Nick Sullivan (@grittygrease)

Filippo Valsorda (@filosottile)

Forward Secrecy in TLS

A Systematic Study

Who am I?

- Cryptography Engineer
- Focused on taking cryptographic concepts and bringing them to the world at scale
- Apple FairPlay: Protecting keys in hostile locations
- Cryptography team at CloudFlare: bringing the security of the Internet giants to everyone

WARNING:

This is a practical talk about the Internet industry.

CloudFlare Reverse Proxy

Bandwidth saved by CloudFlare

Bandwidth you pay for

CLOUDFLARE

CloudFlare Network Map

Application Layer

- DNS
- HTTP(S)

- 5-7% of web requests go through CloudFlare’s network
- We see almost every web user daily
- Low latency is most important feature

Key contributions

- **Keyless SSL**: Terminating HTTPS without the private key
- **Universal DNSSEC**: Digital signatures in the DNS with ECDSA
- **ChaCha20/Poly1305**: djb crypto is not just for Google anymore
- **Deprecation of RC4**: First to drop the creaky cipher
- **Origin CA**: Free certificates for services behind CloudFlare
- **Universal SSL**: Free HTTPS for all sites, ECDSA certificates
- **Global session resumption**: One fewer roundtrip even on new servers

Key contributions

- **Keyless SSL**: Terminating HTTPS without the private key
- **Universal DNSSEC**: Digital signatures in the DNS with ECDSA
- **ChaCha20/Poly1305**: djb crypto is not just for Google anymore
- **Deprecation of RC4**: First to drop the creaky cipher
- **Origin CA**: Free certificates for services behind CloudFlare
- **Universal SSL**: Free HTTPS for all sites, ECDSA certificates
- **Global session resumption**: One fewer roundtrip even on new servers

Forward Secrecy in TLS 1.2

A weak definition

- Compromising a long-term key does not allow an attacker compromise previous connections.

Threat models

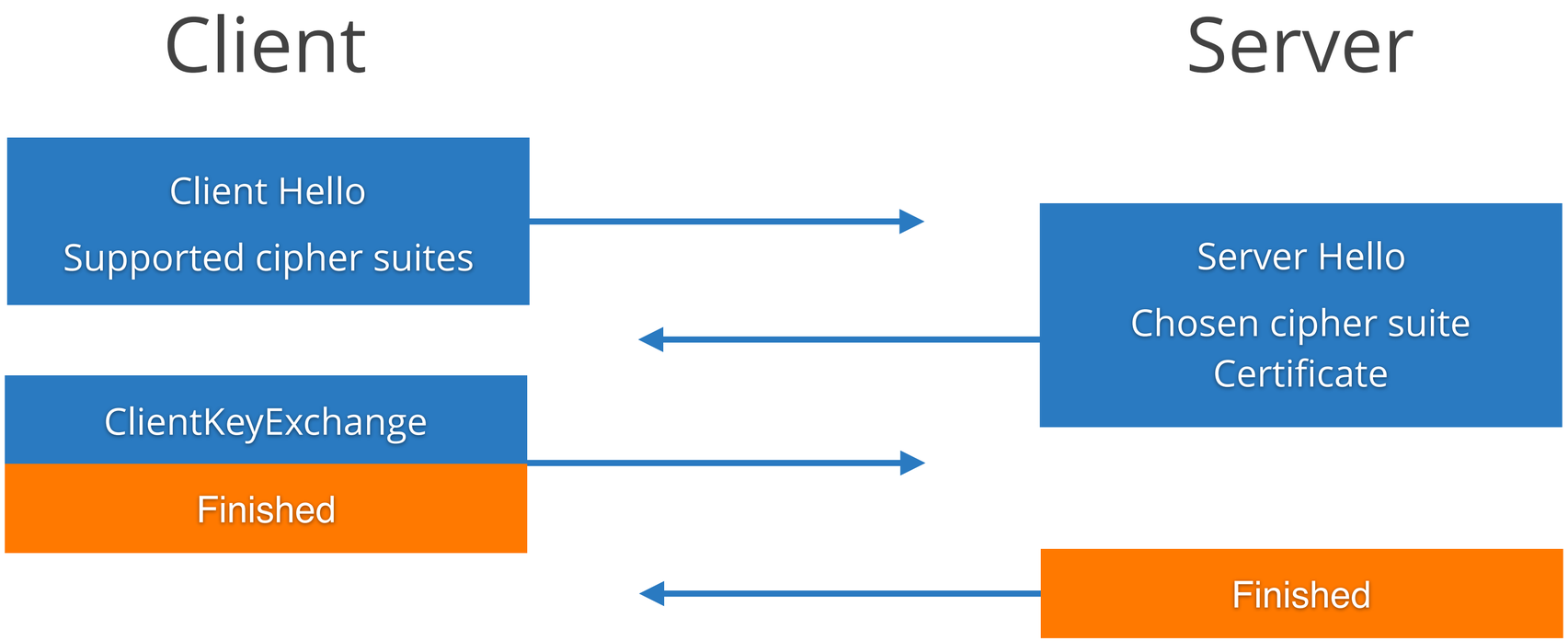
This is the Internet, all wires are tapped

1. Attackers with access to transcript of historical communications
2. Attackers who can place themselves in a MiTM position

We explore what happens when attackers gain access to different keys

TLS 1.2 RSA key exchange

Latency: 2 round-trips

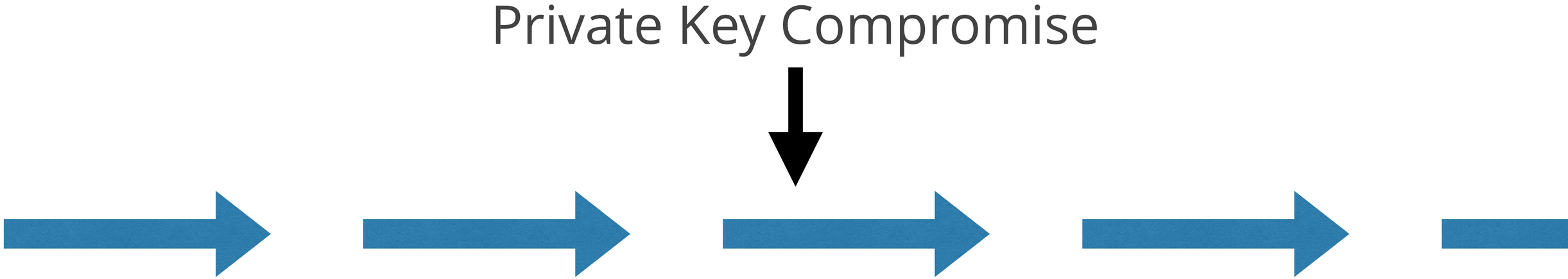


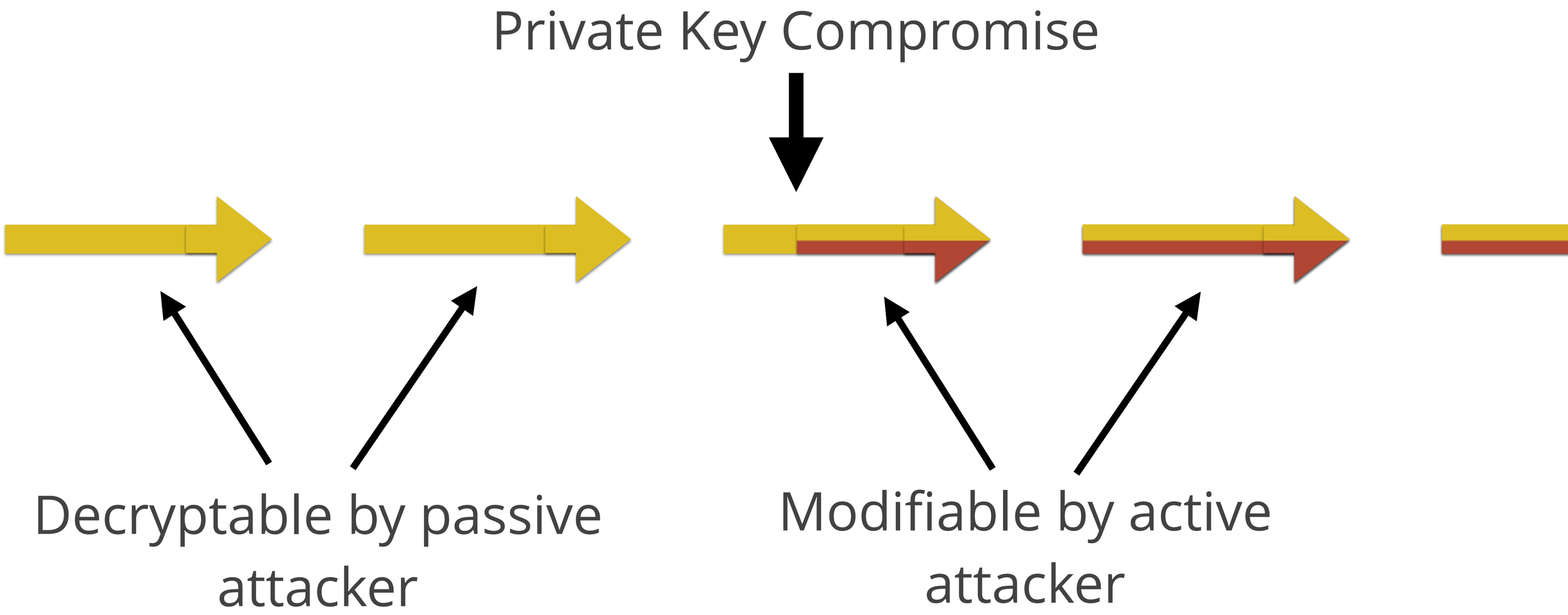
ClientKeyExchange = RSA_encrypt(Certificate_pk, pre_master_secret)

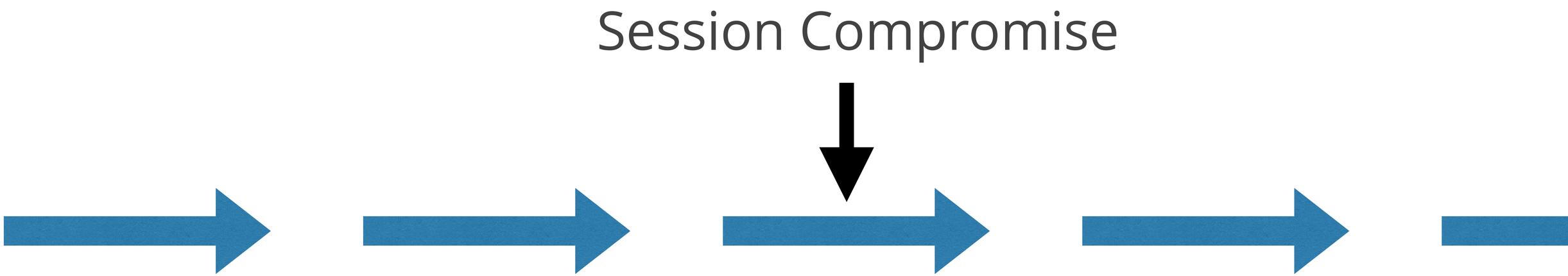
pre_master_secret > master_secret > traffic keys

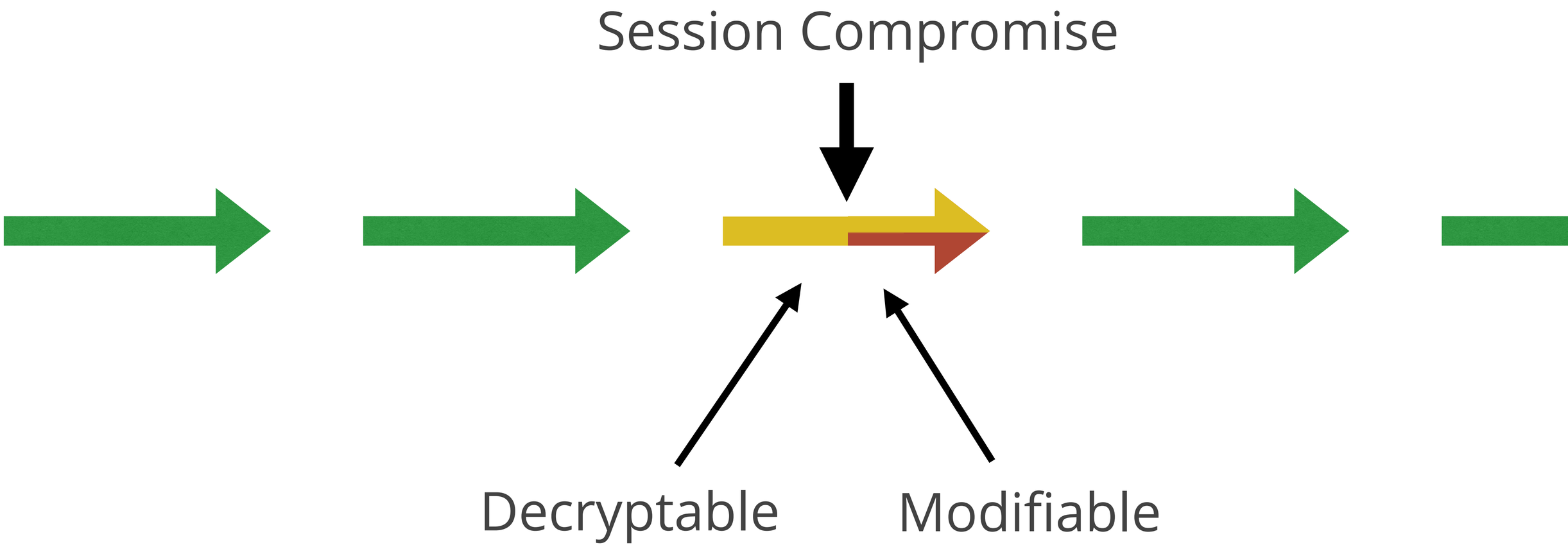
Two keys

- RSA private key
- Session key


Private Key Compromise

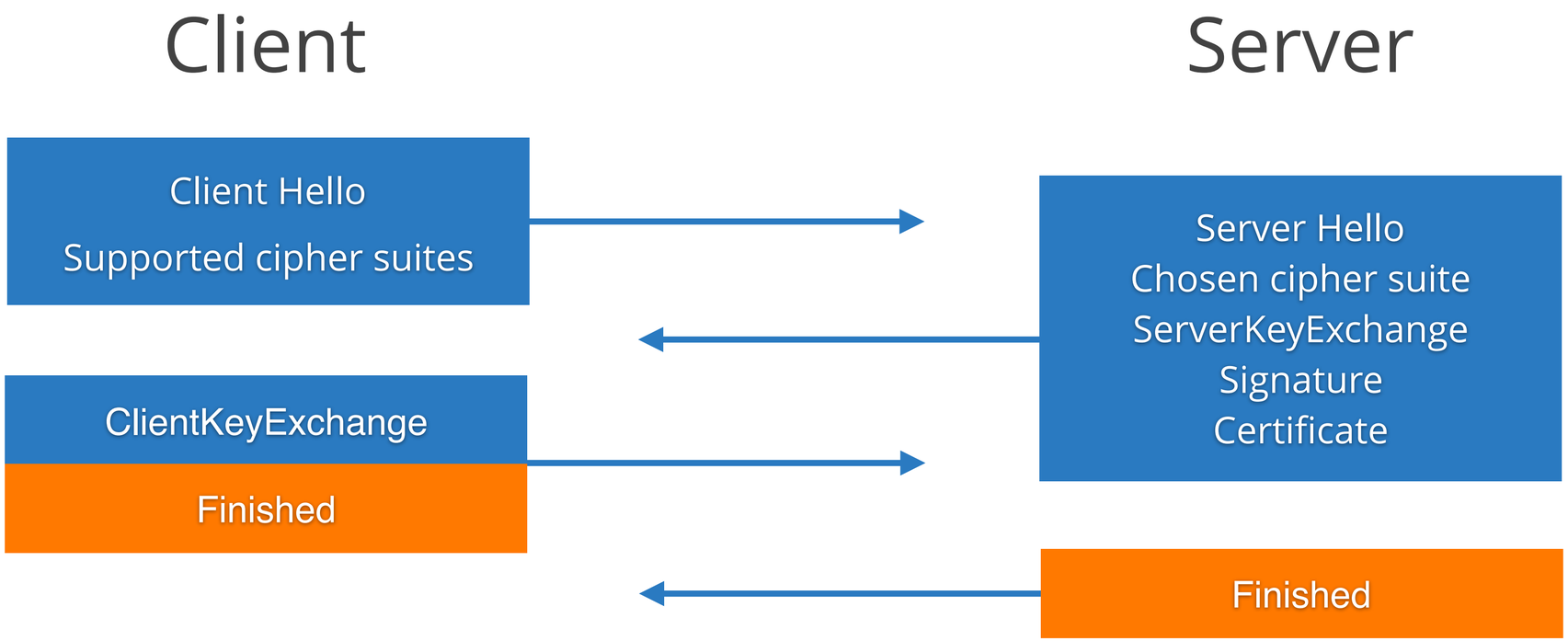

Private Key Compromise
Decryptable by passive attacker
Modifiable by active attacker


Session Compromise


Session Compromise
Decryptable
Modifiable

TLS 1.2 (EC)DHE key exchange

Latency: 2 round-trips



ServerKeyExchange = Sign(Certificate_pk, (EC)DH public key share)
ClientKeyExchange = (EC)DH public key share

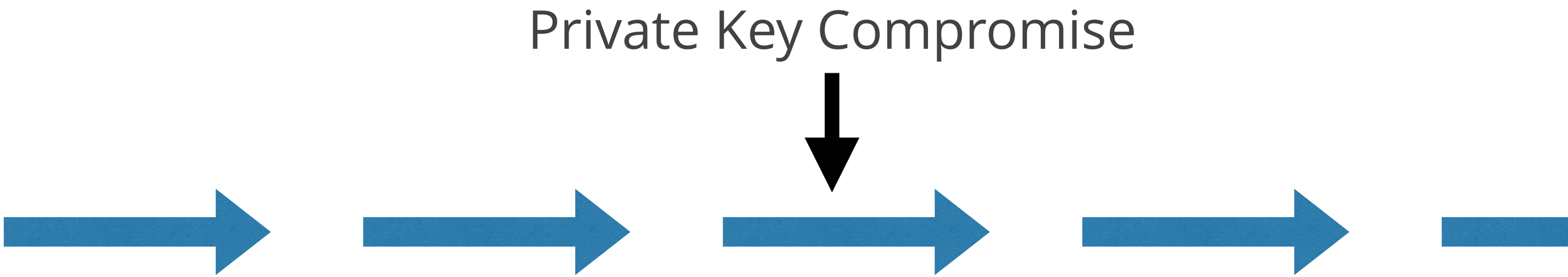
(EC)DH derivation > pre_master_secret > master_secret > traffic keys

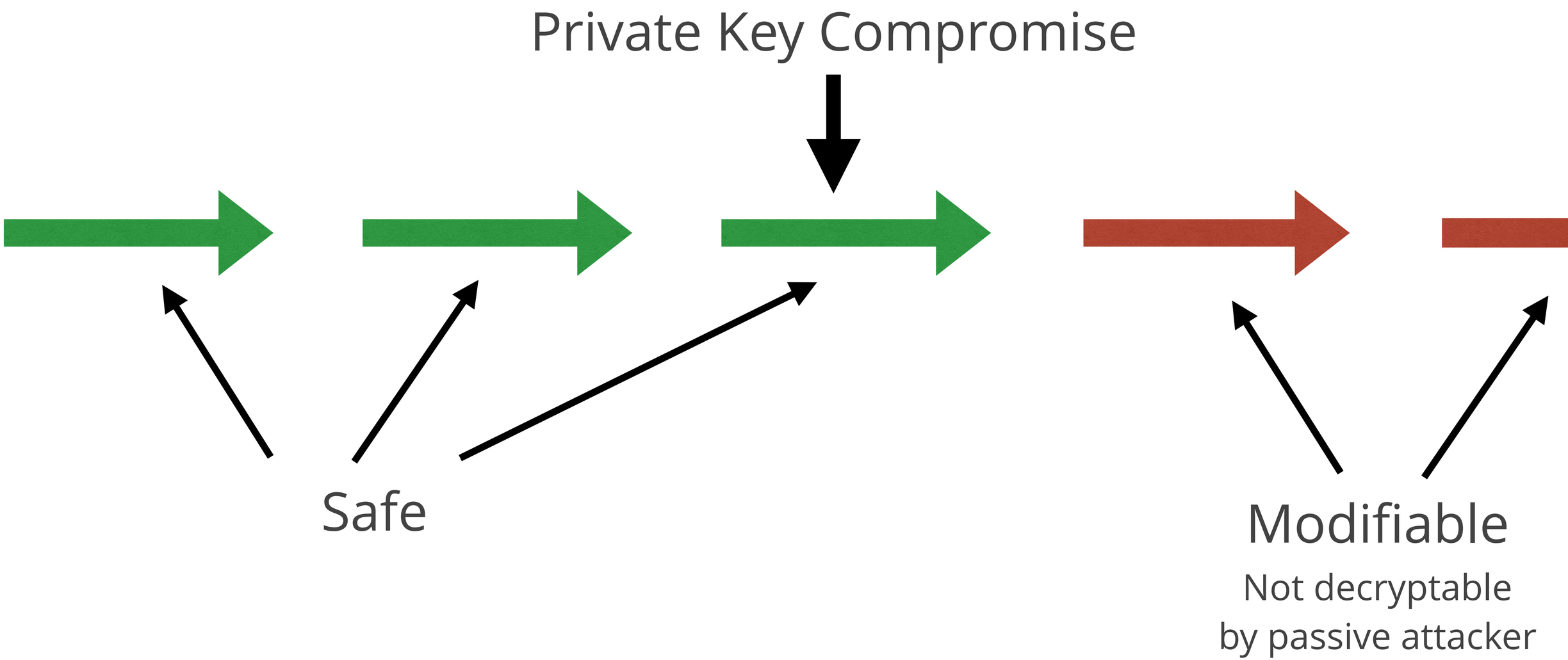
(EC)DHE: the E is for ephemeral

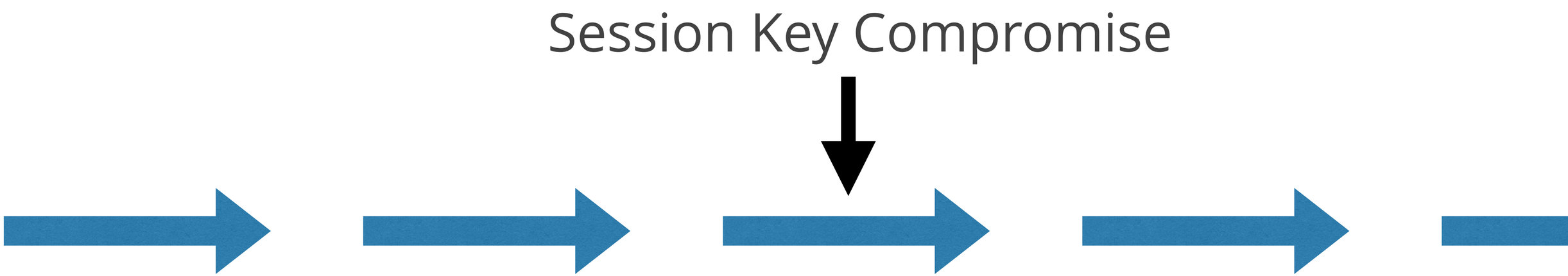
- OpenSSL historically did not always do this
- SSL_OP_SINGLE_DH_USE (required as of 2016)
- We assume it is ephemeral

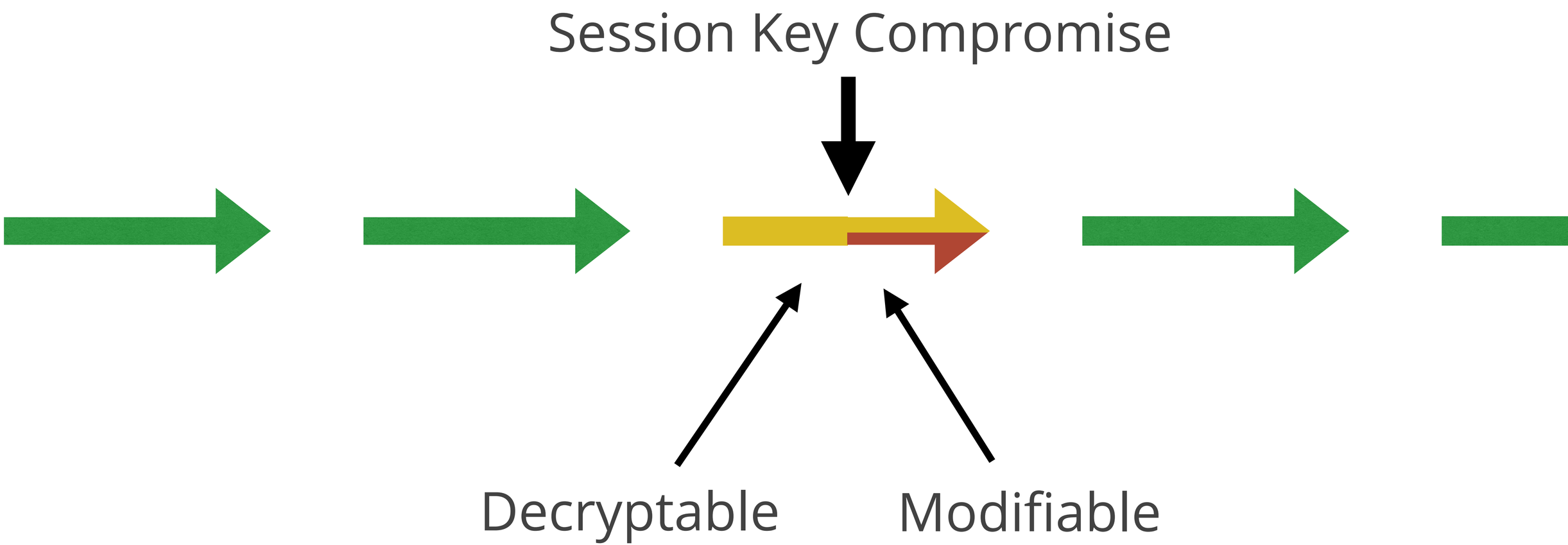
Two keys

- Certificate private key
- Session key


Private Key Compromise


Private Key Compromise
Safe
Modifiable
Not decryptable
by passive attacker


Session Key Compromise

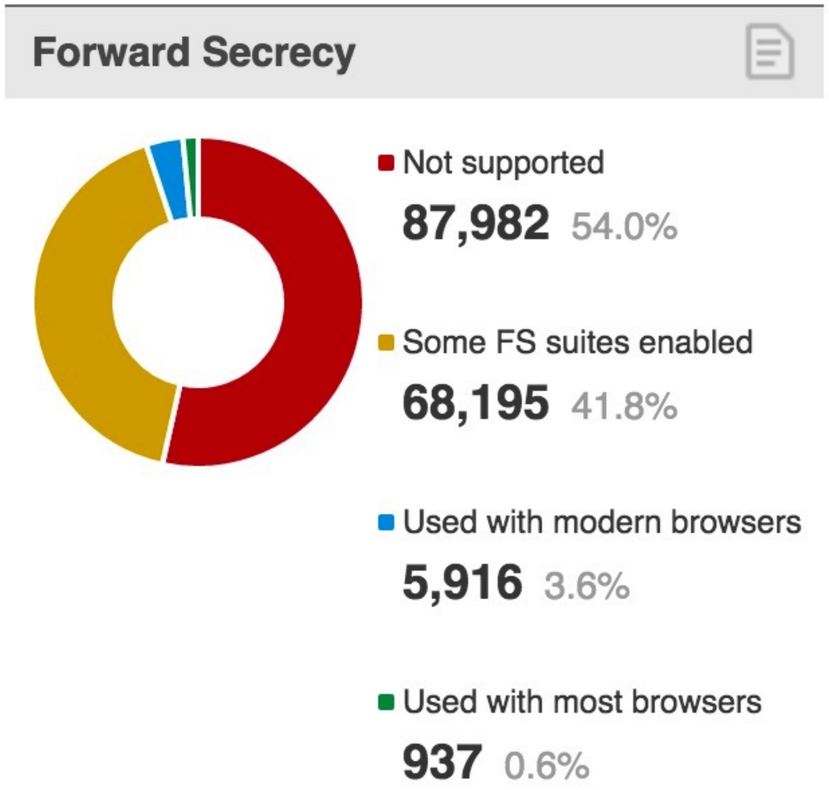

Session Key Compromise
Decryptable
Modifiable

Key contributions

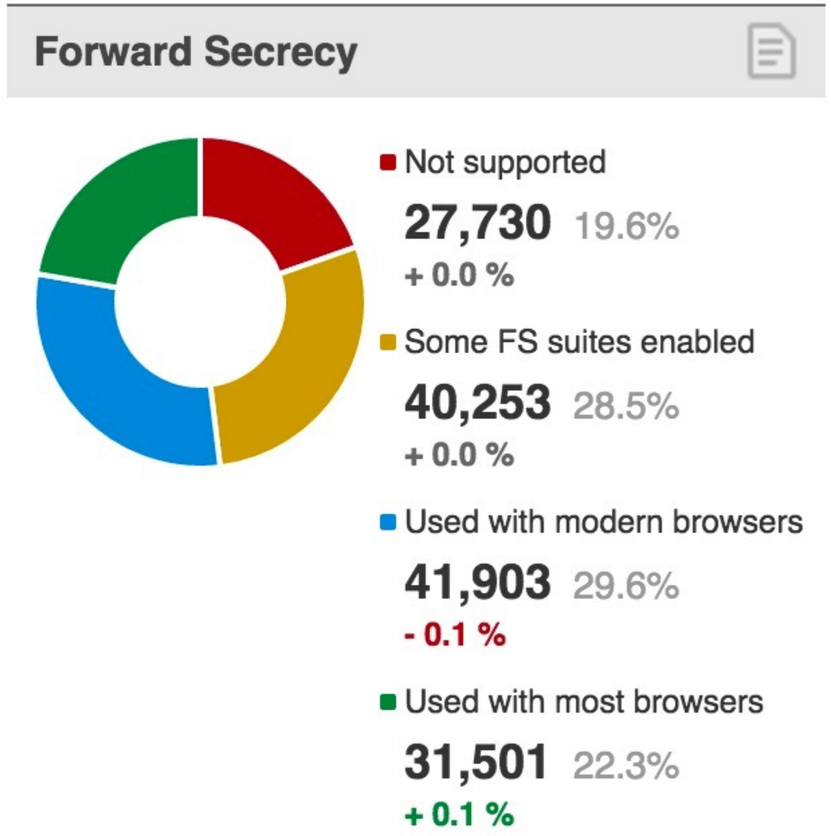
- **Keyless SSL**: Terminating HTTPS without the private key
- **Universal DNSSEC**: Digital signatures in the DNS with ECDSA
- **ChaCha20/Poly1305**: djb crypto is not just for Google anymore
- **Deprecation of RC4**: First to drop the creaky cipher
- **Origin CA**: Free certificates for services behind CloudFlare
- **Universal SSL**: Free HTTPS for all sites, ECDSA certificates ✓
- **Global session resumption**: One fewer roundtrip even on new servers

History of forward secrecy support

October 2013



June 2016

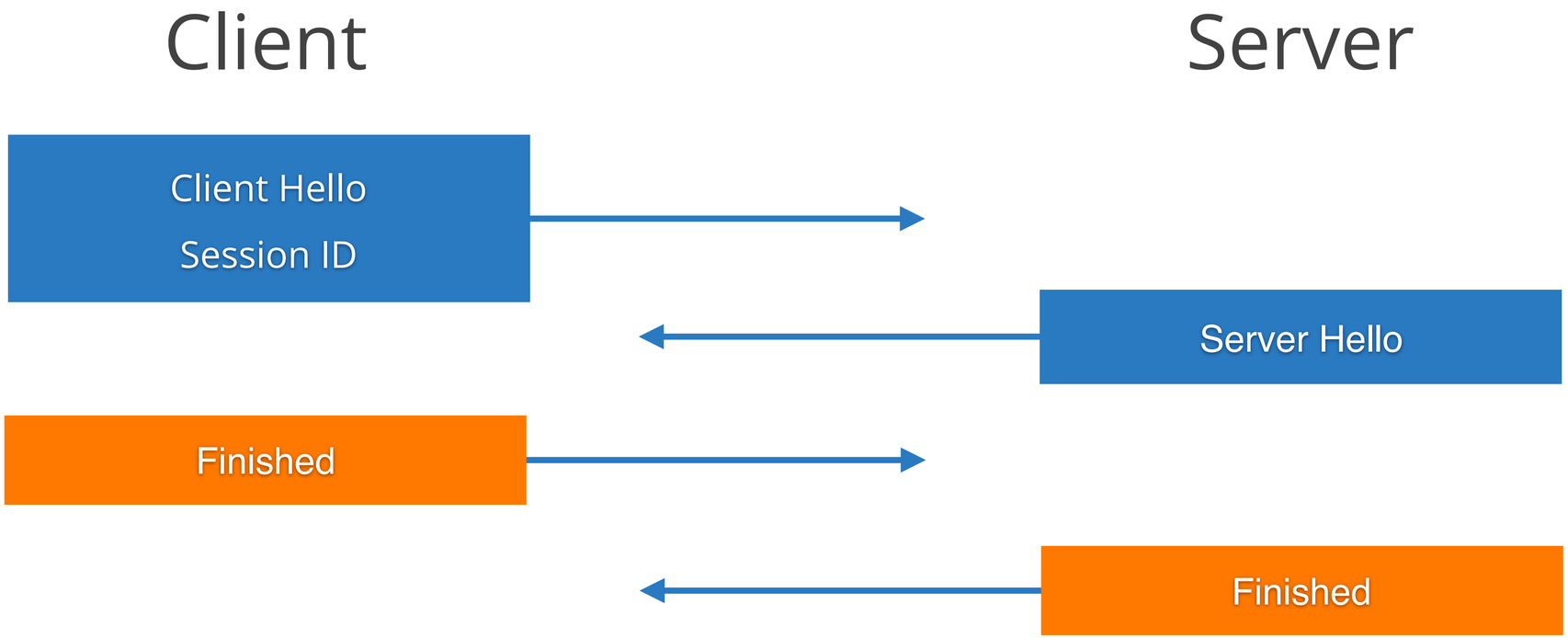


But wait…

- What about session resumption?
- Compromise between performance and secrecy

TLS 1.2 Session Resumption

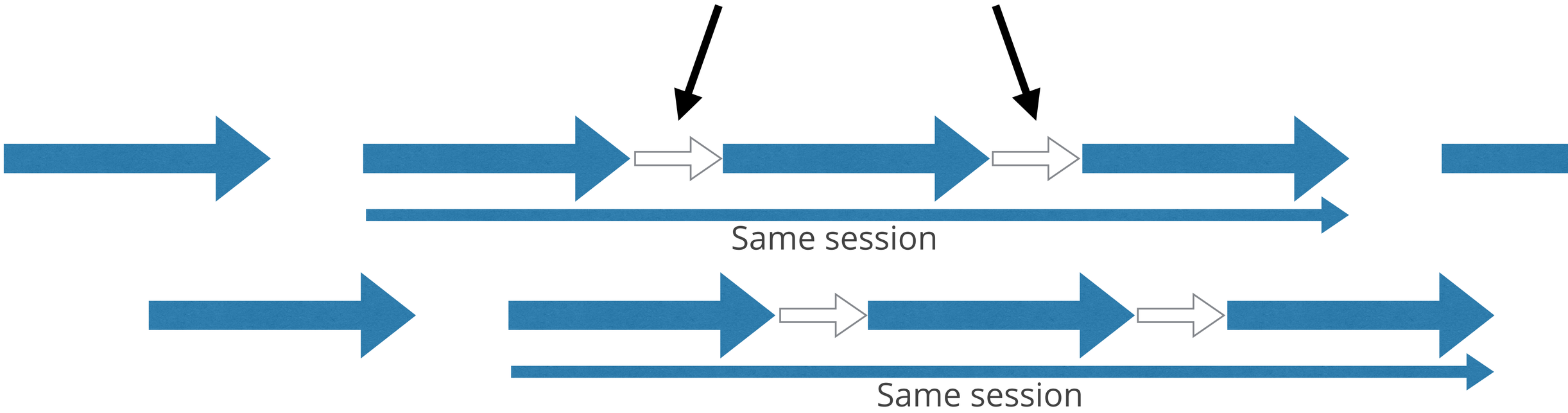
Latency: 1 round-trip



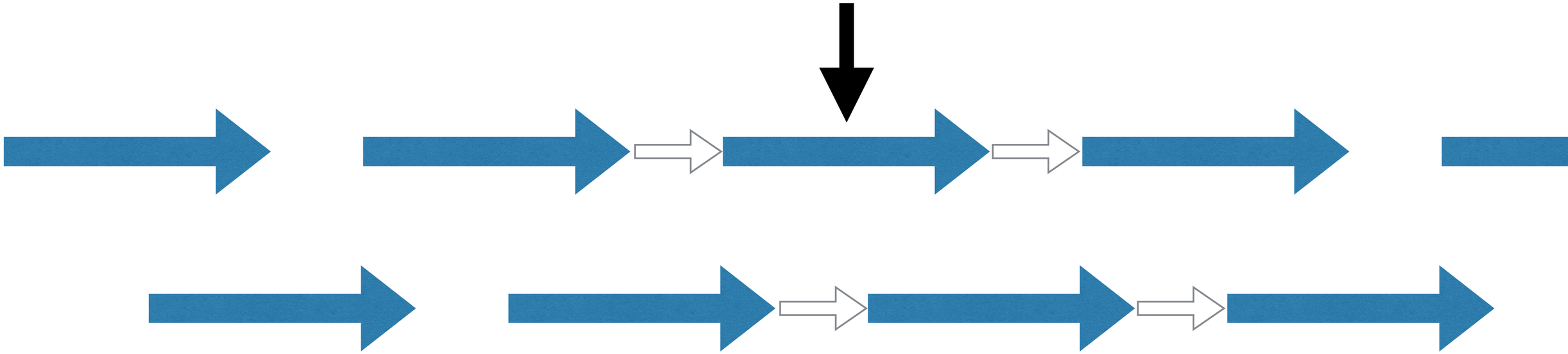
Session = master_secret

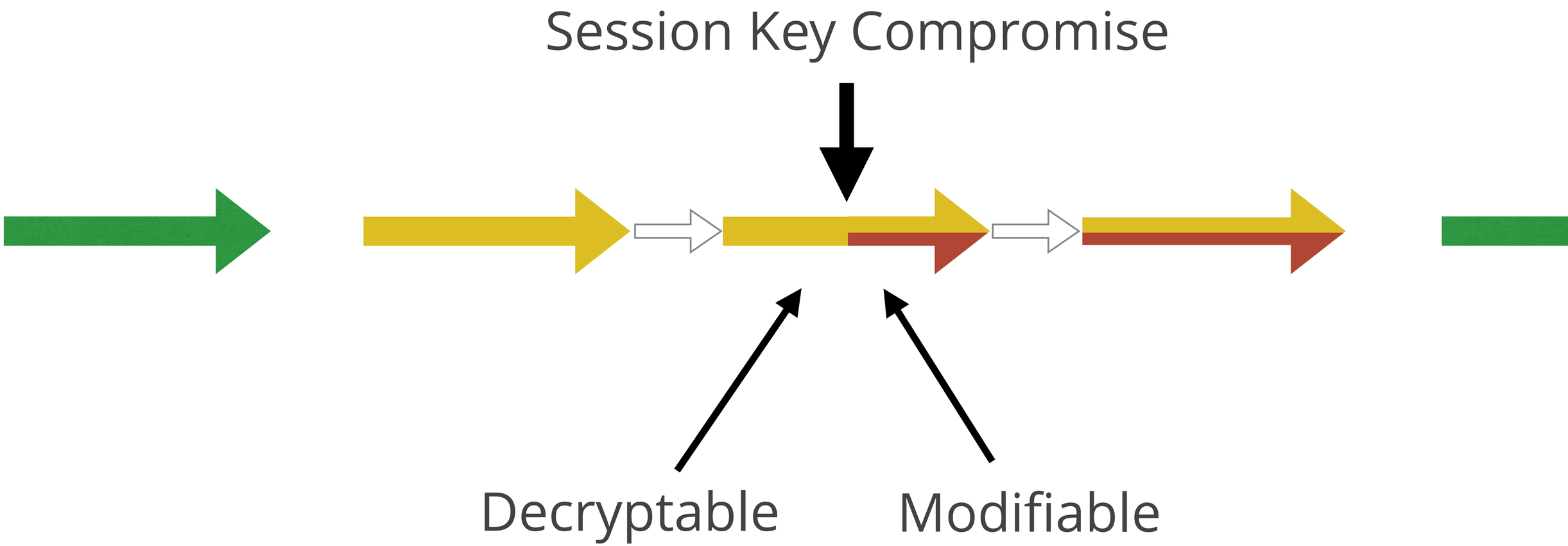
Sessions are saved server-side, indexed by Session ID.

Session Resumption

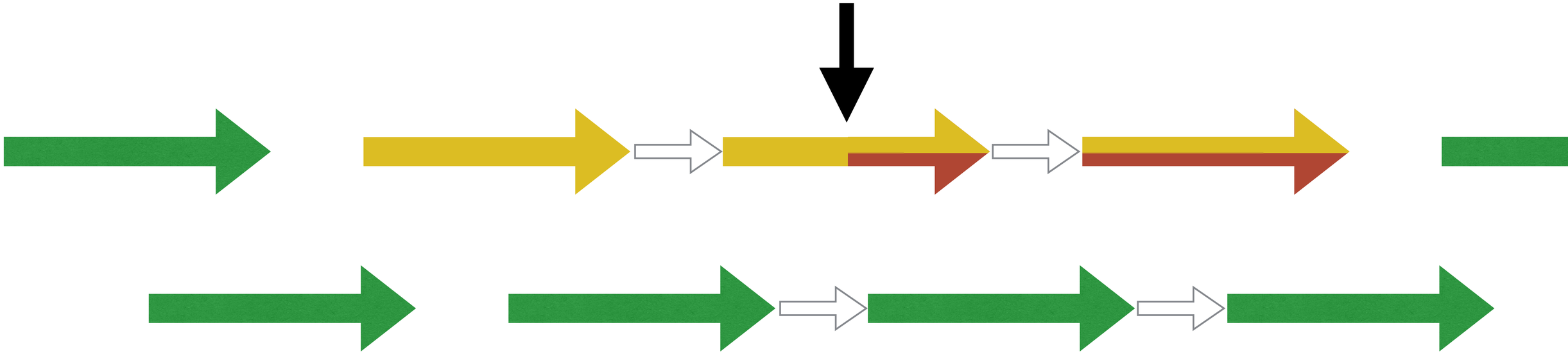


Session Key Compromise



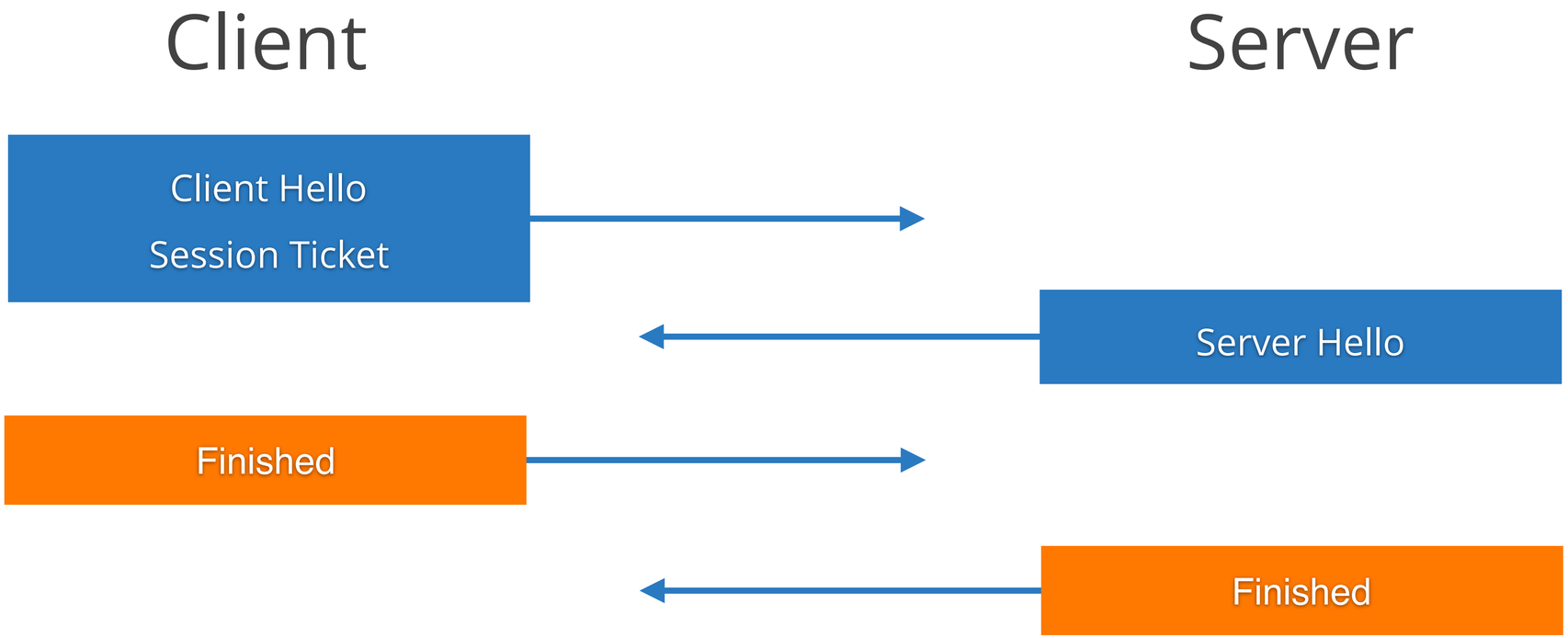

Session Key Compromise
Decryptable
Modifiable

Session Key Compromise



TLS 1.2 Session Tickets RFC5077

Latency: 1 round-trip



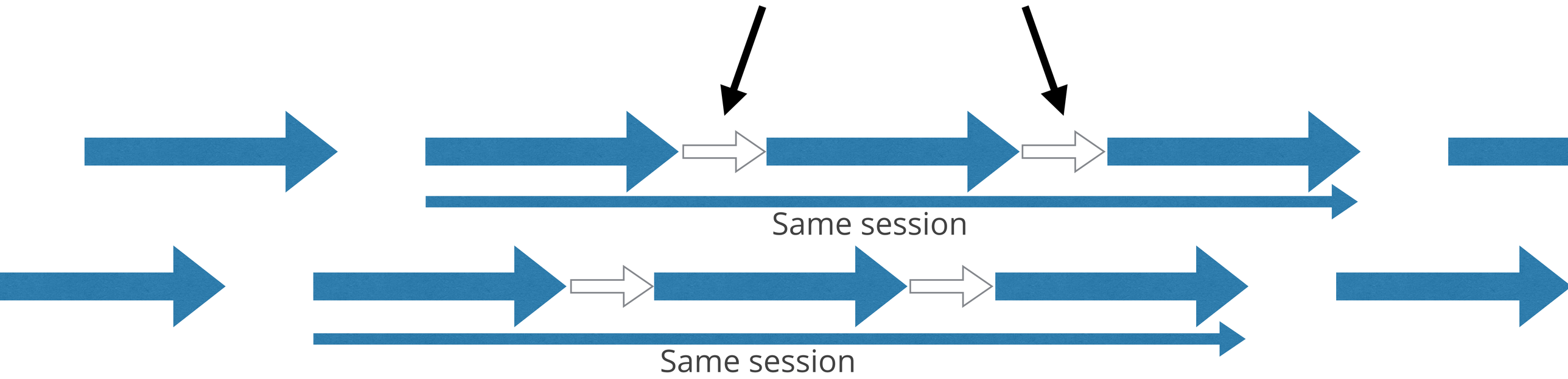
Session Ticket = encrypt(ticket_key, Session)

The server does not store Sessions, holds the ticket_key.

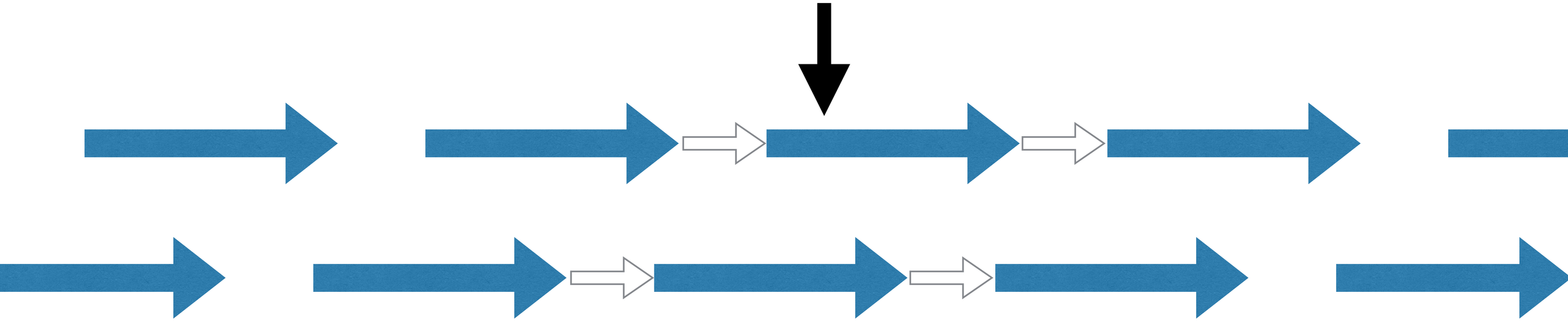
Three keys

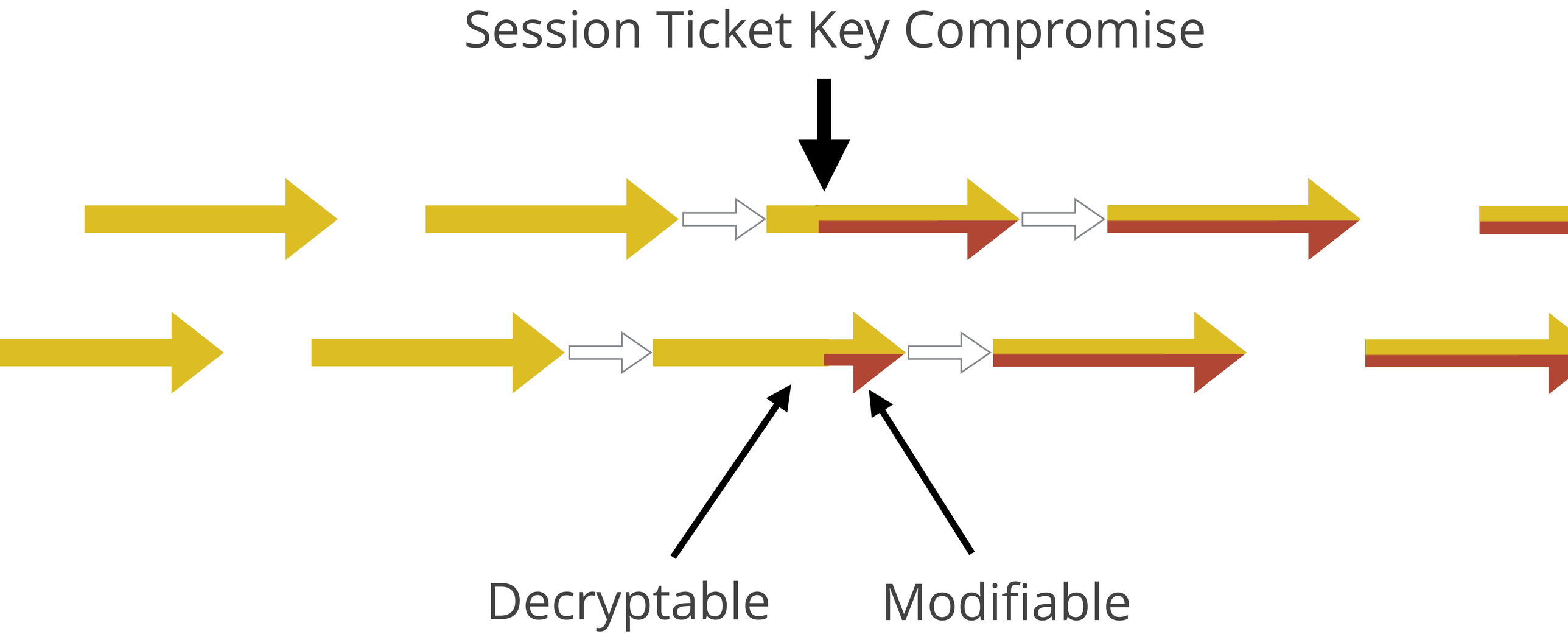
- Certificate private key
- Session key
- Session ticket encryption key

Session Resumption



Session Ticket Key Compromise




Session Ticket Key Compromise
Decryptable
Modifiable

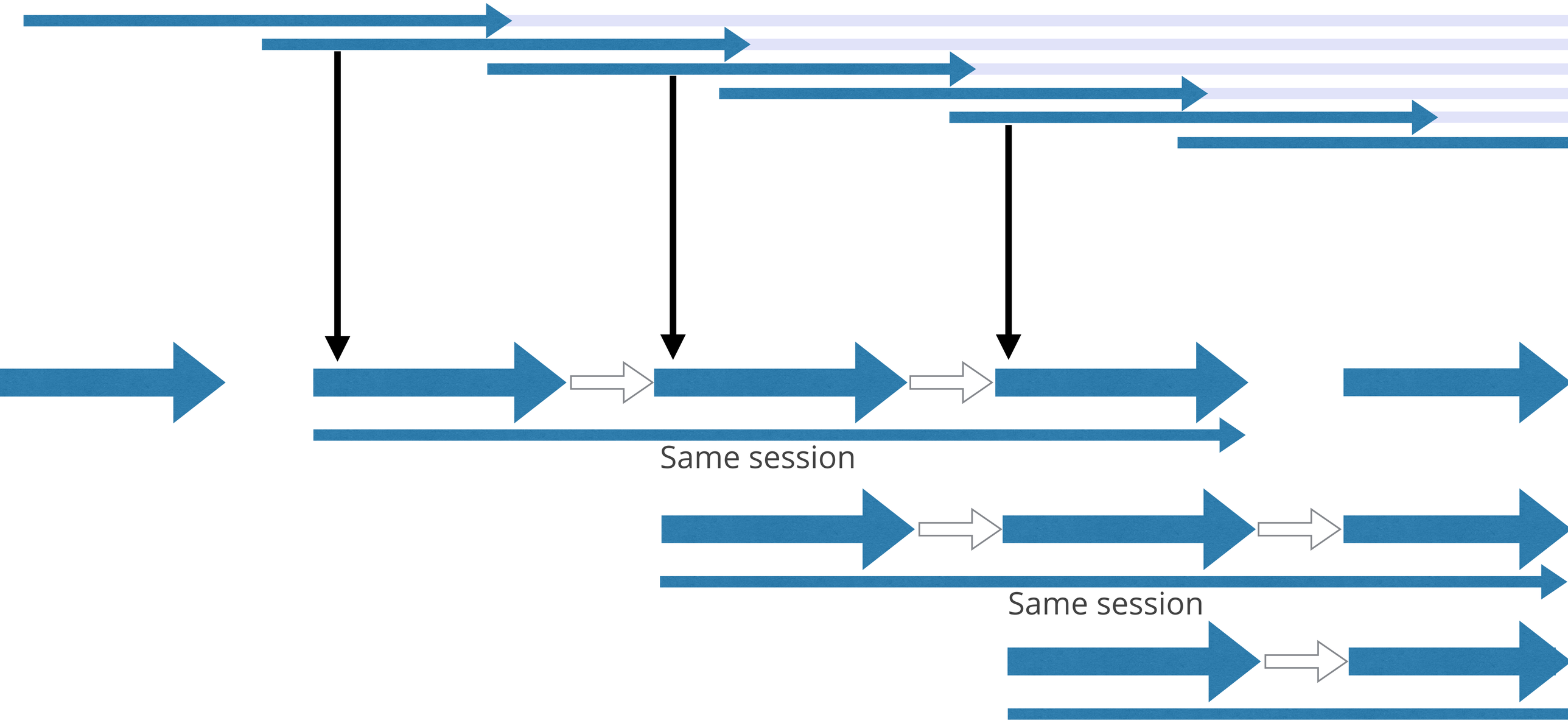
Session resumption + load balancing

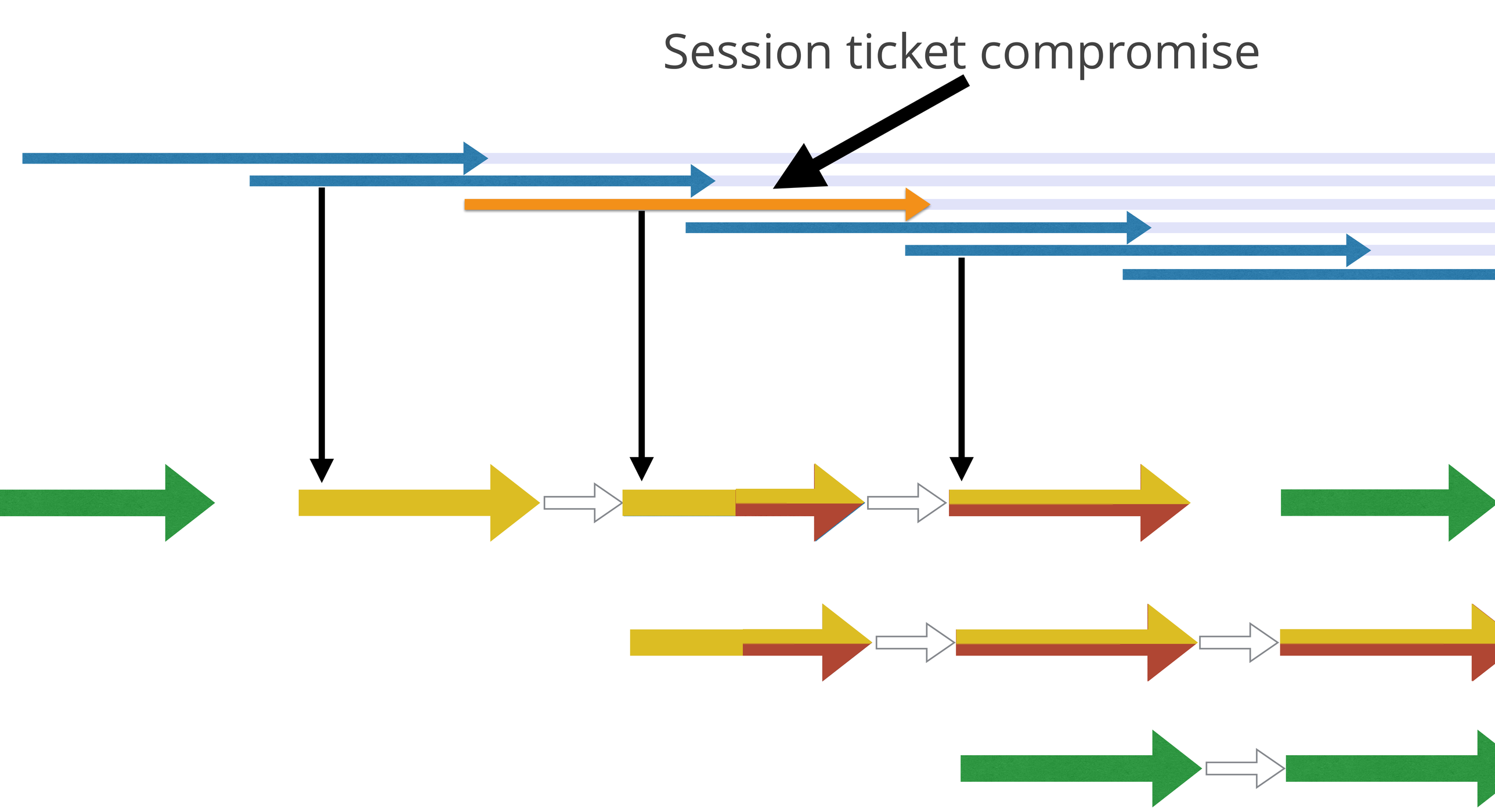
- OpenSSL: session tickets are generated when the server starts
  - Does not let you resume across servers
  - Some servers run for a long time
- Session resumption
  - How long are sessions cached? Is there a shared cache between machines?
  - *Target for measurement*

Session resumption + load balancing

- Distributed session ticket keys
  - Distribute session ticket encryption keys globally
  - CloudFlare: Rotate current session ticket key hourly, resume with old keys for 18 hours
  - *Target for measurement*

Session ticket lifetime




Session ticket compromise

Key contributions

- **Keyless SSL**: Terminating HTTPS without the private key
- **Universal DNSSEC**: Digital signatures in the DNS with ECDSA
- **ChaCha20/Poly1305**: djb crypto is not just for Google anymore
- **Deprecation of RC4**: First to drop the creaky cipher
- **Origin CA**: Free certificates for services behind CloudFlare
- **Universal SSL**: Free HTTPS for all sites, ECDSA certificates ✓
- **Global session resumption**: One fewer roundtrip even on new servers ???

Not so “perfect”

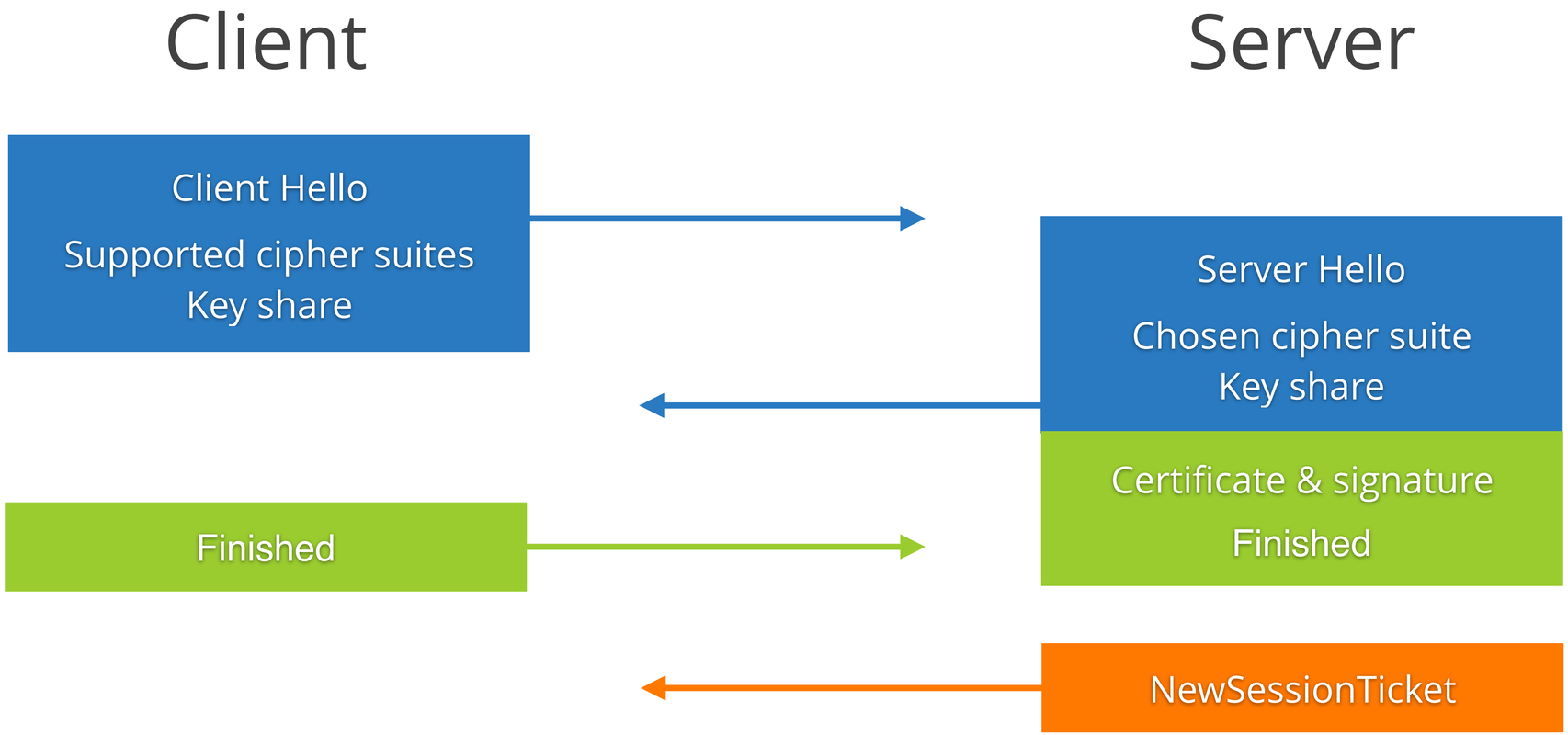
- Sessions persist across resumption, compromise of one session compromises all
- Session ticket compromise compromises all session keys encrypted with the session ticket key

TLS 1.3

Locking down the forward secrecy story

TLS 1.3

Latency: 1 round-trip

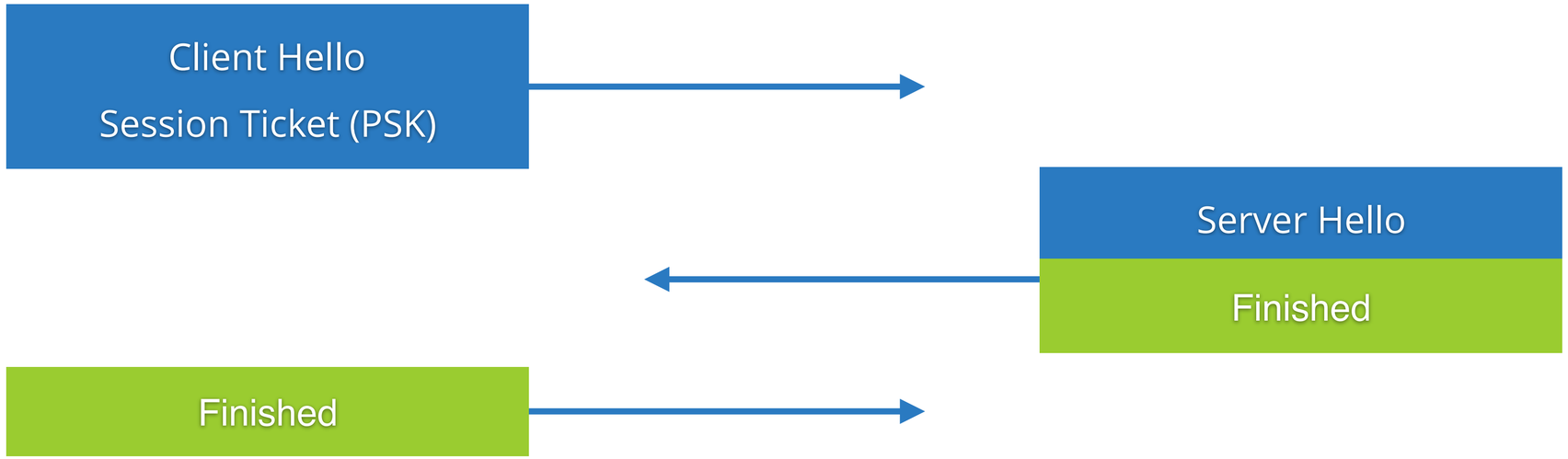


Session resumption improvements

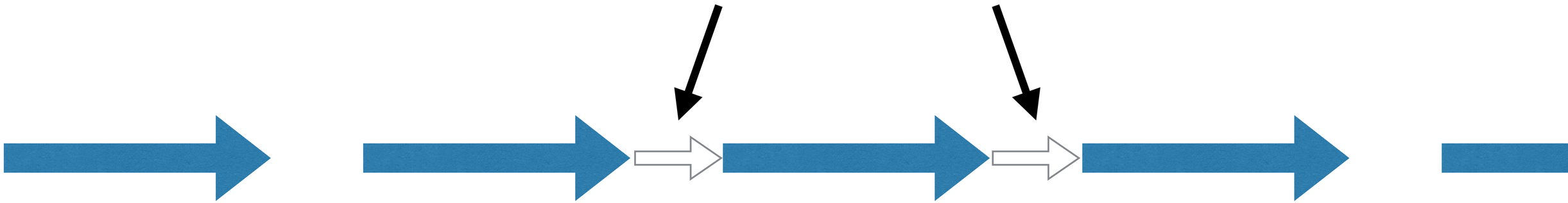
- Resumed sessions use new traffic keys derived from previous master secret and handshake
- PSK mode: **symmetric key ratchet**
- PSK-(EC)DHE mode: **public key ratchet**
- Option to sign resumption handshakes
- Mandatory maximum session ticket lifetime (7 days)

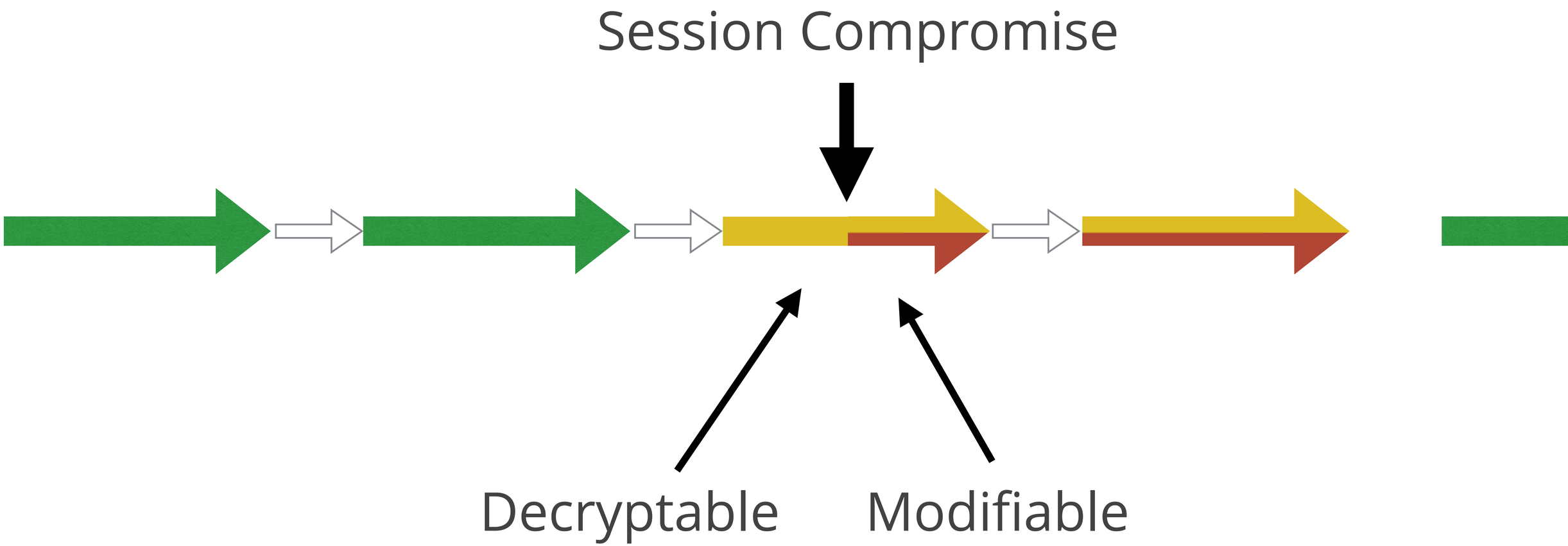
TLS 1.3 PSK Resumption

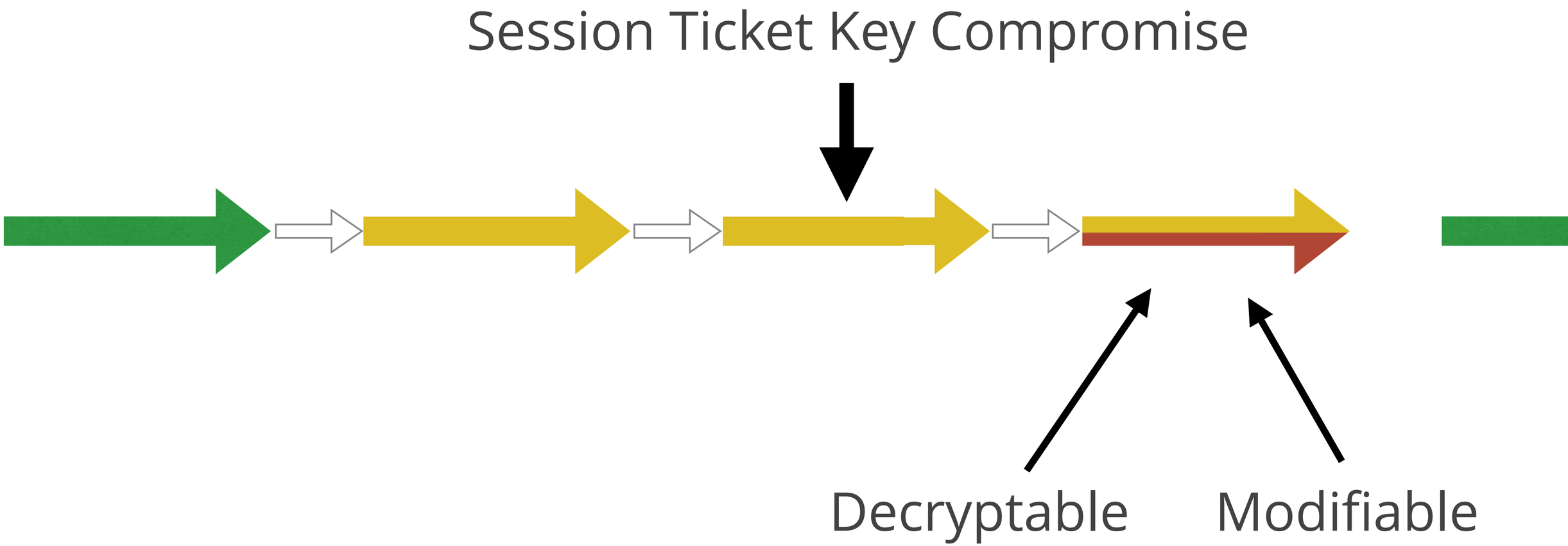
Latency: 1 round-trip



Session Resumption PSK

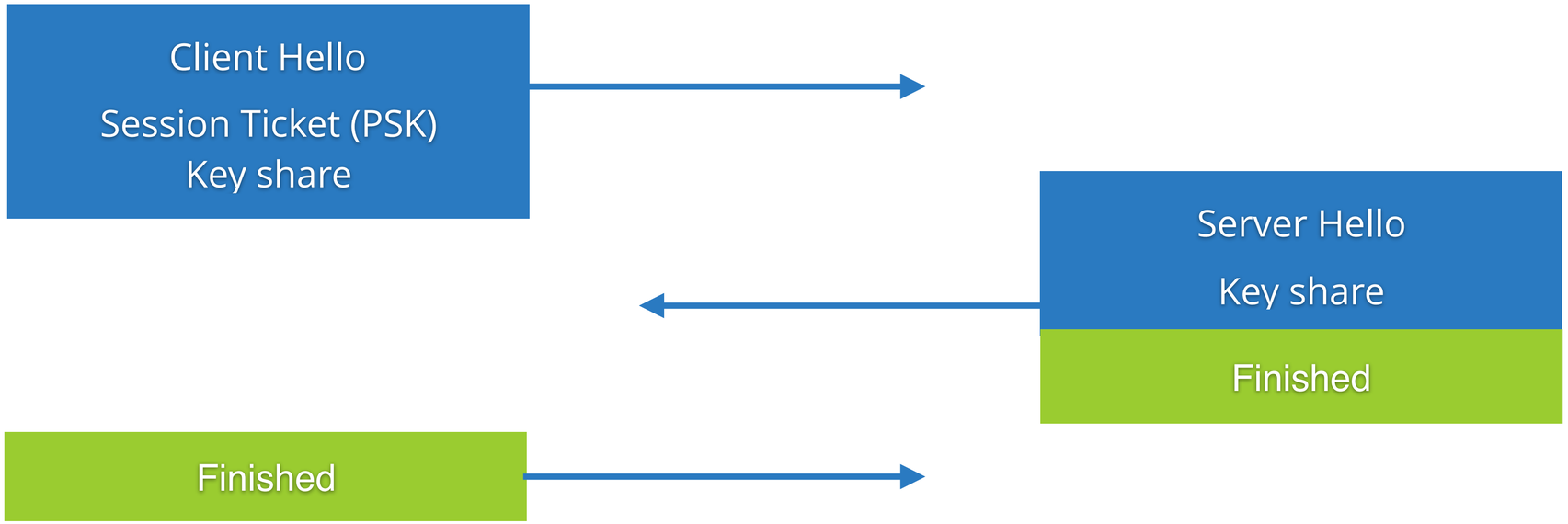



Session Compromise
Decryptable
Modifiable

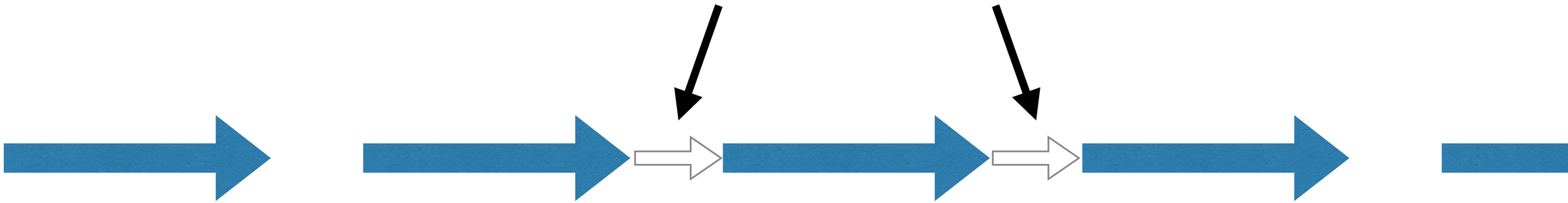

Session Ticket Key Compromise
Decryptable
Modifiable

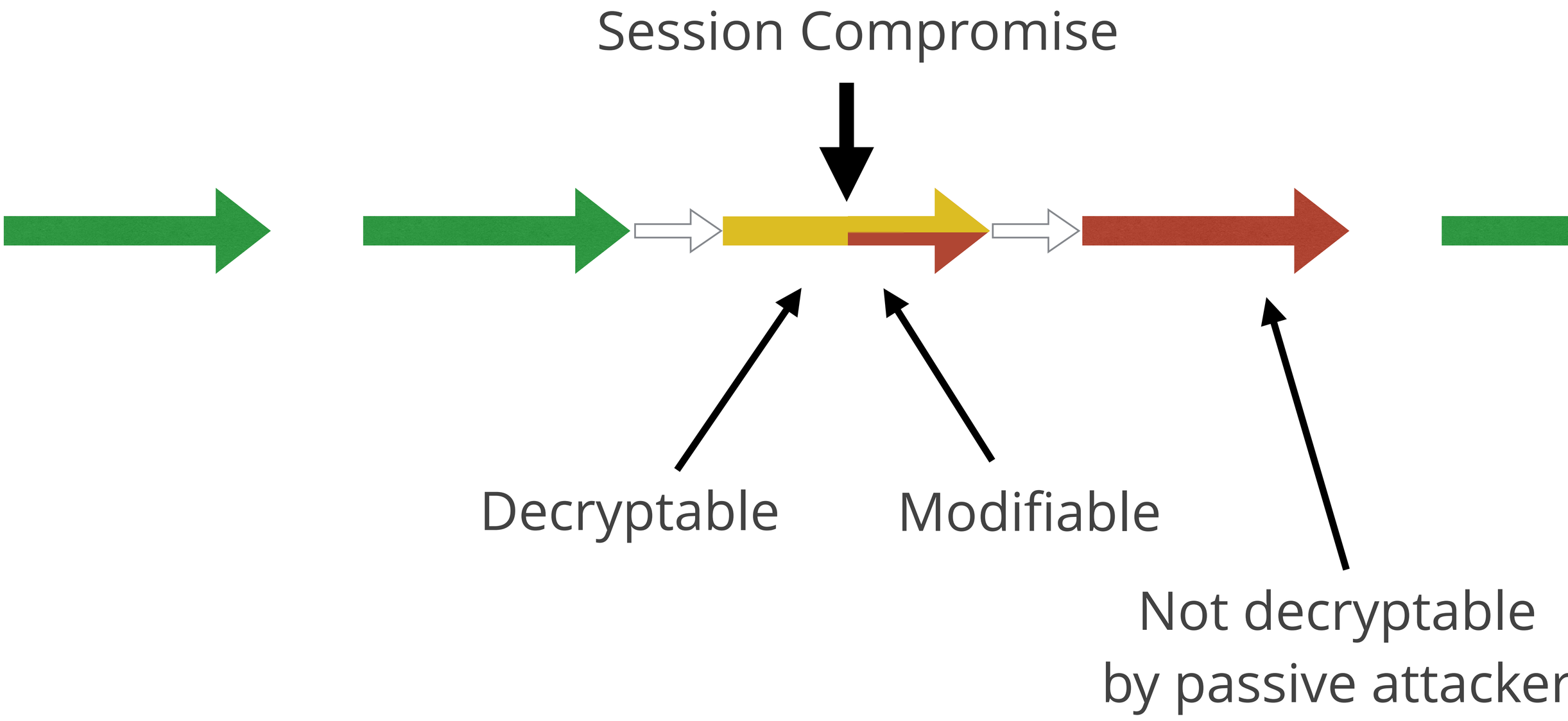
TLS 1.3 PSK-(EC)DHE

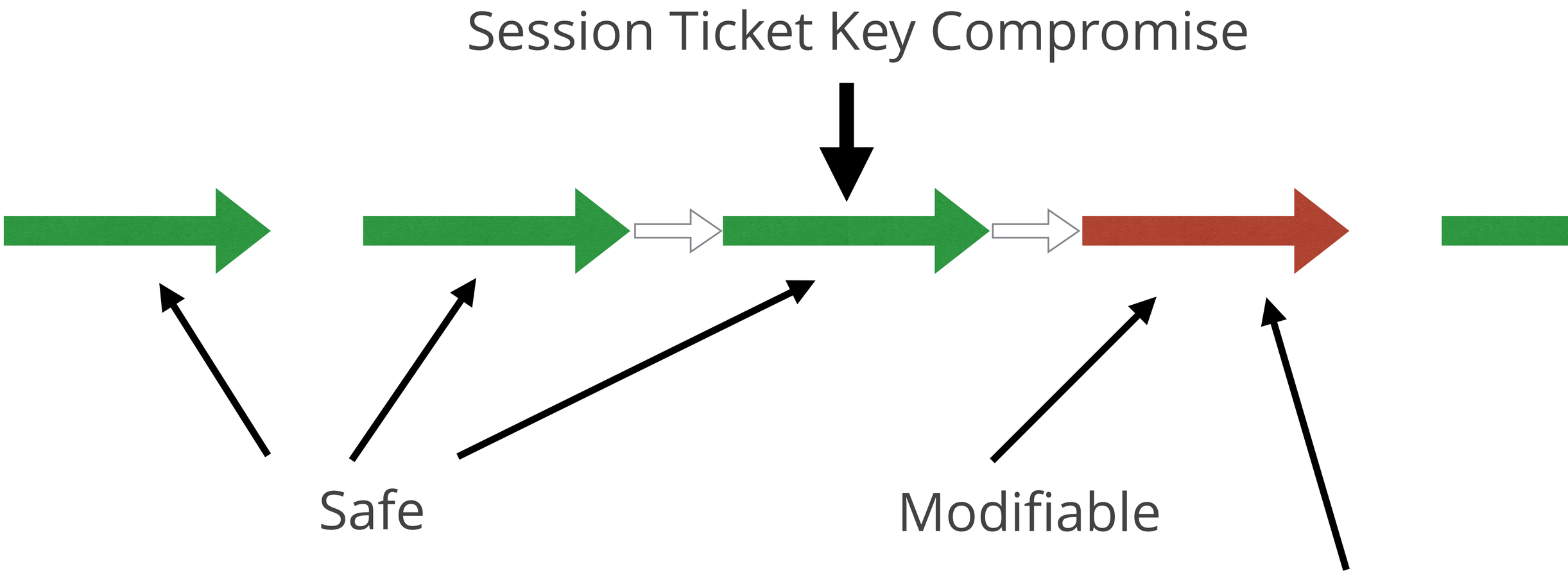
Latency: 1 round-trip



Session Resumption (EC)DHE-PSK

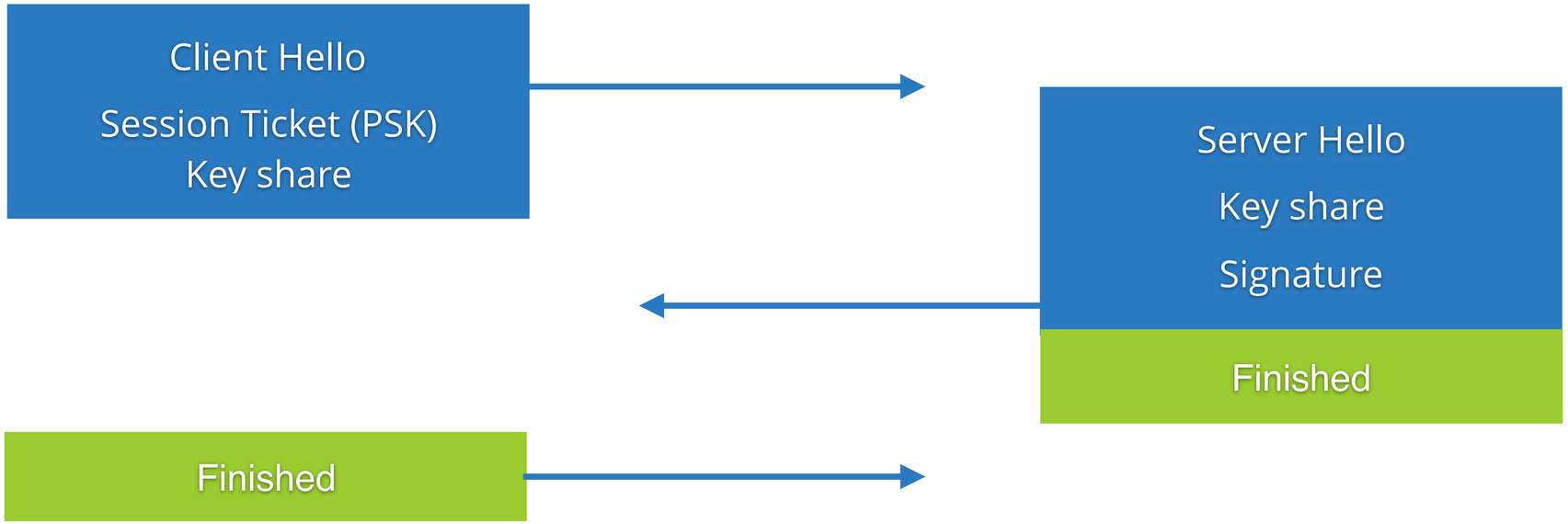



Session Compromise
Decryptable
Modifiable
Not decryptable
by passive attacker

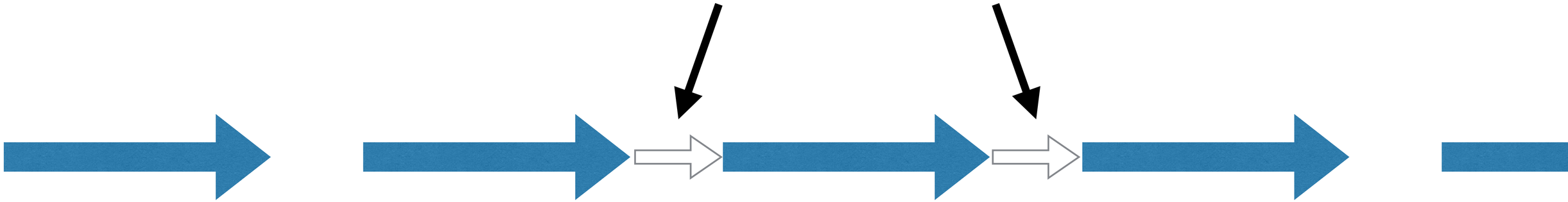

Session Ticket Key Compromise
Safe
Modifiable
Not decryptable
by passive attacker

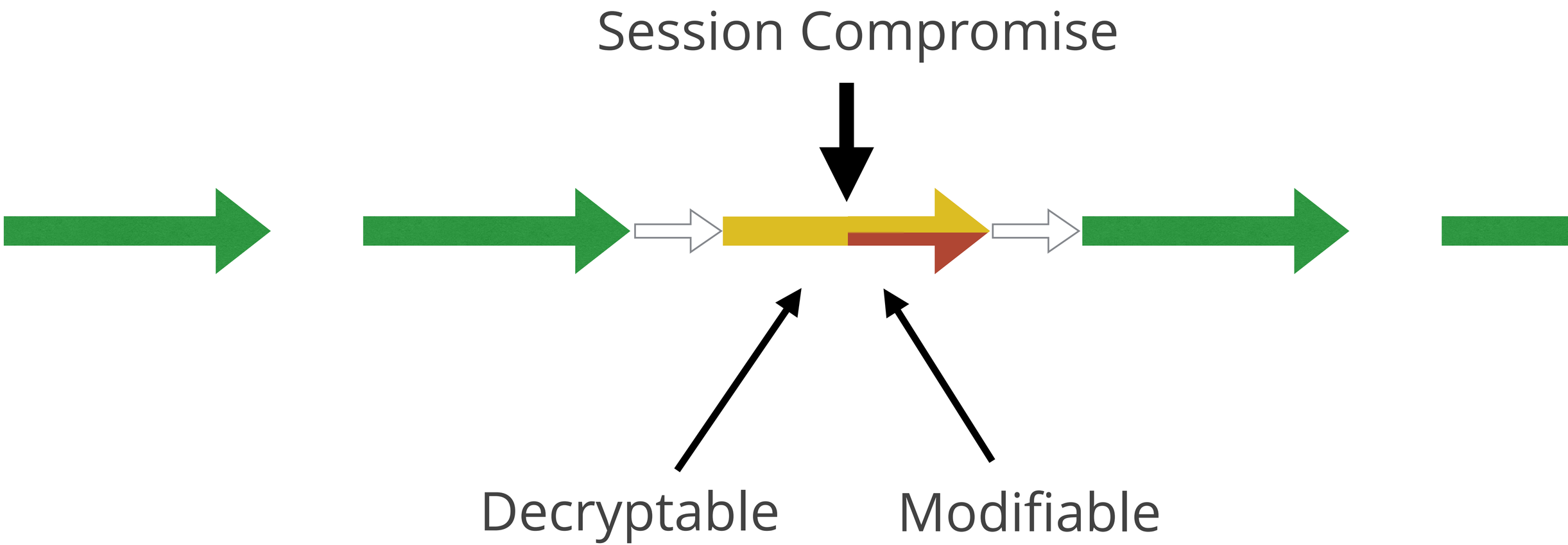
TLS 1.3 PSK-(EC)DHE + Signature

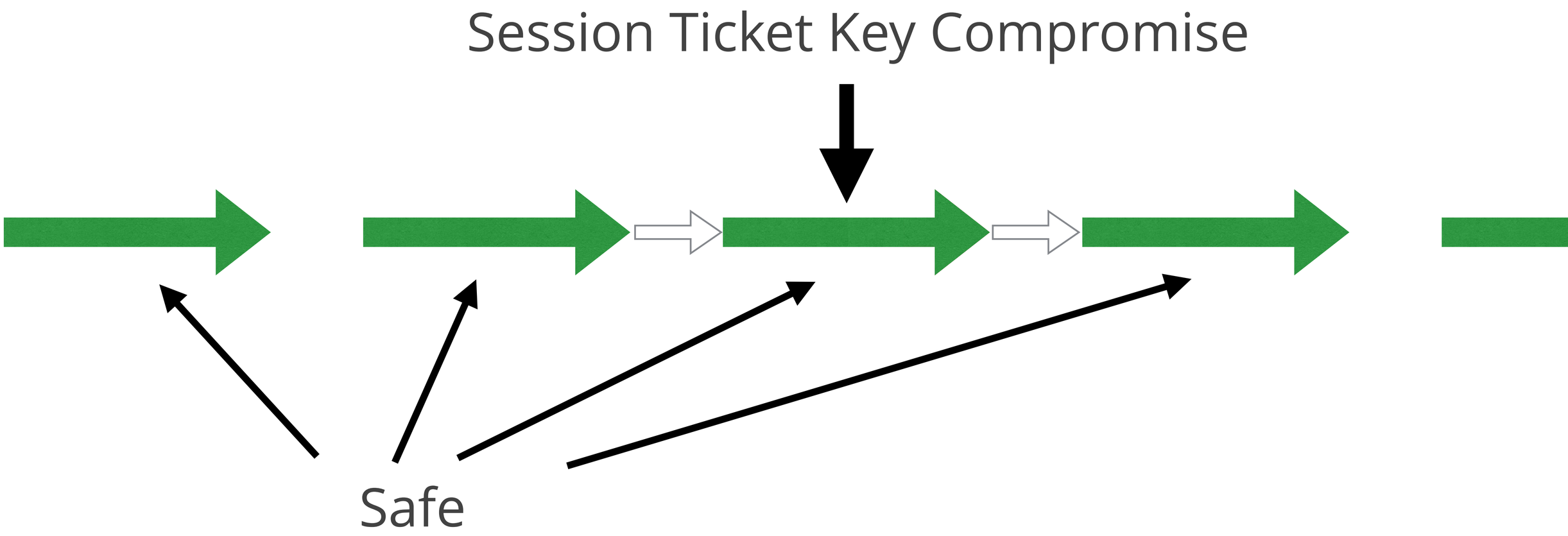
Latency: 1 round-trip



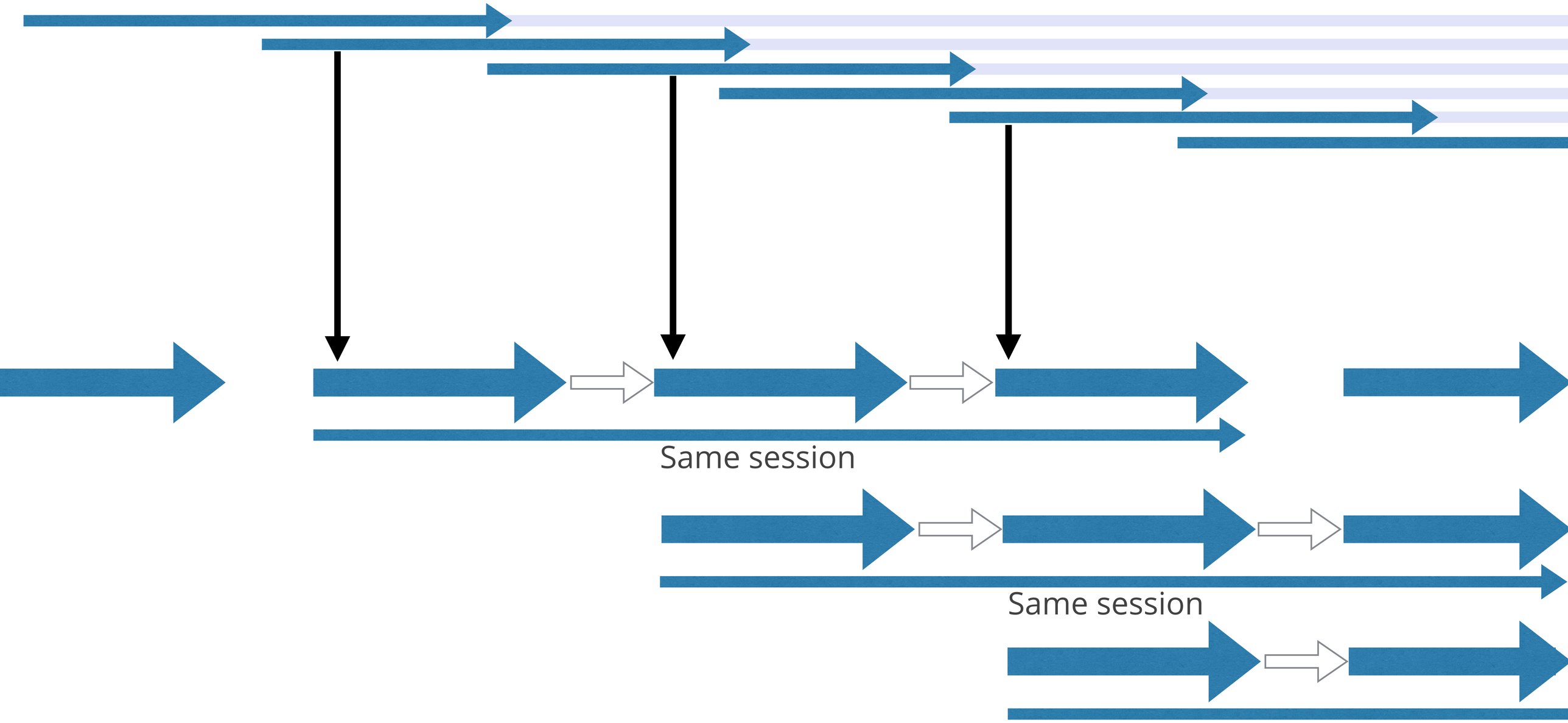
Session Resumption (EC)DHE-PSK + Signature

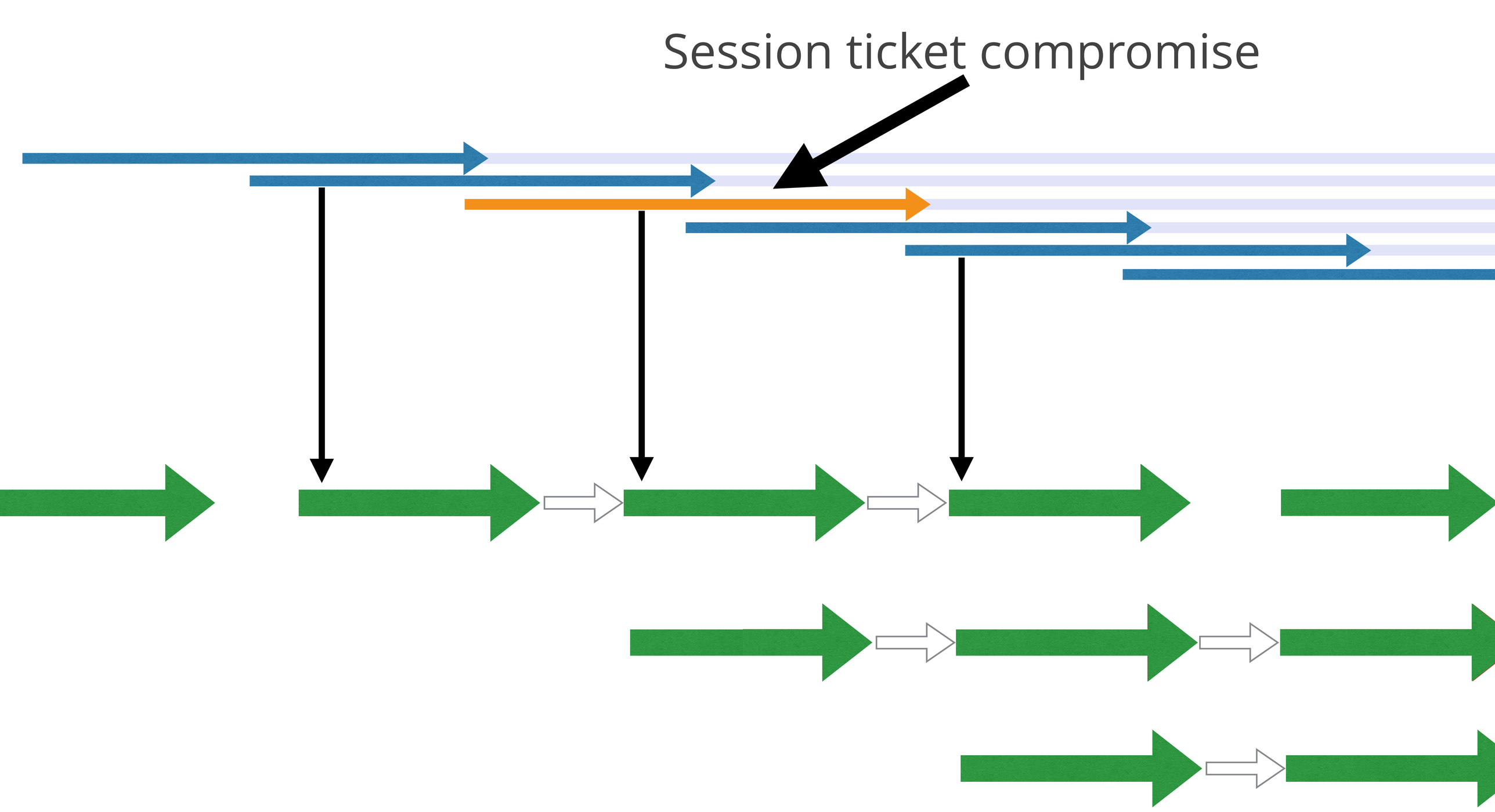



Session Compromise
Decryptable
Modifiable


Session Ticket Key Compromise
Safe

Session ticket lifetime



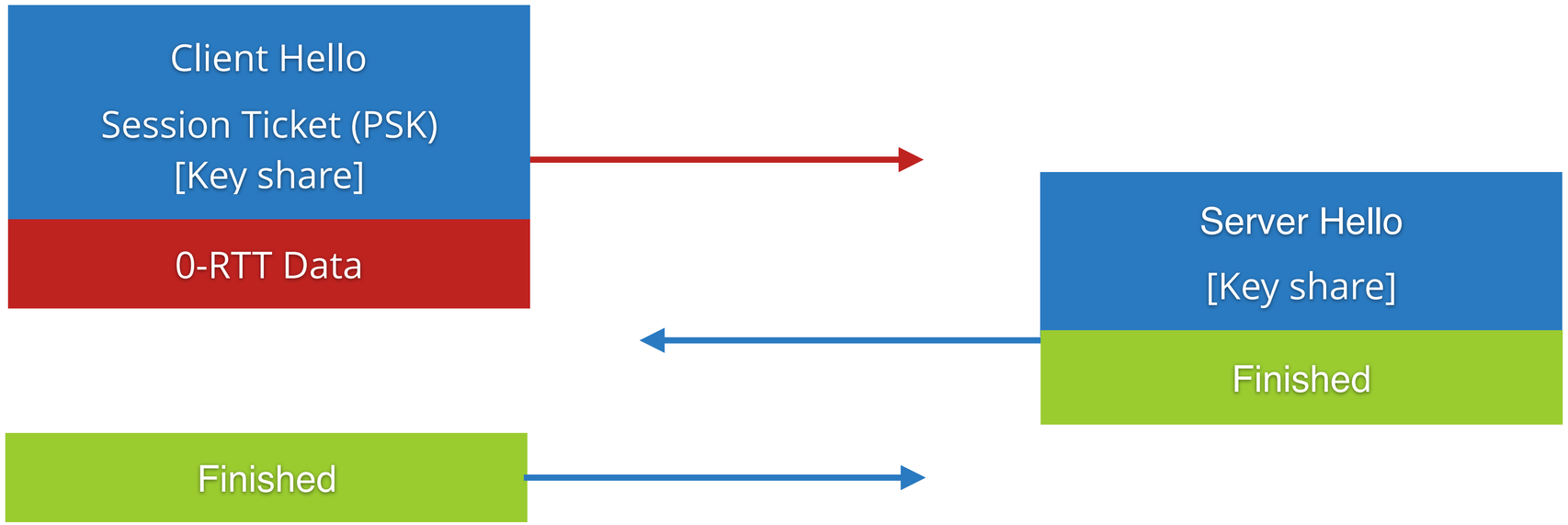

Session ticket compromise

Enter 0-RTT

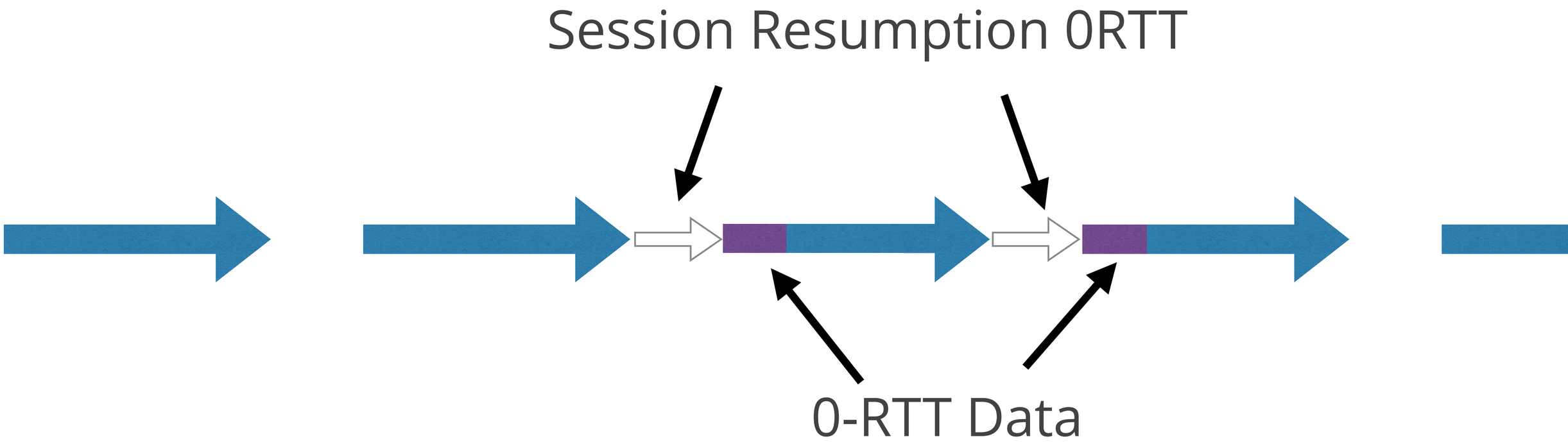
Performance is king

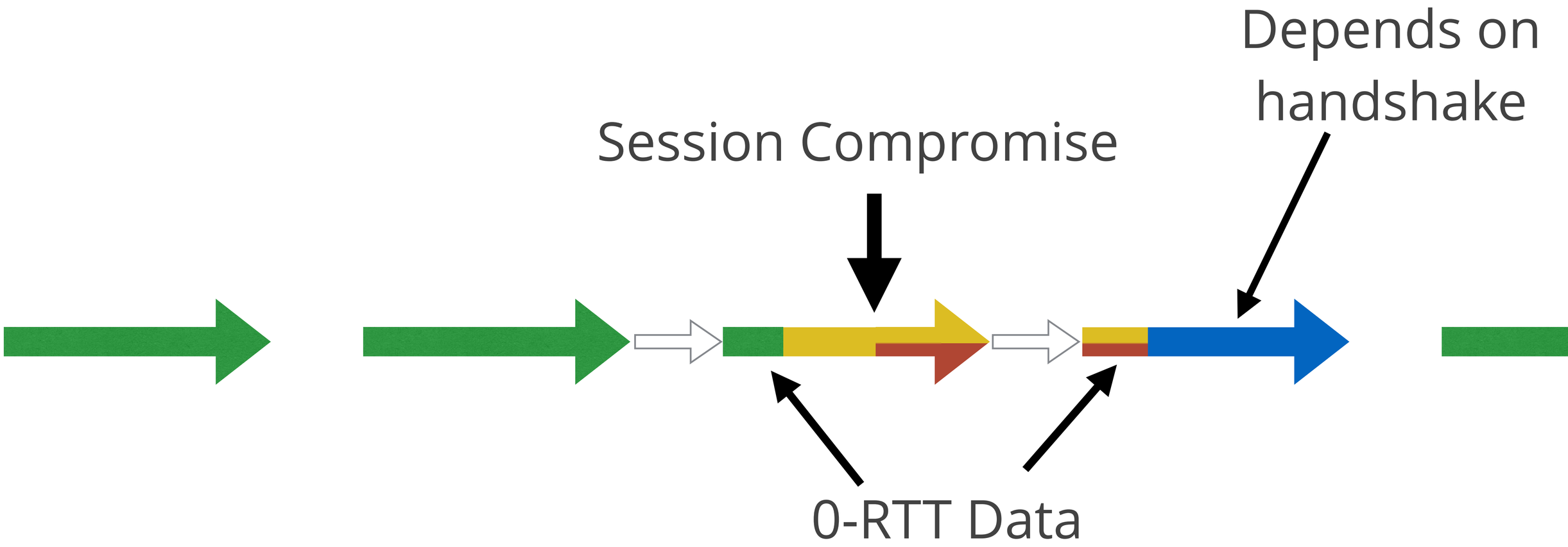
TLS 1.3 0-RTT

Latency: 0 round-trips

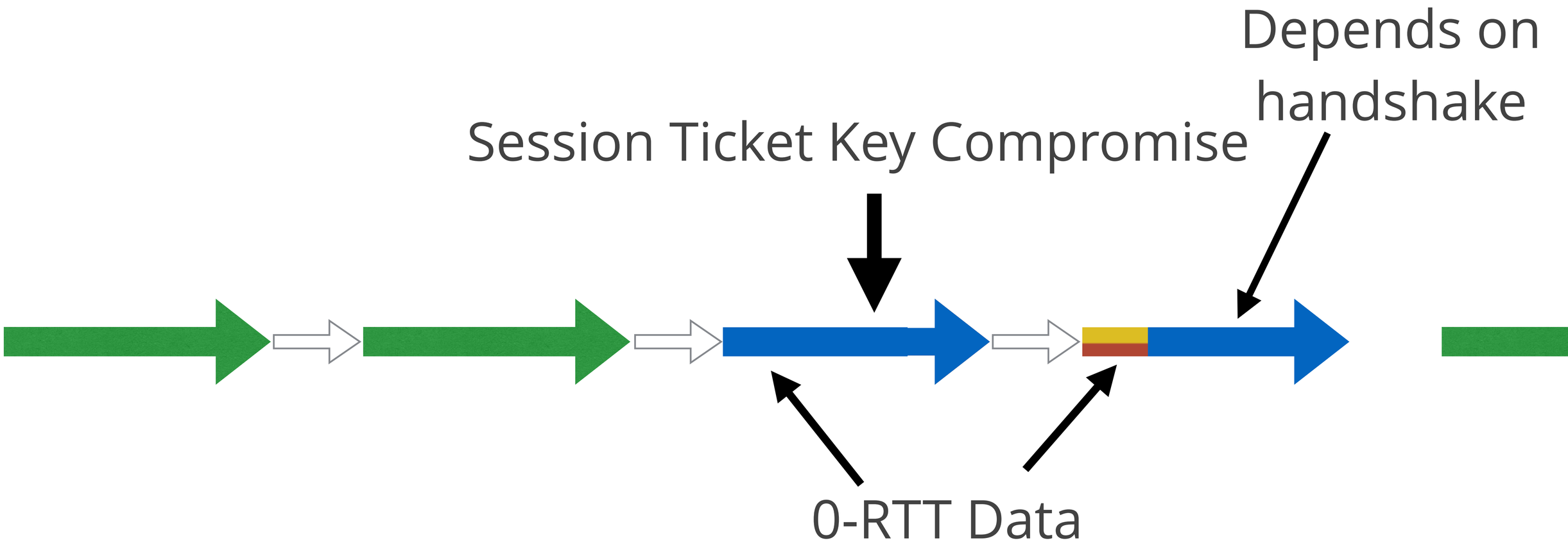


0-RTT data is encrypted with key derived from PSK/Resumption Secret


Session Resumption 0RTT
0-RTT Data



0-RTT Data has same properties as previous connection



0-RTT Data is as safe as the application data from a PSK resumption

Key protection

Three keys

- ECDSA private key
- Session
- Session ticket encryption key

- Client compromise of session has same properties as server compromise

Physical/Logical attack

- ECDSA private key -> Can be stored in HSM, or soft HSM (Keyless SSL)
- Session -> In-memory only or server cache for load balancing
- Session ticket encryption key -> Centrally distributed

- Only works as long as key is “alive”

Cryptographic advancement

- ECDSA private key -> Quantum computing
- Session -> Quantum computing for breaking DH, AES
- Session ticket encryption key -> Symmetric crypto

- Key can be stolen from cryptanalysis of transcript

In conclusion

Summary

- Forward secrecy in TLS is definitely not perfect
- Session resumption provides a complicated tradeoff
- Lifetimes of sessions and session ticket keys are important
  - Measurement needs to be done
- TLS 1.3 provides more protection at the cost of public key operations
- 0-RTT is not “forward secret” it’s as secure as the resumption secret

Nick Sullivan (@grittygrease)

Filippo Valsorda (@filosottile)

Forward Secrecy in TLS

A Systematic Study