OpenSSL Foundation

OpenSSL's TLS Implementation

Inside the State Machine and Record Layer

Matt Caswell · OpenSSL Foundation · 13 October 2026

34,000 lines

SSL/STATEM/

24,700 lines

SSL/RECORD/

9,700 lines

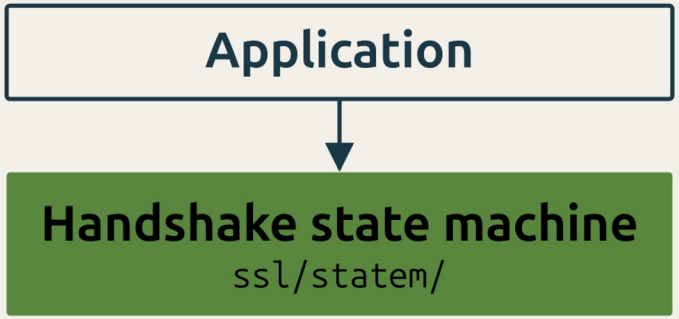
58 handshake states

You won't leave knowing libssl.
You'll leave knowing where to start looking.

Two subsystems

The whole picture

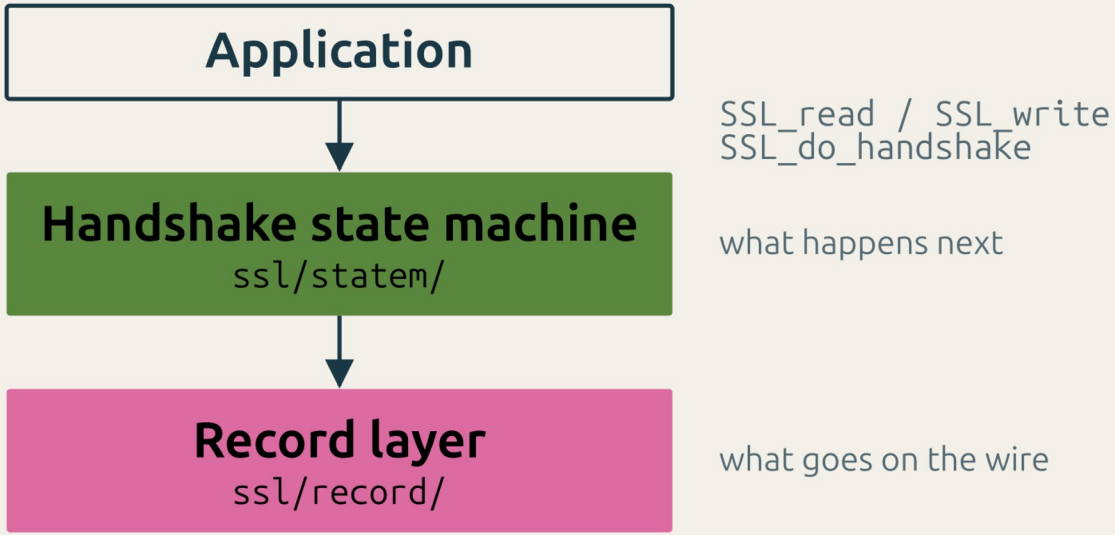
The whole picture



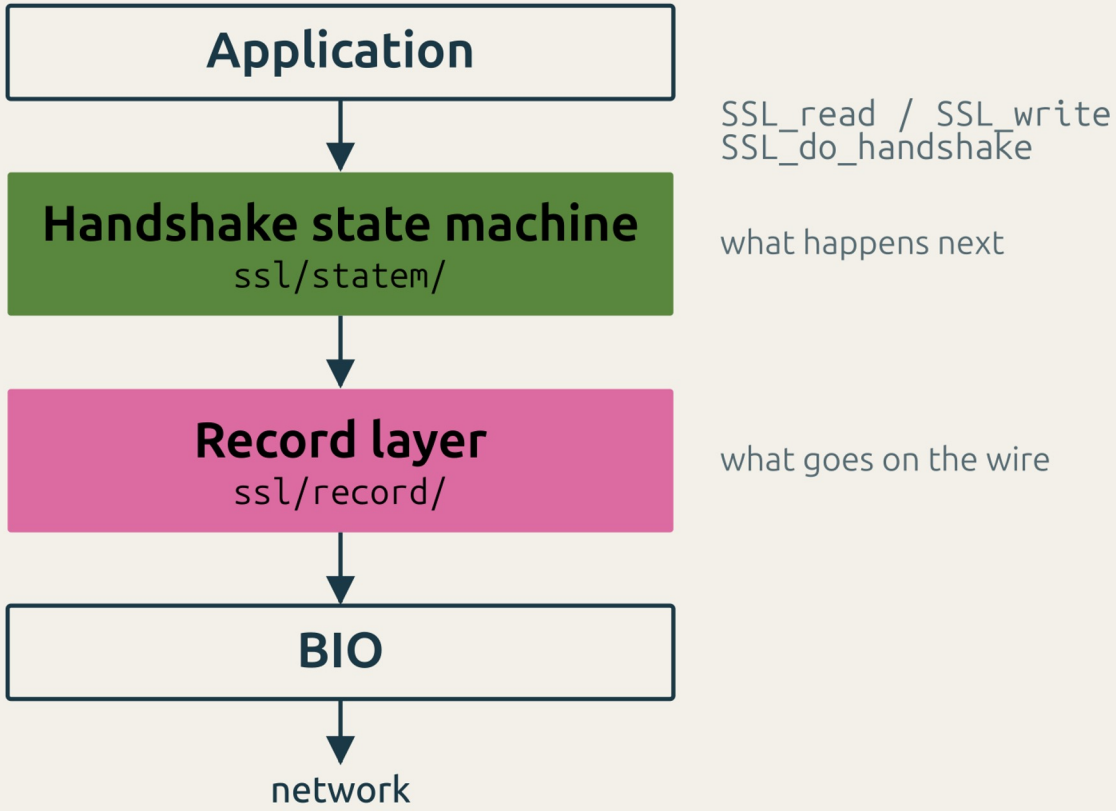
`SSL_read / SSL_write`
`SSL_do_handshake`

what happens next

The whole picture



The whole picture



Different questions

STATE MACHINE

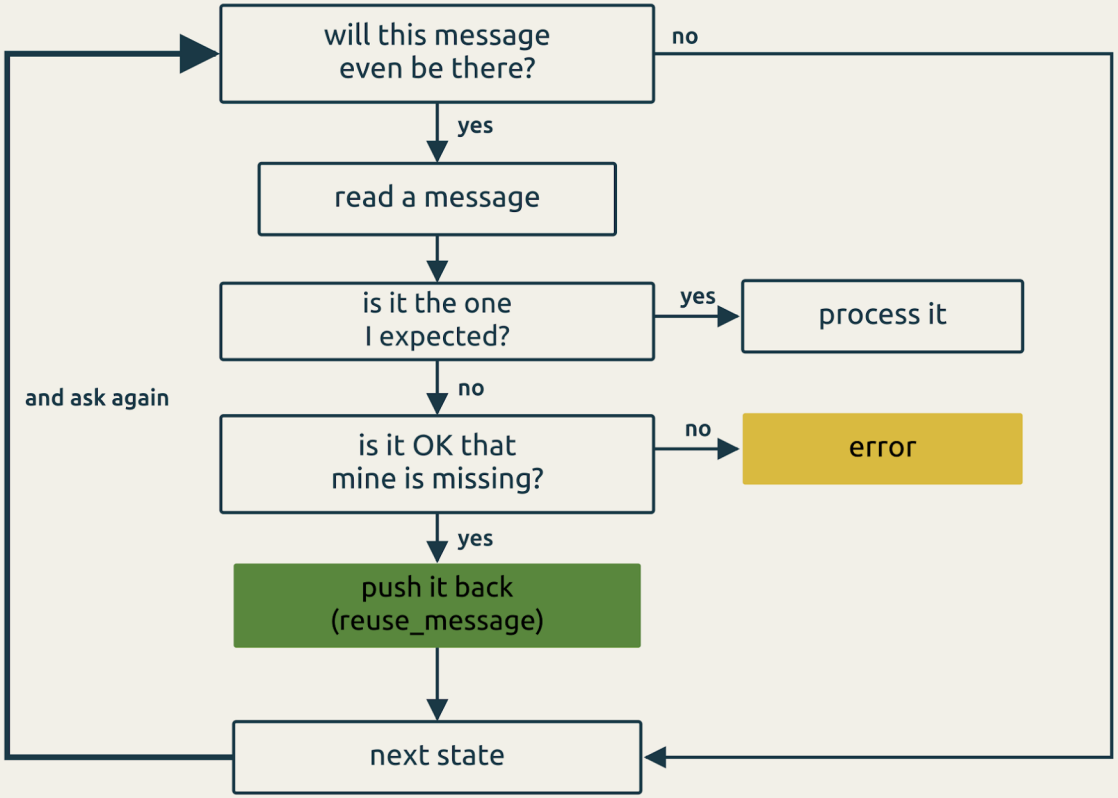
Which message next?
Is this transition legal?
What work before / after?
Where were we when we stopped?

RECORD LAYER

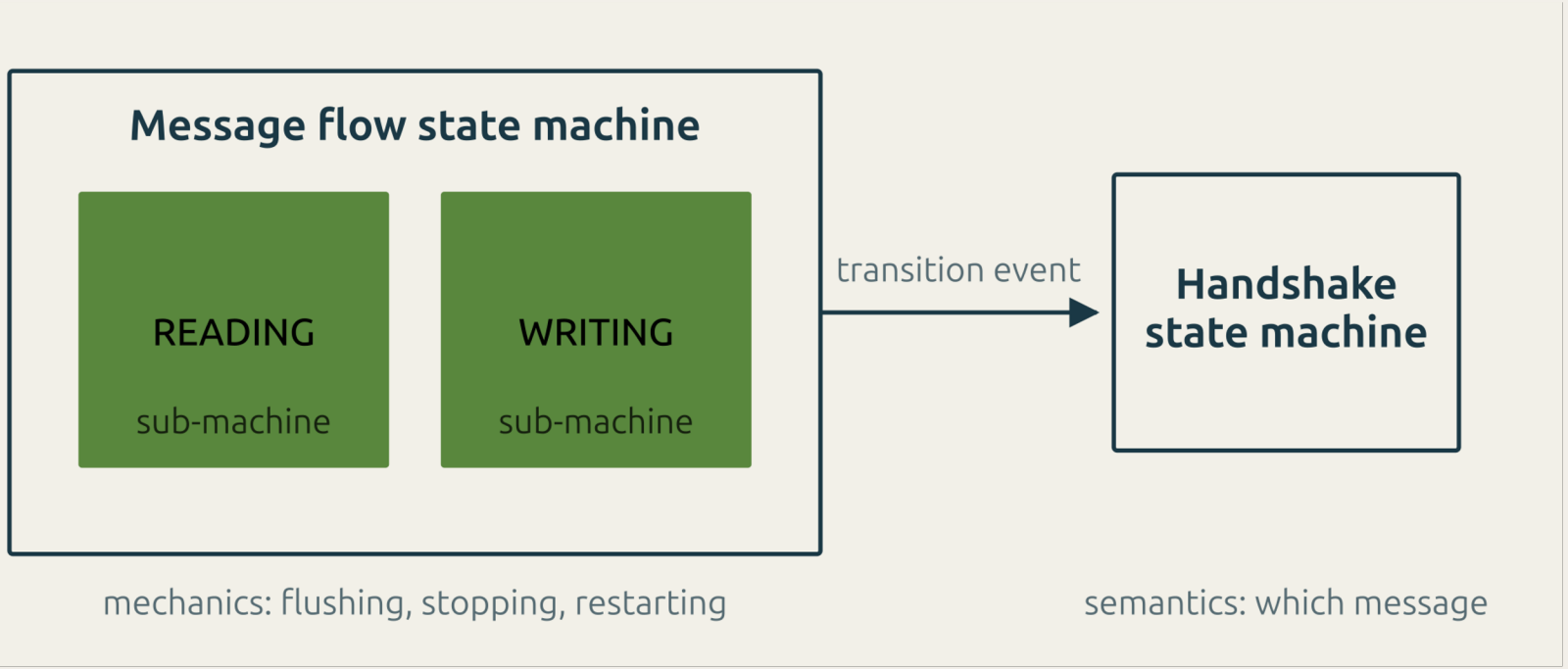
Framing
Protection
Buffering
Key changes in flight

The state machine

How the old parser worked



Two levels


Message flow state machine
READING
sub-machine
WRITING
sub-machine
transition event
Handshake state machine
mechanics: flushing, stopping, restarting
semantics: which message

The outer machine is five values

UNINITED → READING ⇄ WRITING → FINISHED

└── ERROR

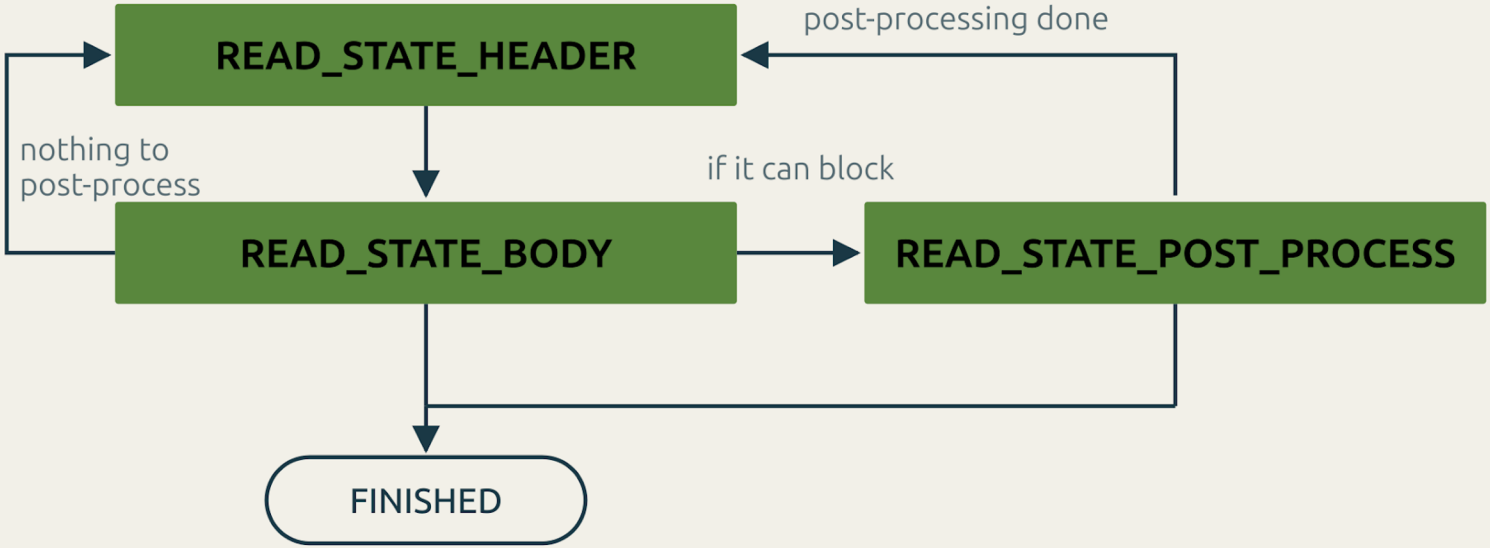
MSG_FLOW_STATE — include/internal/statem.h

Idea #1 — parse what arrived, not what you expected

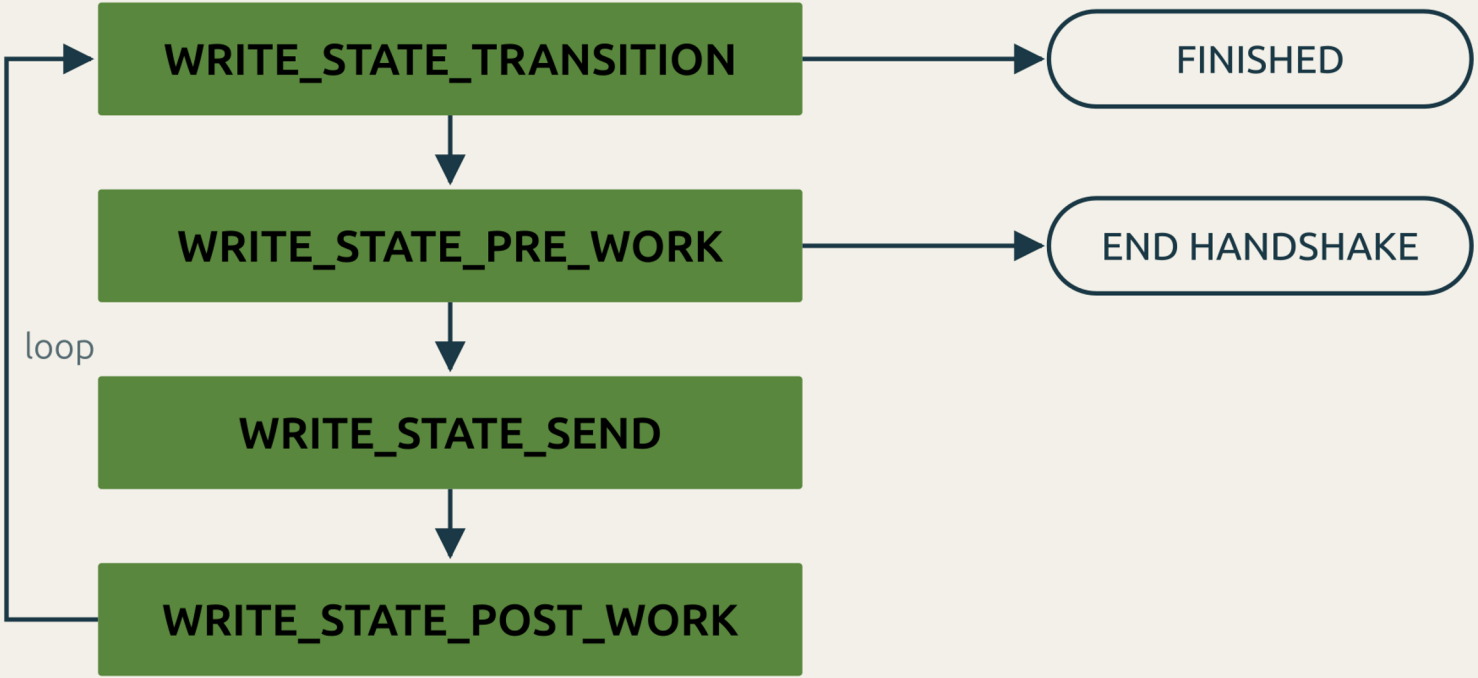
```
ret = tls_get_message_header(s, &mt);


if (!transition(s, mt))
    return SUB_STATE_ERROR;
```

— ssl/statem/statem.c — READ_STATE_HEADER

Reading a message



Writing a message



Idea #2 — every point you might stop at needs a name

```
WORK_MORE_A,

WORK_MORE_B,

WORK_MORE_C
```

Every handshake message is eight functions

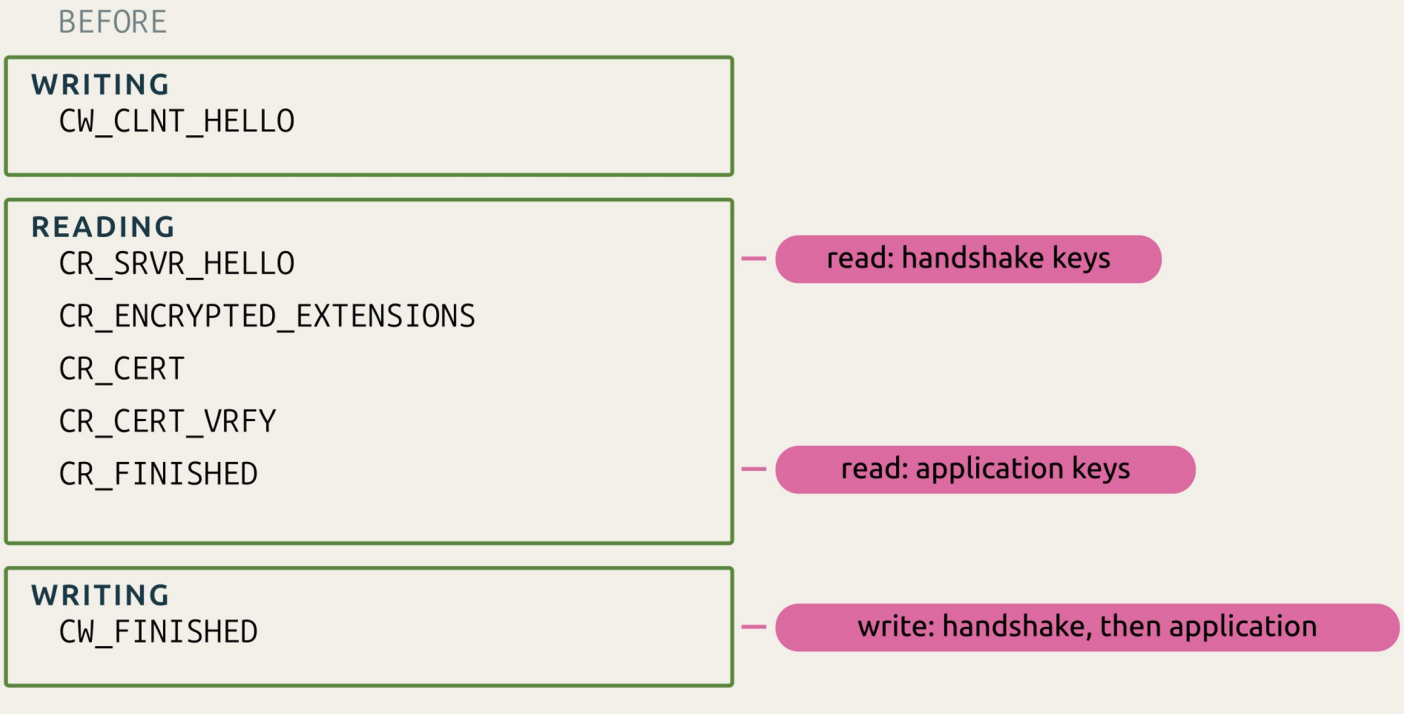
READING

read_transition()
max_message_size()
process_message()
post_process_message()

WRITING

write_transition()
pre_work()
construct_message()
post_work()

A TLS 1.3 handshake, as the machine sees it



Key changes do not line up with the band boundaries.

The record layer

Frame · Protect · Buffer · Re-key in flight

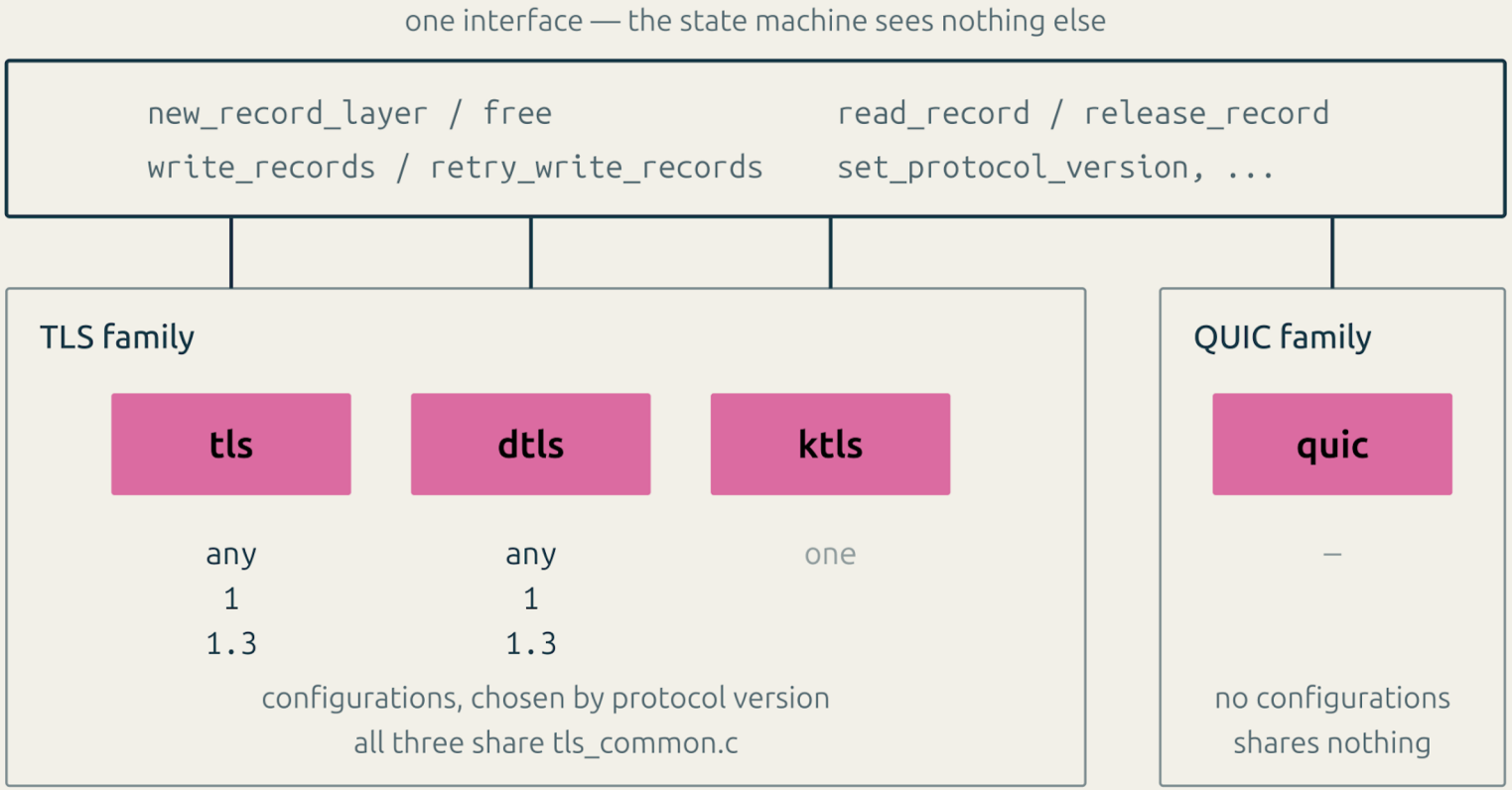
Two rewrites

1.1.0 separated the record layer — to an extent

3.2 made the interface pluggable

TLS · DTLS · KTLS · QUIC

OSSL_RECORD_METHOD



Idea #3 – keys don't change. Record layers do.

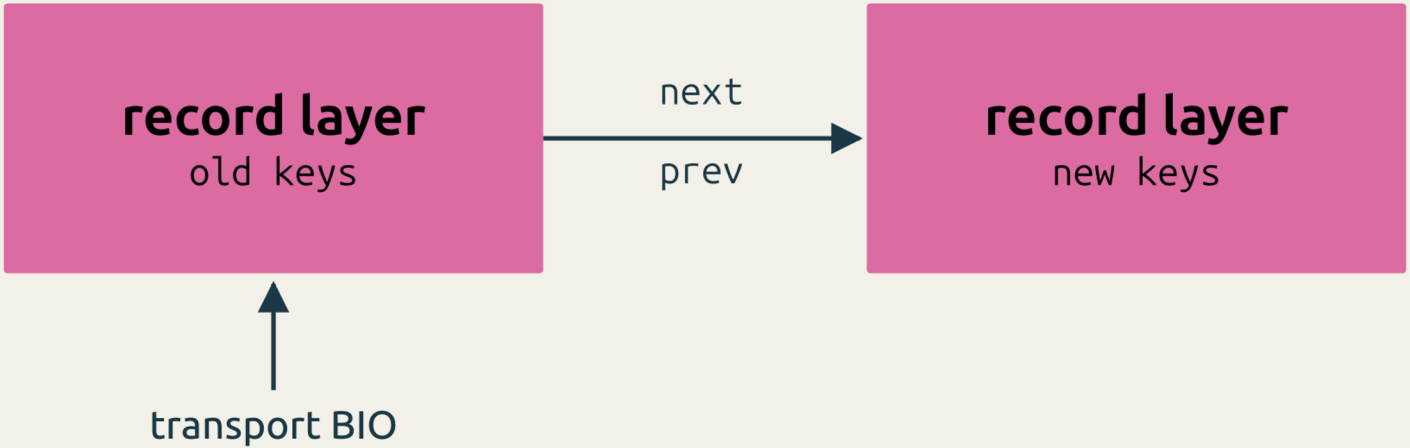
```
ssl_set_new_record_layer(s, version, direction,
                         level, secret, key, iv, ...);
```

Is this even TLS?

```
if (HAS_PREFIX(p, "GET ")  || HAS_PREFIX(p, "POST ")
 || HAS_PREFIX(p, "HEAD ") || HAS_PREFIX(p, "CONNE"))
    RLAYERfatal(rl, SSL_AD_NO_ALERT, SSL_R_HTTP_REQUEST);
```

— ssl/record/methods/tlsany_meth.c

The handover problem



A layer torn down at a key change may still hold bytes that belong to its successor.

Borrowed buffers

```
read_record(rl, &rechandle, &rversion, &type,
            &data, &datalen, ...);

release_record(rl, rechandle, length);
```

Idea #4 – the record layer doesn't know what a connection is

```
skip_early_data
msg_callback
security
```

```
padding
get_urxe_packet
release_urxe_packet
block_for_write
```

```
ossl_ssl_set_custom_record_layer(sc,
    &quic_tls_record_method, qtls);
```

DTLS 1.3 – record layer tweaked again

ACK logic | listener | demux | connection lookup

the protocol work — all of it above the record layer

`OSSL_RECORD_METHOD`

`snkey · sniph`	record number encryption
`epoch · sequence_number`	name a record, so it can be ACKed
`get_sequence / set_sequence / get_epoch`	same epoch, new layer — don't restart
`set_prev_epoch_rl`	survive a lost ACK

four additions to the interface

Final thoughts

Symptom → file

STATE MACHINE	SSL_state_string_long()
"unexpected message", wrong order	*_read_transition()
Message contents rejected	*_process_message()
Stalls mid-handshake	pre_work / post_work, WORK_MORE_*
Extension behaviour	extensions_clnt.c / _srvr.c
RECORD LAYER	**SSL_rstate_string_long()**
"wrong version number", "http request"	tlsany_meth.c
"bad record mac"	tls13_meth.c / tls1_meth.c
Wrong keys at the wrong time	grep ssl_set_new_record_layer

Thank you to our leading supporters!

PREMIER SUPPORTERS

Sovereign Tech
Fund

CODE PROTECTORS

craig newmark philanthropies

openSSL Foundation

Data privacy. Secure connections. For everyone.

Questions

ssl/statem/README.md

github.com/openssl/openssl

matt@openssl.foundation