PLAYBACK: A TLS 1.3 STORY

WHO ARE WE?

Alfonso García Alguacil

Alejo Murillo Moya

INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
- Traffic inspection not as easy as it is currently done

INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
- Traffic inspection not as easy as it is currently done

INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
- Traffic inspection not as easy as it is currently done

INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
- Traffic inspection not as easy as it is currently done

INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
- Traffic inspection not as easy as it is currently done

INTRODUCING TLS 1.3

The **Bad**

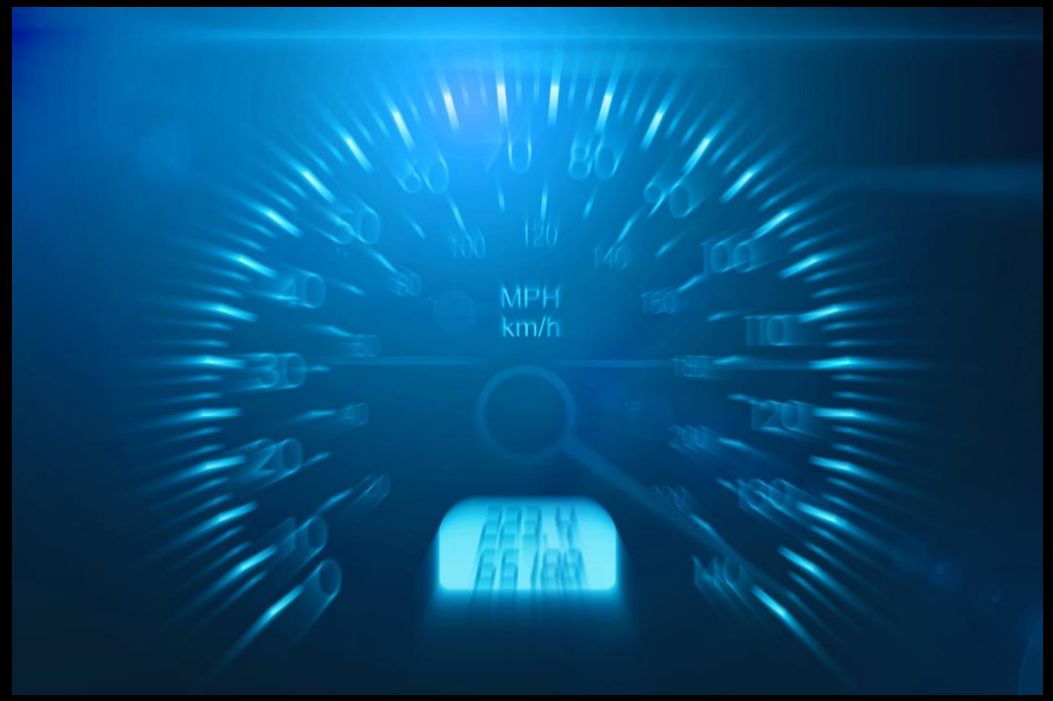
- Protocol tainted due to “compatibility issues” ☹

INTRODUCING TLS 1.3

The **Ugly**

- 0-RTT (this talk ☺)

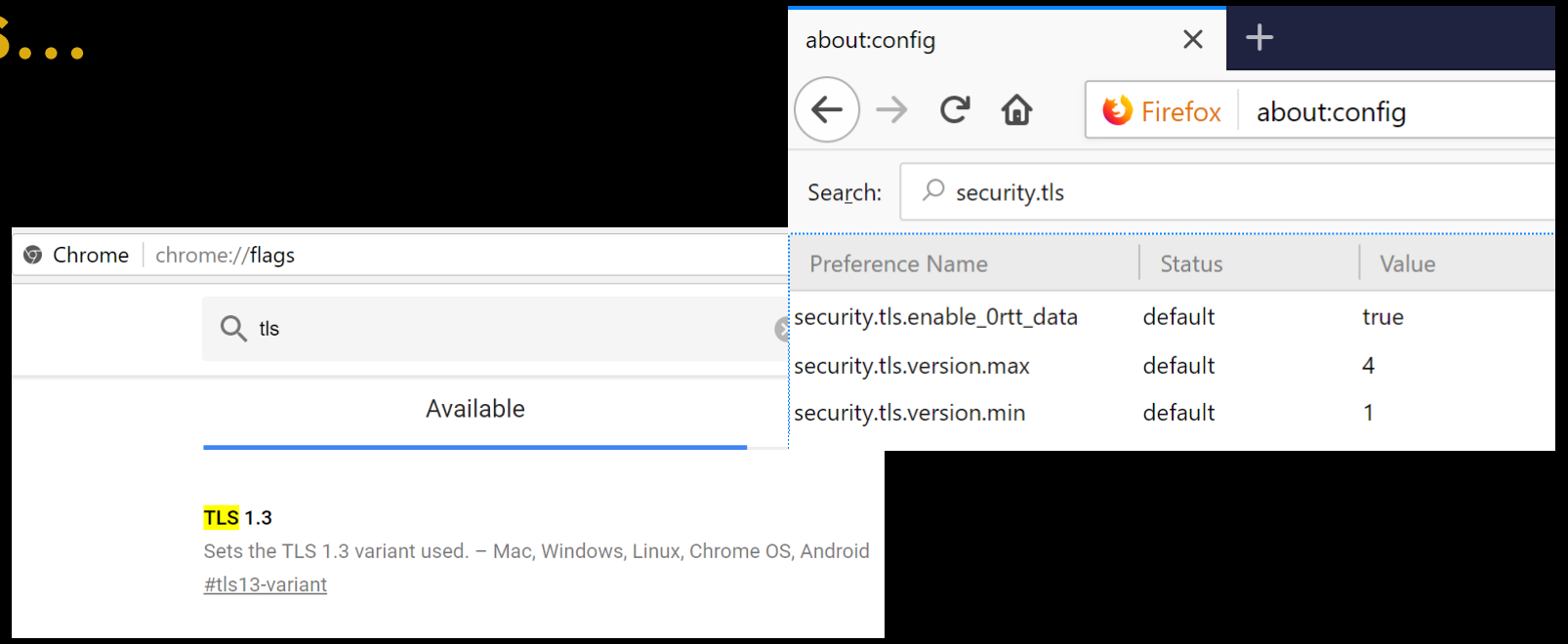
0-RTT: SPEED AT A COST



vs


TELL ME AGAIN...
WHY SHOULD I CARE?
memes.com

Your browsers…



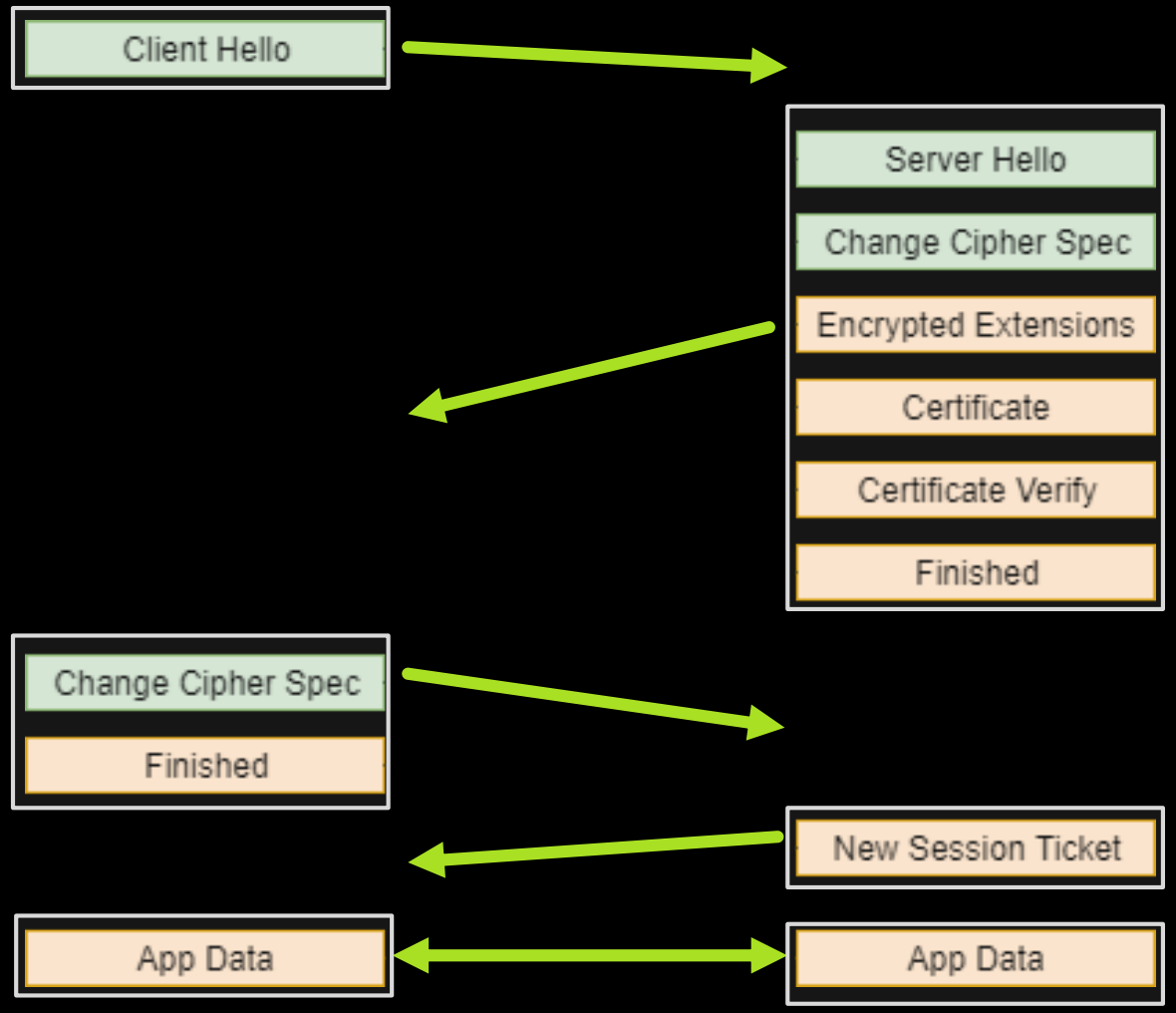
… implementations …

OpenSSL
Cryptography and SSL/TLS Toolkit

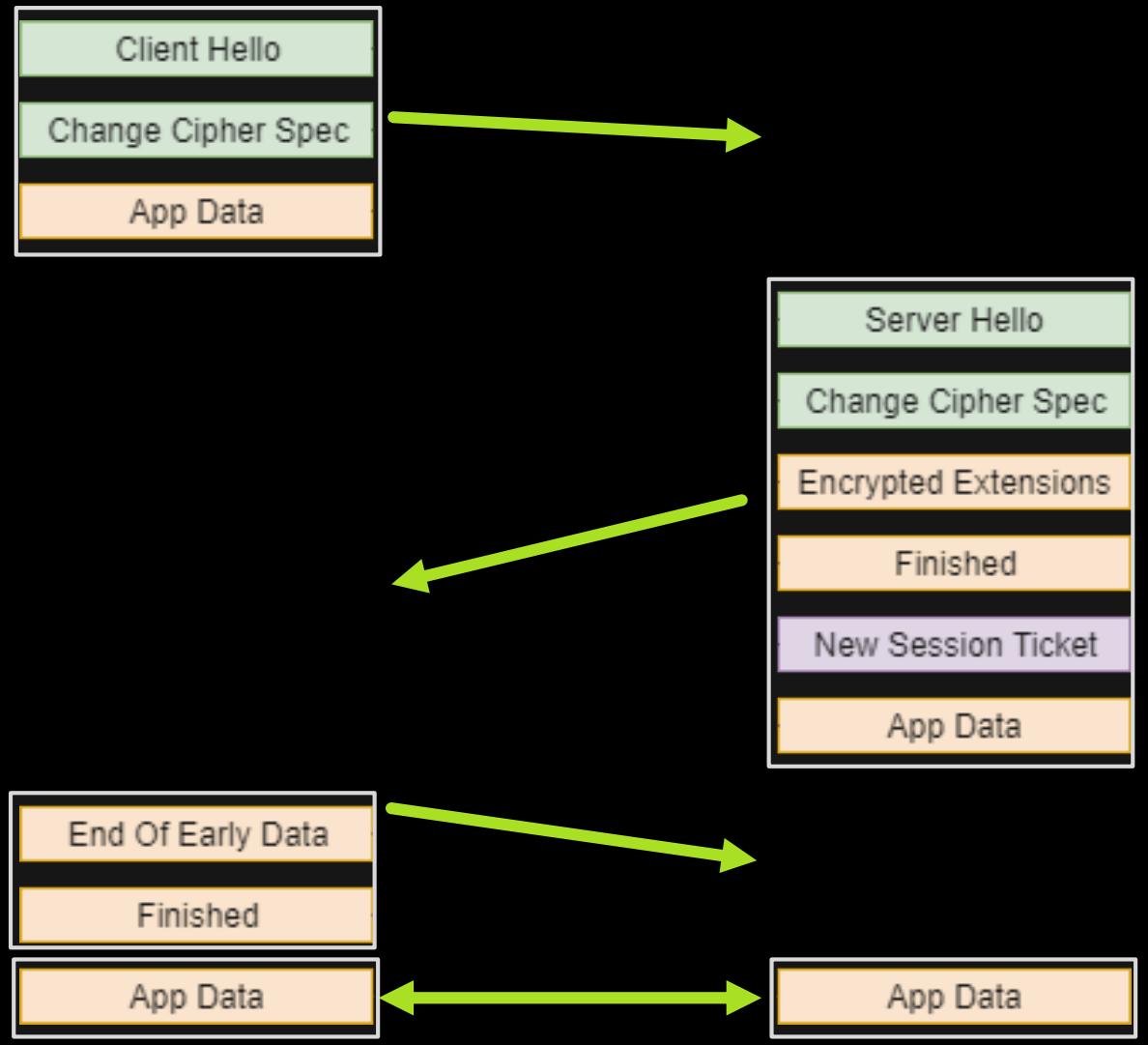
NGINX

… and CDNs may already be supporting TLS 1.3 0-RTT!

TLS 1.3 HANDSHAKE



TLS 1.3 0-RTT


Client Hello
Change Cipher Spec
App Data
Server Hello
Change Cipher Spec
Encrypted Extensions
Finished
New Session Ticket
App Data
End Of Early Data
Finished
App Data
App Data

As you can see…

it may be possible to do **REPLAY**

REPLAY

REPLAY

REPLAY

REPLAY attacks!

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

Client-Hello Recording

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

Application profiles

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

Application profiles

Separate API

ANTI-REPLAY PROTECTIONS (JUL-2018)

	Single-Use TIckets	Client-Hello Recording	"Freshness"	Application Profile	Other protections
OpenSSL Cryptography and SSL/TLS Toolkit	✓				Different API for handling 0-RTT
BoringSSL					0-RTT without protections
CDN [1]					0-RTT only on "safe" methods, no params
					0-RTT not available
					0-RTT only on "safe" methods

ARE THOSE PROTECTIONS ENOUGH?

ANATOMY OF AN ATTACK

- **Vantage point** in the network
- Browser and server with TLS 1.3 and **0-RTT enabled**
- GET not being a "**safe method**" (a.k.a. RFC meets reality)

IMPROVING OUR ATTACK

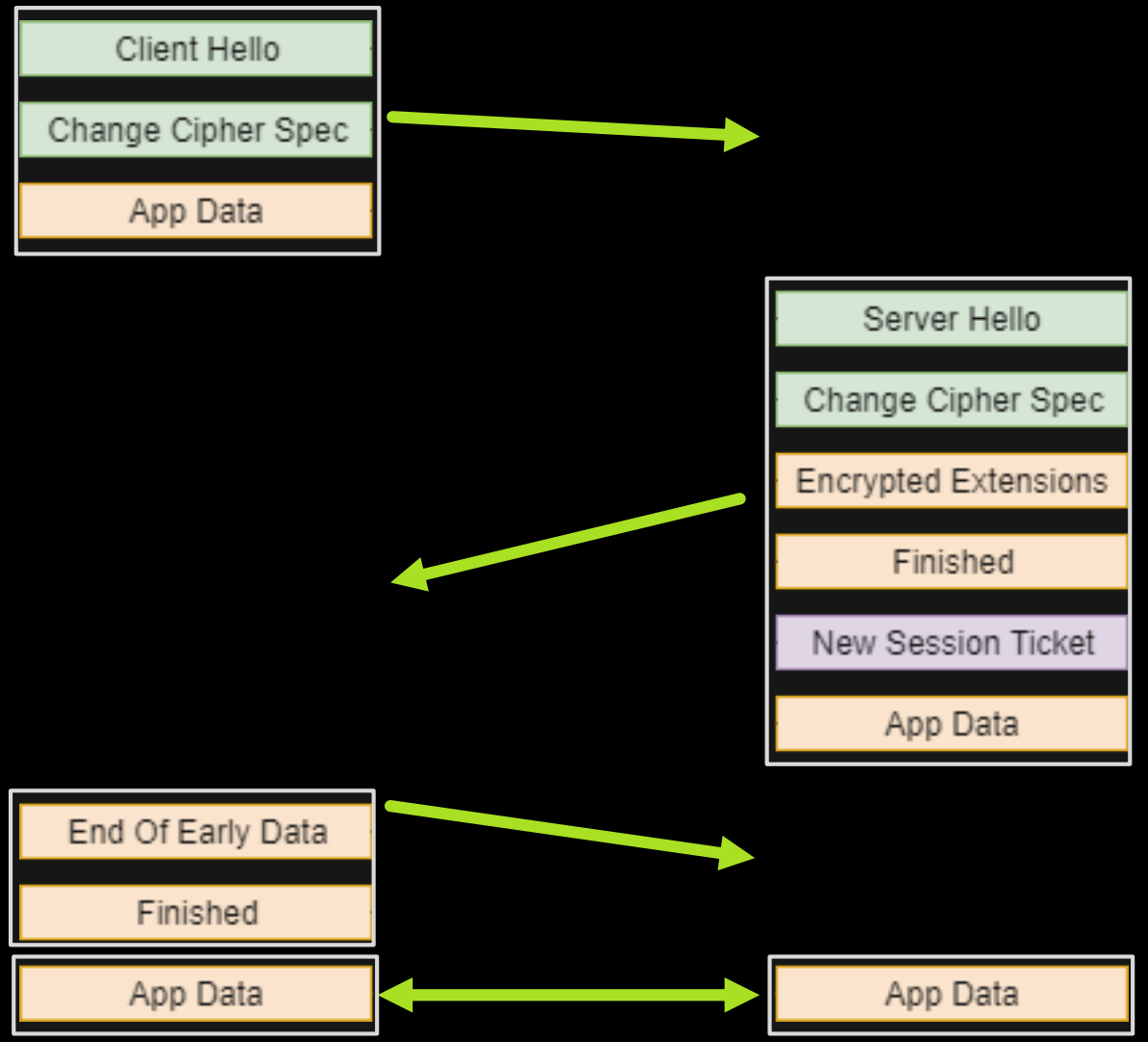
- **The browser decides** when to send 0-RTT data, which reduces the window for attacks
- Could it be possible to control when to send 0-RTT data?

IMPROVING OUR ATTACK

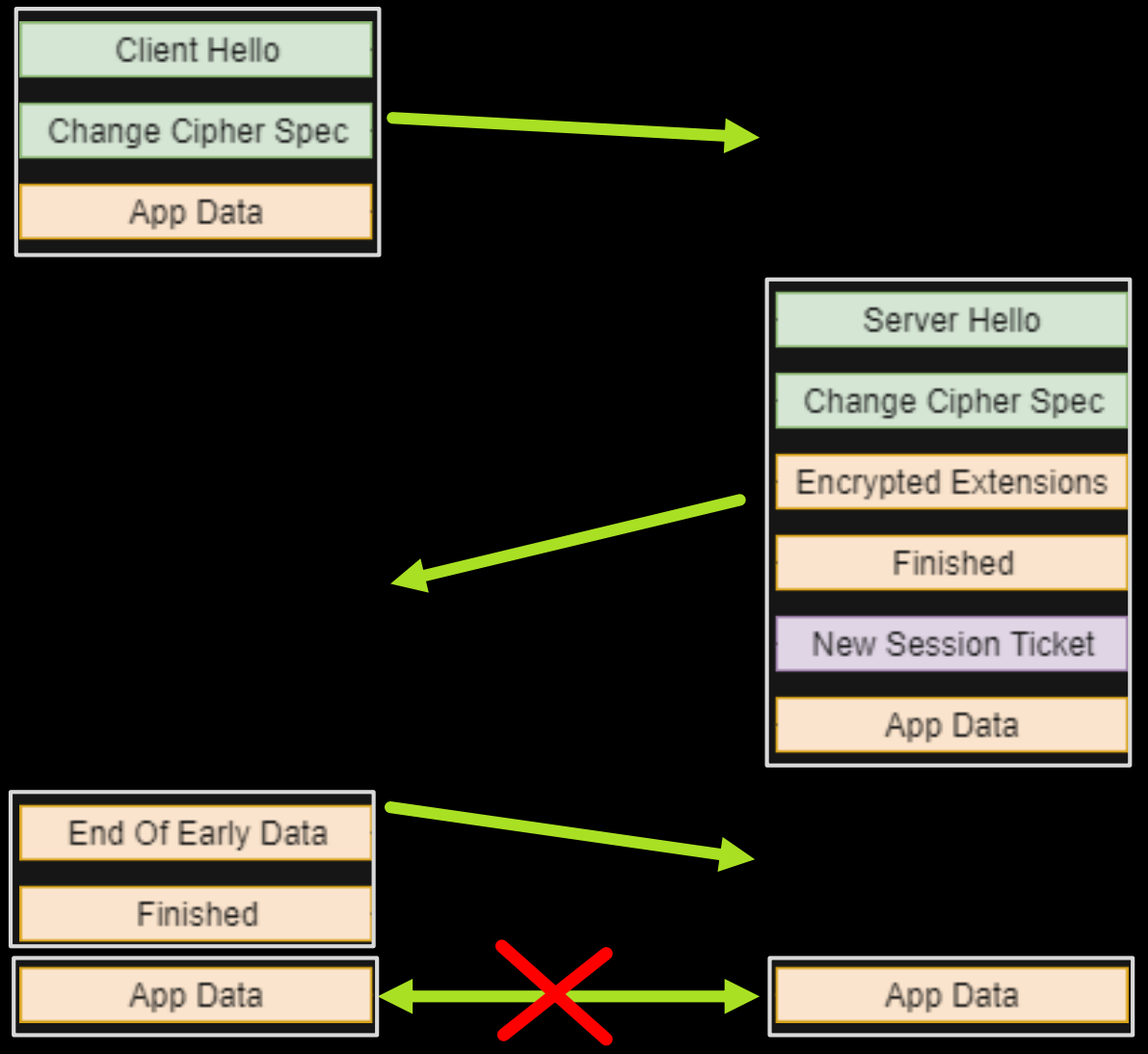
- **The browser decides** when to send 0-RTT data, which reduces the window for attacks
- Could it be possible to control when to send 0-RTT data?

YES!!!

CONTROLLING THE BROWSER



CONTROLLING THE BROWSER



DEMO

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

Application profiles

Separate API

IMPROVING OUR ATTACK (AGAIN)

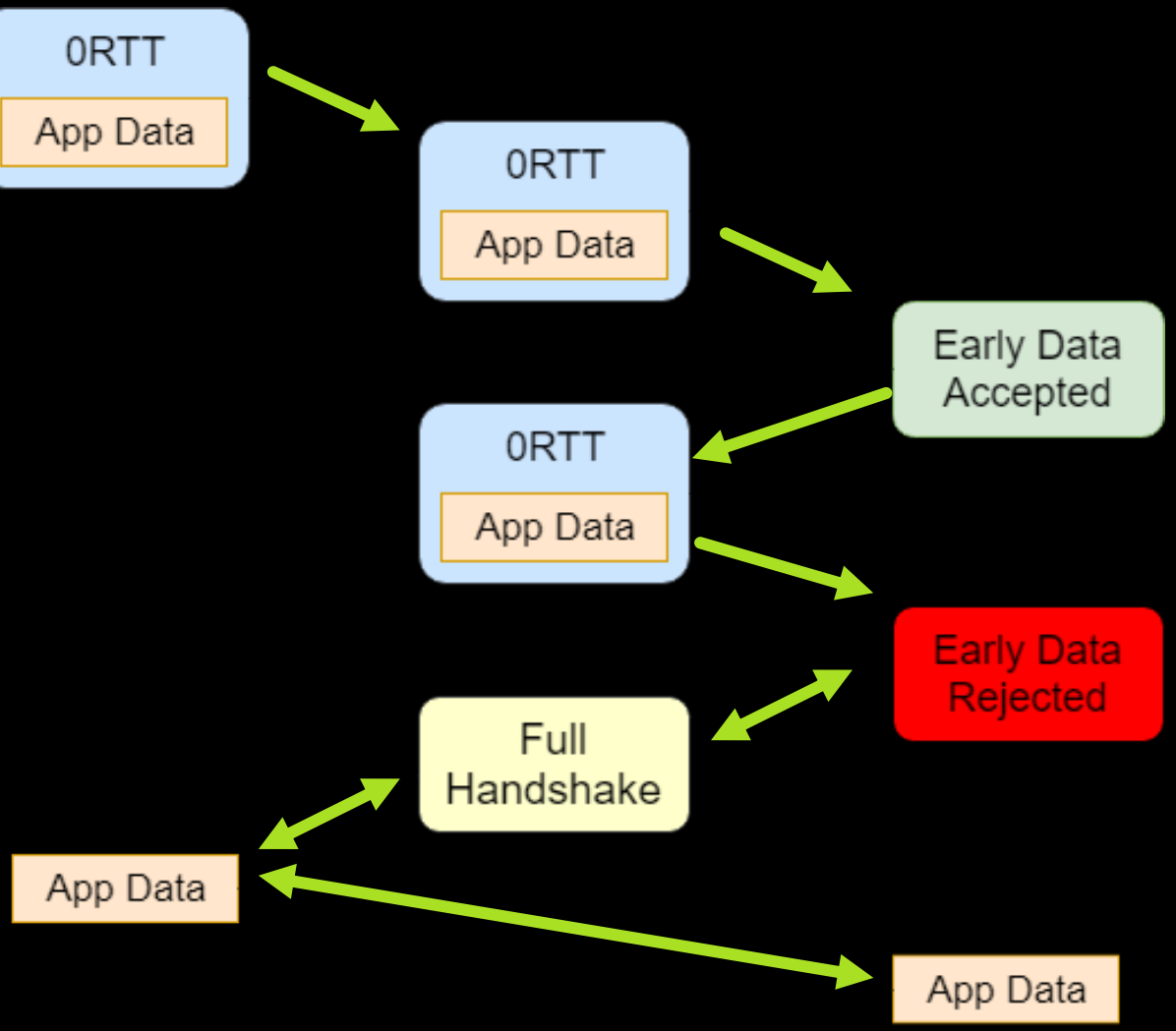
- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT.
- Could it be possible to do replay attacks?

IMPROVING OUR ATTACK (AGAIN)

- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT.
- Could it be possible to do replay attacks?

YES!!!

UNIVERSAL REPLAY ATTACK



DEMO

TOOL: HIGH-LEVEL DESCRIPTION

- Assumes a vantage point in the network
- Provides creation of templates for encrypted traffic.
- Supports the two attacks described on this presentation.
- It has support for three modes:
  - Mode monitor
  - Active - No protections
  - Active - Protections
- Available at https://github.com/portcullislabs/tlsplayback

SIDE EFFECTS OF 0-RTT

- It is important to understand that 0-RTT creates a **dependency between the application and** the underlying TLS 1.3 **protocol**
- The application will need to be **0-RTT aware**.
- Enabling 0-RTT could leave you application vulnerable to **replay attacks**
- Ultimately, the **last line of defence** would be the application itself.

MITIGATIONS

- **Disable 0-RTT**
- Ensure that your application does not allow replays (e.g. strict **CSRF**). Ensure that REST services are developed properly
- Create an strict **application profile** after careful analysis

KEY TAKEAWAYS

- **TLS 1.3 is awesome**, but could led to a vulnerable application if 0-RTT is being used.
- Your application (not just webapps) needs to be **0-RTT-aware** to prevent side effects
- You may need to change your application or server/CDN configuration to **protect against replay attacks**

Thanks!