Overview of TLS v1.3

What's new, what's removed and what's changed?

OWASP

The Open Web Application Security Project

About Me

OWASP
The Open Web Application Security Project

- Andy Brodie
  - Worldpay Principal Design Engineer.
  - Based in Cambridge, UK.
  - andy.brodie@owasp.org
- Neither a cryptographer nor a mathematician!
  - This means no maths in this presentation.

worldpay

Agenda

OWASP
The Open Web Application Security Project

- History & Background.
- What's Been Removed.
- What's New & Changed.
  - Cipher Suites.
  - Handshake Changes.
  - Hashed-Key Derivation Function.
  - Session Resumption.
- Summary.

OWASP
The Open Web Application Security Project

The Goals and Basics of TLS

HISTORY & BACKGROUND

How SSL became TLS

OWASP
The Open Web Application Security Project

When	Who	What	Comments
1994	Netscape	SSL 1.0 designed.	Never published as security flaws were found internally.
1995	Netscape	SSL v2.0 published.	Flaws found pretty quickly, which led to…
1996	Netscape	SSL v3.0 published.	SSL becomes ubiquitous.
1999	IETF	TLS v1.0 published (SSL v3.1)	Incremental fixes, political name change and IETF ownership.
2006	IETF	TLS v1.1 published (SSL v3.2)	Incremental fixes and capabilities.
2008	IETF	TLS v1.2 published (SSL v3.3)	What we should all be using!
2014	IETF	TLS v1.3 draft 1 (SSL v3.4)	
2018	IETF	TLS v1.3 draft 23	Expires July 15

OWASP
The Open Web Application Security Project

A Client and Server can have a secure conversation over an insecure medium having never met before.

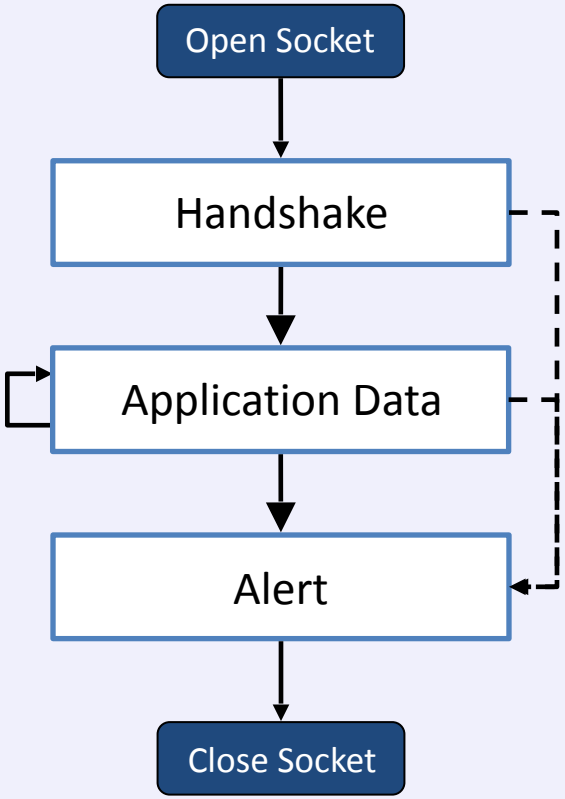
What is a secure conversation?

OWASP
The Open Web Application Security Project

- Privacy
  - Conversation must be encrypted.
  - Prevent eavesdropping attacks.
- Integrity
  - Client & Server must be able to detect message tampering.
  - Prevent Man In The Middle (MITM) attacks.
- Authentication
  - Client needs to trust they're talking to the intended server.
  - Prevent impersonation attacks.

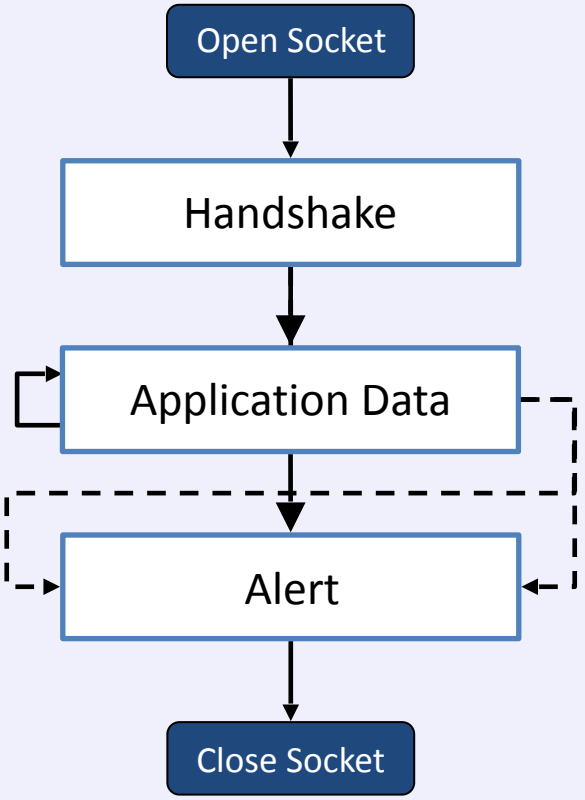
- Privacy
  - Symmetric key encryption for application data.
  - Typically Advanced Encryption Standard (AES).
- Integrity
  - Authenticated Encryption with Additional Data (AEAD).
  - Usually AES-GCM (Galois/Counter Mode) cipher mode.
- Authentication
  - X509 certificates signed by a mutually trusted third party.
  - Typically server authenticated only.

Flow of messages in a TLS conversation



Flow of messages in a TLS conversation

- Handshake
  - Agree a cipher suite.
  - Agree a master secret.
  - Authentication using certificate(s).
- Application Data
  - Symmetric key encryption.
  - AEAD cipher modes.
  - Typically HTTP.
- Alerts
  - Graceful closure, or
  - Problem detected.



https://tlswg.github.io/tls13-spec/draft-ietf-tls-tls13.html

TLS V1.3

Key Goals of TLS v1.3

- Key Goals of TLS v1.3:
  - Clean up - Remove unsafe or unused features.
  - Security - Improve security w/modern techniques.
  - Privacy - Encrypt more of the protocol.
  - Performance – 1-RTT and 0-RTT handshakes.
  - Continuity – Backwards compatibility.

OWASP
The Open Web Application Security Project

WHAT'S REMOVED IN TLS V1.3?

What's removed in TLS v1.3

- Key Exchange
  - RSA
- Encryption algorithms:
  - RC4, 3DES, Camellia.
- Cryptographic Hash algorithms:
  - MD5, SHA-1.
- Cipher Modes:
  - AES-CBC.
- Other features:
  - TLS Compression & Session Renegotiation.
  - DSA Signatures (ECDSA ≥ 224 bit).
  - ChangeCipherSpec message type & "Export" strength ciphers.
  - Arbitrary/Custom (EC)DHE groups and curves.

This has mitigated quite a few attacks…

OWASP
The Open Web Application Security Project

RC4

- Roos's Bias 1995
- Fluhrer, Martin & Shamir 2001
- Klein 2005
- Combinatorial Problem 2001
- Royal Holloway 2013
- Bar-mitzvah 2015
- NOMORE 2015

RSA-PKCS#1 v1.5 Encryption

- Bleichenbacher 1998
- Jager 2015
- DROWN 2016

Renegotiation

- Marsh Ray Attack 2009
- Renegotiation DoS 2011
- Triple Handshake 2014

3DES

- Sweet32

AES-CBC

- Vaudenay 2002
- Boneh/Brumley 2003
- BEAST 2011
- Lucky13 2013
- POODLE 2014
- Lucky Microseconds 2015

Compression

- CRIME 2012

MD5 & SHA1

- SLOTH 2016
- SHAttered 2017

WHAT'S NEW AND CHANGED?

What's New and Changed?

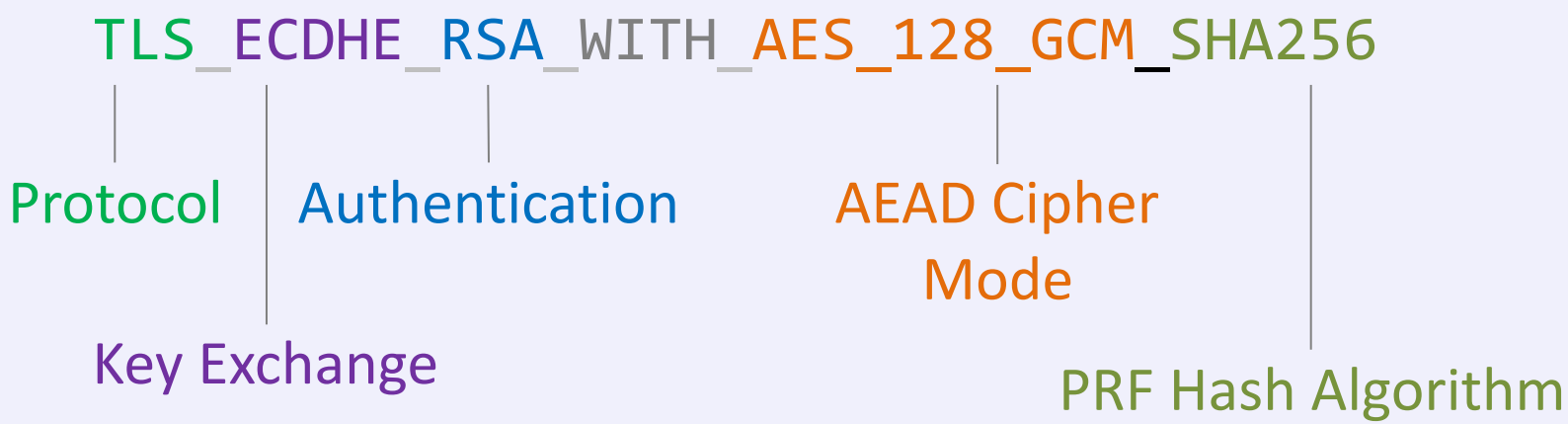
OWASP
The Open Web Application Security Project

- Cipher Suites.
- Handshake.
- Hashed-Key Derivation Function (HKDF).
- Key Schedule.
- Sessions.

OWASP
The Open Web Application Security Project

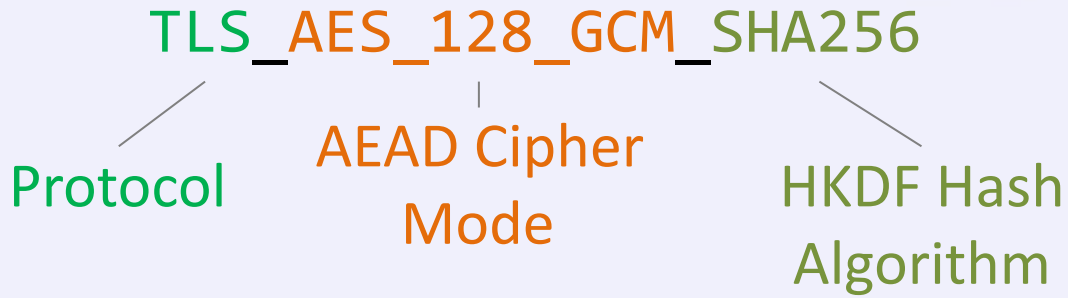
CIPHER SUITES

TLS v1.2 provides 37 Cipher Suites



- TLS 1.2 specifies 37 cipher suites.
  - Add previous versions in: 319 cipher suites.

TLS 1.3 Cipher Suites



- TLS v1.3 supports 5 cipher suites.
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  - TLS_AES_128_CCM_SHA256
  - TLS_AES_128_CCM_8_SHA256

OWASP
The Open Web Application Security Project

- Key Exchange algorithms:
  - DHE & ECDHE
    - Only 5 ECDHE curve groups supported
    - Only 5 DHE finite field groups supported
  - Pre-Shared Key (PSK)
  - PSK with (EC)DHE

- Digital Signature (Authentication) algorithms:
  - RSA (PKCS#1 variants)
  - ECDSA / EdDSA

OWASP
The Open Web Application Security Project

HANDSHAKE CHANGES

TLS Handshake

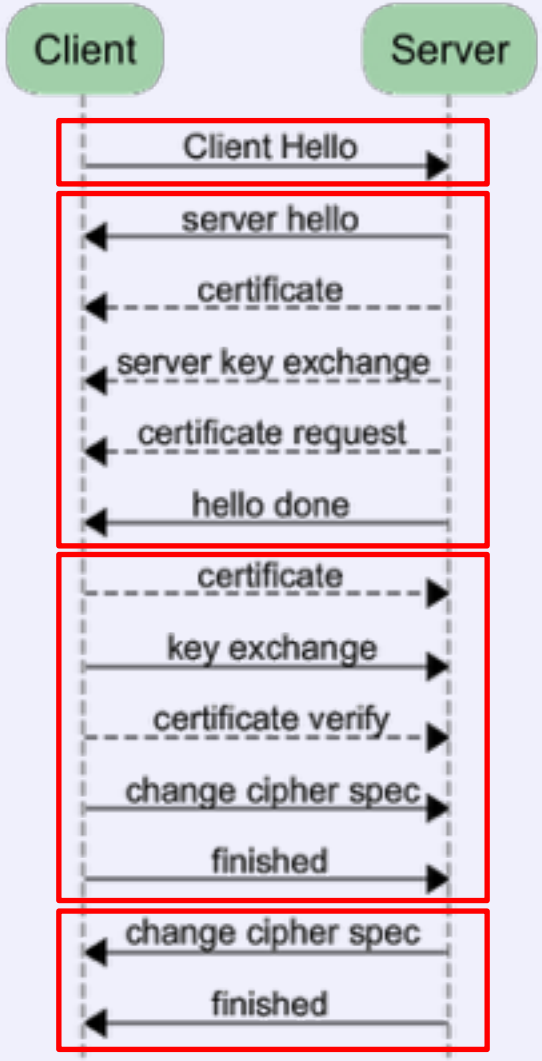
OWASP
The Open Web Application Security Project

- The handshake has three goals:
  - Agree a cipher suite.
  - Agree a master secret.
  - Establish trust between Client & Server.

- Optimise for the most common use cases.
  - Everyone* wants a secure conversation.
  - Same cipher suites used across websites repeatedly.
  - Clients connect to the same sites repeatedly.

* ok, *almost* everyone!

TLS 1.2 Handshake

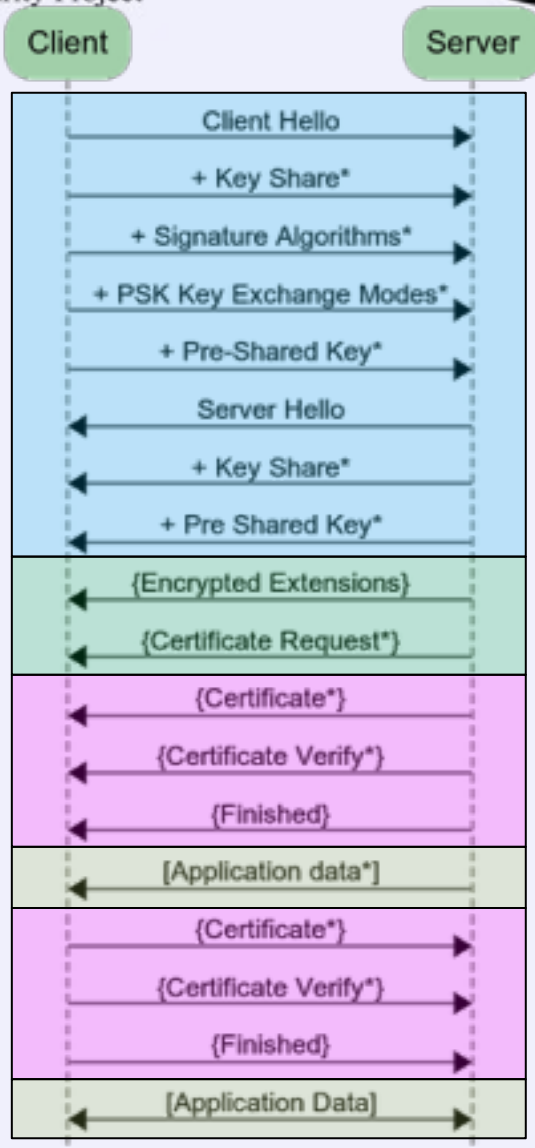


Three Stages of a TLS 1.3 Handshake

OWASP
The Open Web Application Security Project

Key Exchange

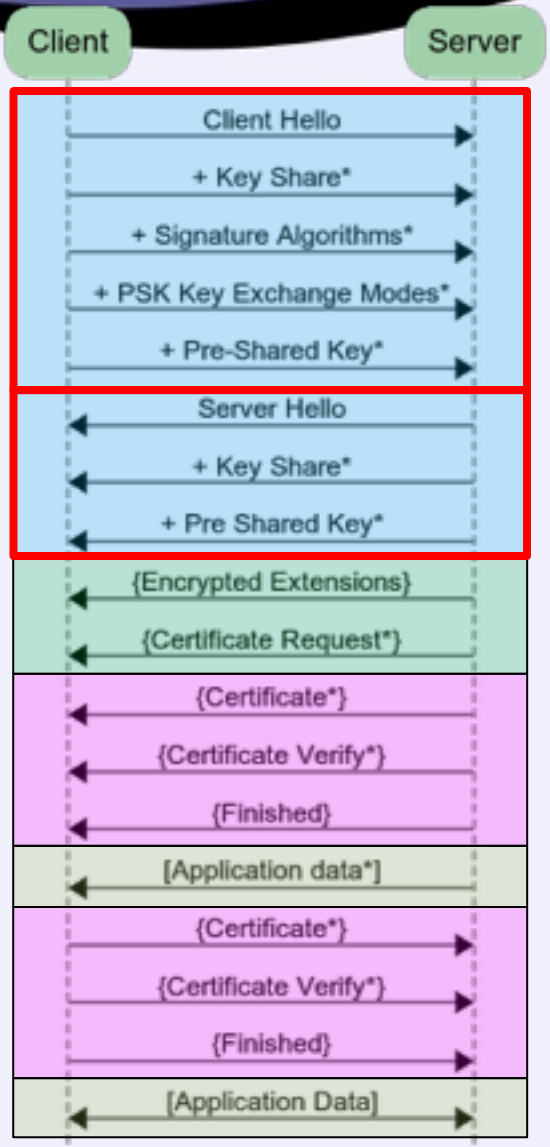
Authentication



Server Parameters

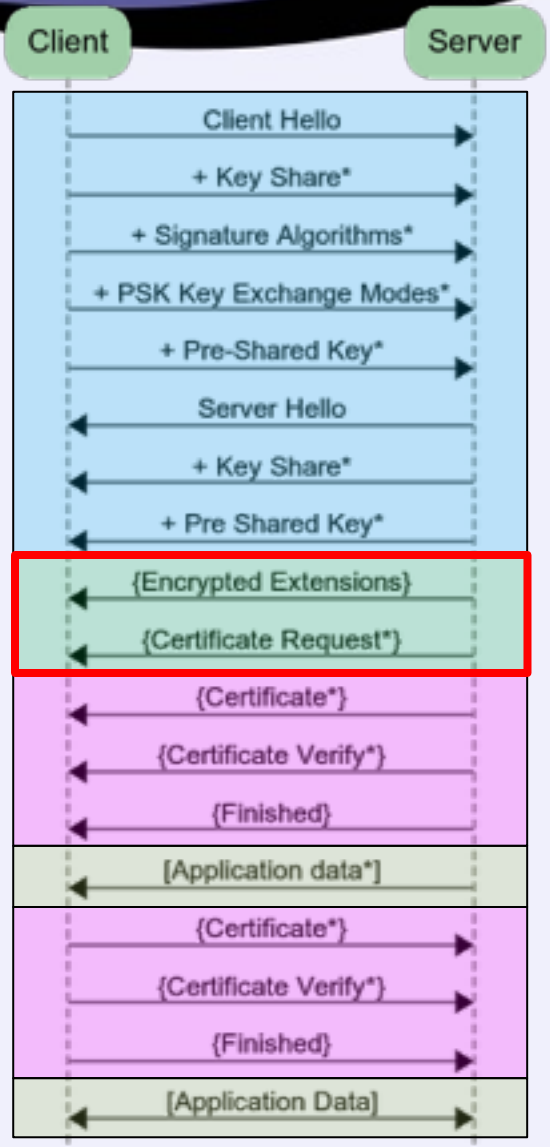
Client now makes assumptions about server support.

- Client sends:
  - Cipher Suite options.
  - List of supported groups/curves.
  - (EC)DHE Key Share(s).
- Server sends:
  - Cipher suite selection.
  - (EC)DHE Key Share
- Client and Server now share a key.



The rest of the handshake is encrypted.

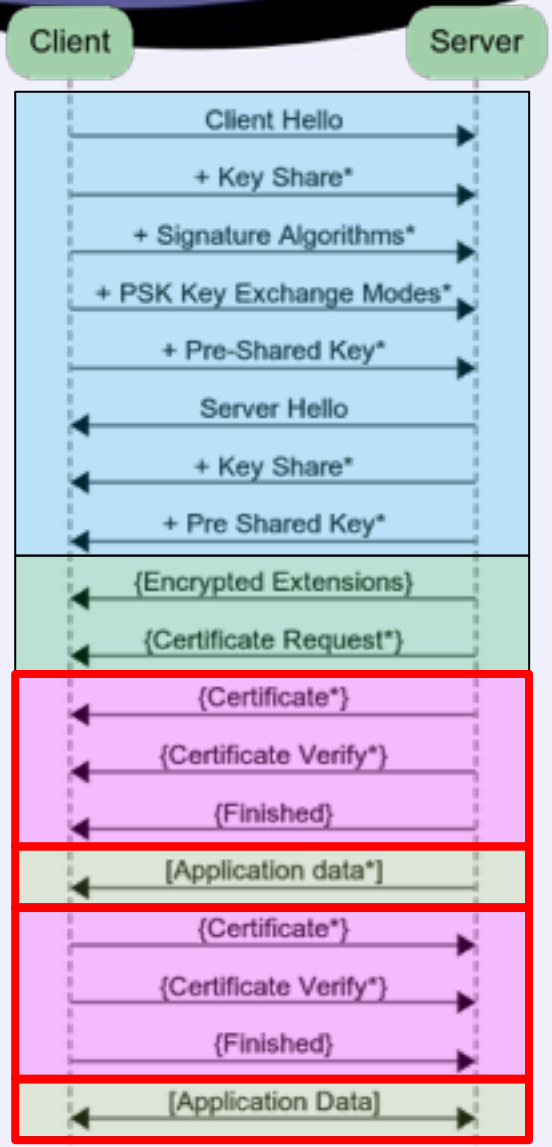
- Server sends:
  - Encrypted Extensions
    - Server Name
    - Message Length
    - *…and optionally many more*
  - Certificate Request
    - Supported signature algorithms.



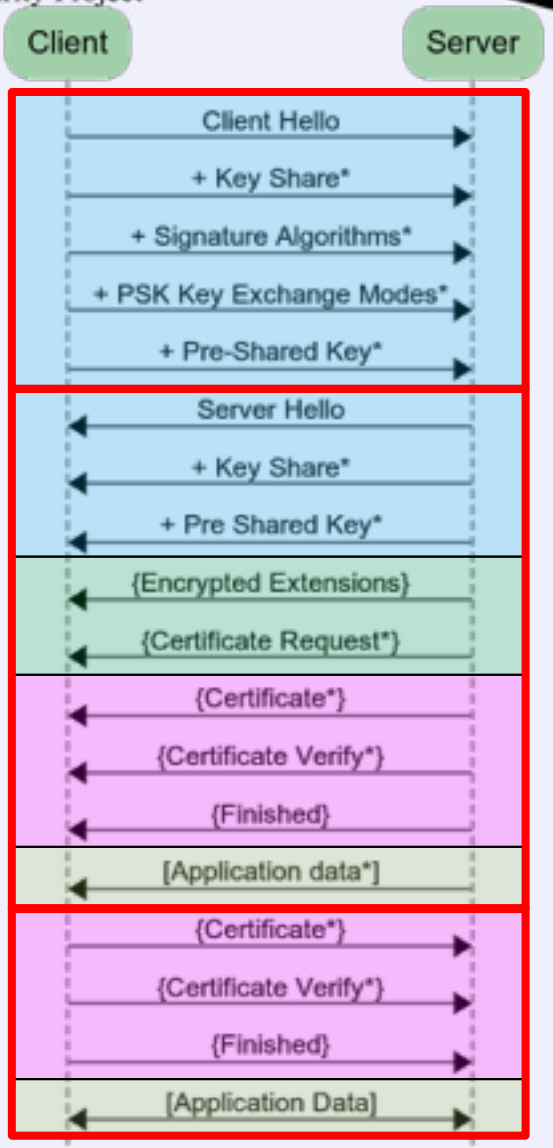
Client now makes assumptions about server support.

OWASP
The Open Web Application Security Project

- Server sends:
  - Certificate.
  - Proof of private key possession.
  - Finished.
  - Application Data
- Client responds:
  - Certificate.
  - Proof of private key possession.
  - Finished.



Efficiency Gains



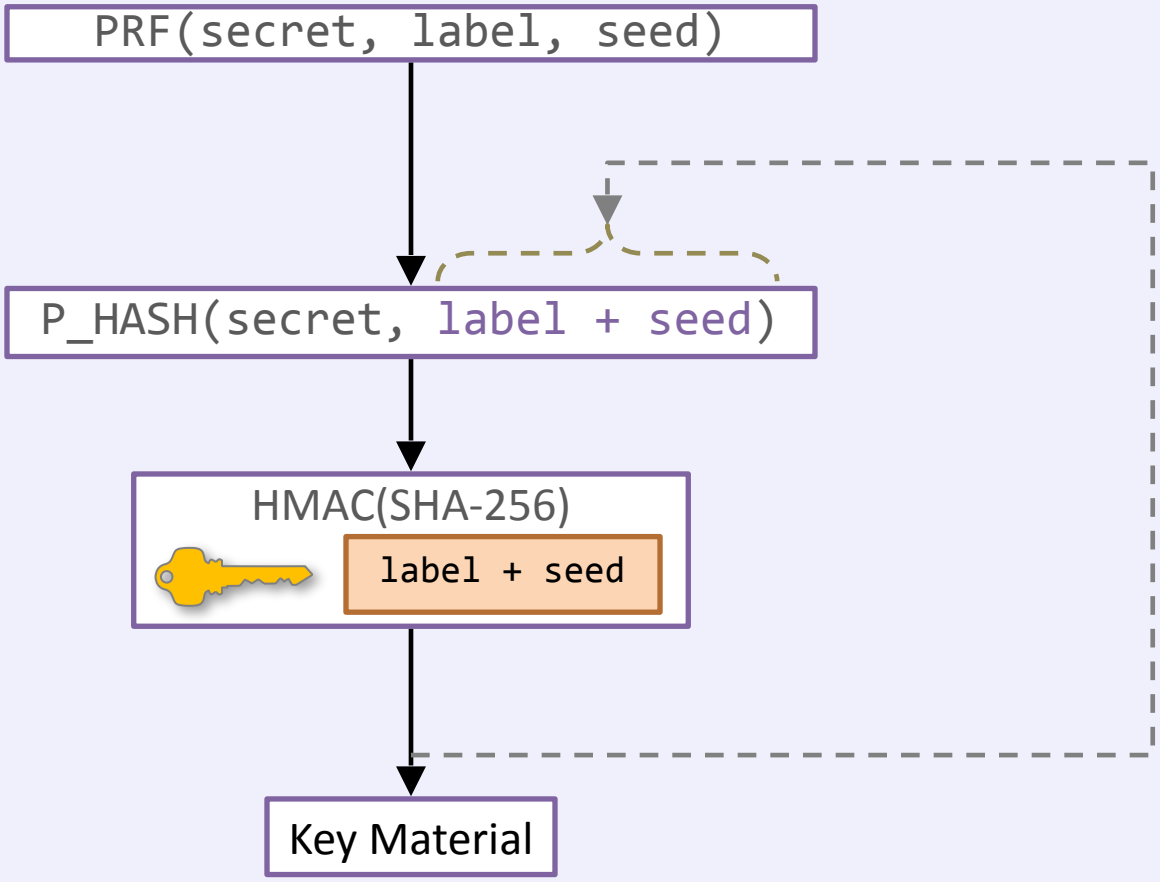
GENERATING KEYS USING HKDF

OWASP
The Open Web Application Security Project

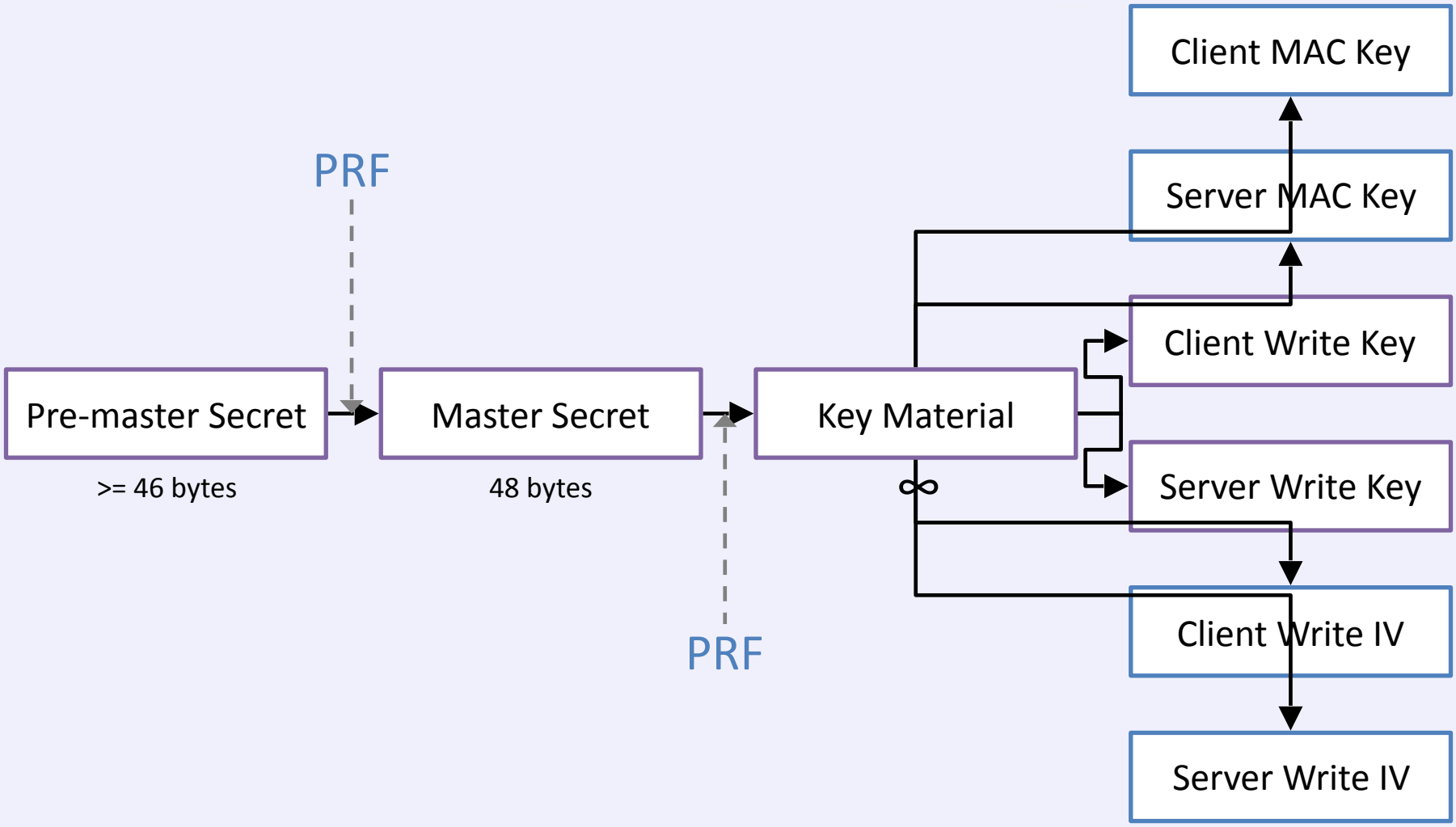
HKDF (RFC5869) HMAC-based Key Derivation Function

- TLS <= v1.2 defines PRF algorithm.
- TLS v1.3 replaces this with HKDF.
  - HKDF encapsulates how TLS uses HMAC.
  - Re-used in other protocols.
  - Separate cryptographic analysis already done.
- Provides 2 functions:
  - Extract - create a pseudo-random key from inputs.
  - Expand - create more keys from the extract output.
- HMAC is integral to HKDF.
  - HMAC requires the Cryptographic Hash algorithm specified in the cipher suite (SHA256 or SHA384).

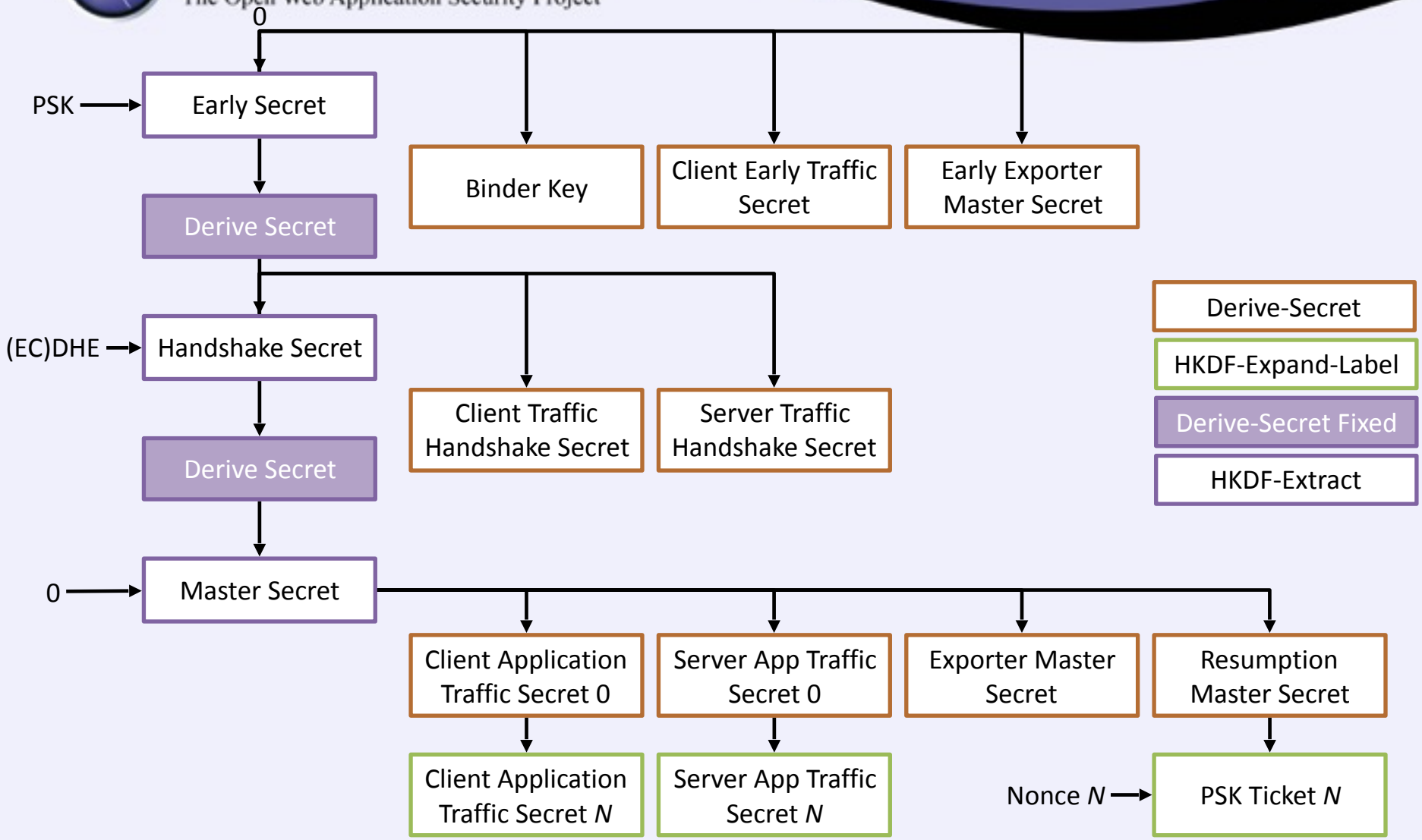
How the PRF is implemented



TLS <= v1.2 Creating Key Material from a master secret



TLS v1.3 Key Schedule Generation



What's the difference?

PRE-SHARED KEYS AND SESSIONS

Why do we need sessions?

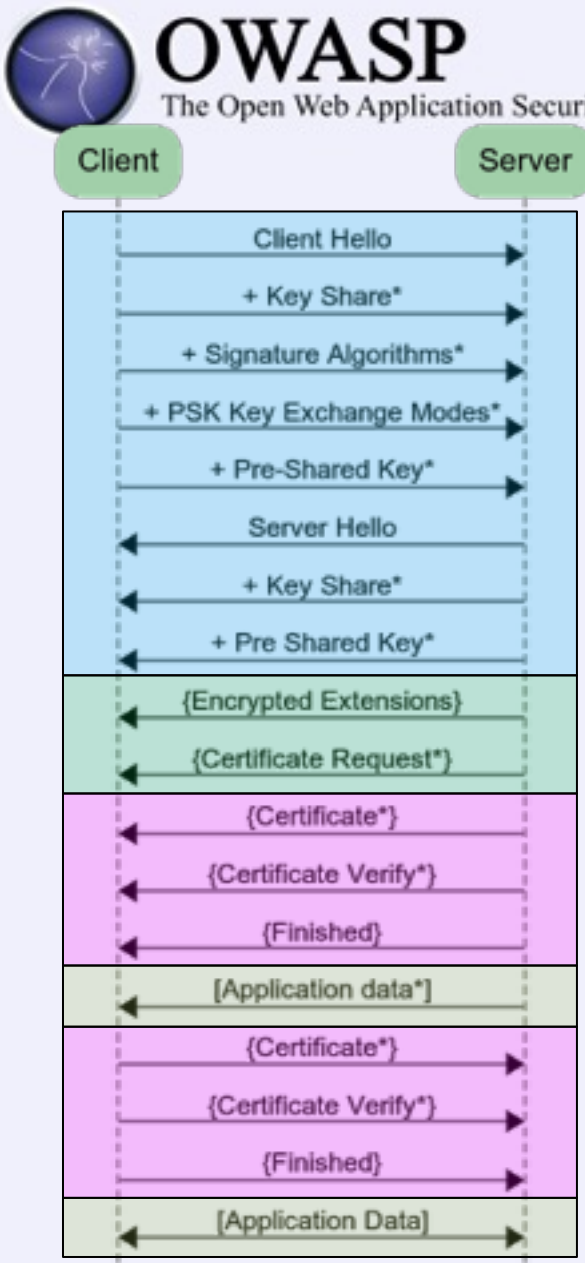
OWASP
The Open Web Application Security Project

- Full handshakes are expensive.
  - Key generation.
  - Server (& Client) Authentication.
- Many HTTP clients need it.
  - Download web page resources (JS, CSS, images).
  - Dynamic web pages (XHR).
  - May not be feasible to keep connection open.

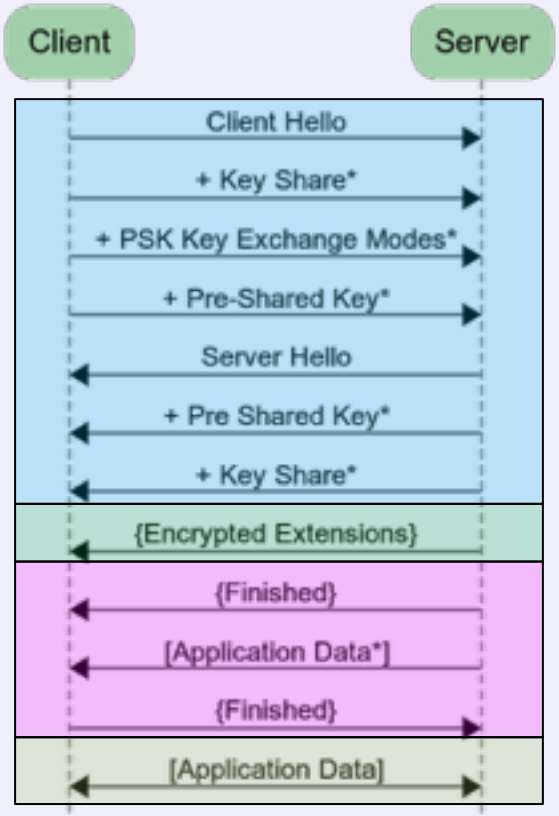
How do we establish a PSK?

- Out-of-band
  - Added to TLS in 2006 via RFC4279.
- During Handshake
  - Client announces it supports session resumption.
  - Server provides a PSK *identities* during handshake.
- After handshake, Server sends “New Session Ticket”
  - Contains PSK identity, nonce and max age.
  - The PSK is derived from master secret.
  - Server can send multiple tickets.

So, TLS v1.3 supports PSK-based session resumption



becomes…



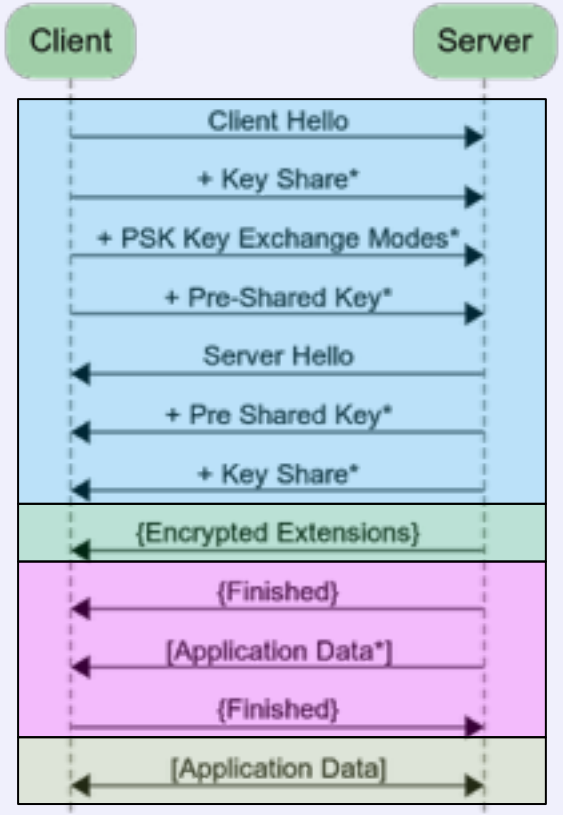
OWASP
The Open Web Application Security Project

- PSK means the key is known to both sides.
  - Does this mean Client can send data immediately?
  - Can we have a zero round trip time handshake?

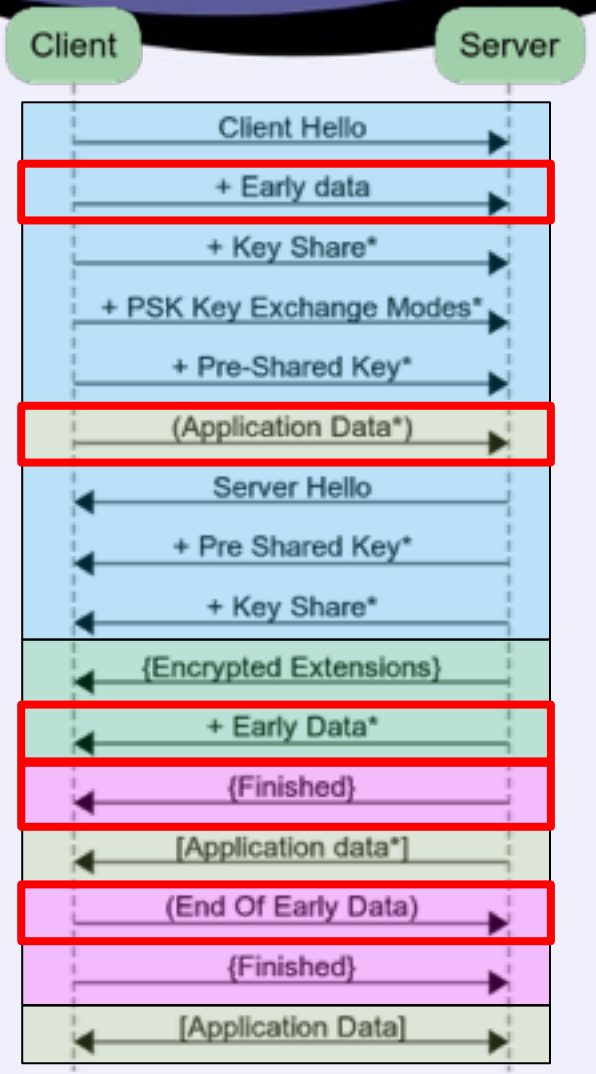
Yes, we can!

- But…
  - No forward secrecy for the “early data” sent by client.
  - No guarantees of non-replay.

So, TLS v1.3 supports PSK-based session resumption



becomes…



Extensions… Extensions everywhere!

BACKWARDS COMPATIBILITY

Backwards Compatibility

OWASP
The Open Web Application Security Project

- Backwards compatibility is important
  - TLS v1.3 clients need to talk to TLS v1.2 servers.
  - TLS v1.2 clients need to talk to TLS v1.3 servers.
- Structure of Hello messages is maintained.
  - 12 extensions defined in the RFC.
  - 9 extensions defined in other RFCs.
- E.g. server key exchange message replaced with key_share extension.

All the extensions

OWASP
The Open Web Application Security Project

Extension	TLS 1.3
server_name [RFC6066]	CH, EE
max_fragment_length [RFC6066]	CH, EE
status_request [RFC6066]	CH, CR, CT
supported_groups [RFC7919]	CH, EE
signature_algorithms [RFC5246]	CH, CR
use_srtp [RFC5764]	CH, EE
heartbeat [RFC6520]	CH, EE
application_layer_protocol_negotiation [RFC7301]	CH, EE
signed_certificate_timestamp [RFC6962]	CH, CR, CT
client_certificate_type [RFC7250]	CH, EE
server_certificate_type [RFC7250]	CH, CT
padding [RFC7685]	CH
key_share	CH, SH, HRR
pre_shared_key	CH, SH
psk_key_exchange_modes	CH
early_data	CH, EE, NST
cookie	CH, HRR
supported_versions	CH
certificate_authorities	CH, CR
oid_filters	CR
post_handshake_auth	CH

Acronym	Message
CH	Client Hello
SH	Server Hello
EE	Encrypted Extensions
CT	Certificate
CR	Certificate Request
NST	New Session Ticket
HRR	Hello Retry Request

- Protocol Version is mentioned in every message.
  - Now deprecated/fixed to old version values
  - Handshake claims 1.2, App Data claims 1.0.
  - New extension specifies list of supported versions.

- Fixed values to prevent downgrade attacks.
  - Server “Random” has fixed last 8 bytes
    - `DOWNGRD[0x01]` for TLS 1.2 clients.
    - `DOWNGRD[0x00]` for <= TLS 1.1 clients.

And that's TLS v1.3!

OWASP
The Open Web Application Security Project

- Removed
  - Anything that was unused, unsafe or didn't offer significant value.
- Added
  - Handshake encryption.
  - 1-RTT and 0-RTT PSK / Session Resumption.
- Changed
  - Cipher Suites.
  - Handshake.
  - Hashed-Key Derivation Function (HKDF).
  - Key Schedule.
  - Sessions.

OWASP
The Open Web Application Security Project

THANK YOU FOR LISTENING!

My own thoughts?

OWASP
The Open Web Application Security Project

- The Good:
  - Massive efficiency gains*.
  - Fewer choices for Client & Server means reduced attack vectors.

- The Bad:
  - “Extensions…. extensions everywhere” (21)
  - A lot of added complexity for backwards compatibility.
  - Specification consumability is questionable.

* 0-RTT has a “whiff of future regret” about it.

Unused Slides

APPENDIX

What's the point of the master secret?

- Client and Server need:
  - Keys for symmetric encryption.
  - Initialisation Vectors for AEAD Cipher Modes.
- Keys & IVs generated from a master secret.
- TLS defines a "Key Schedule"
  - How HKDF algorithm is used.
  - How to generate an infinite amount of secure key material.
- So, how does HKDF work?

HMAC-based Extract-and-Expand Key Derivation Function

HMAC (IS THE NEW PRF)

What is HKDF used for?

OWASP
The Open Web Application Security Project

- Key Schedules
  - Handshake Secrets.
  - Early Traffic Secrets.
  - Master Secret.
  - Application Data Secrets.
  - Initialisation Vectors.

- Transcript Hashes
  - Certificate Verification.
  - Handshake “Finished” Keys.

- TLS <= v1.2 defines PRF algorithm.
  - HKDF encapsulates how TLS uses HMAC.
  - Re-used in other protocols.
  - Separate cryptographic analysis already done.

- Provides 2 functions:
  - Extract - create a pseudo-random key from inputs.
  - Expand - create more keys from the first key.

- HMAC is integral to HKDF.

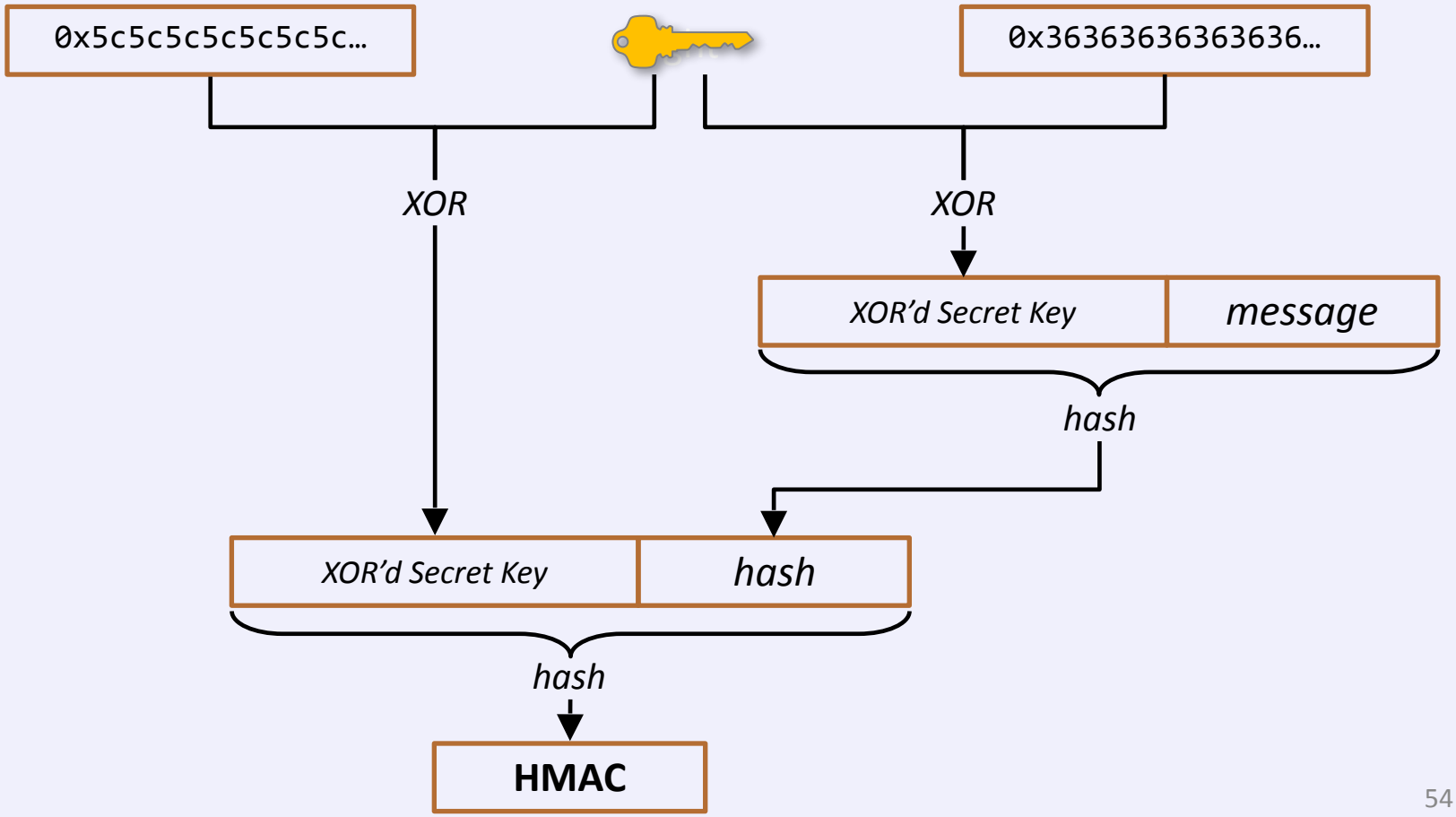
OWASP
The Open Web Application Security Project

- It creates a Message Authentication Code using:
  - Message data.
  - A shared key.
  - A cryptographic hash algorithm (set in cipher suite).
    - SHA256 or SHA384.

$$HMAC(K, m) = H((K' \oplus opad)||H((K' \oplus ipad)||m))$$

OWASP
The Open Web Application Security Project

- Keyed-Hash Message Authentication Code



HKDF Extract & Expand

OWASP
The Open Web Application Security Project

- Extract
  - Creates a Pseudo-Random Key (PRK)

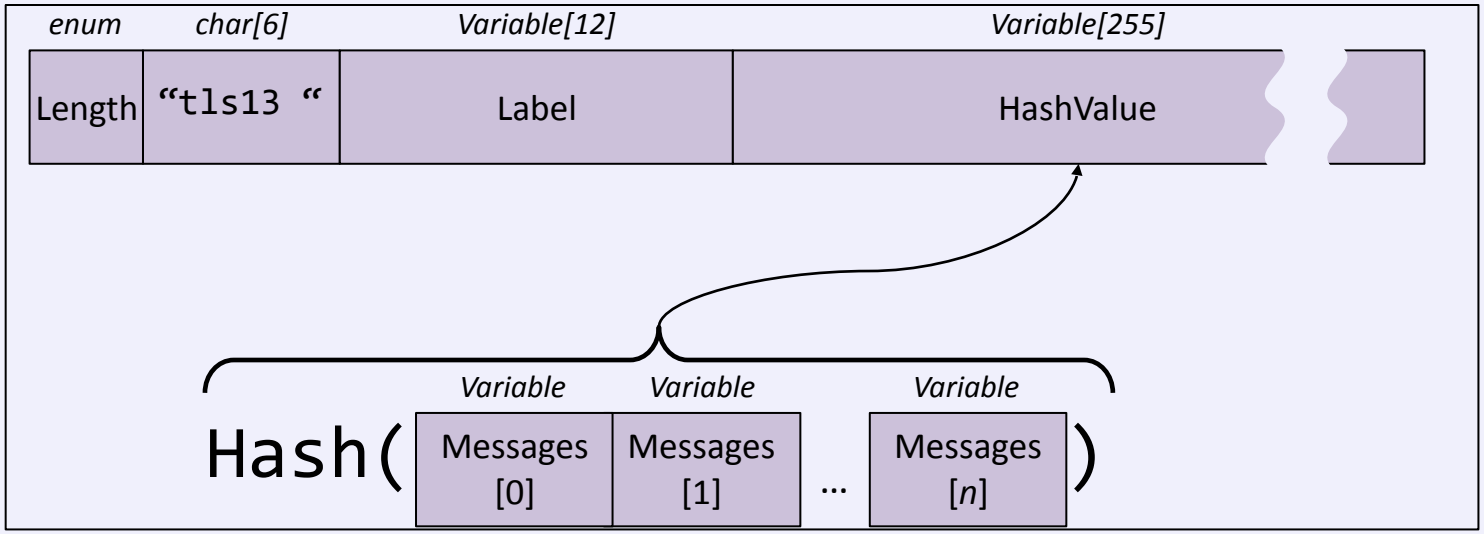
```
HKDF-Extract(salt, IKM) -> PRK
PRK = HMAC-Hash(salt, IKM)
```

Expand

- Creates infinite key material from the PRK.
- Iteratively calls HMAC with an increasing counter.

```
HKDF-Expand(PRK, info, L) -> OKM
T(0) = empty string (zero length)
T(1) = HMAC-Hash(PRK, T(0) | info | 0x01)
T(2) = HMAC-Hash(PRK, T(1) | info | 0x02)
…
```

```
Derive-Secret(Secret, Label, Messages[]) =
    HKDF-Expand(
        Secret,
```




```
        Hash.Length)
```

Client says Hello

OWASP
The Open Web Application Security Project

CH Parameter	Description	Notes
Protocol Version	Legacy slot for protocol version.	0x0303 TLS v1.2
Random	The Client Random	No more Unix time
Session ID	Session ID	Forced 0 byte length
Cipher Suites	Symmetric cipher options	One of Five
Compression Methods	N/A	Must specify not supported.
Supported Versions	List of uint16	0x0304 (TLS v1.3)
Signature Algorithms	List of supported	Required for Client Cert Auth
Negotiated Groups	Required for (EC)DHE	
Key Share	Required for (EC)DHE	
Pre-Shared Key	Required for PSK (incl. session resumption)	

First Contact: Client Hello

- Client initiates the connection.
- Contents:
  - Version (Legacy)
    - Unused, must be set to 0x0303 (TLS v1.2)
  - Client Random
    - Used in PRF to create master secret.
  - Session ID (Legacy)
    - Ignored, kept for backwards compatibility.
  - Supported Cipher Suites
    - What cipher suites this client can support.
  - Compression (Legacy)
    - Ignored, kept for backwards compatibility
  - Extensions (TLS v1.3)
    - List of supported TLS versions (mandatory)
  - Extensions (Others)
    - Other extensions, e.g. SNI

- The problem with RSA key exchange:
  - The pre-master secret is always encrypted with the public certificate key in the certificate.
  - The certificate doesn't change (often).
  - If the private key was ever compromised, Eve could read every conversation.

SHA-1 & MD5 Weaknesses

OWASP
The Open Web Application Security Project

- Cryptographic hash algorithm features:
  - Find any *m* and *m'* such that *hash(m)=hash(m')*
  - Find *m'* given m such that *hash(m)=hash(m')*
  - Find *m* given *x* such that *hash(m)=x*

- MD5 vulnerabilities:
  - Collision attack – done.
  - Theoretical attack on pre-image (2^{123} operations).

- SHA-1 vulnerabilities:
  - Collisions attack – given 6500 CPU-years or 1000-GPU years.
  - Reduced cryptographic strength from 160 bits to 77 bits.

Renegotiation Attacks [RRDO10]