

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

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

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

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

INTRODUCING TLS 1.3

The **Bad**

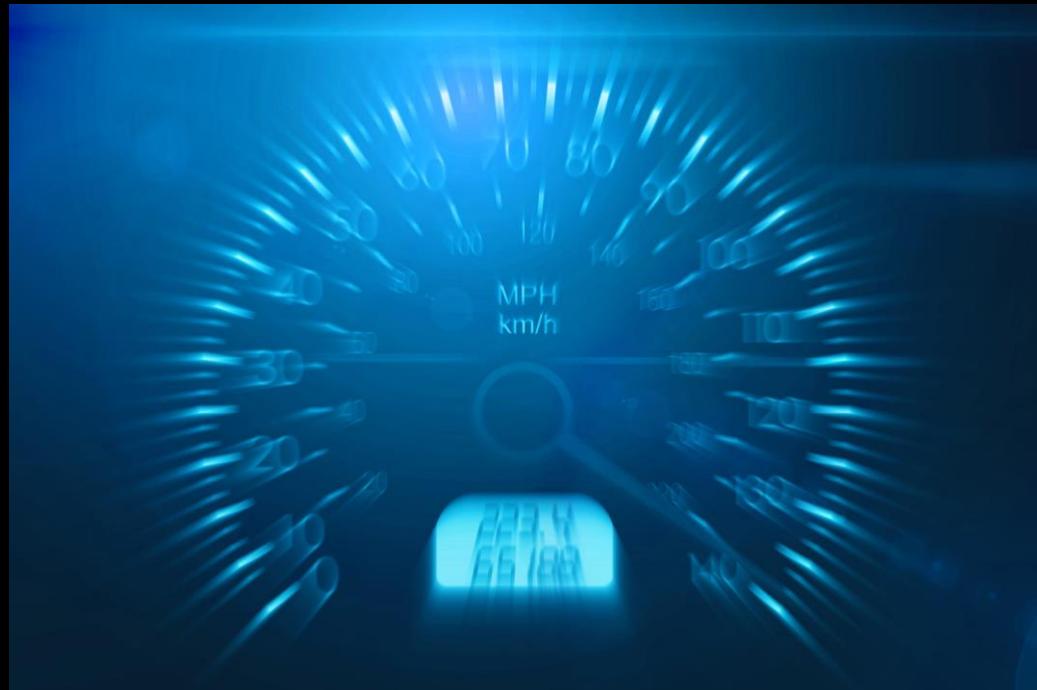
- Protocol tainted due to “compatibility issues” 😞

INTRODUCING TLS 1.3

The **Ugly**

- 0-RTT (this talk ☺)

0-RTT: SPEED AT A COST



vs

 Secure | <https://>

TELL ME AGAIN...

WHY SHOULD I CARE?

Your browsers...

The image shows three browser windows side-by-side:

- Chrome**: chrome://flags search results for "tls". A blue bar highlights the "Available" section. Below it, a yellow box highlights the "TLS 1.3" section, which includes a description and a link to "#tls13-variant".
- Firefox**: about:config search results for "security.tls". A table lists three preferences:

Preference Name	Status	Value
security.tls.enable_0rtt_data	default	true
security.tls.version.max	default	4
security.tls.version.min	default	1
- Edge**: Edge DevTools Network tab showing a request to "https://www.google.com". The status bar at the bottom indicates "TLS 1.3".

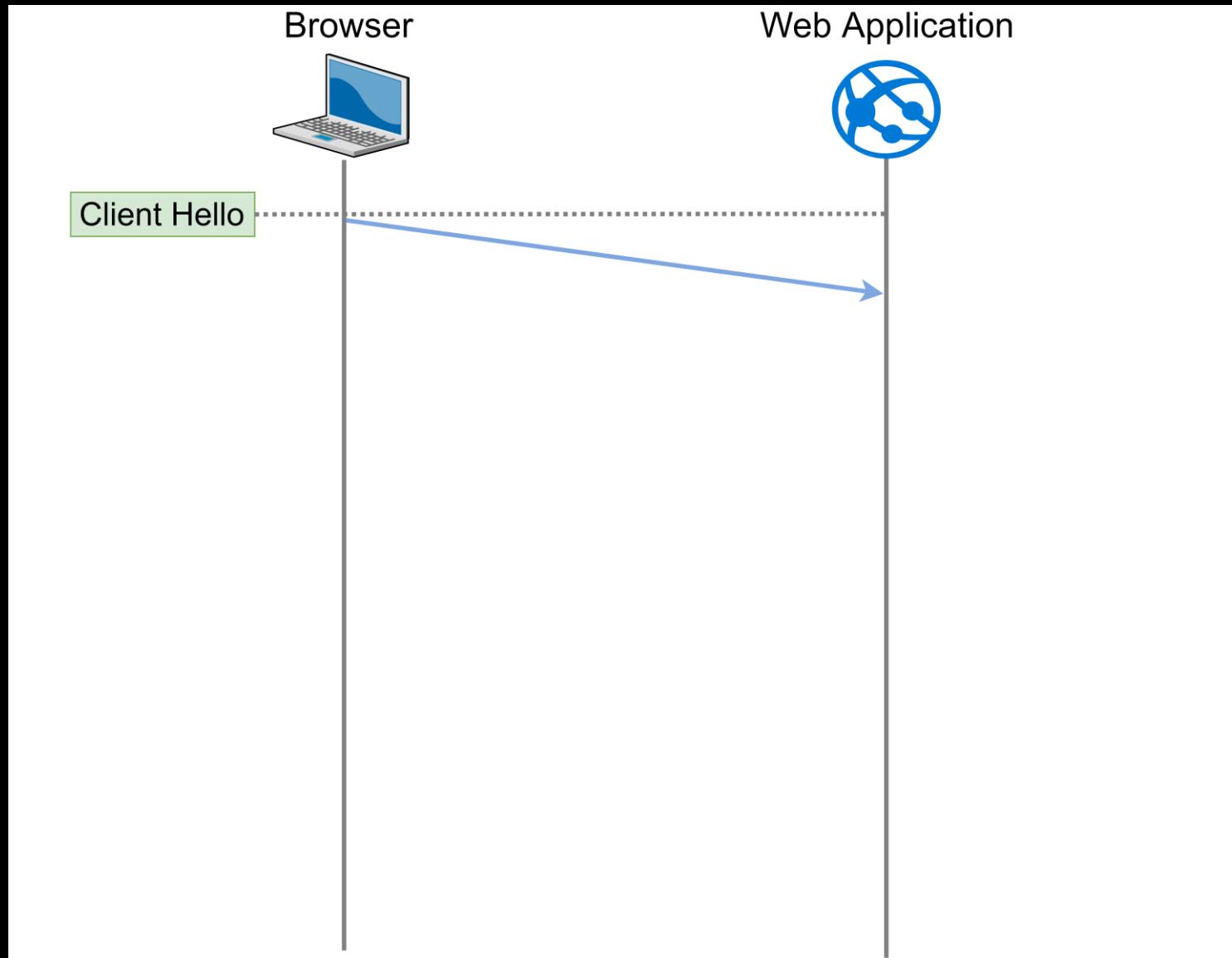
... implementations ...



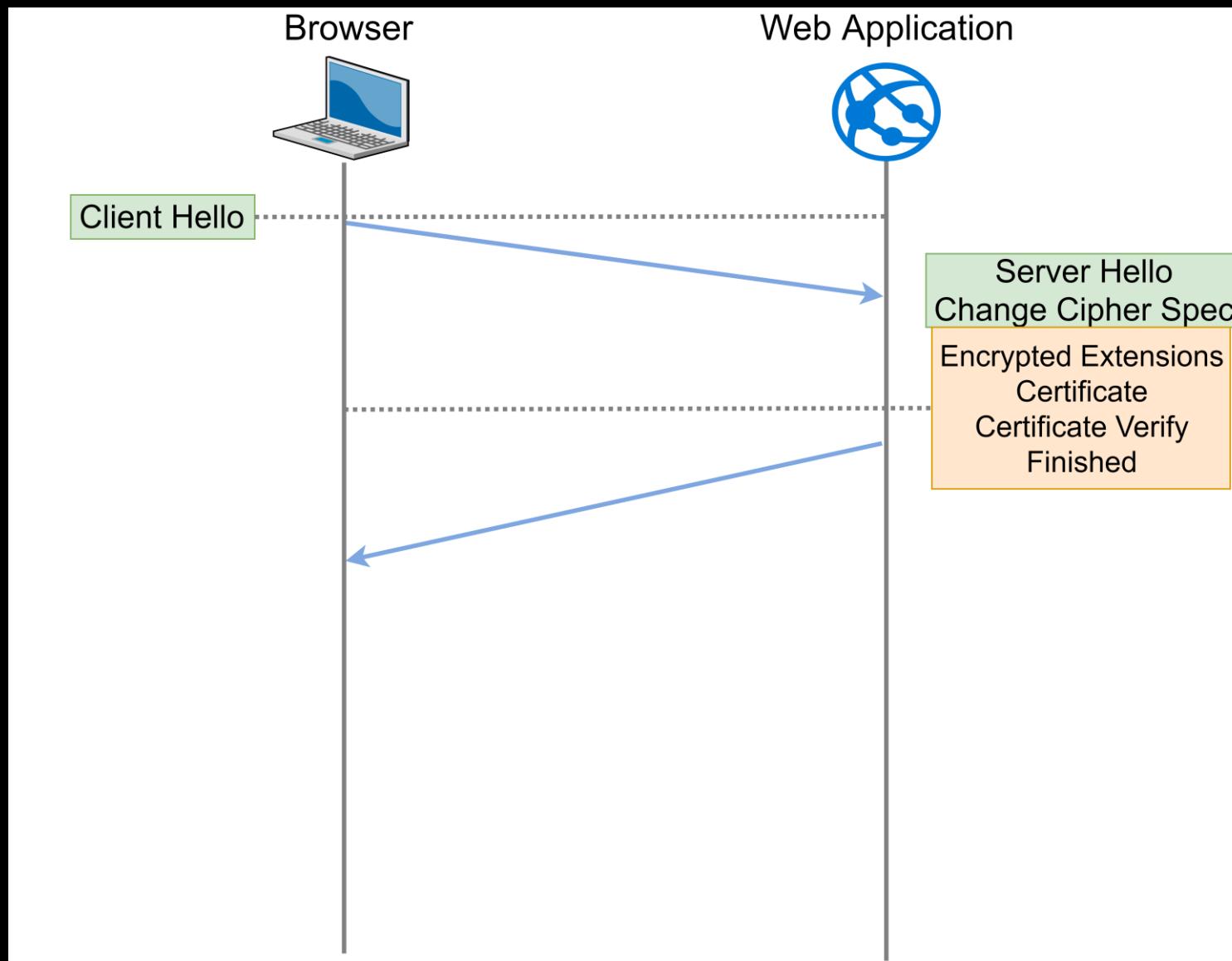
BoringSSL

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

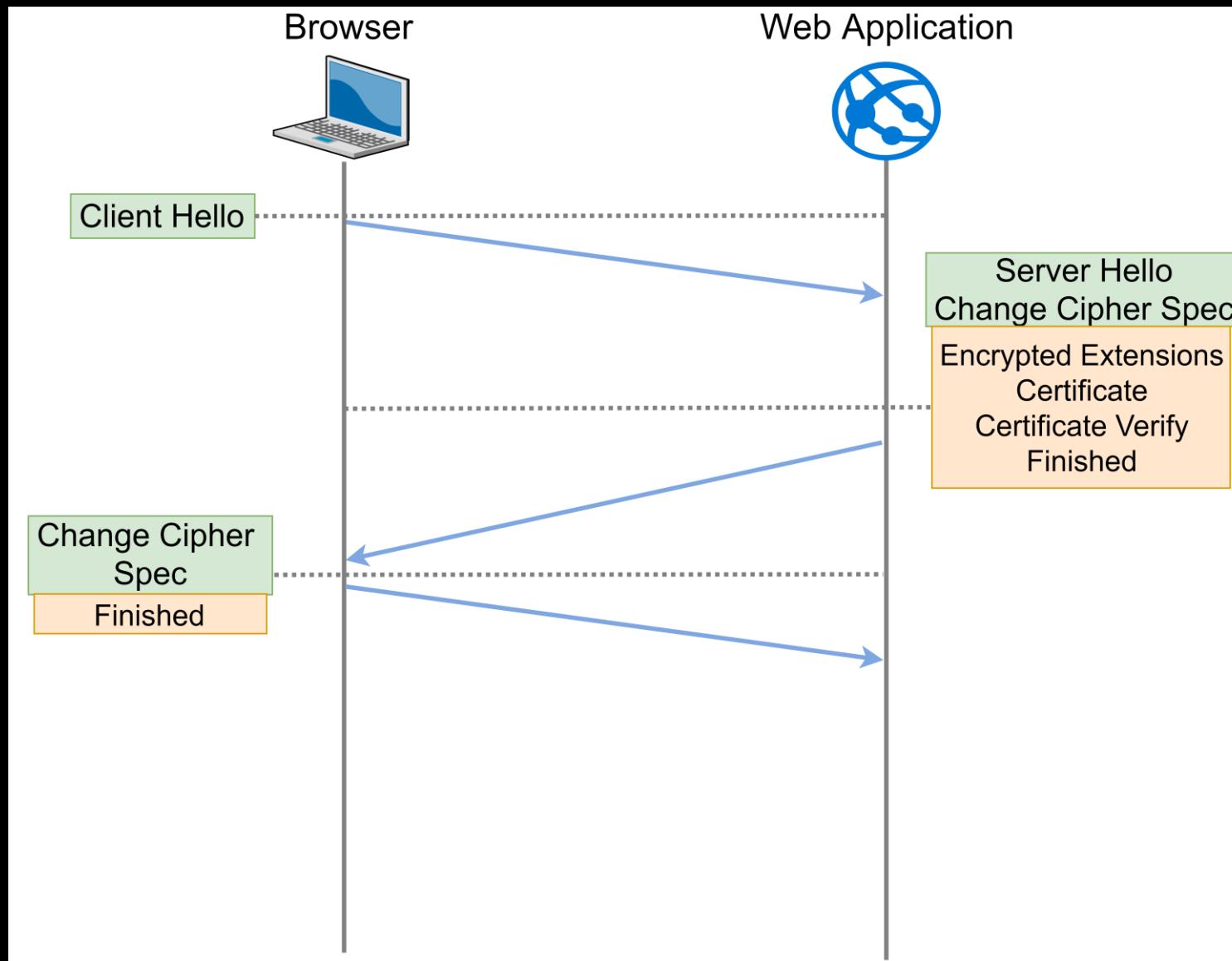
TLS 1.3 HANDSHAKE



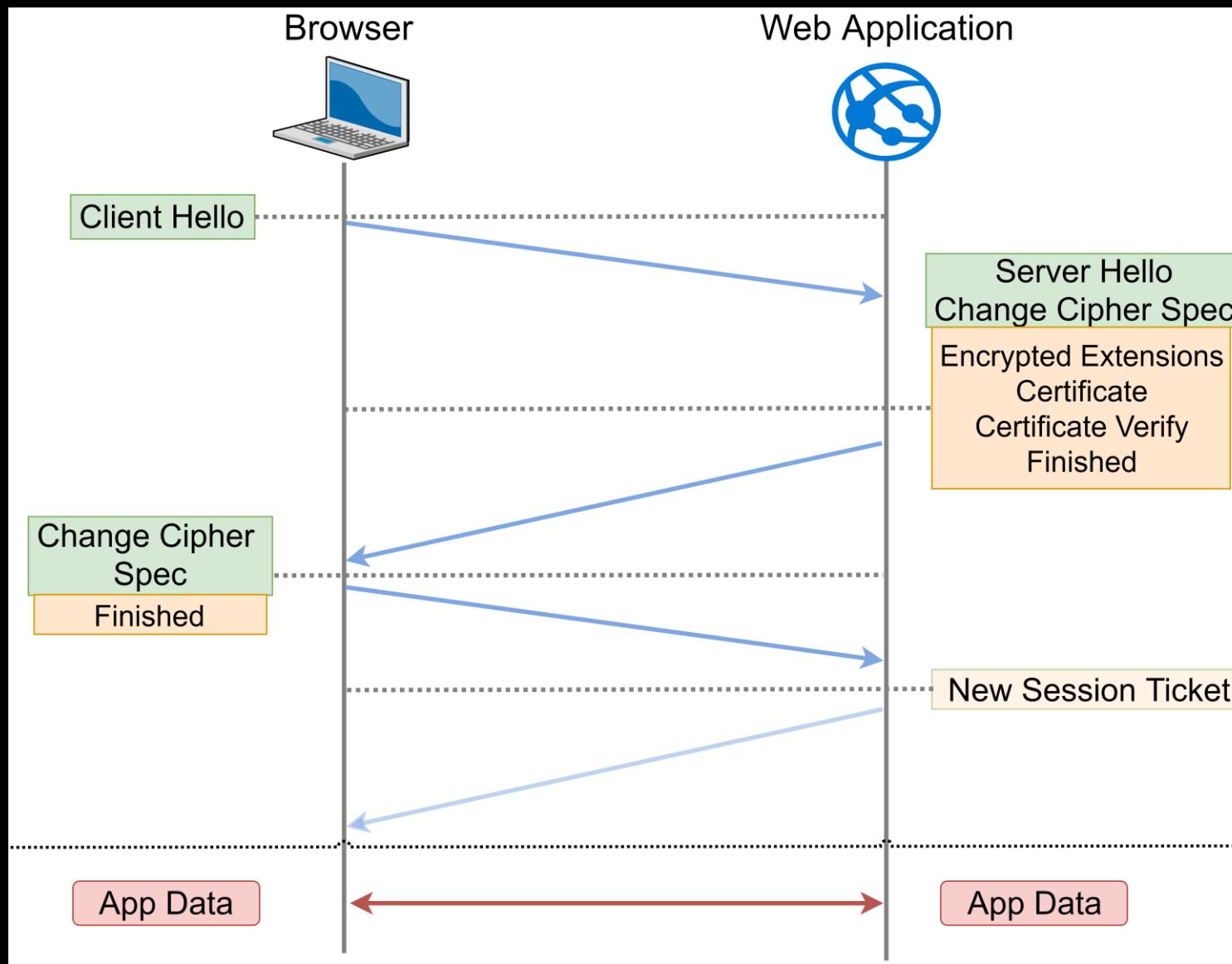
TLS 1.3 HANDSHAKE



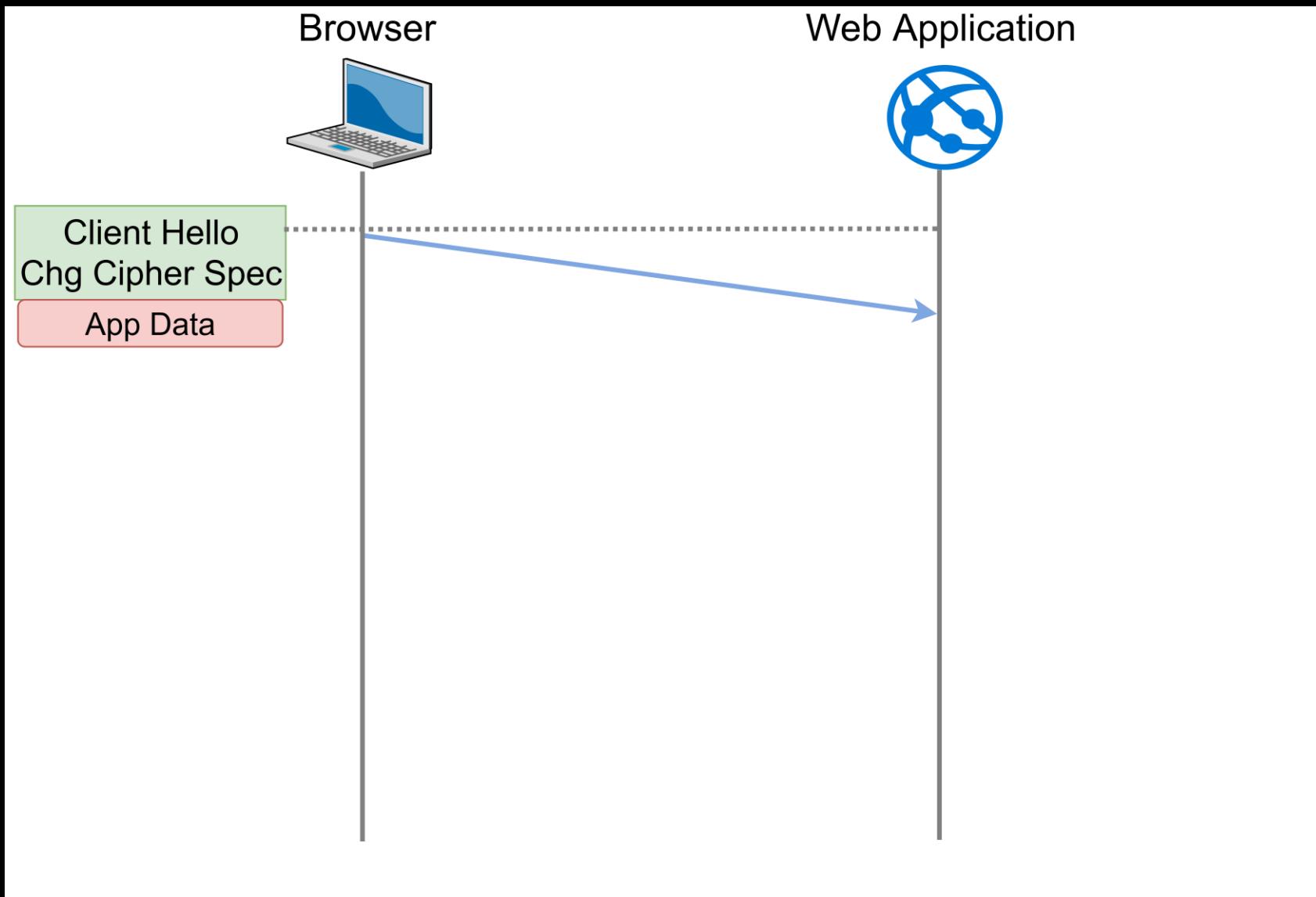
TLS 1.3 HANDSHAKE



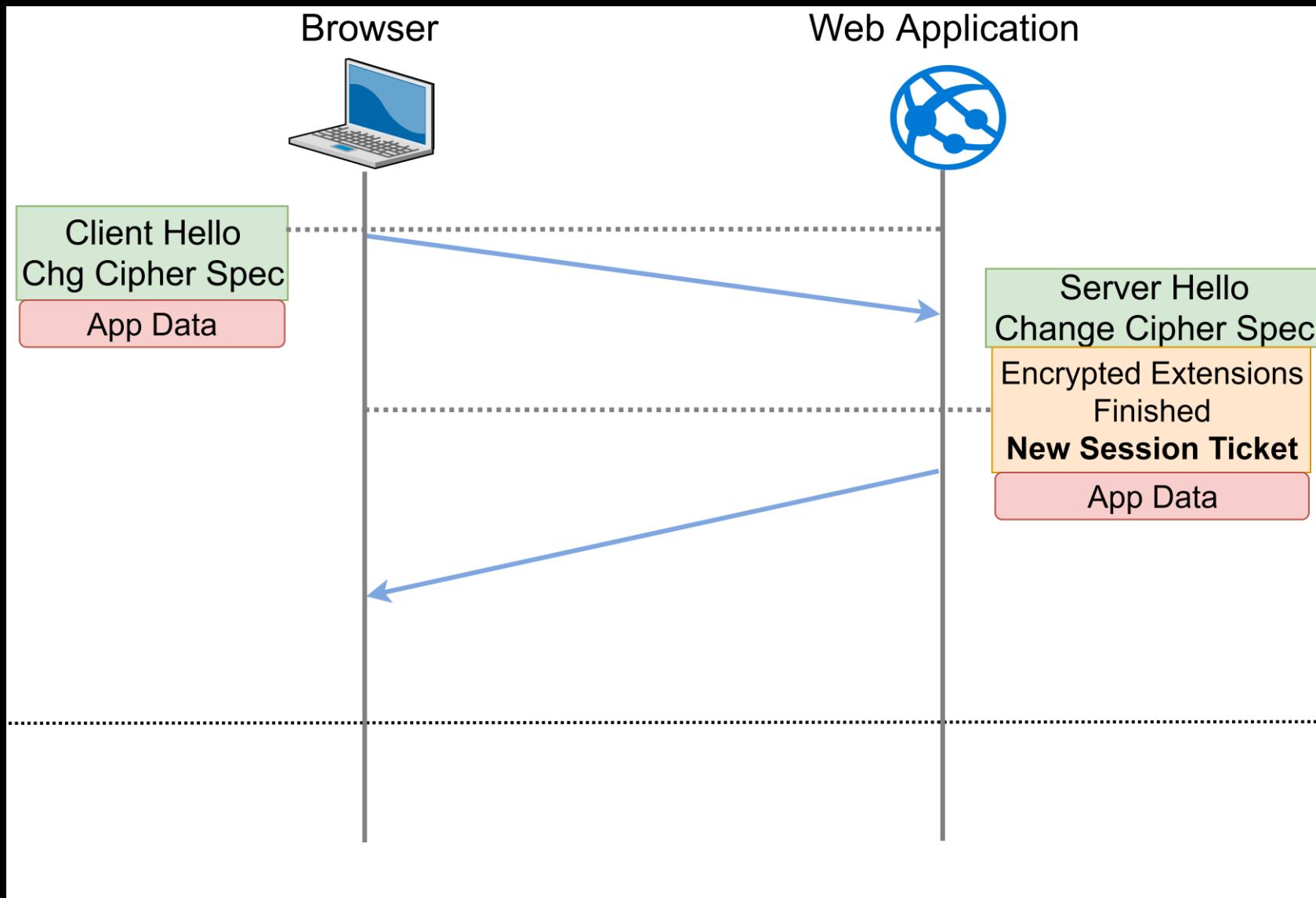
TLS 1.3 HANDSHAKE



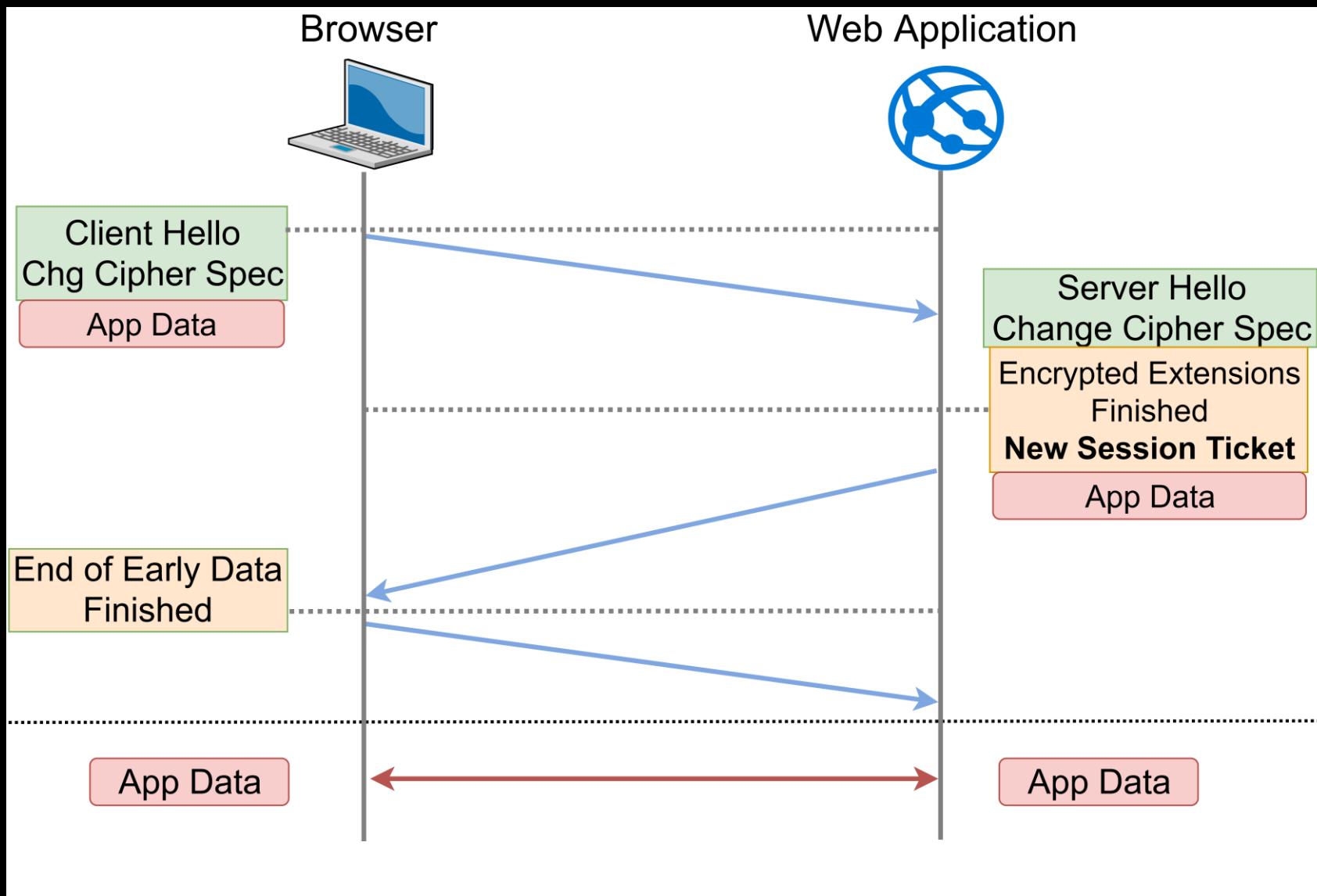
TLS 1.3 0-RTT



TLS 1.3 0-RTT



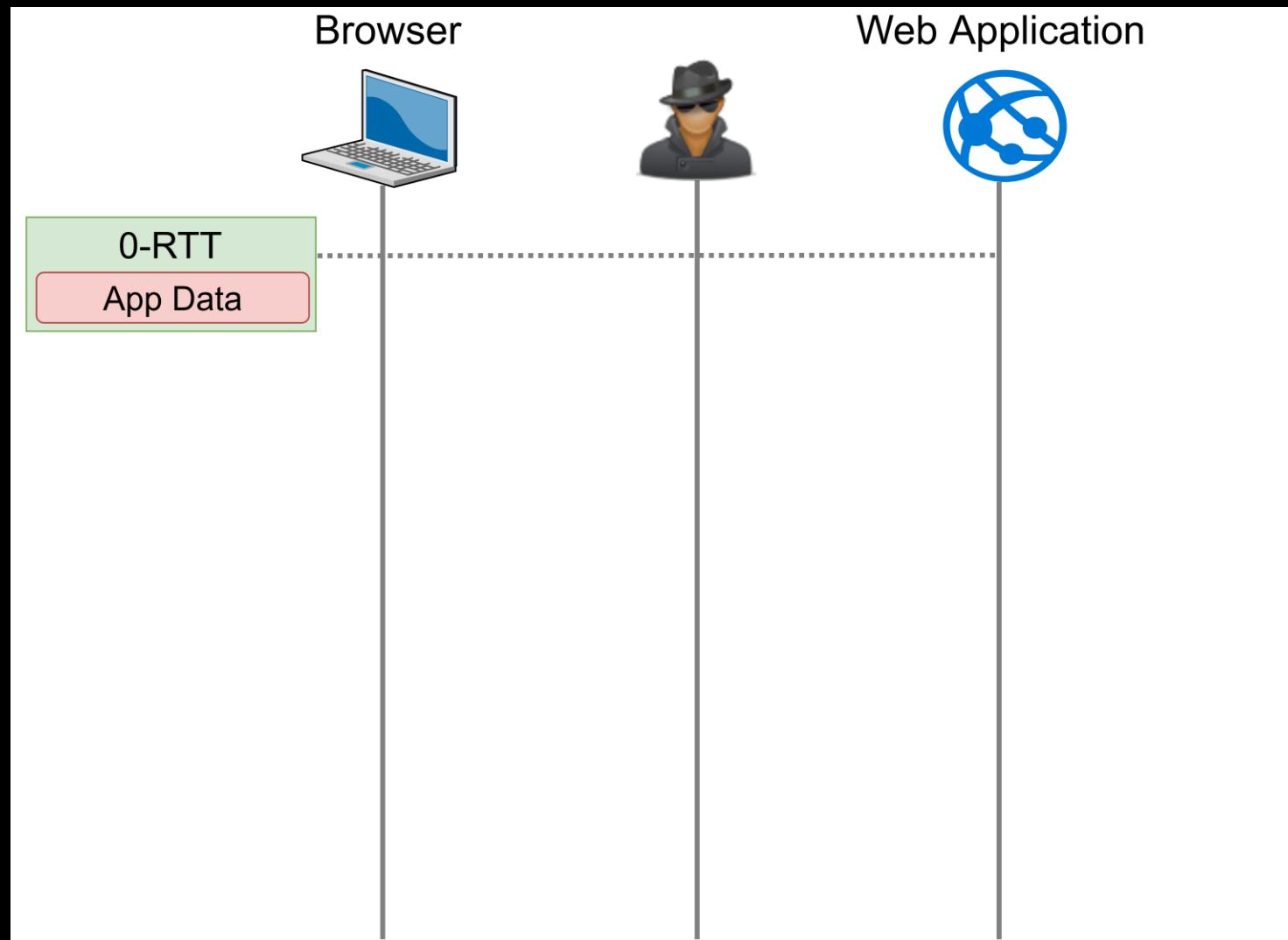
TLS 1.3 0-RTT



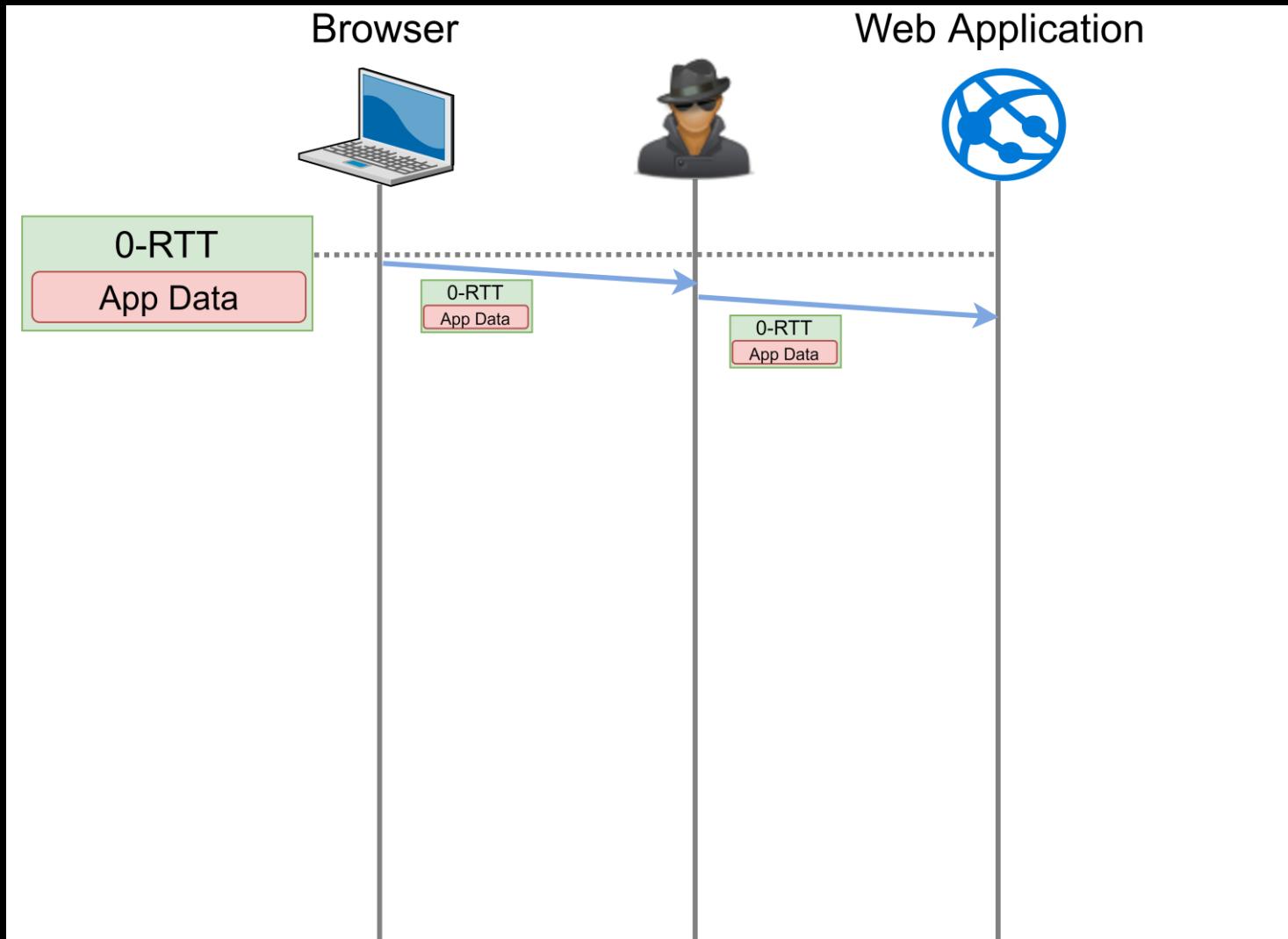
As you can see...

it may be possible to do **REPLAY**
REPLAY
REPLAY
REPLAY
REPLAY attacks!

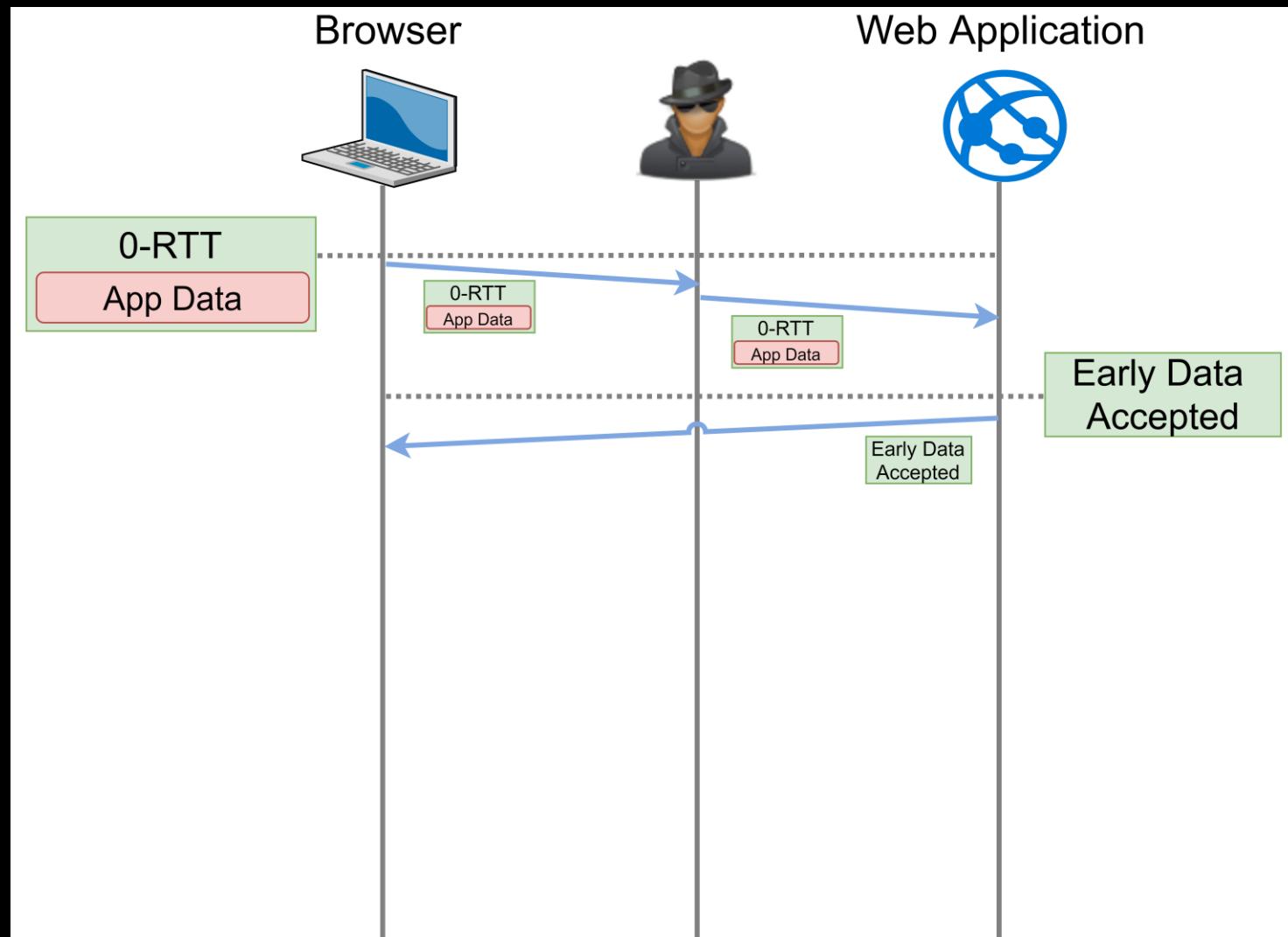
TLS 1.3 0-RTT REPLAY



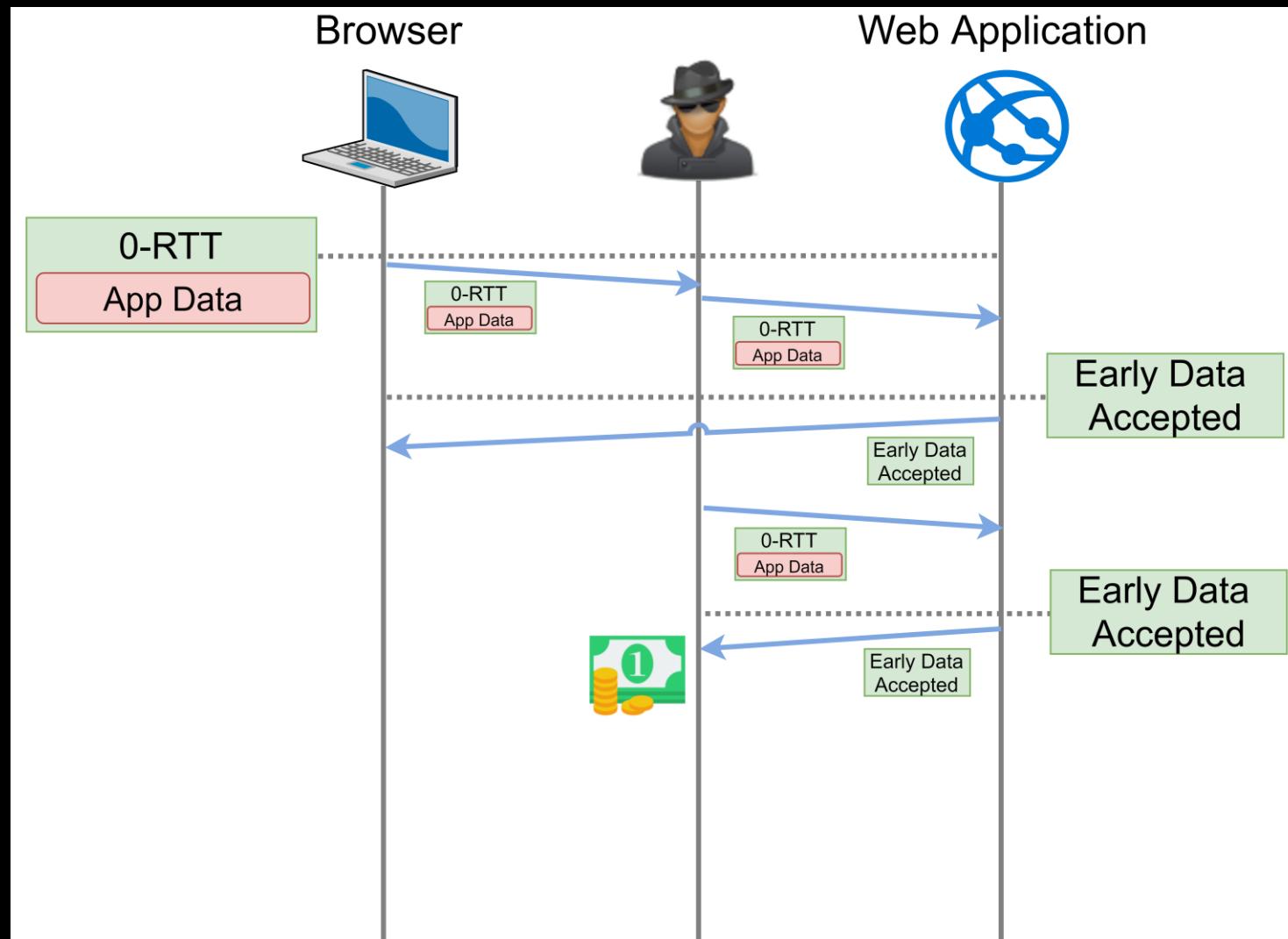
TLS 1.3 0-RTT REPLAY



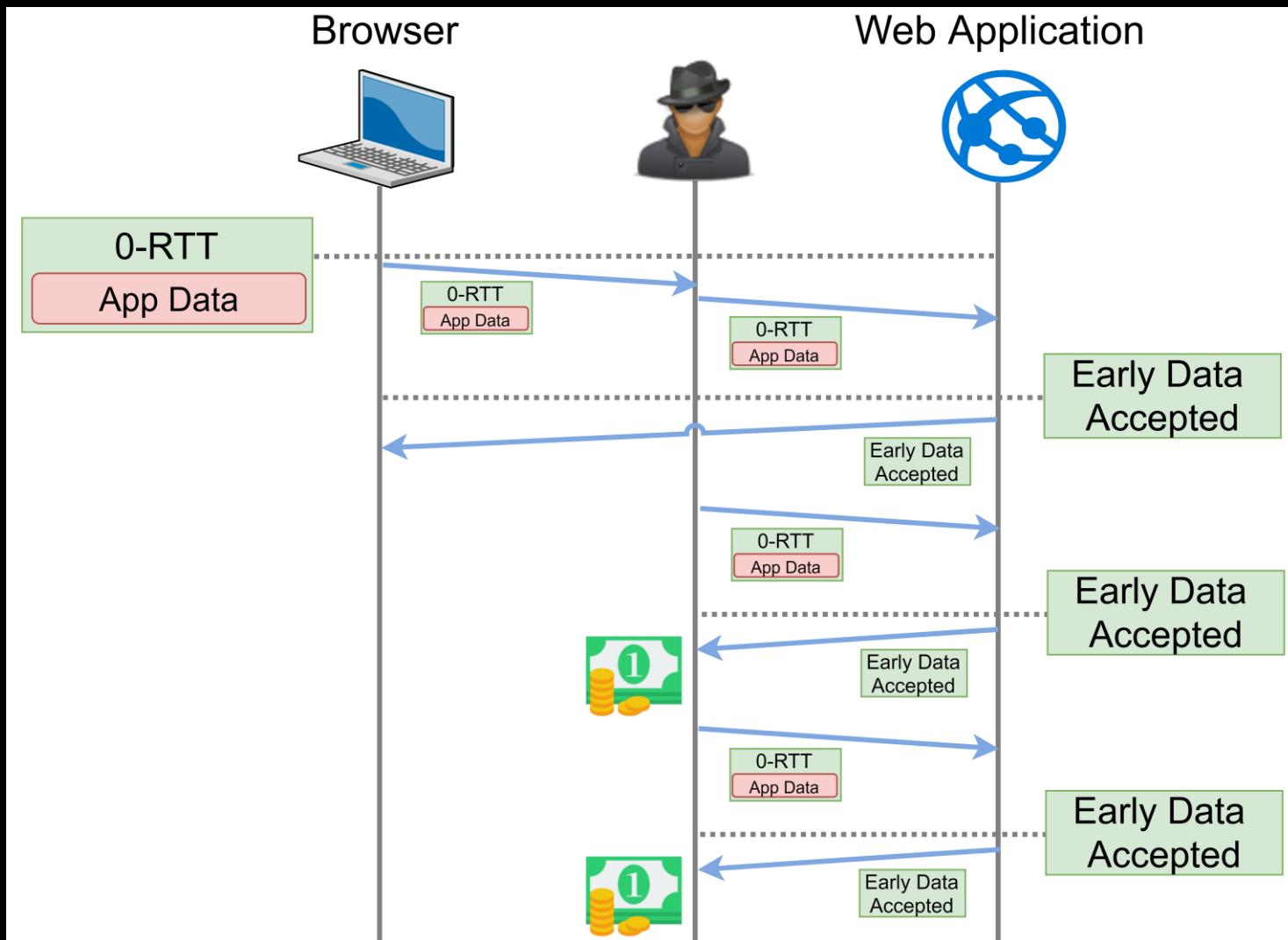
TLS 1.3 0-RTT REPLAY



TLS 1.3 0-RTT REPLAY



TLS 1.3 0-RTT REPLAY



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

“Freshness” checks

Client-Hello Recording

Application profiles

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

“Freshness” checks

Separate API

Client-Hello Recording

Application profiles

ANTI-REPLAY PROTECTIONS (JUL-2018)

	Single-Use Tickets	Client-Hello Recording	Application Profile	Other protections
OpenSSL Cryptography and SSL/TLS Toolkit	✓		n/a	Different API for handling 0-RTT
BoringSSL			n/a	0-RTT without protections
	✓		Partial (HTTP Header)	0-RTT only on “safe” methods, no params
		n/a	n/a	0-RTT not available
		n/a	n/a	0-RTT only on “safe” methods

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)

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)

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)

THE BROWSER BEHAVIOUR

- The **browser decides** when to send 0-RTT data, which reduces the window for attacks

DEMO

IMPROVING OUR ATTACK

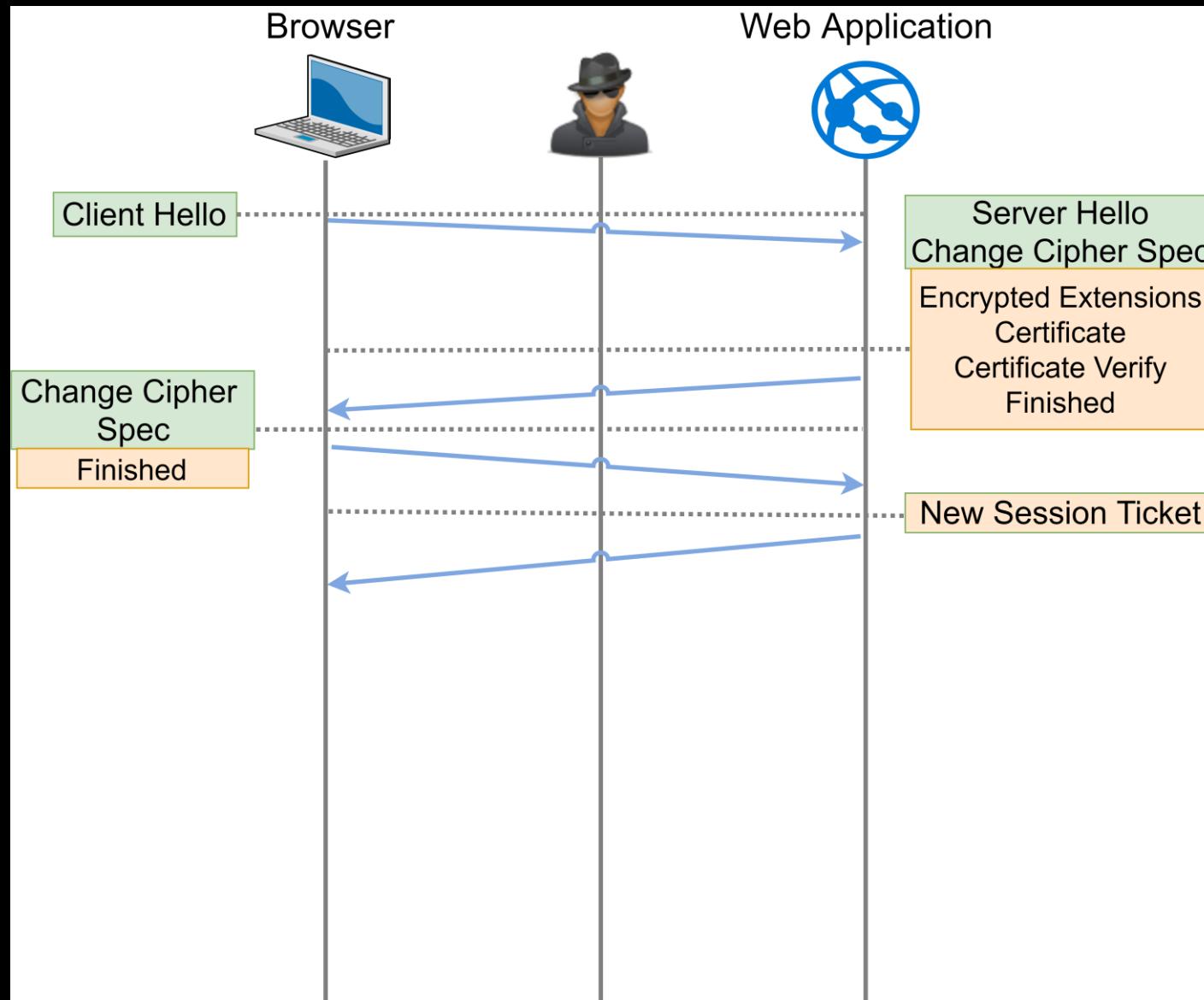
- The **browser decides** when to send 0-RTT data, which reduces the window for attacks
- Could it be possible to **control** when to send 0-RTT data?

IMPROVING OUR ATTACK

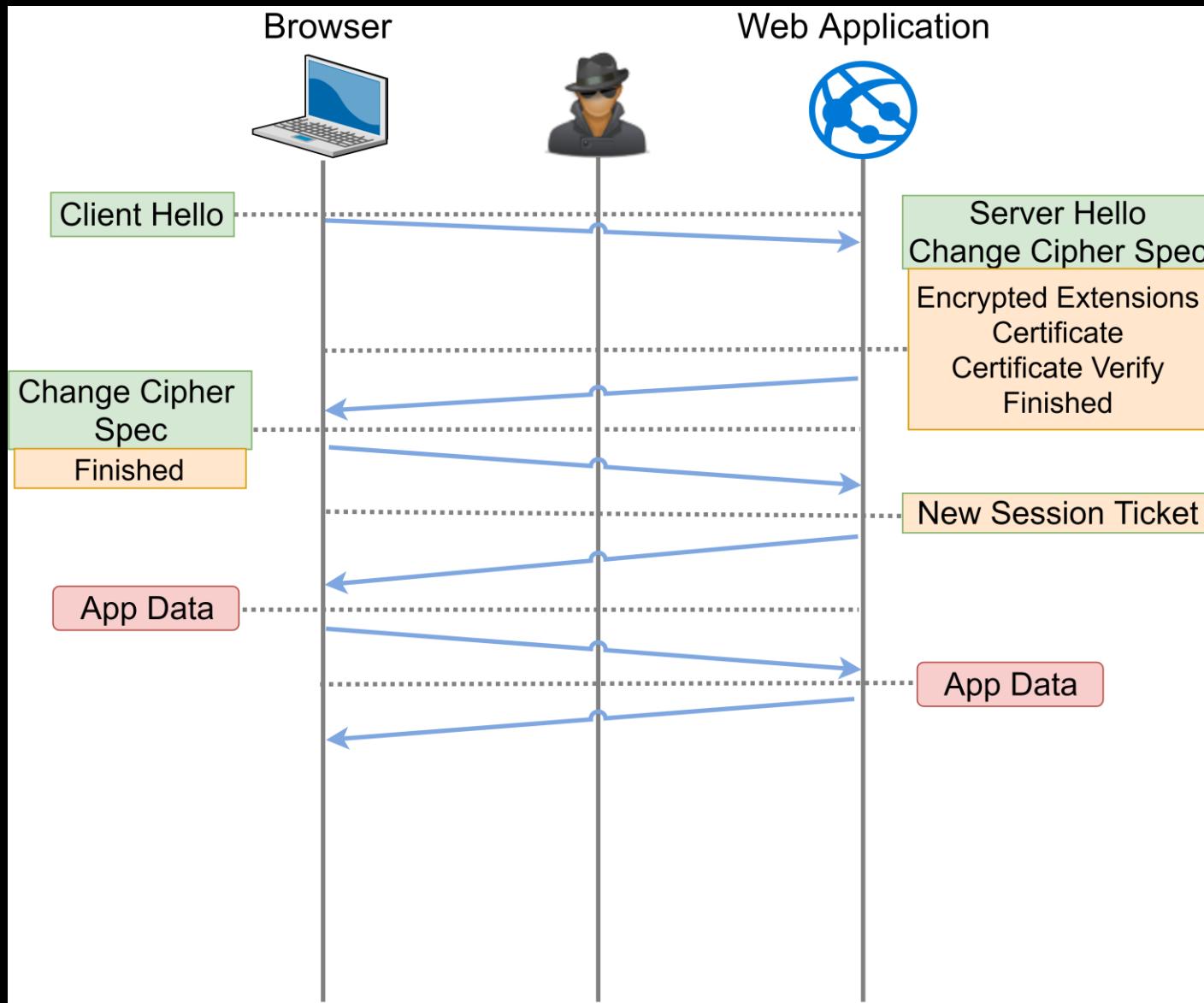
- The **browser decides** when to send 0-RTT data, which reduces the window for attacks
- Could it be possible to **control** when to send 0-RTT data?

YES!!!

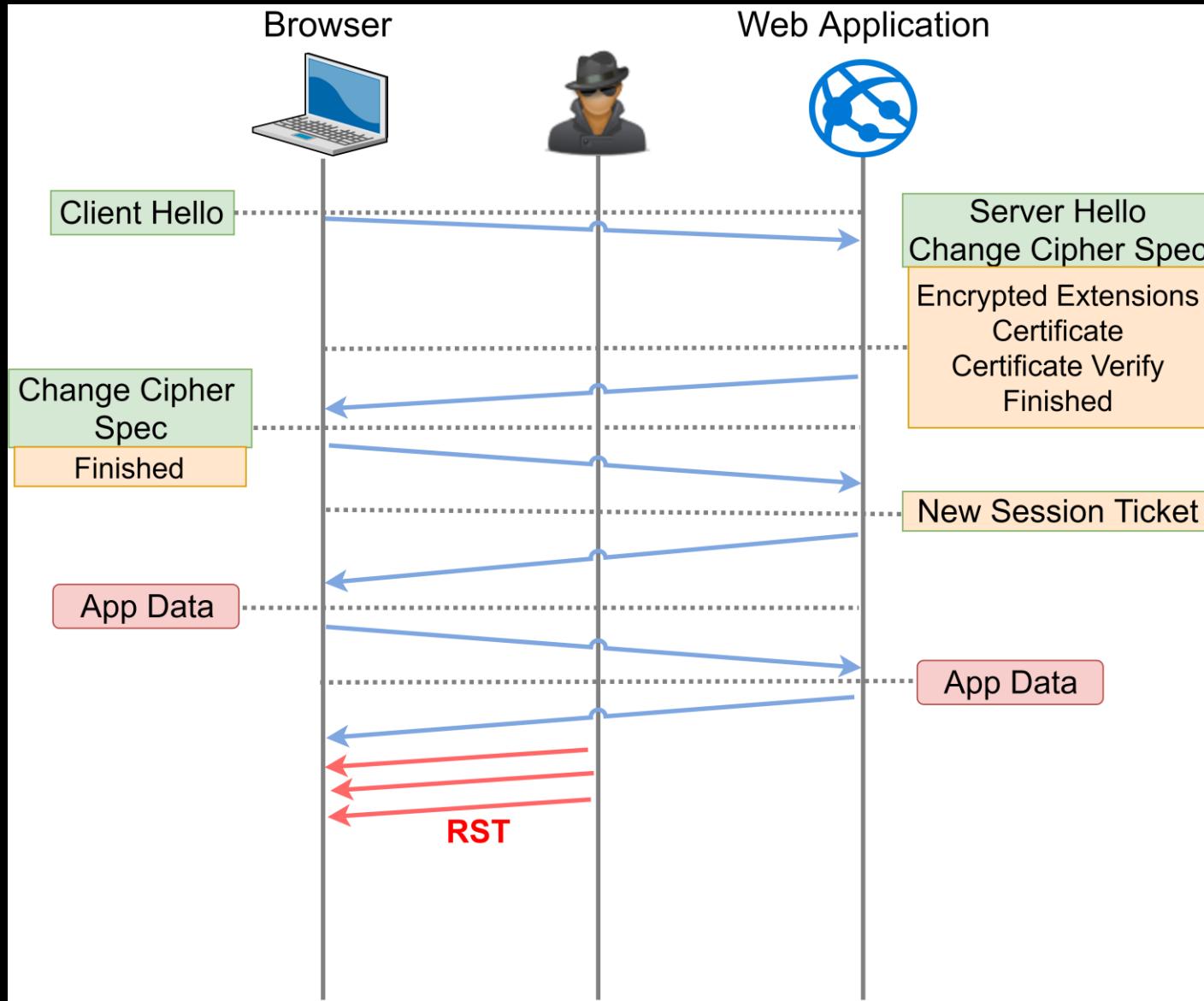
CONTROLLING THE BROWSER



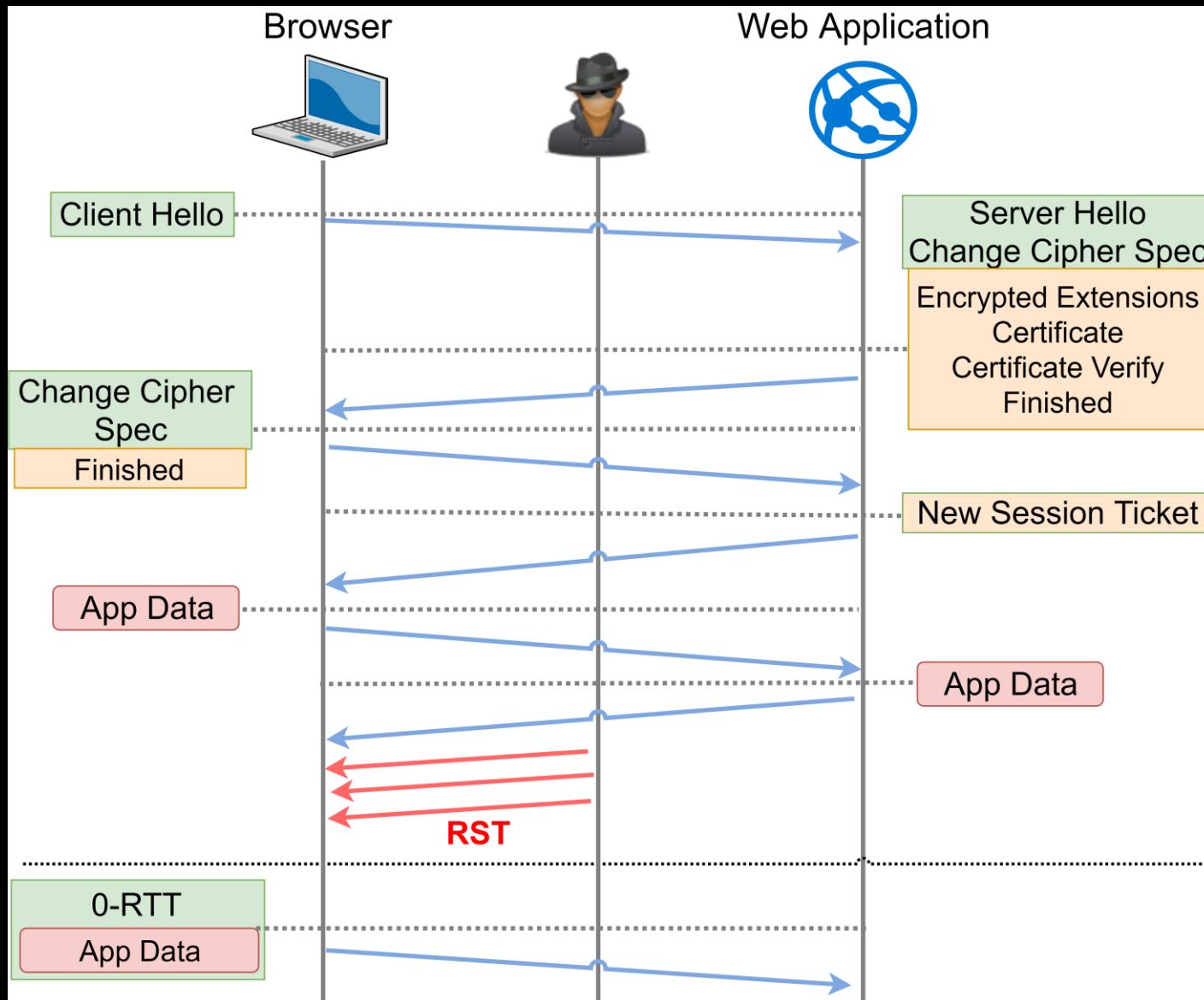
CONTROLLING THE BROWSER



CONTROLLING THE BROWSER



CONTROLLING THE BROWSER



DEMO

ANTI-REPLAY PROTECTIONS

Single-Use Tickets

“Freshness” checks

Separate API

Client-Hello Recording

Application profiles

IMPROVING OUR ATTACK (AGAIN)

- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT.

IMPROVING OUR ATTACK (AGAIN)

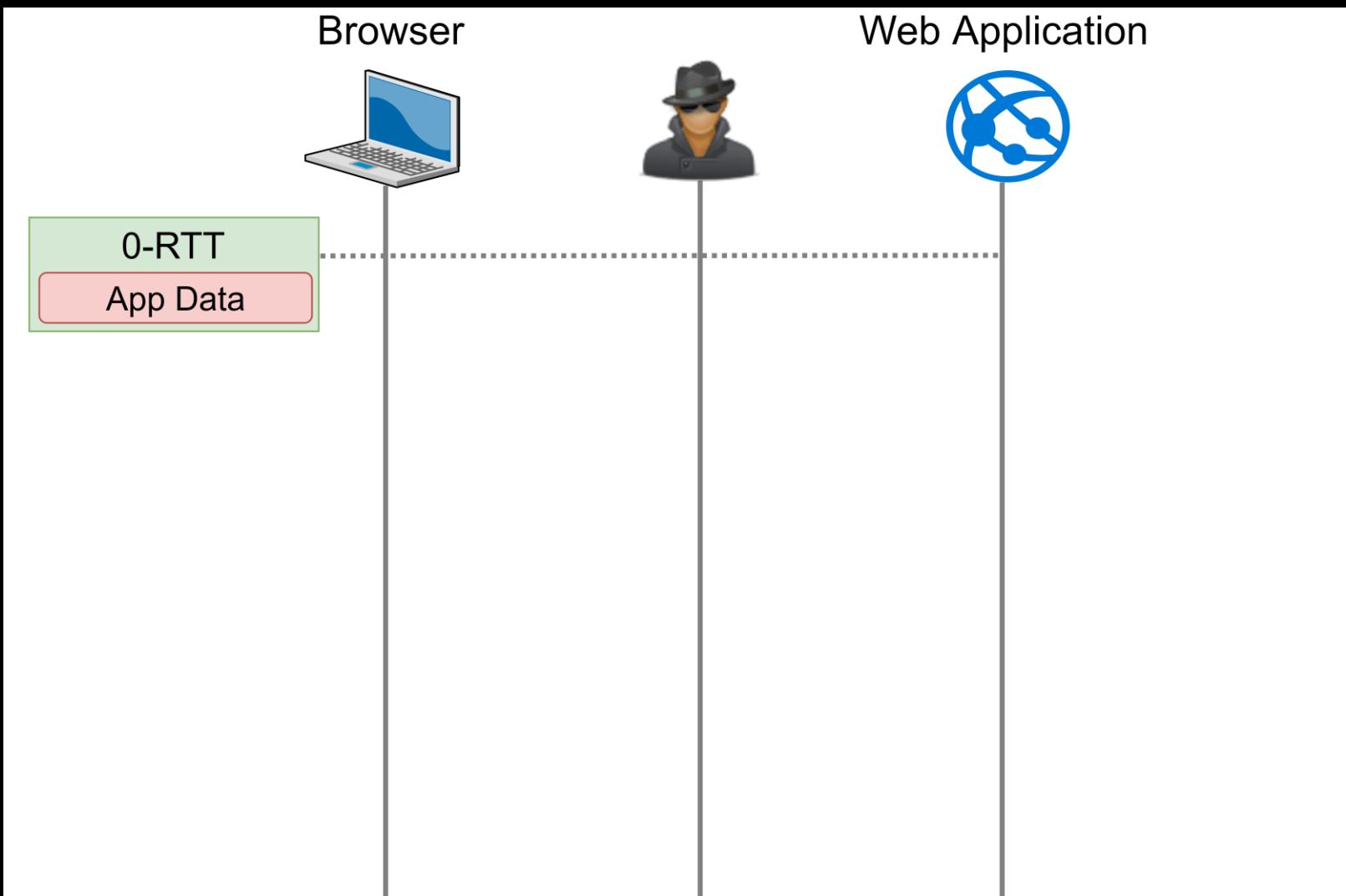
- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT.
- Could it be possible to do replay attacks?

IMPROVING OUR ATTACK (AGAIN)

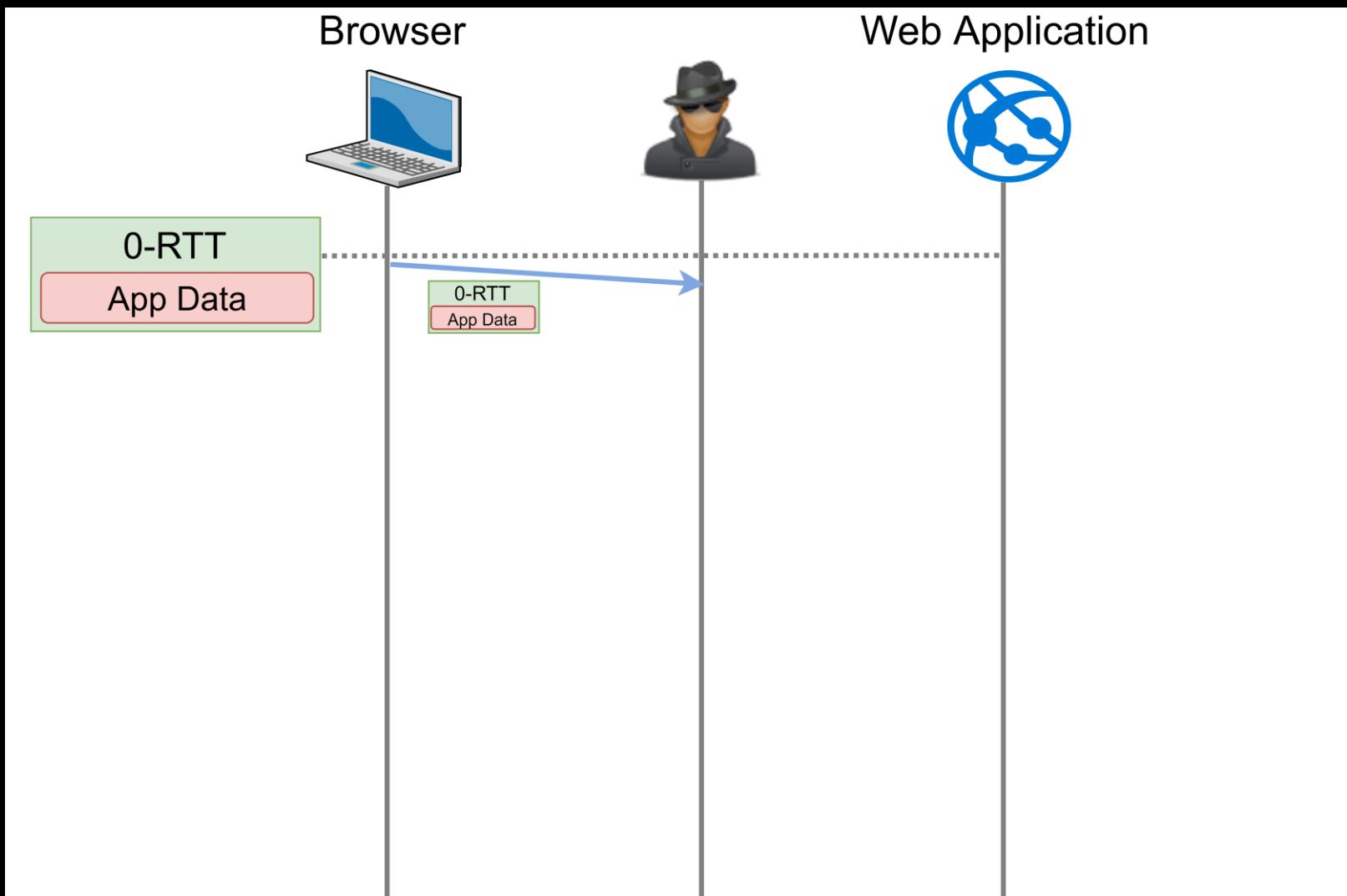
- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT.
- Could it be possible to do replay attacks?

YES!!!

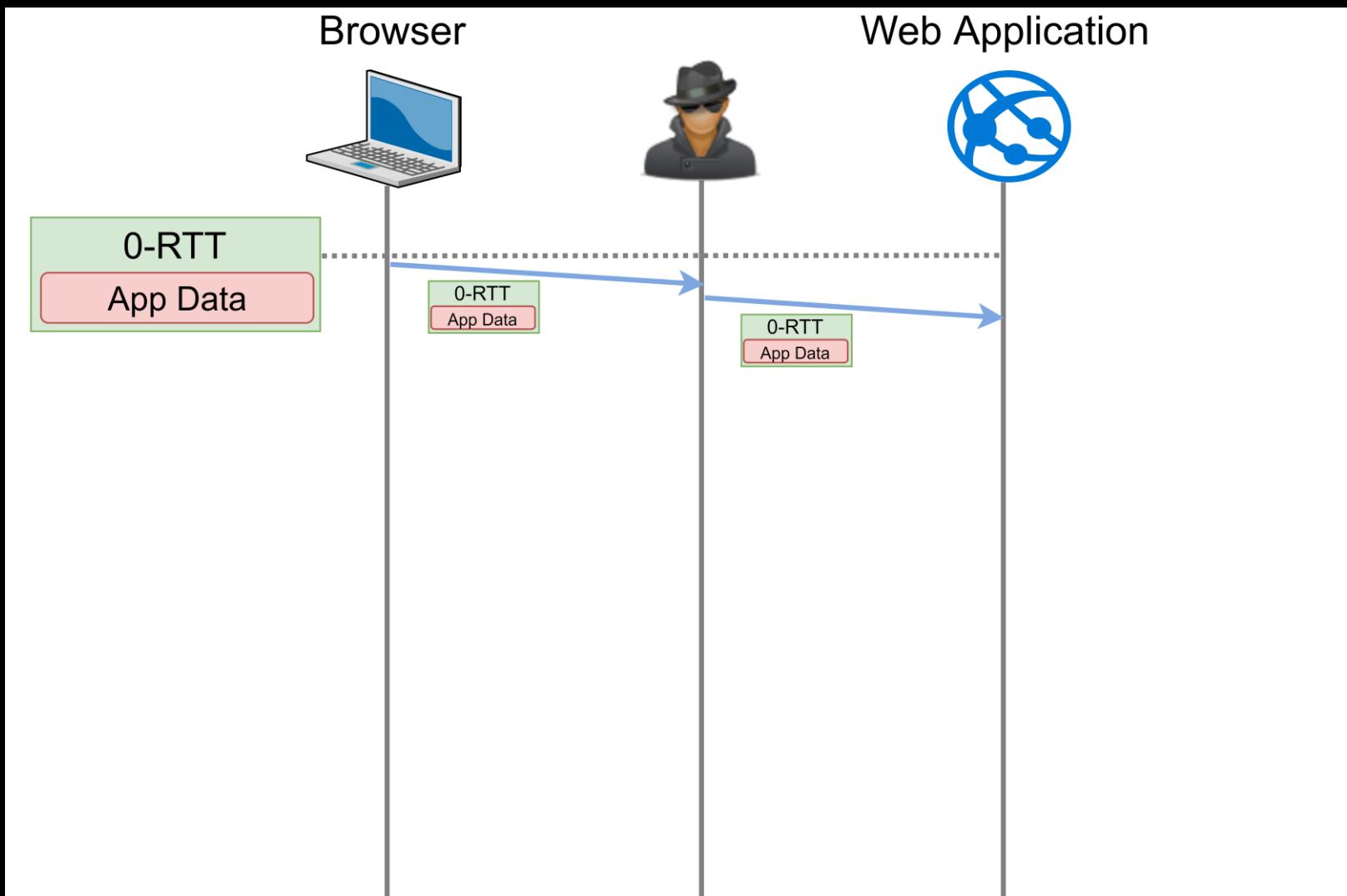
UNIVERSAL REPLAY ATTACK



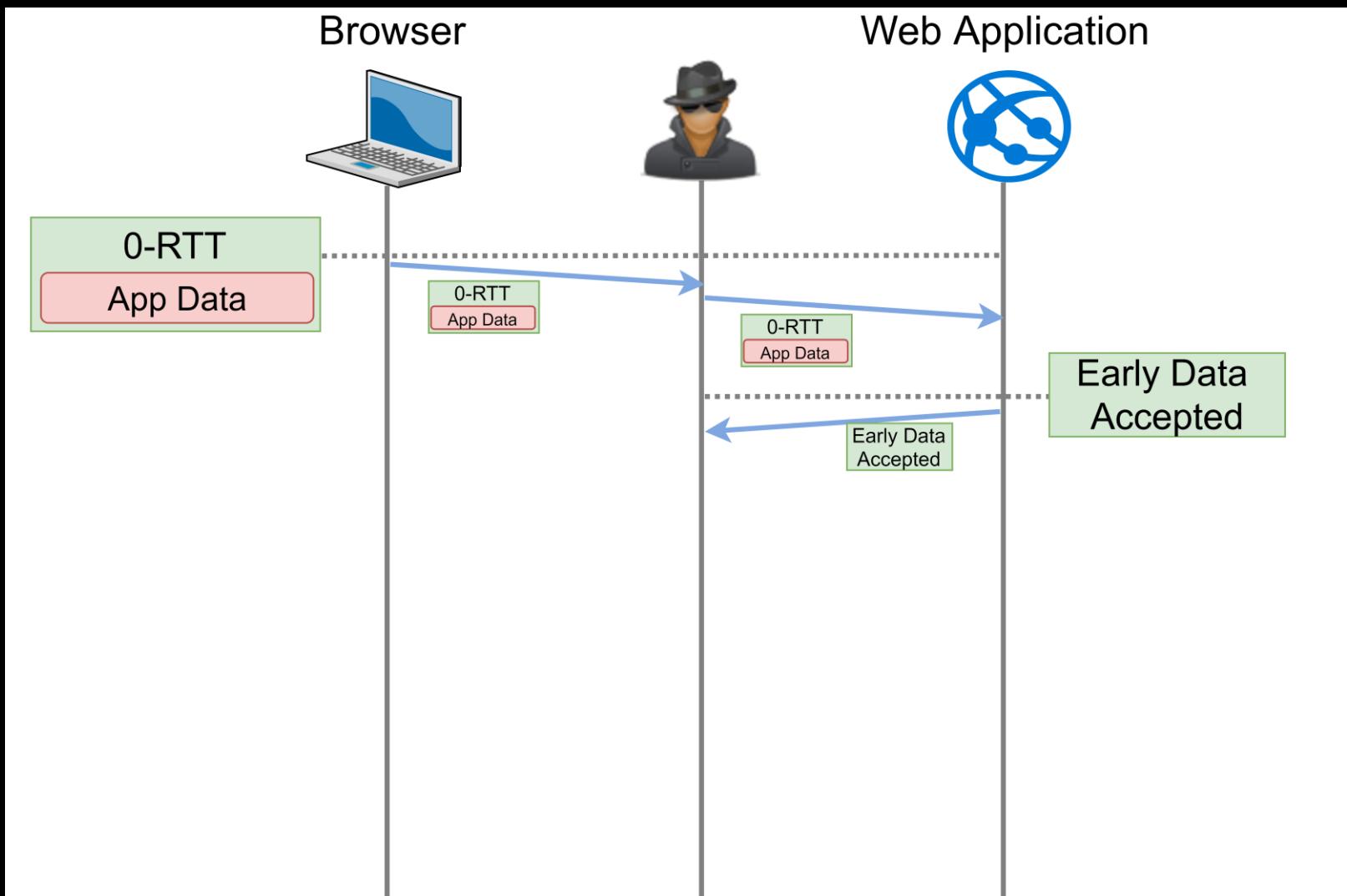
UNIVERSAL REPLAY ATTACK



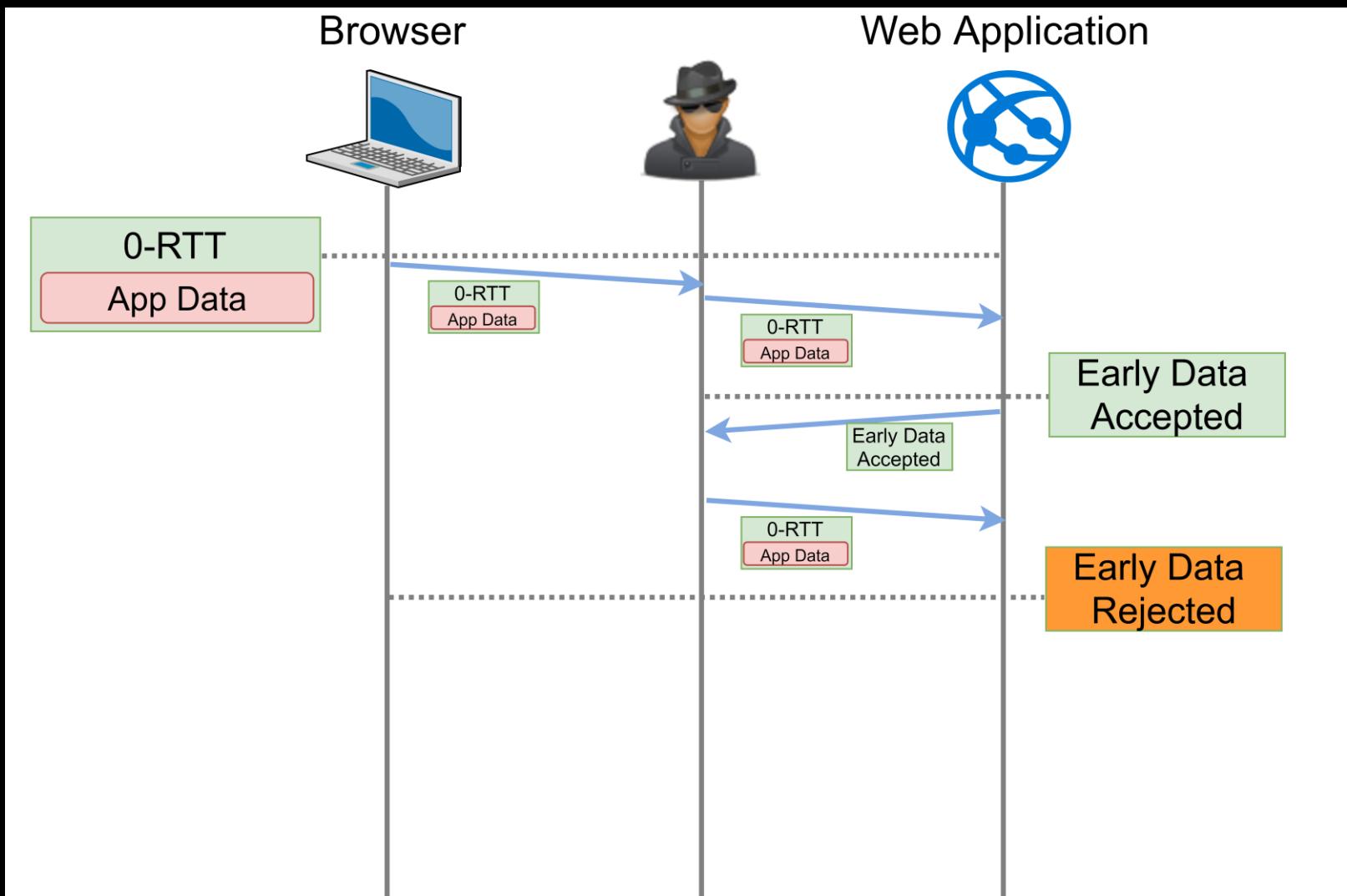
UNIVERSAL REPLAY ATTACK



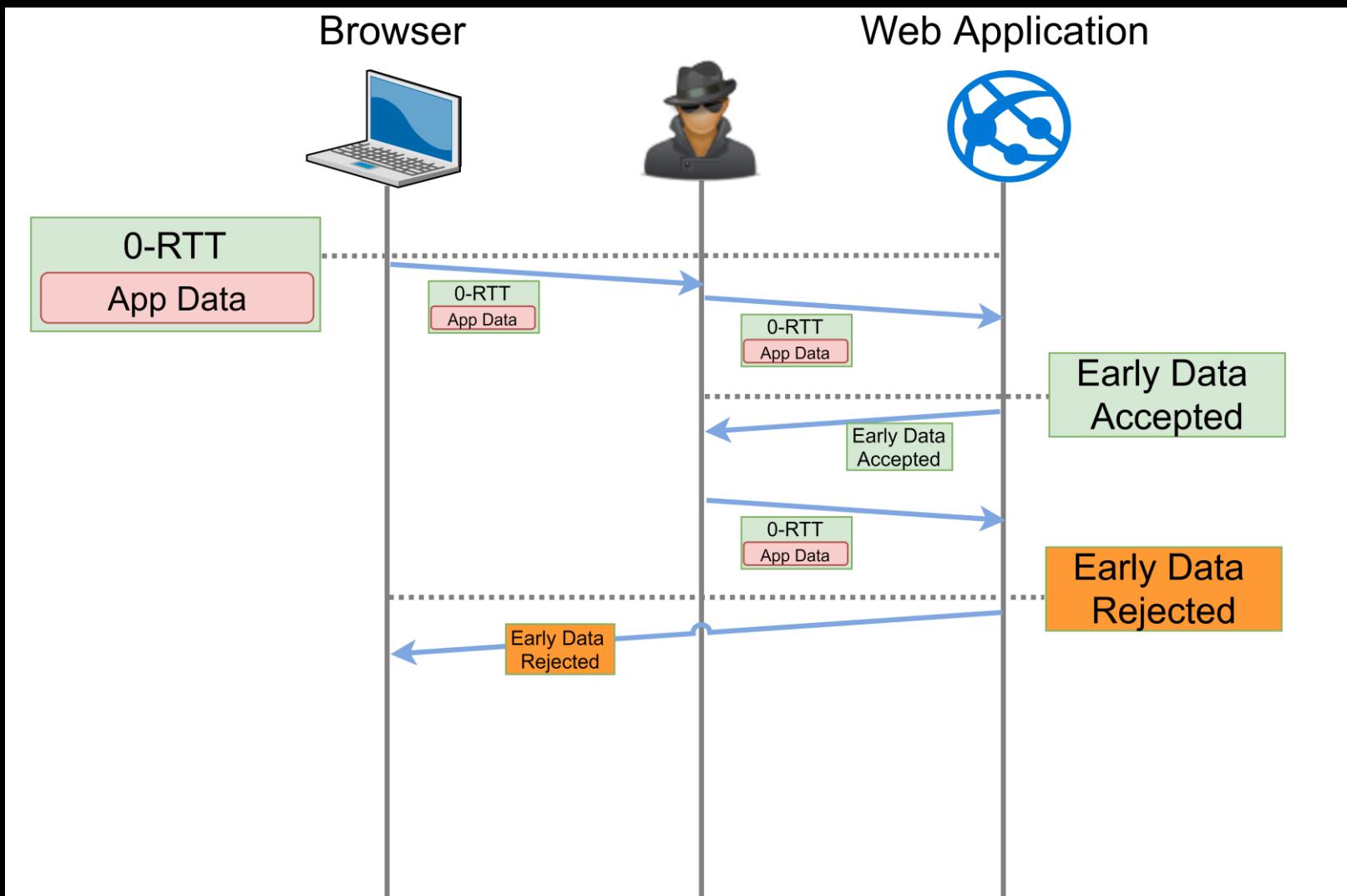
UNIVERSAL REPLAY ATTACK



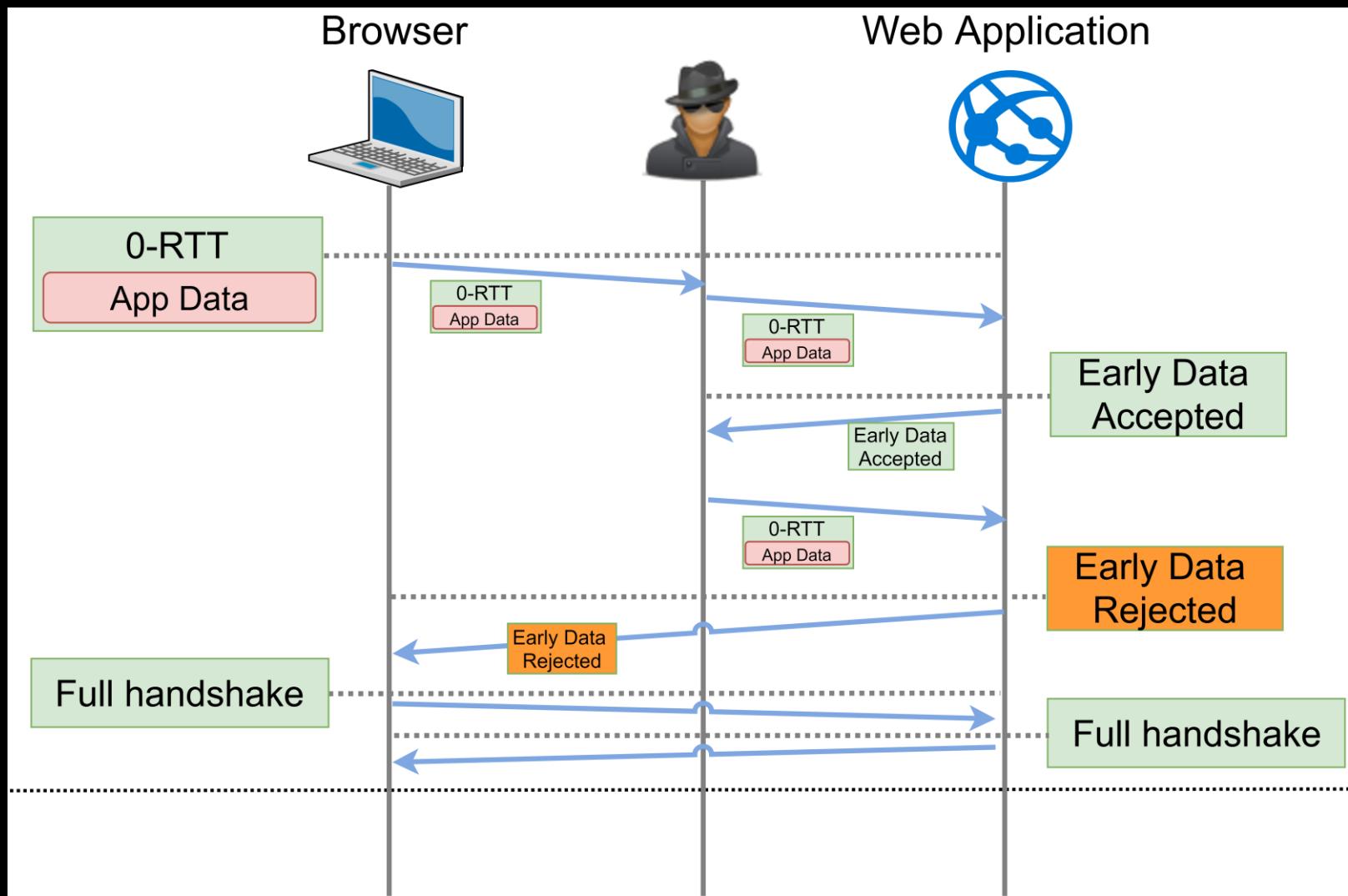
UNIVERSAL REPLAY ATTACK



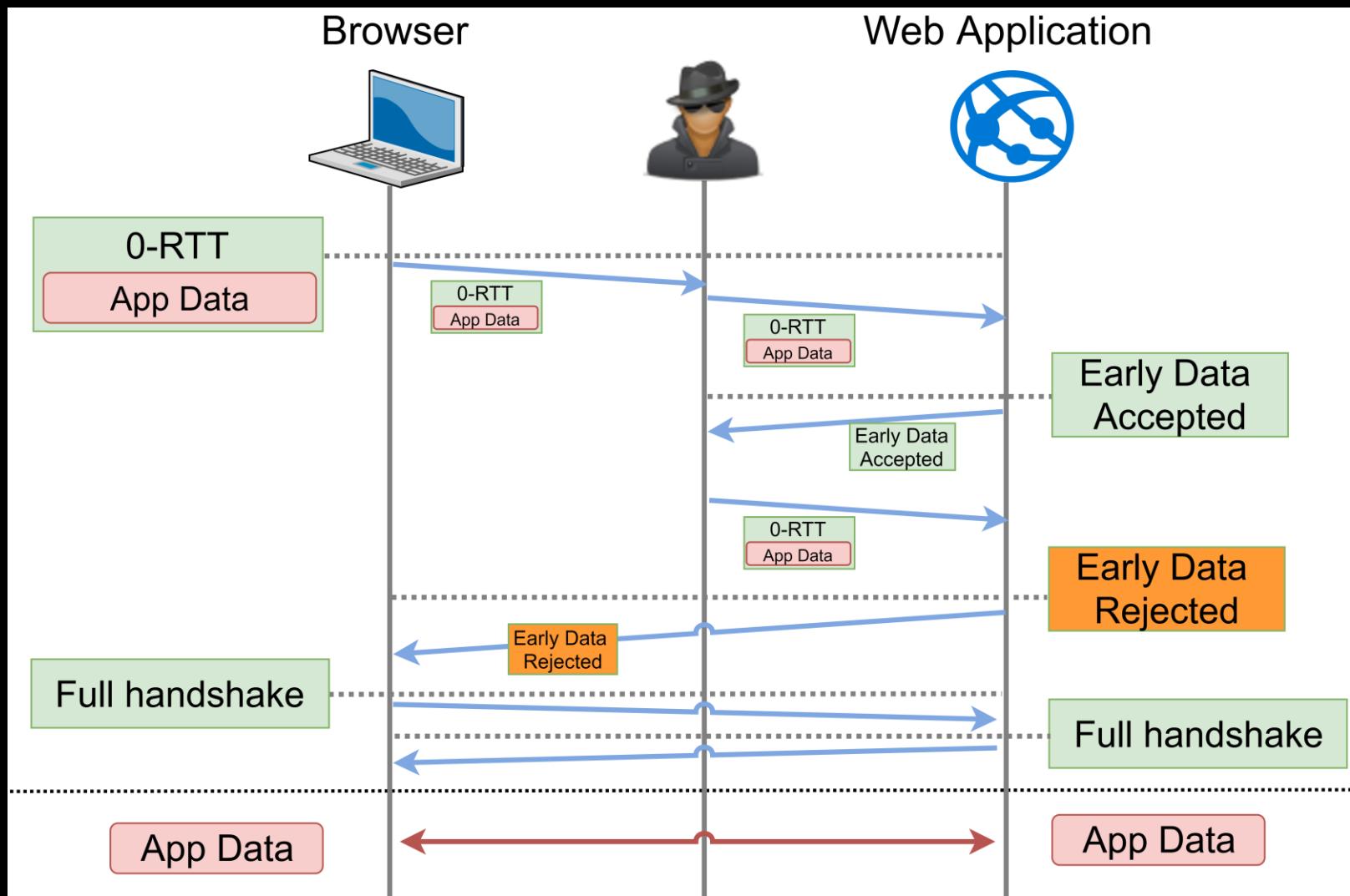
UNIVERSAL REPLAY ATTACK



UNIVERSAL REPLAY ATTACK



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.
- 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 your 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 lead 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!