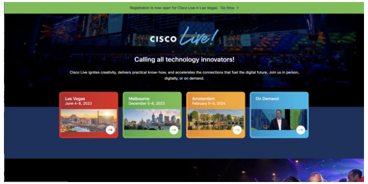
/c/cdcrSwitch.js

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```


Head of Line Blocking HTTP/1.1

Browser

GET /c/cdcrSwitch.js



/c/cdcrSwitch.js

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```

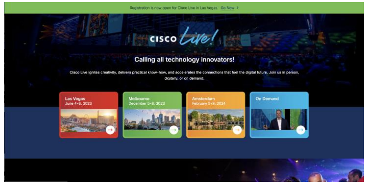
HTTP 1.1 200 OK
Content-Length : 2000

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```

Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js

Browser



/c/cdcrSwitch.js

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```

HTTP 1.1 200 OK
Content-Length : 2000

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```

TCP PKT 1

HTTP 1.1 200 OK
Content-Length : 2000

```
var revisionsFix =
document.createElement('style
');
revisionsFix.innerHTML = '.cdc-
eot-toc .docId
```

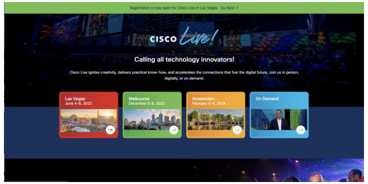
TCP PKT 2

```
.docHeaderComponent
.infobar .seeRevisions {
display:none; }';
document.head.appendChild(r
evisionsFix);
```

Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js GET /c/cisco.png

Browser



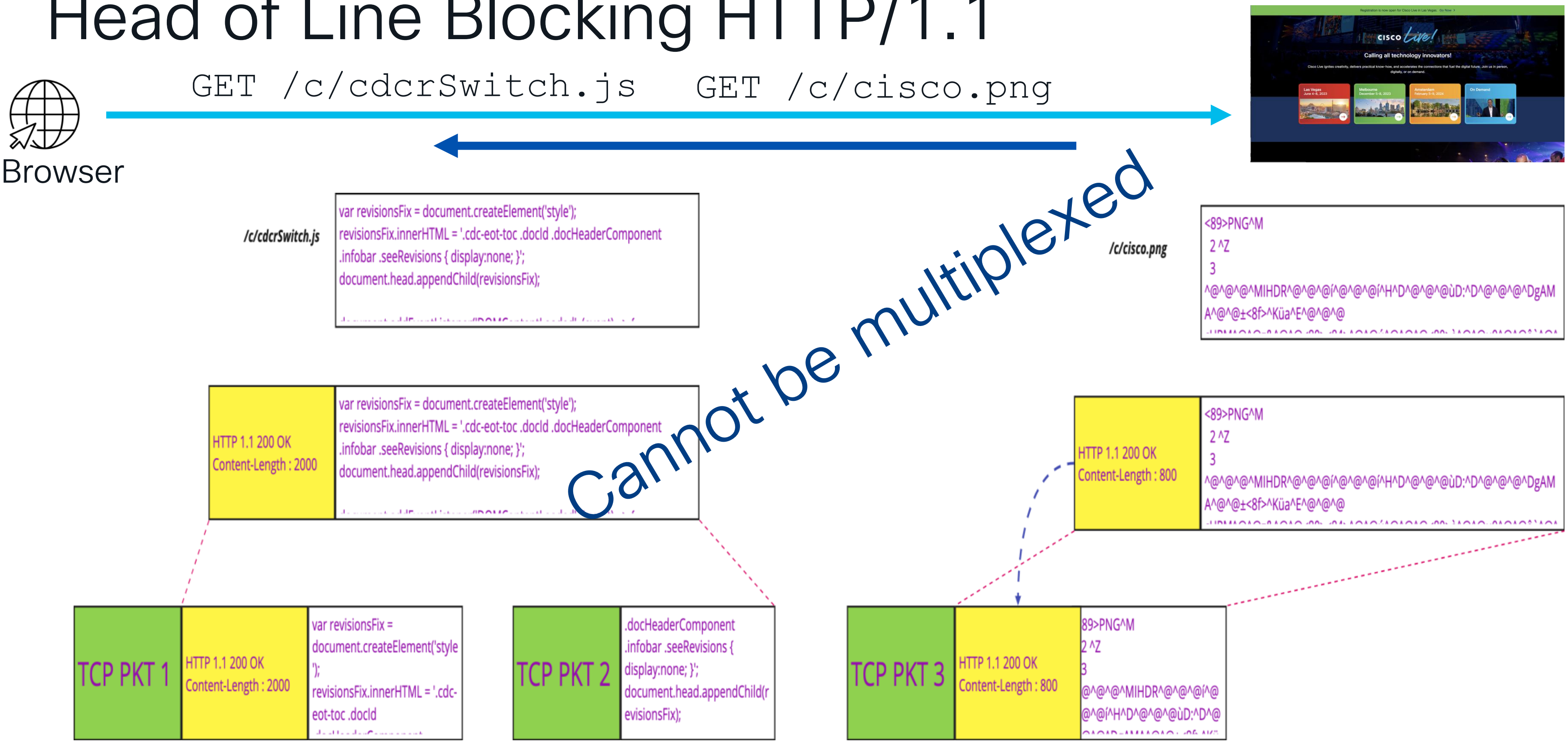
/c/cdcrSwitch.js

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```

/c/cisco.png

```
<89>PNG^M
 2 ^Z
 3
^@^@^@^MIHDR^@^@^@í^@^@^@í^H^D^@^@^@ùD:^D^@^@^@^DgAM
A^@^@±<8f>^Küa^E^@^@^@
```

Head of Line Blocking HTTP/1.1



Head of Line Blocking HTTP/1.1

- Work around:
  - Browsers open multiple TCP connections, typically 6
  - Shard content over multiple domains for load distribution
- Drawback
  - TCP setup overhead
    - 3-way handshake
  - TLS setup overhead for HTTPS connection

The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.

/c/cdcrSwitch.js

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);

document.addEventListener('DOMContentLoaded', (event) => {
```

The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.

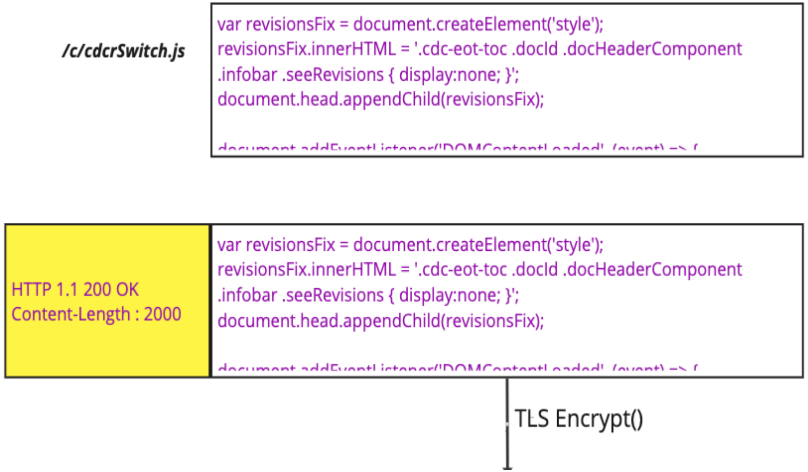
/c/cdcrSwitch.js

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
.infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);

document.addEventListener('DOMContentLoaded', (event) => {
```

HTTP 1.1 200 OK
Content-Length : 2000
	var revisionsFix = document.createElement('style');
	revisionsFix.innerHTML = '.cdc-eot-toc .docId .docHeaderComponent
	.infobar .seeRevisions { display:none; }';
	document.head.appendChild(revisionsFix);
	
	document.addEventListener('DOMContentLoaded', (event) => {

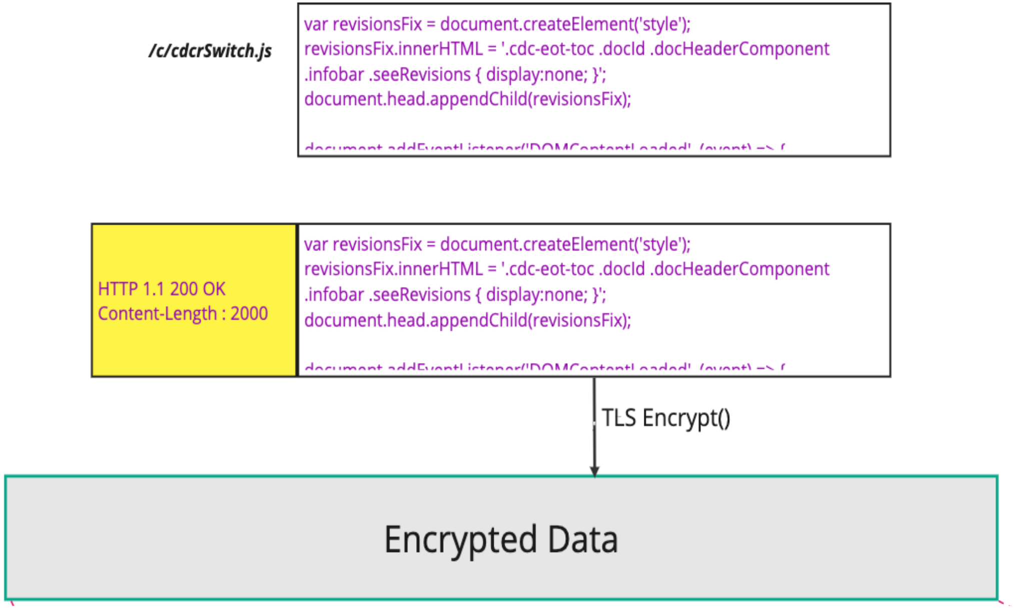
The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.



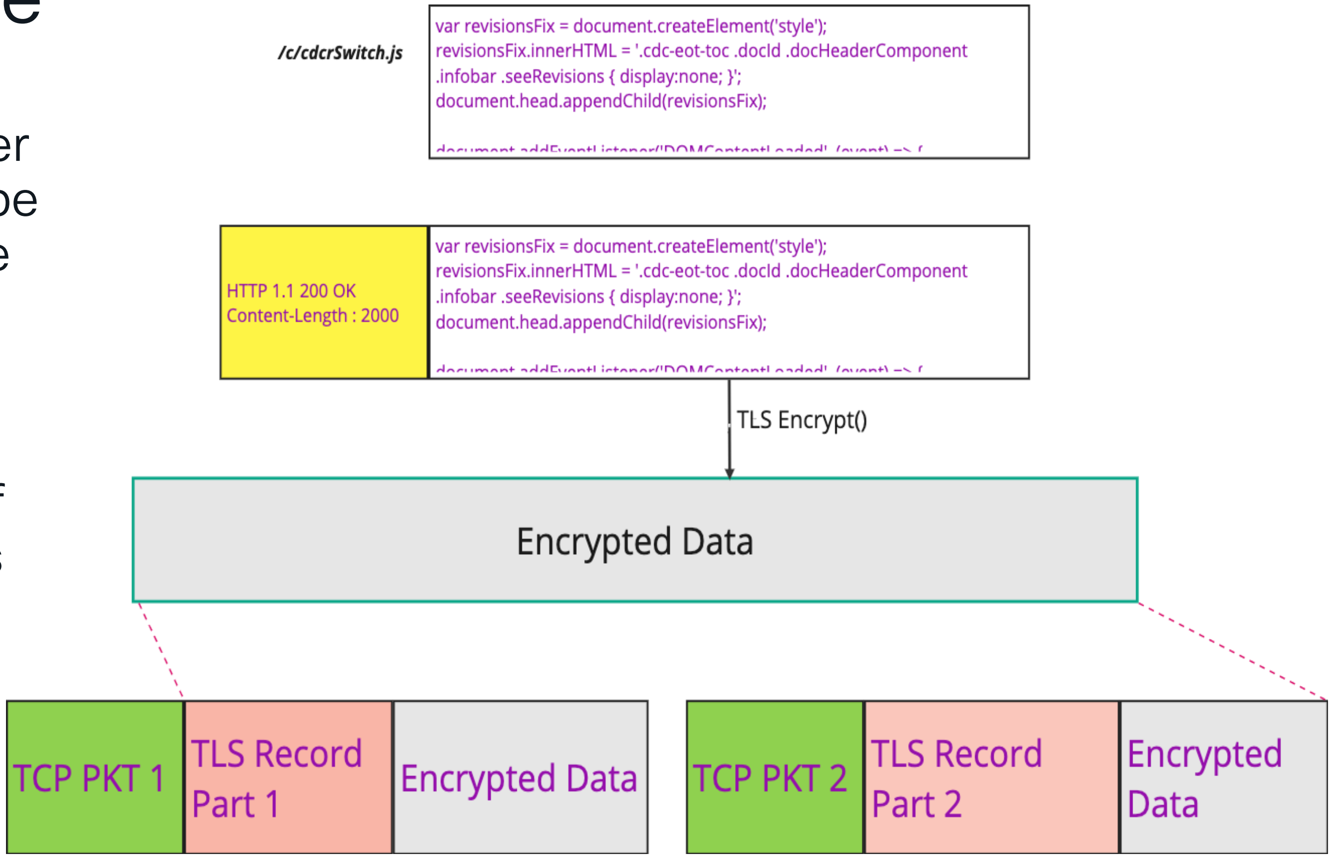
The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.



The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.

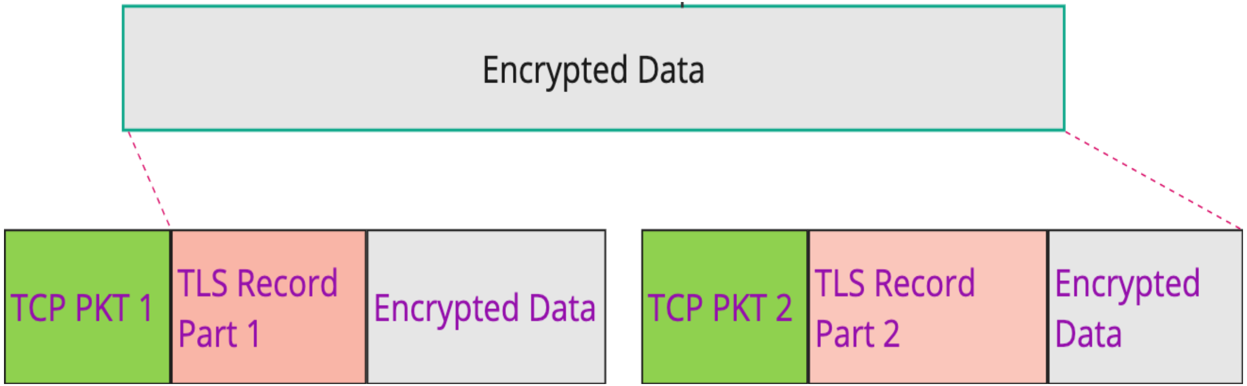


The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption
- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.

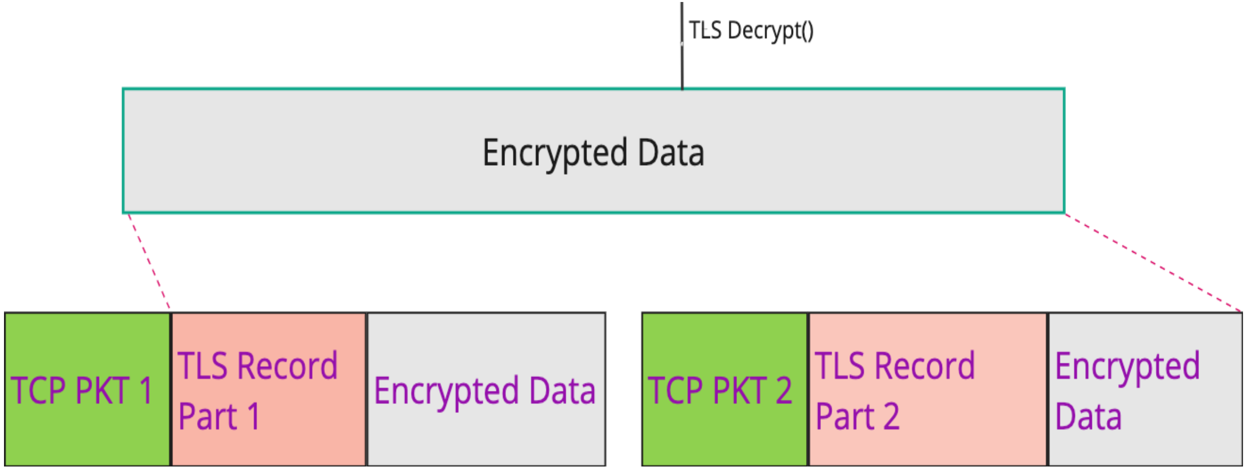
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption
- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.



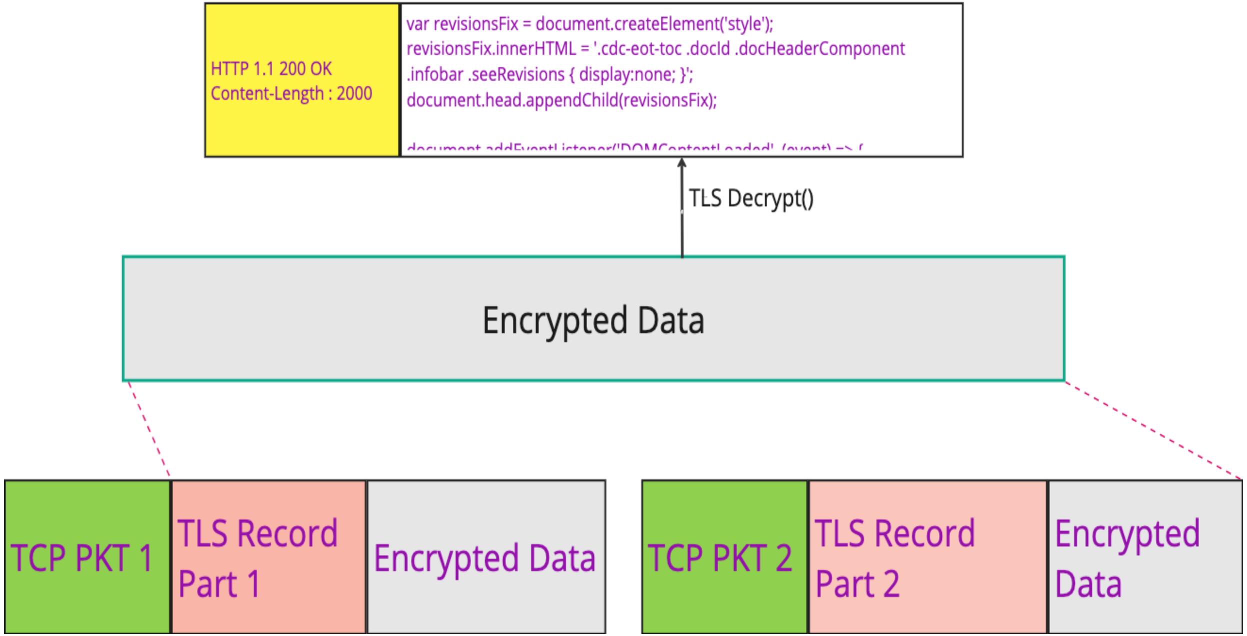
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption
- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.



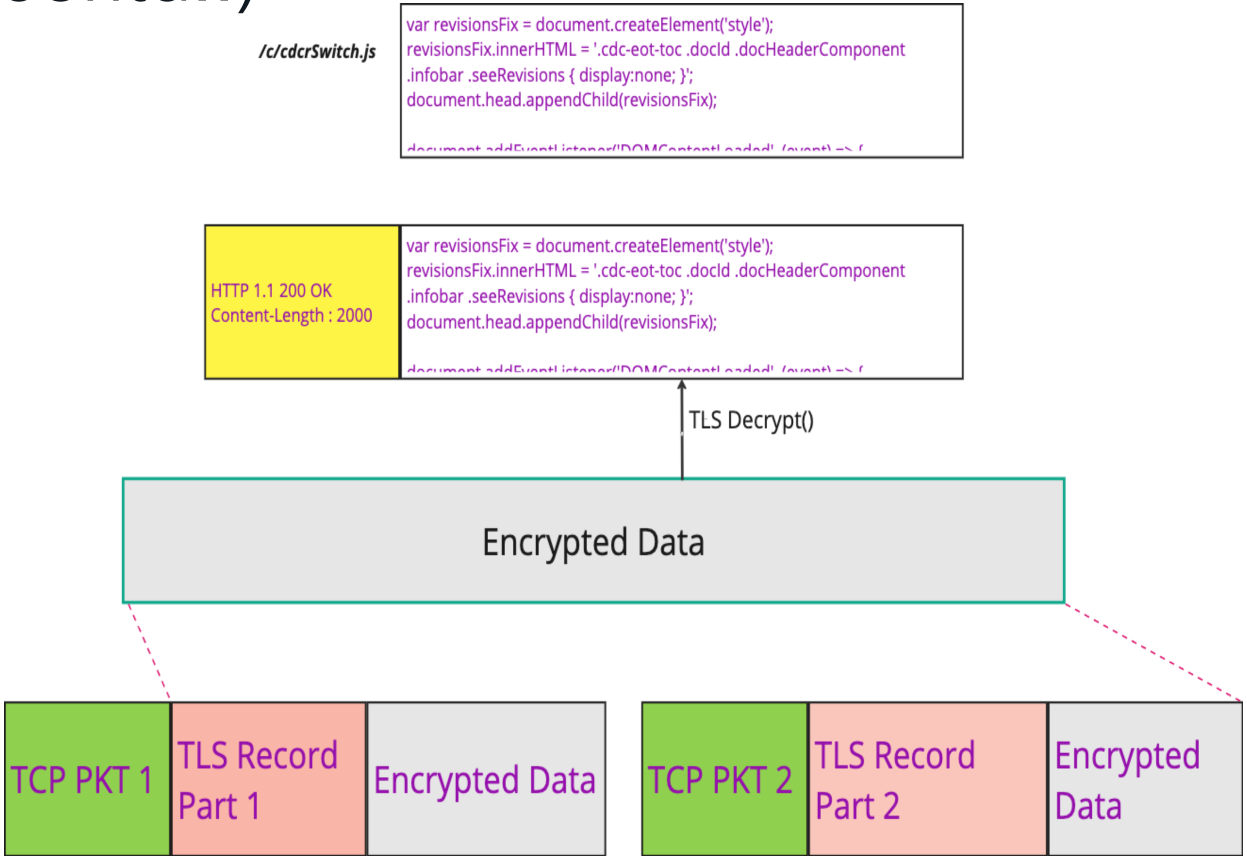
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption
- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.



The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption
- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.



Problem Summary

HTTP 1.1

- Data sent before blocks data sent later
- Cannot be multiplexed

TLS

- Transmission errors cause delayed decrypt
- Setup times are huge

ADVENT OF HTTP/2

HTTP/2

- Stream based protocol
- Multiple HTTP content Streams
- Truly multiplexed
- One TCP connection

HTTP/2

- Stream based protocol
- Multiple HTTP content Streams
- Truly multiplexed
- One TCP connection

TCP PKT 1	HEADERS: Stream ID:1	status 200 Content-Length : 100	DATA: Stream ID:1	status 200 Content-Length : 900	var revisionsFix = document.createElement('style '); revisionsFix.innerHTML = '.cdc-eot-toc .docId
TCP PKT 2	Data: Stream ID:2	status 200 Content-Length : 50	DATA: Stream ID:2	status 200 Content-Length : 500	if (typeof ctm === "undefined") ctm = {}; //Function encapsulates all GDPR releated features and attributes

TCP PKT 3	Data:
	Stream ID:1
	Length:600
		document.addEventListener('DOMContentLoaded', (event) => {	DATA:
			Stream ID:2
			Length : 200
				//Event Queue for call backs if not defined
				var eventqueue = [];
				//Optanon Wrapper Function Counter	

Problem Recap

HTTP 1.1

- Data sent earlier blocks data sent later
- Cannot be multiplexed

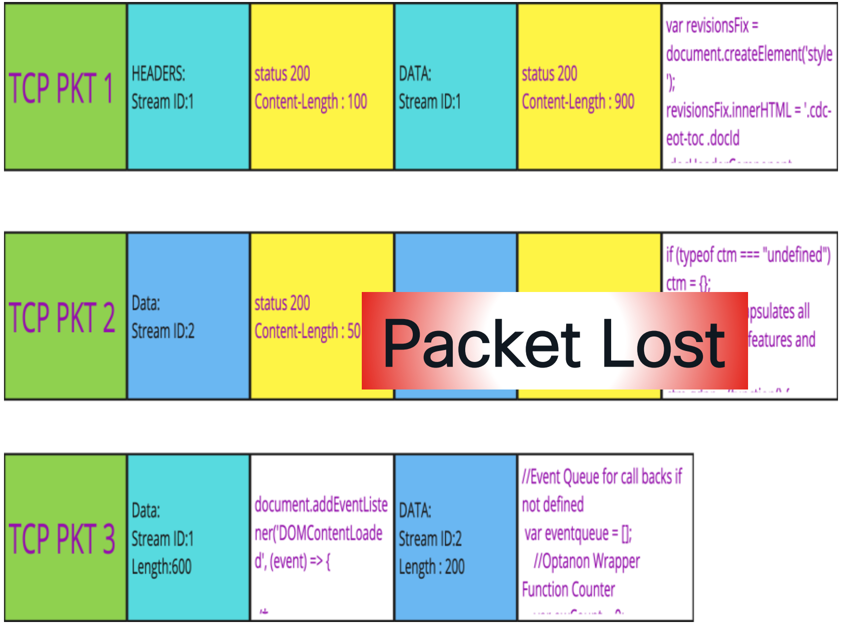
SOLVED

TLS

- Transmission errors cause delayed decrypt
- Setup times are huge

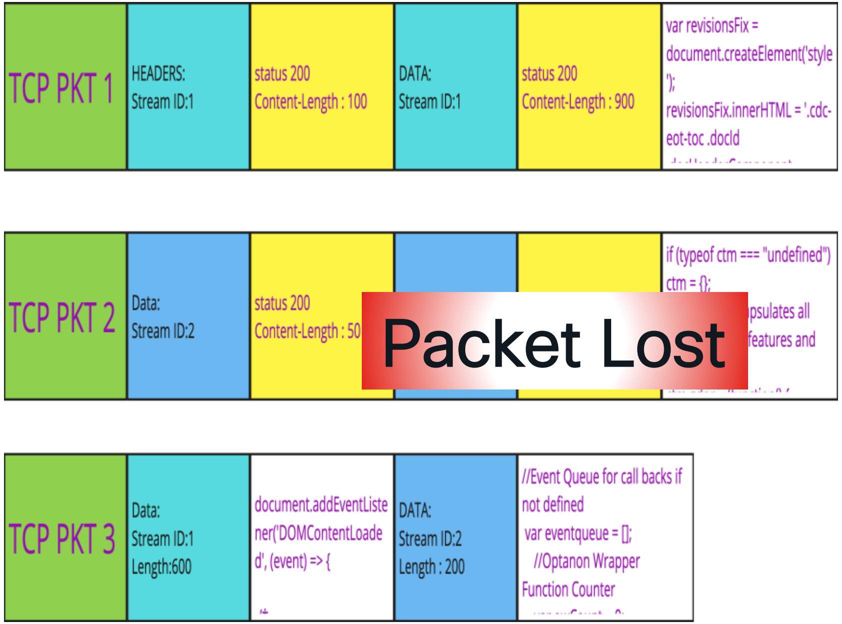
NOT SOLVED

A Not so New Problem .... Yet Again



- TCP guarantees in order packet delivery
- Assume packet ‘2’ lost
- Packet ‘1’ delivered to application
- Packet ‘3’ buffered
  - Until packet ‘2’ retransmitted and received successfully
- Packet ‘3’ may be containing a different stream, but blocked by lost packet ‘2’
- TCP is unaware of payload data
- Remember: TLS problem still exists

A Not so New Problem .... Yet Again



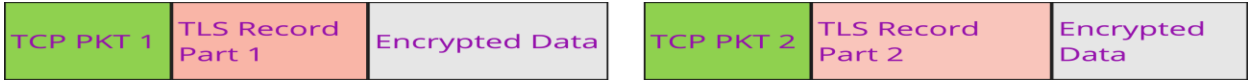
<u>Head of Line Blocking at TCP Level</u>

- TCP guarantees in order packet delivery
- Assume packet ‘2’ lost
- Packet ‘1’ delivered to application
- Packet ‘3’ buffered
  - Until packet ‘2’ retransmitted and received successfully
- Packet ‘3’ may be containing a different stream, but blocked by lost packet ‘2’
- TCP is unaware of payload data
- Remember: TLS problem still exists

HTTP/3 over QUIC PROTOCOL

Need for HTTP/3

HTTP/1.1



HTTP/2

- Multiplexed streams
- Reduced connection times
- Increased Privacy
- Eliminate TCP head of line blocking

HTTP/3

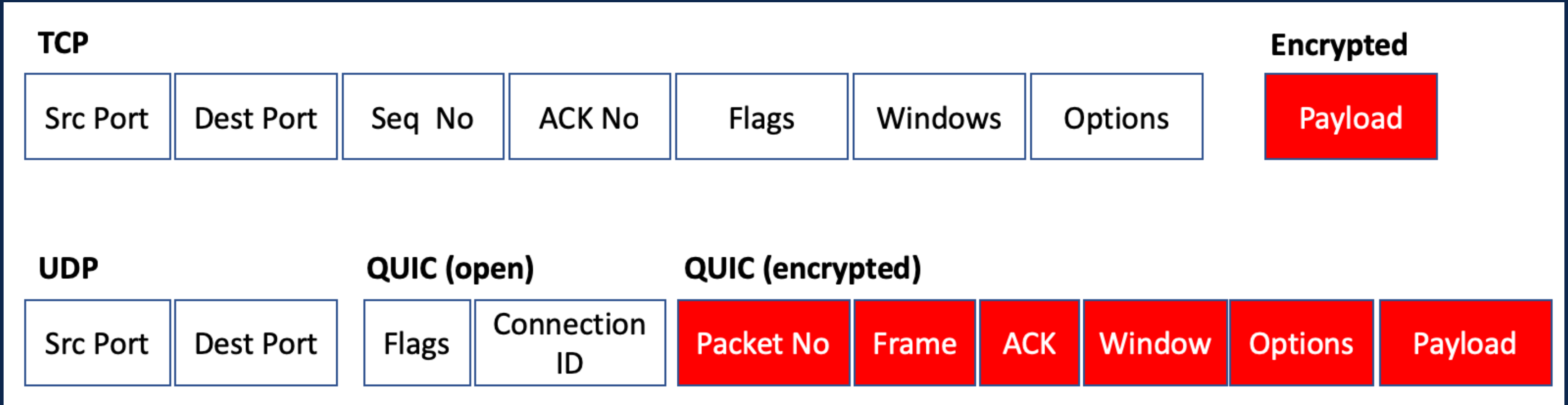
- Removal of streams at Application layer
- Simple compared to HTTP/2
- In some ways similar to HTTP/1
  - Multiple connections are instead carried over multiple streams on the same connection.
- Works only on QUIC
- Streaming protocol → streams from HTTP/2 moved down to transport layer

Quick UDP Internet Connections - QUIC

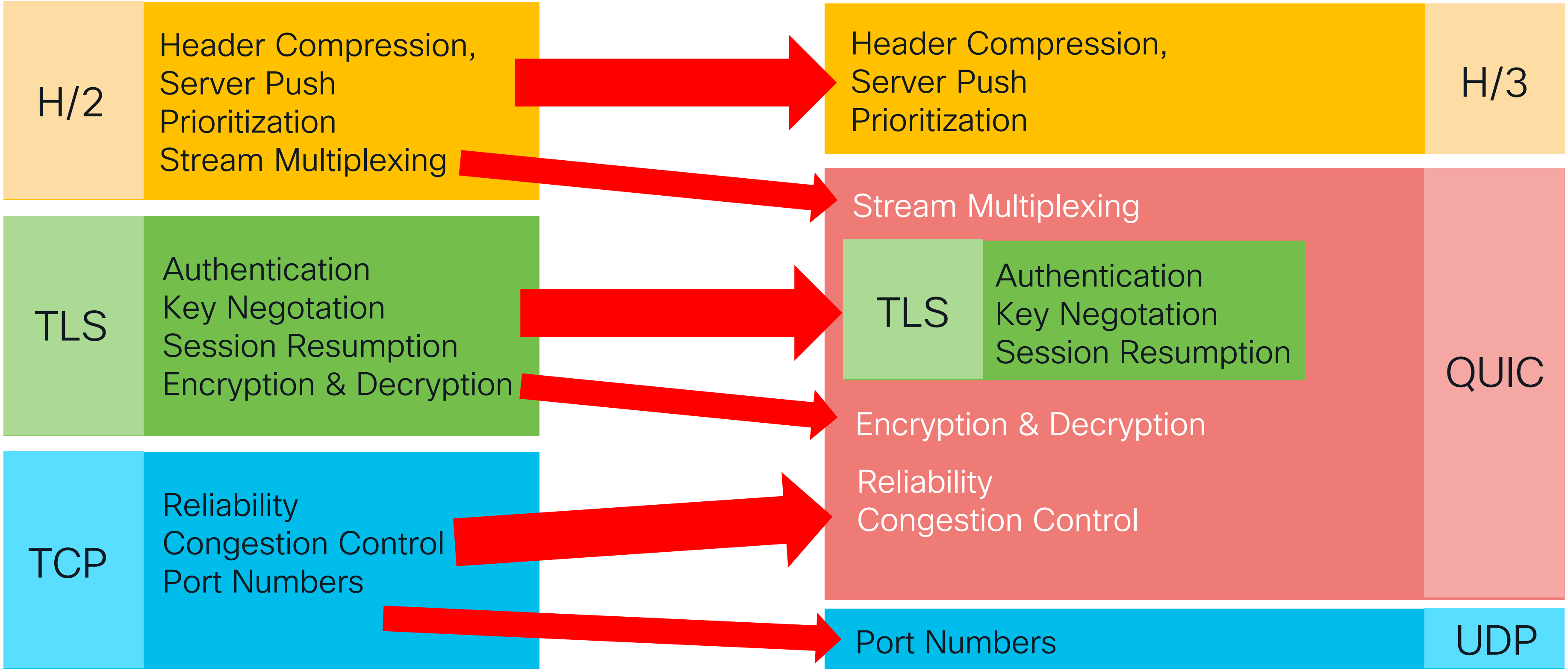
- Generic for all kinds of traffic
- NOT ONLY HTTP

- Always encrypted
  - No opt-in, no opt-out

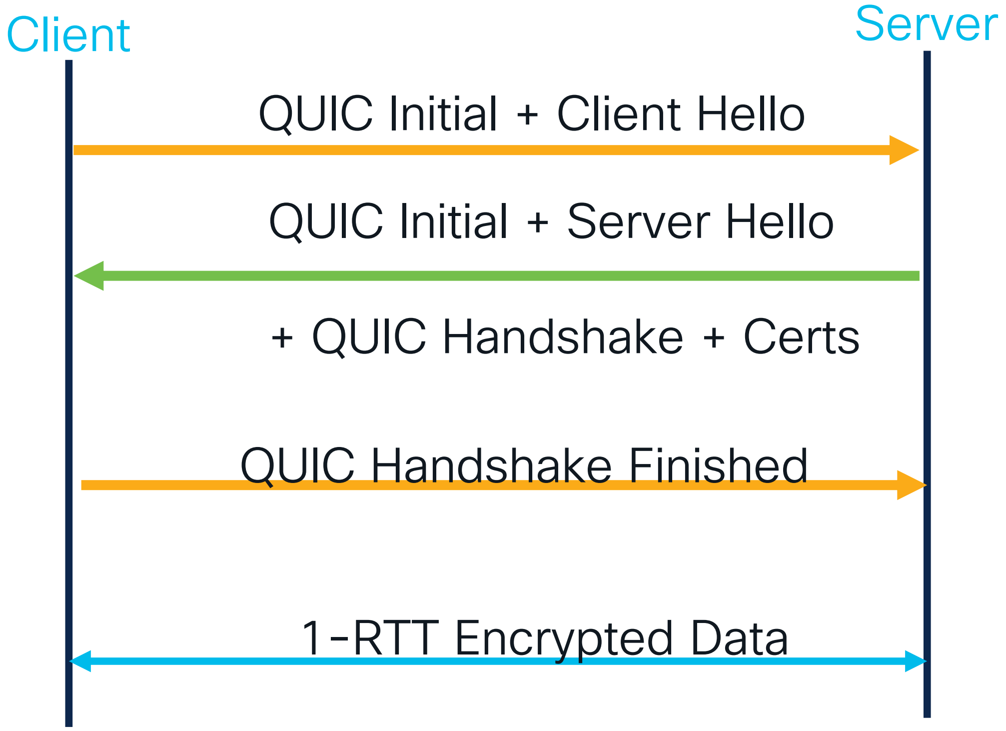
TCP vs QUIC


TCP
Src Port
Dest Port
Seq No
ACK No
Flags
Windows
Options
Encrypted
Payload
UDP
Src Port
Dest Port
QUIC (open)
Flags
Connection ID
QUIC (encrypted)
Packet No
Frame
ACK
Window
Options
Payload

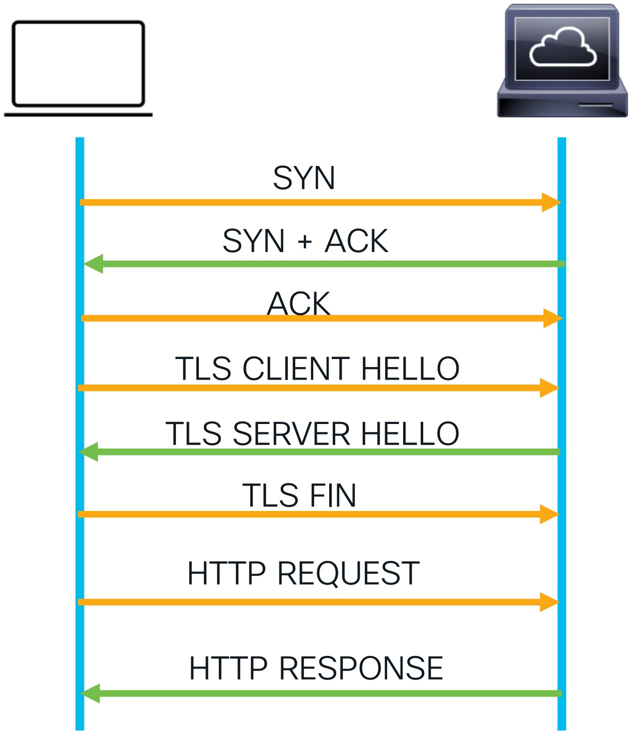
Where Does QUIC Fit in the Protocol Stack?



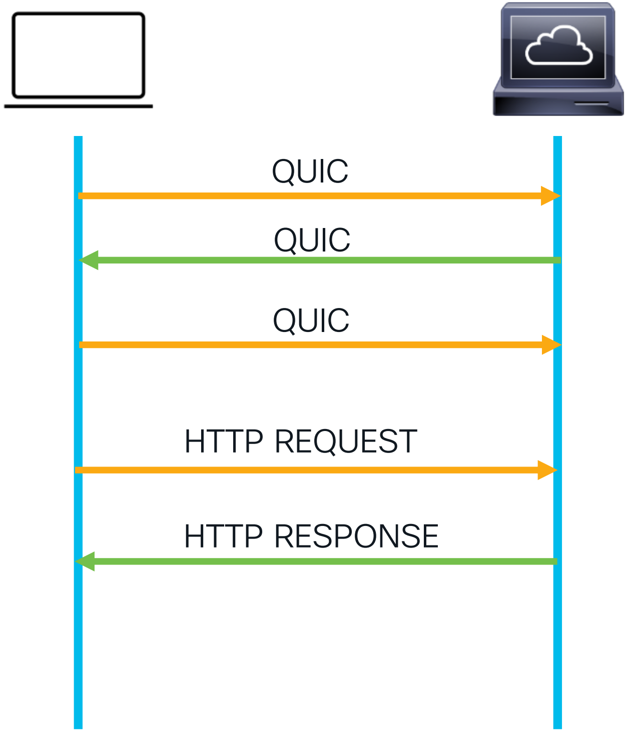
QUIC Connection Setup



Connection Comparison

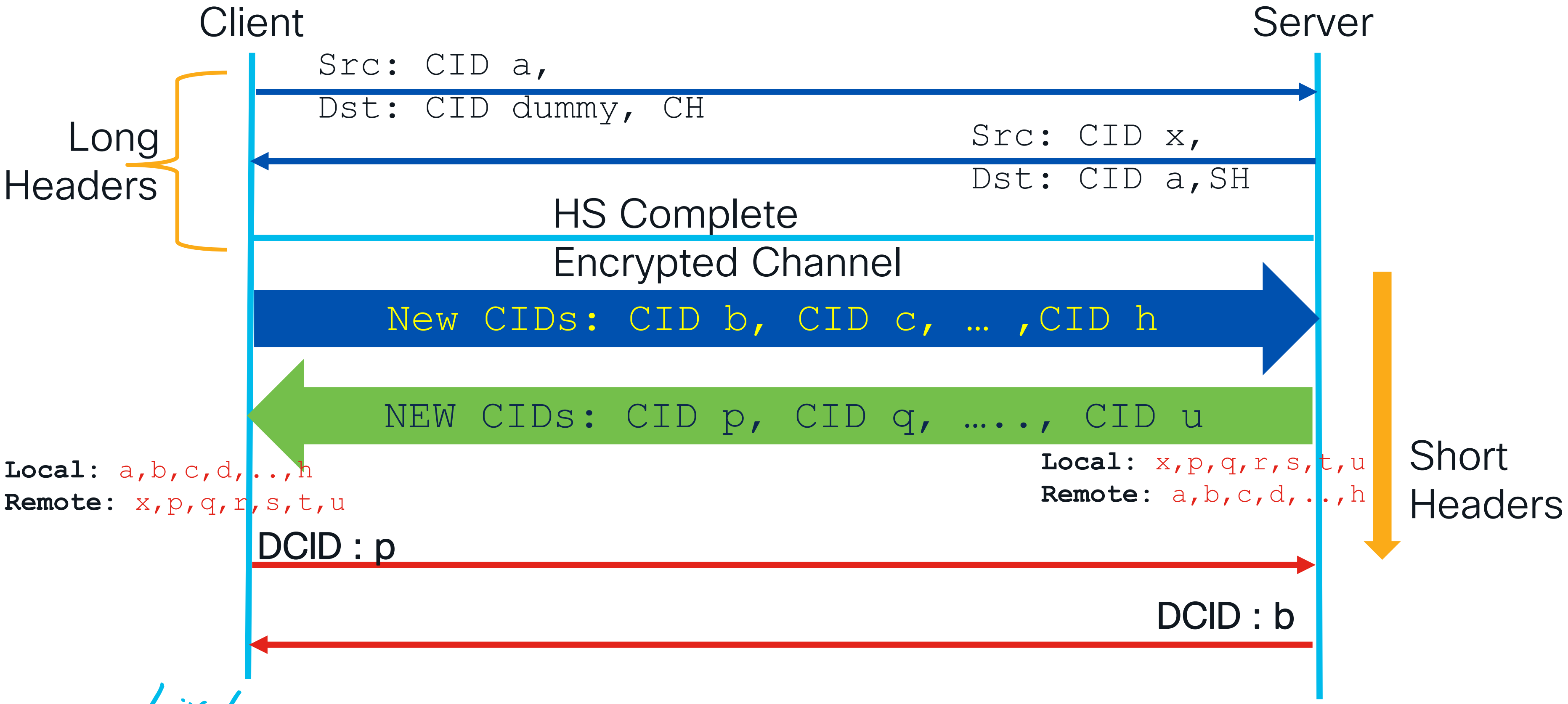


HTTP over TCP+TLS



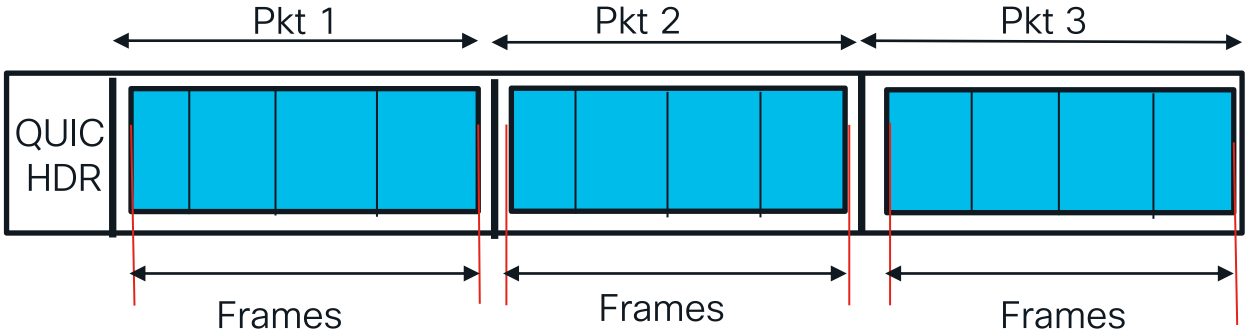
HTTP over QUIC

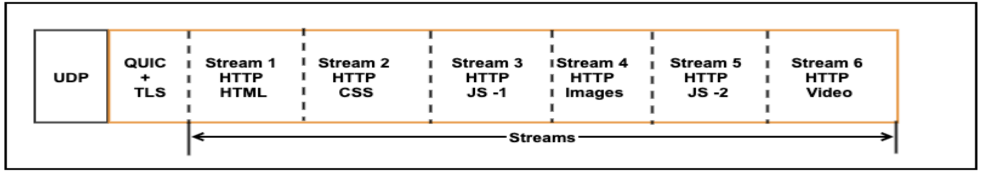
Connection Identifiers



QUIC Packets

- One UDP payload can have multiple QUIC packets
- Each packet has multiple frames

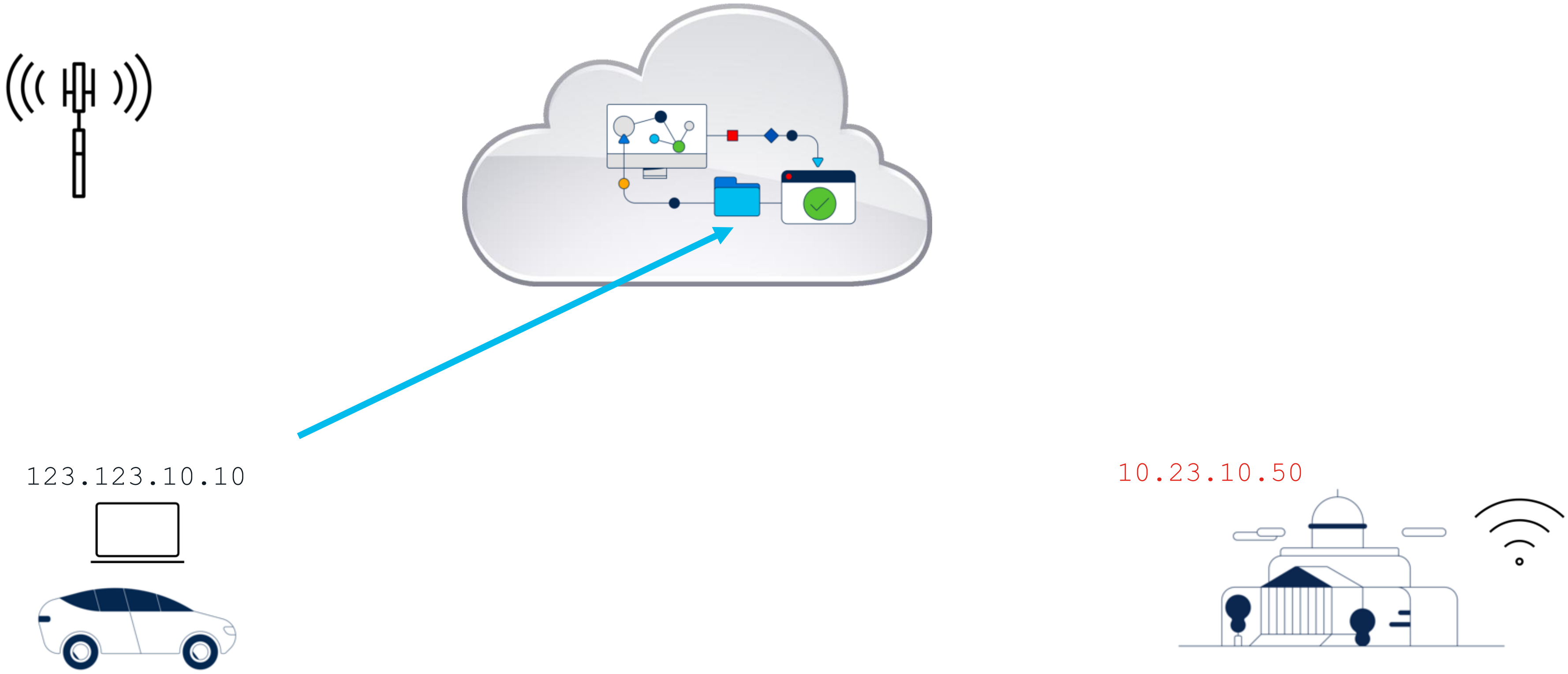




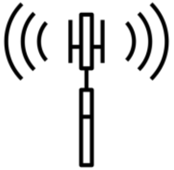
Connection Migration

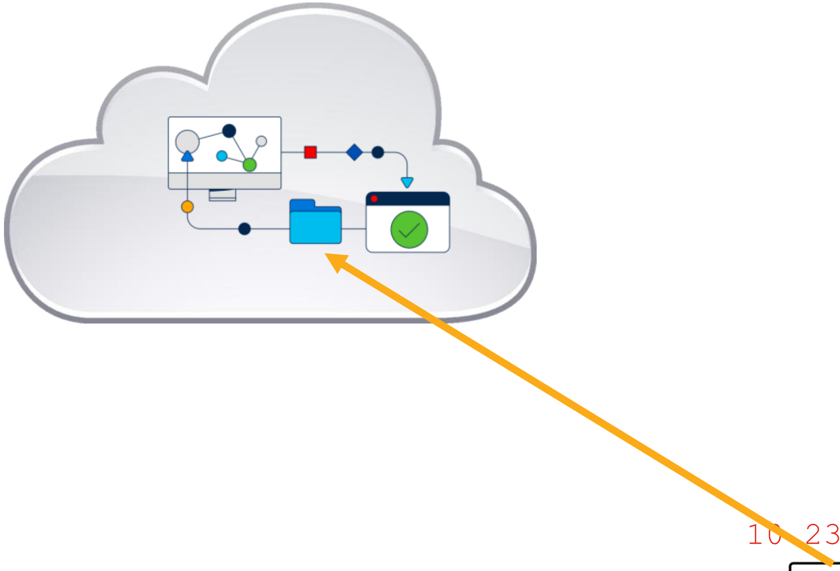
- QUIC uses Connection Identifiers (CID) for identifying connections, not IP
- Any CID could be used to identify the connection
- The set of CIDs, except the first pair, are exchanged over encrypted channel
- IP of peers can change any time, CIDs help keep the connections alive
- For privacy, new CID to be used on IP migration
- Migration can happen from IPv4 to IPv6 and vice-versa

Connection Management with TCP



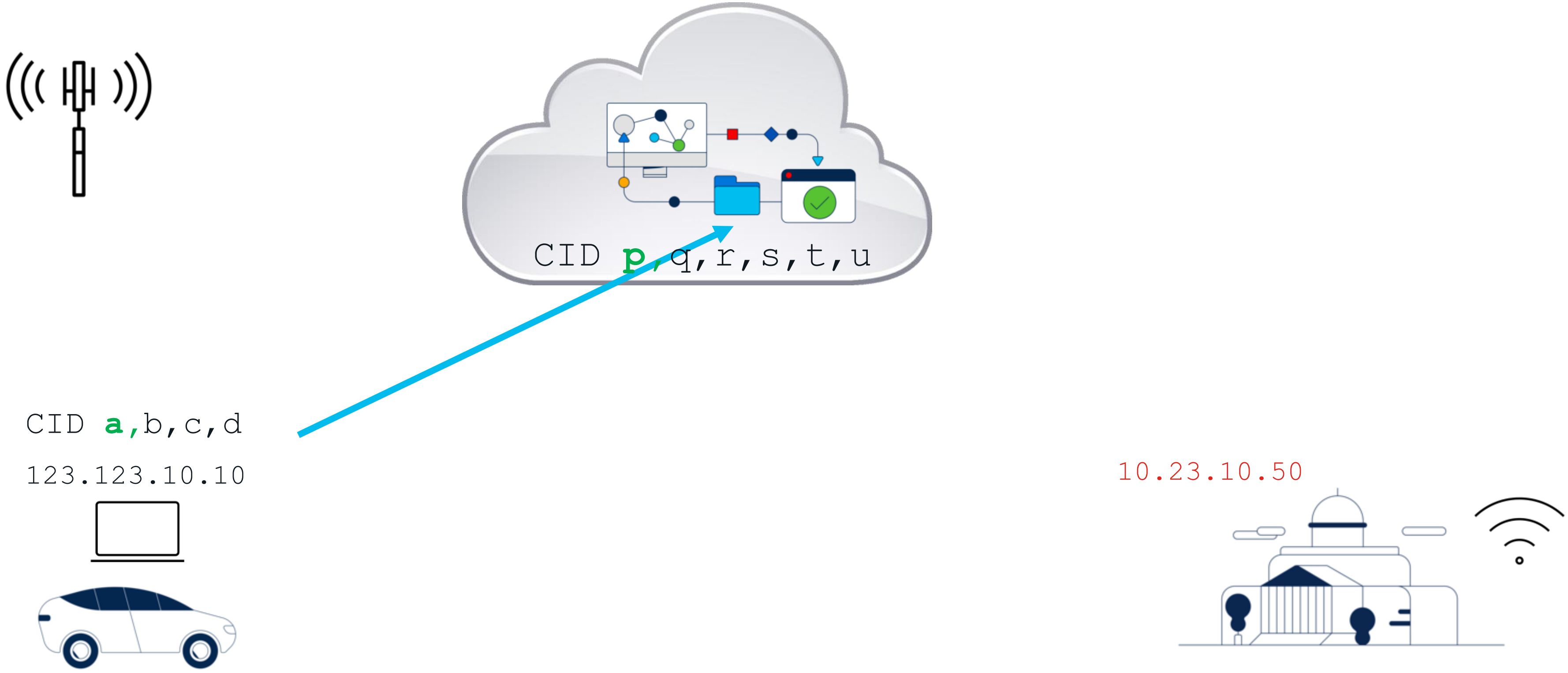
Connection Management with TCP



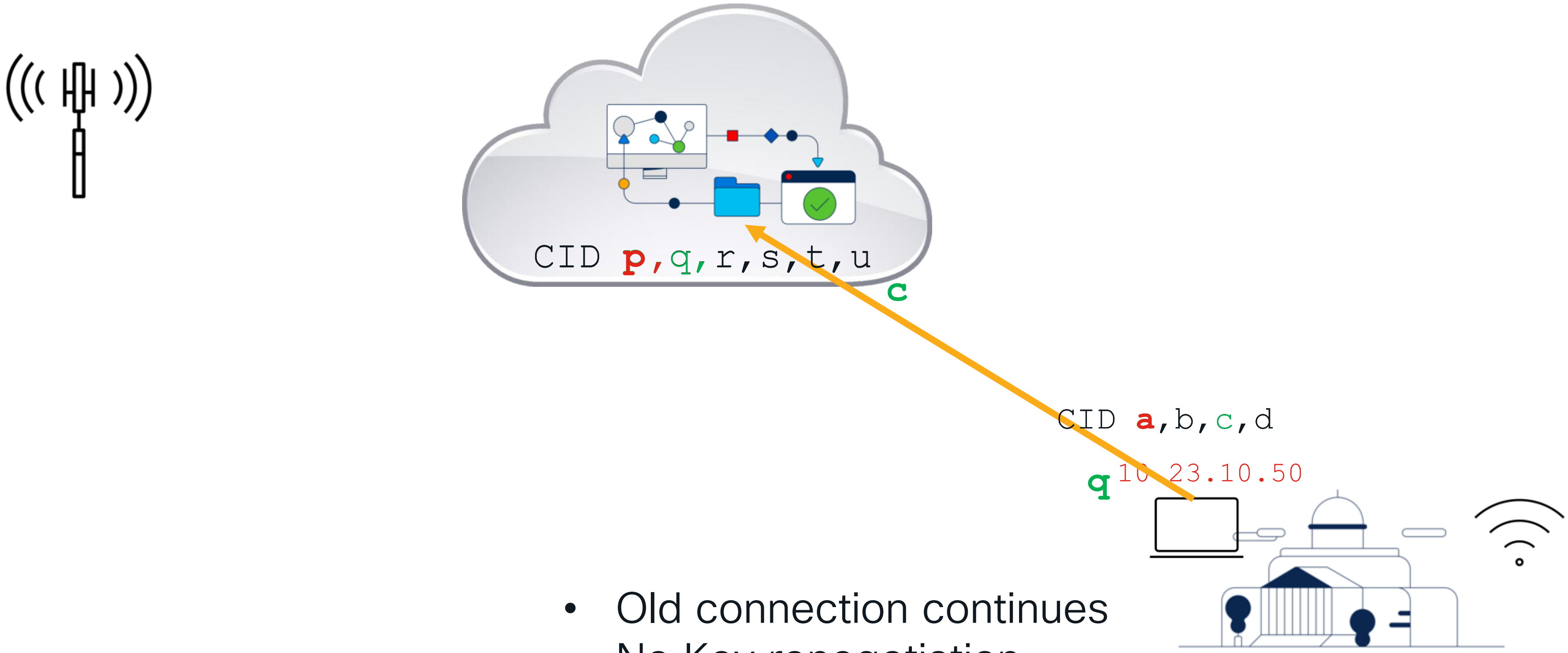


- New TCP + TLS Connection has to be established.
- Increases Latency

Connection Management with QUIC

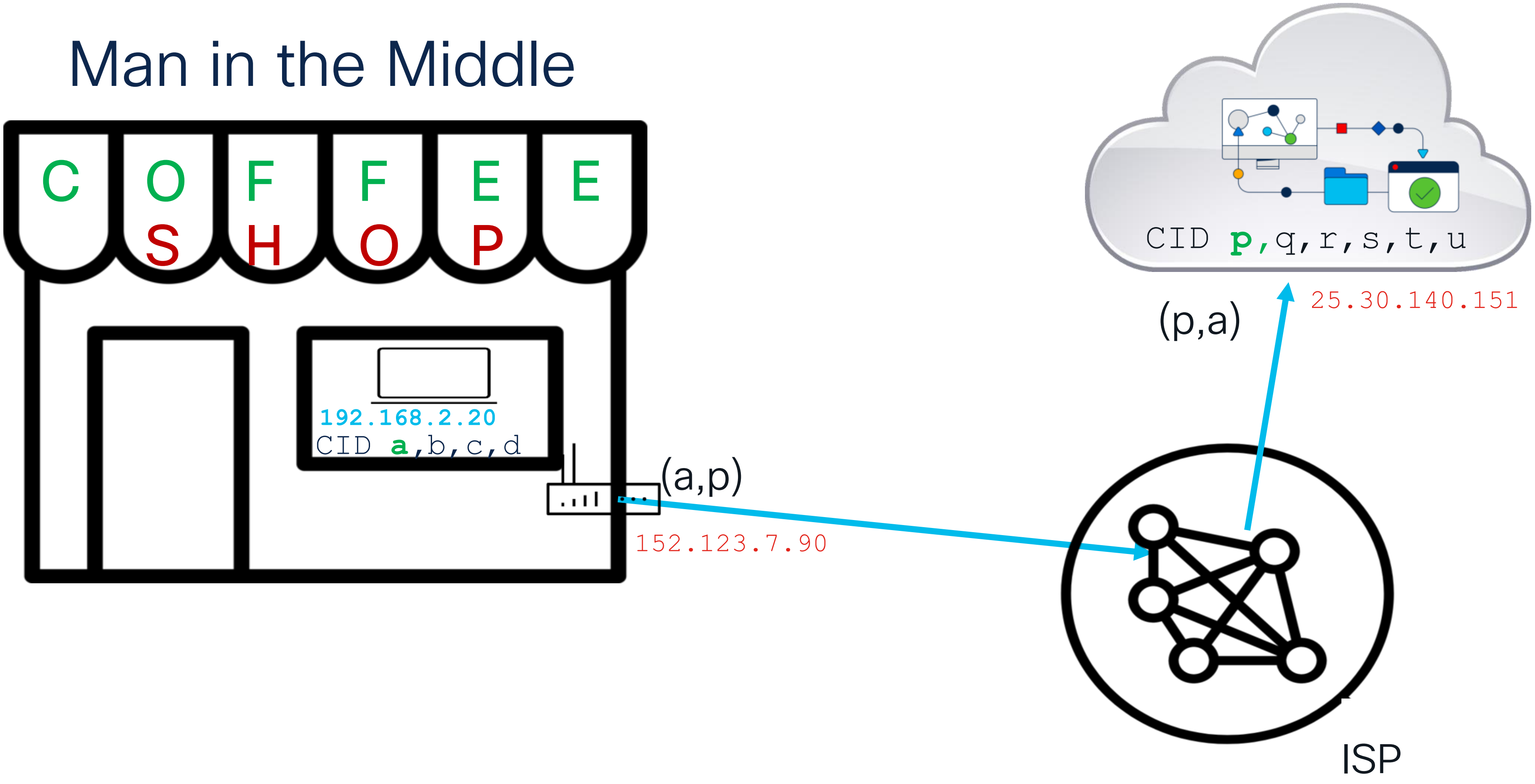


Connection Management with QUIC

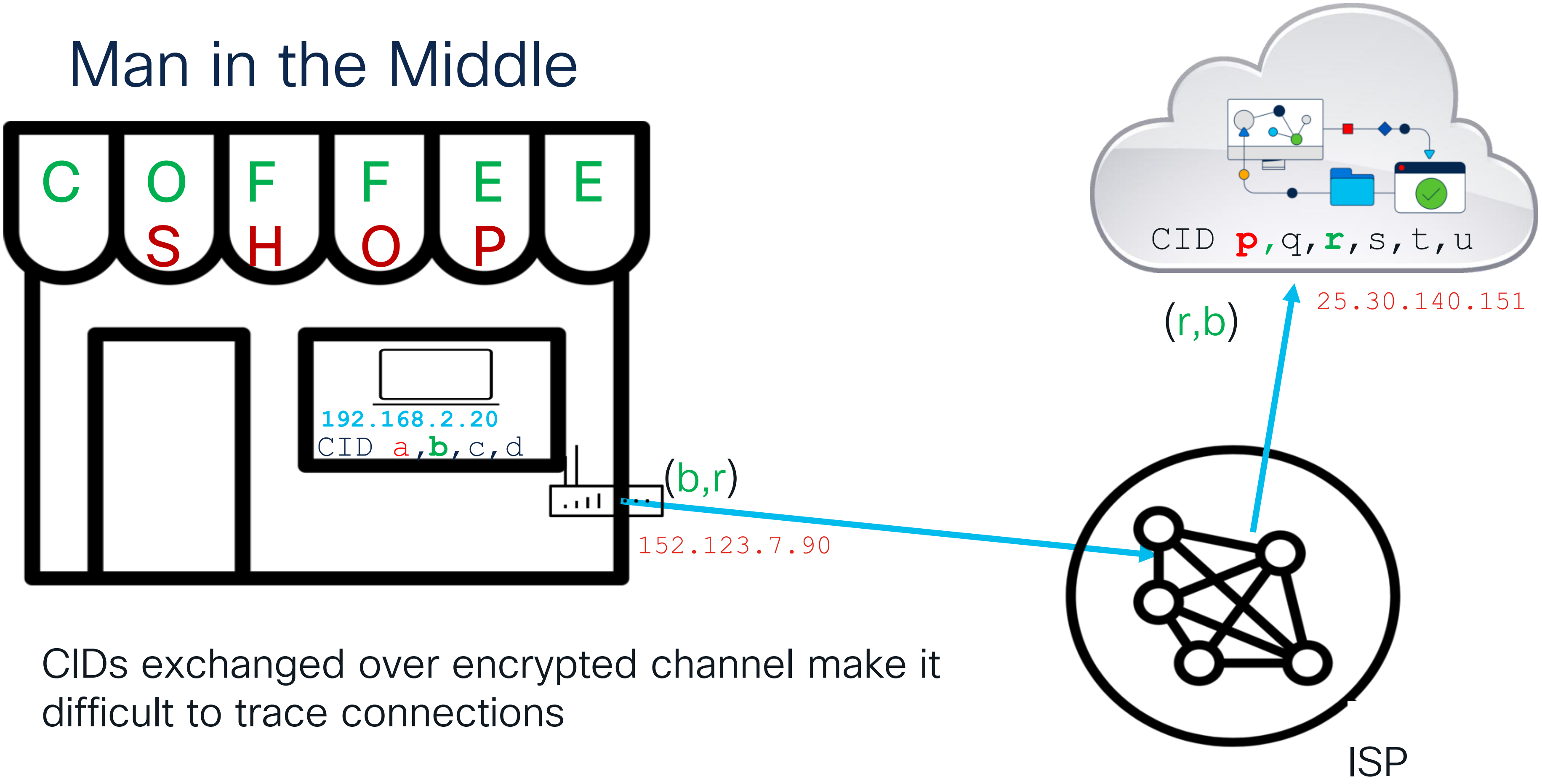


- Old connection continues
- No Key renegotiation

Man in the Middle



Man in the Middle



CIDs exchanged over encrypted channel make it difficult to trace connections

Other Properties of QUIC

Unique Packet numbers

Even retransmitted packets have different packet numbers

Connection Resumption

Connecting to a server which negotiated a 0-RTT secret in the previous session

Leverages TLS 1.3 0-RTT

Congestion Control at stream layer

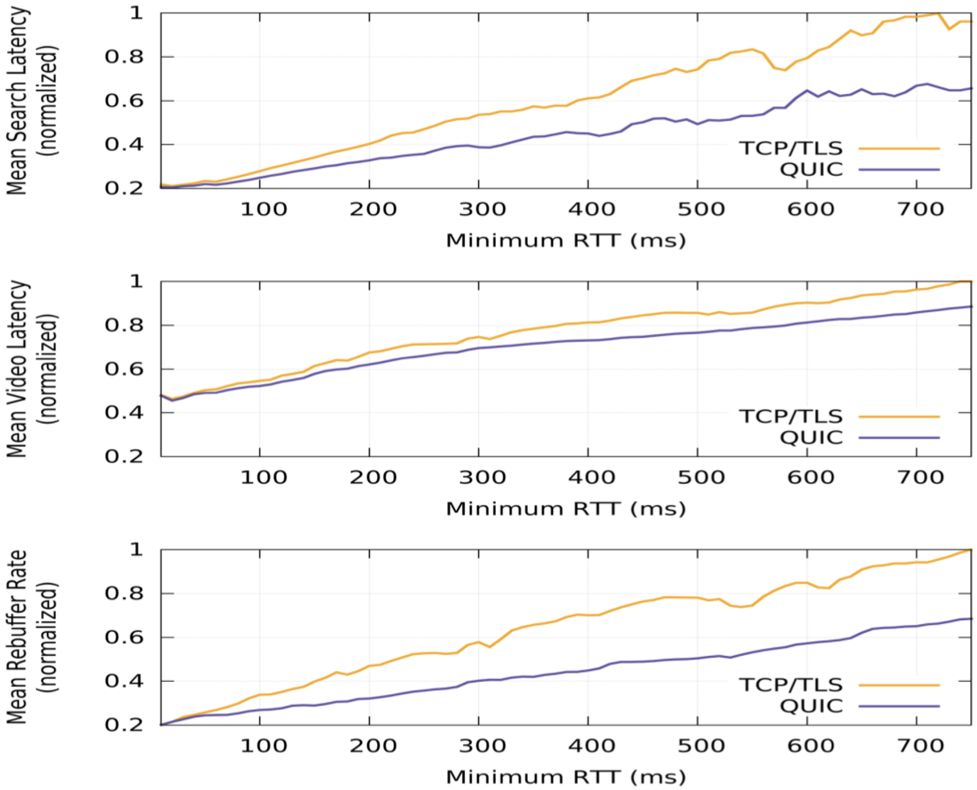
Optimised ACKing

Performance Benefits

- Mileage varies
  - Implementation
  - Congestion control options used

- Low improvement in stable/reliable networks

- Moderate improvement in unstable networks



Source :https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46403.pdf

Challenges for Firewalls

Challenges for existing Security devices

FW is blind to QUIC traffic

- Can at most block UDP/443
- No mechanism to inspect encrypted traffic

Only option to block QUIC traffic

- Recommended by almost all FW vendors

IP cannot be used to identify connections

- Would be difficult to create ACLs and other transport level parameters based rules
- Much of the content would be in the streams

Challenges to adding QUIC support

QUIC is IP agnostic

- Firewalls use IP to track and maintain connection states. Changing IPs require Firewalls to decide if they should allow or block QUIC traffic.

Stream Based

- Reassembly of streams can be done only after decryption of packets.

Challenges to adding QUIC support

Connection ID Based

- One flow can have multiple Connection IDs.
- Source Connection ID is optional in data packets.
- Firewalls have to understand and manage Connection IDs

All packets Encrypted

- Encryption at packet level – 2 levels of decryption
- More packets to be decrypted.
- Unlike TLS where TLS records span multiple packets which needs lesser crypto invocations
- Compute intensive

Work based on QUIC

Follow up Poll

slido

Join at

slido.com

#1546 305

Passcode: **clusquic**

QUIC Inspection in Cisco Secure Firewall

Coming soon

- Inspects content per stream
- Block individual streams

Encrypted Visibility Engine

TLS Decryption is Expensive

Over 90% of Internet traffic being encrypted with Transport Layer Security (TLS)

Mobile app and SaaS flows are not decryptable

TLS 1.3 encrypts Server Certificate Information

TLS decryption has a steep performance impact

Use techniques to make decisions on TLS traffic without decryption

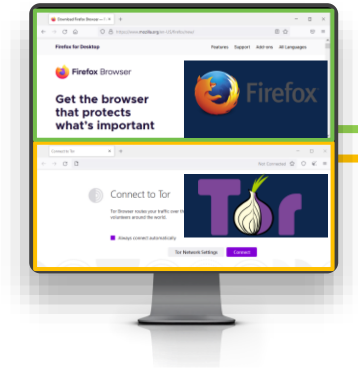
Almost 63% of servers prefer TLS 1.3 to other protocols (Aug/2021)

Encrypted Visibility Engine uses TLS Fingerprinting

TLS ClientHello

```
Cipher Suites (18 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)
    Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
    Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
```

Confidence: 99.94%
Process: firefox.exe
Version: 76.0.1
Category: browser
OS: Windows 10 19041.329
Typical FQDN: cisco.com



TCP/TLS 192.168.2.110/34624->172.16.45.200/443

TCP/TLS 192.168.2.110/21013->203.0.113.154/443

FTD

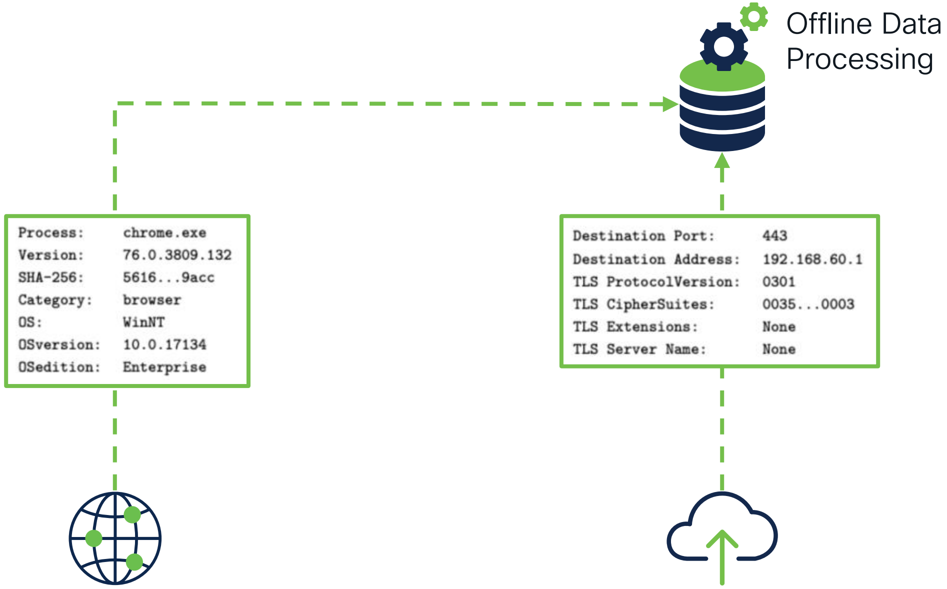
Generate unique fingerprints for client applications based on TLS, TCP and other clear text fields to use for context enrichment

TLS ClientHello

```
Cipher Suites (19 suites)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
    Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
```

Confidence: 100%
Process: tor.exe
Version: 9.0.2
Category: anonymizer
OS: Windows 10 19041.329
Typical FQDN: nsksdlkoup.me

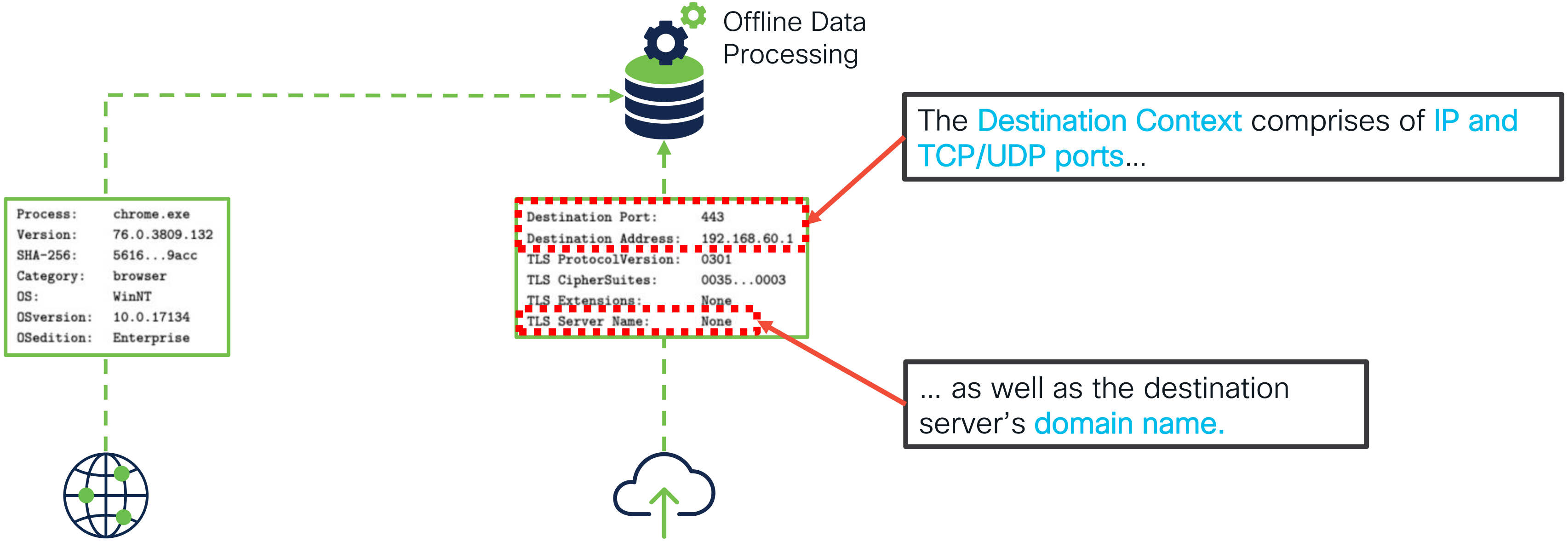
Machine Learning Requires a Comprehensive Data Set



Process data from 80,000 AnyConnect endpoints **everyday**

Over 30 DCs around the globe collecting 1B TLS fingerprints **daily**

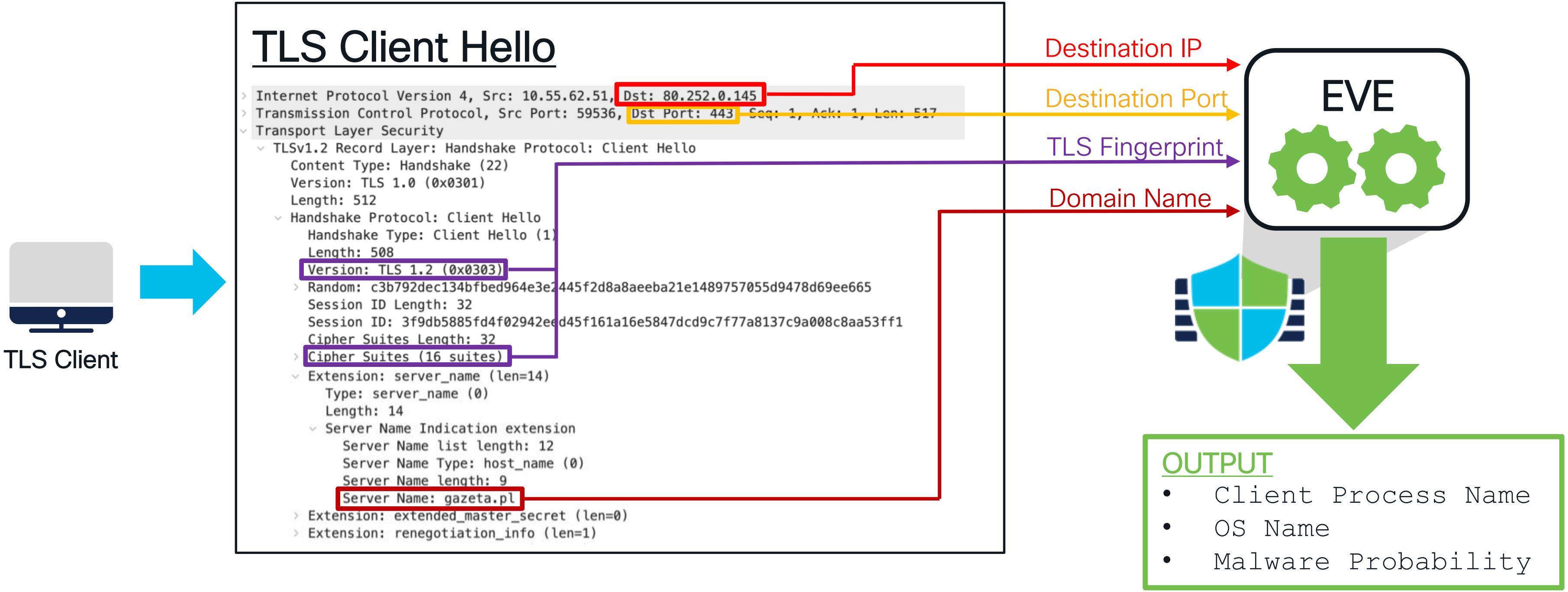
Machine Learning Requires a Comprehensive Data Set



Process data from 80,000 AnyConnect endpoints **everyday**

Over 30 DCs around the globe collecting 1B TLS fingerprints **daily**

Fingerprinting Analysis at the Firewall



Offload Your Firewall with ML-Based Technology

Encrypted Visibility Engine

- Detects malware in encrypted flows
- Can be used to allow/block in firewall policy
- Triggers Indications of Compromise
- Minimal performance impact
- Enriches Endpoint DB with Application and OS

Learn more about EVE

Using the Cisco Secure Firewall with the Encrypted Visibility Engine

- BRKSEC-2099
- By Costas Kleopa
    - Wednesday, Jun 71:00 PM - 2:00 PM PDT
    - Level 2, Breakers DJ

Fill out your session surveys!

Attendees who fill out a minimum of four session surveys and the overall event survey will get **Cisco Live-branded socks** (while supplies last)!

Attendees will also earn 100 points in the **Cisco Live Game** for every survey completed.

These points help you get on the leaderboard and increase your chances of winning daily and grand prizes

Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

The bridge to possible

Thank you

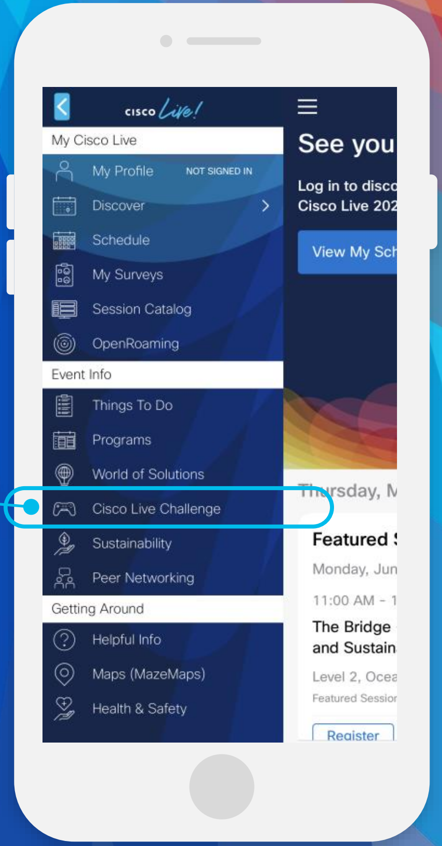
Cisco Live **Challenge**

Gamify your Cisco Live experience!

Get points for attending this session!

How:

1. Open the Cisco Events App.
2. Click on 'Cisco Live Challenge' in the side menu.
3. Click on View Your Badges at the top.
4. Click the + at the bottom of the screen and scan the QR code:



cisco Live!

cisco Live!

Let's go