

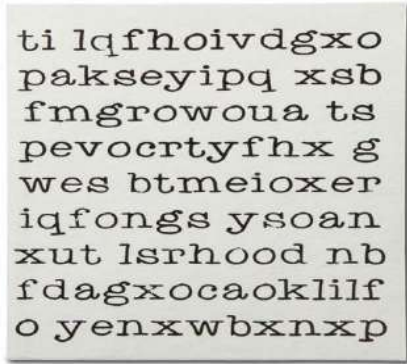
TLS the OTP way

The background of the slide is a dark blue, abstract digital landscape. It features numerous glowing blue lines and streaks that curve and flow across the scene, suggesting data paths or network connections. In the upper center, there is a bright, glowing light source that illuminates the surrounding area, creating a sense of depth and focus.

By Ingela Anderton Andin - Ericsson AB

#CodeBEAMSF

Transport Layer Security



ti lqfhoivdgxo
pakseyipq xsb
fmgrowoua ts
pevocrtyfhx g
wes btmeioxer
iqfongs ysoan
xut lsrhood nb
fdagxocaoklif
o yenxwbxnxp

Data privacy
Symmetric encryption
of payload data

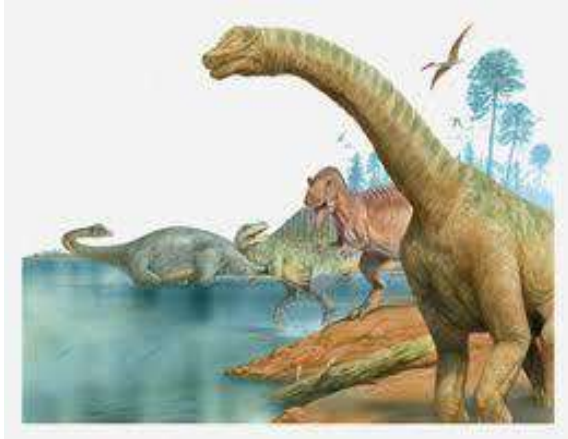


Integrity - MAC
(Message
authentication code)



Authenticity - Certificates

SSL 2.0 (1995) | SSL-3.0 (1996)



TLS 1.0 (1999) | TLS 1.1 (2006)



Fairly similar to SSL-3.0

TLS-1.2 (2008) TLS 1.3 (2018)

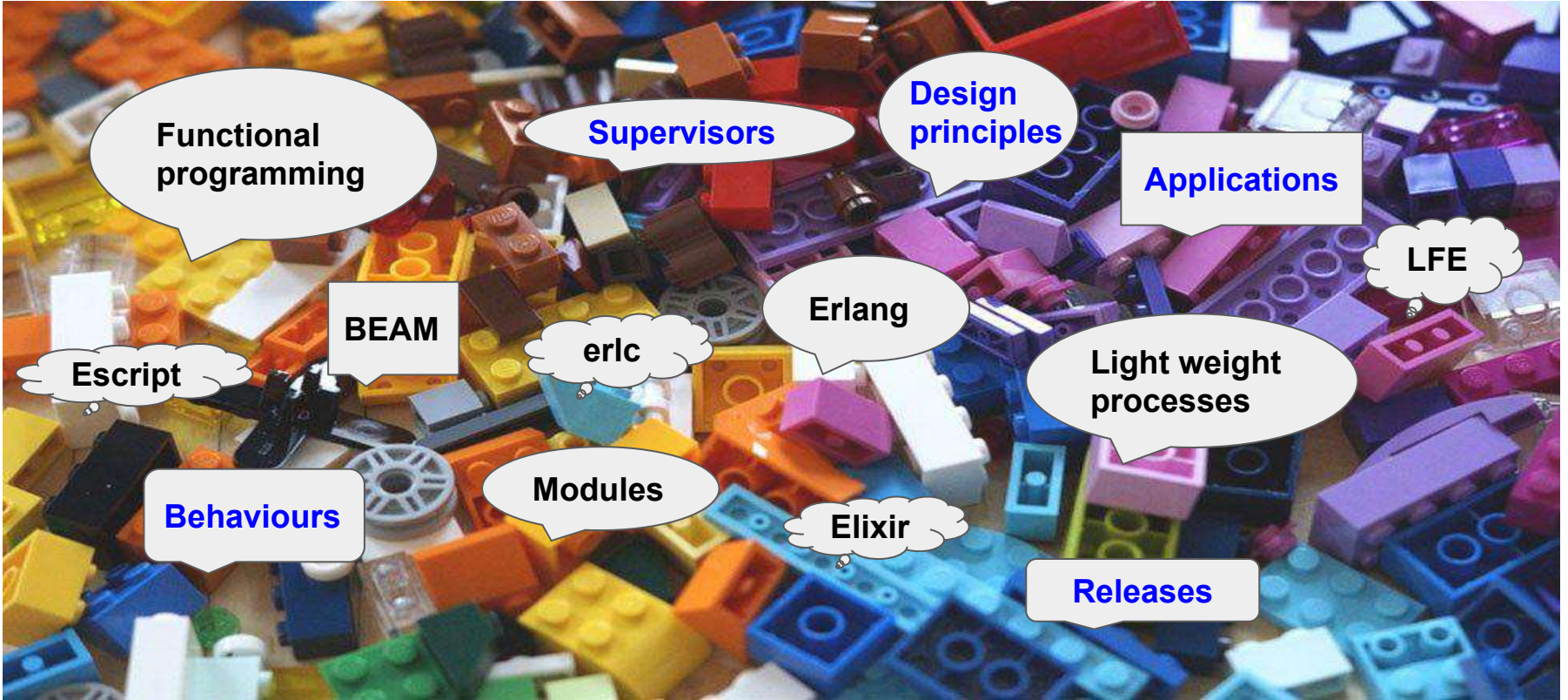


- New algorithms
- New extensions
- Backwards compatible



- New protocol message flow
- Encrypts earlier
- Removes legacy support

OTP



#CodeBEAMSF

Erlang/OTP Building Blocks

Releases

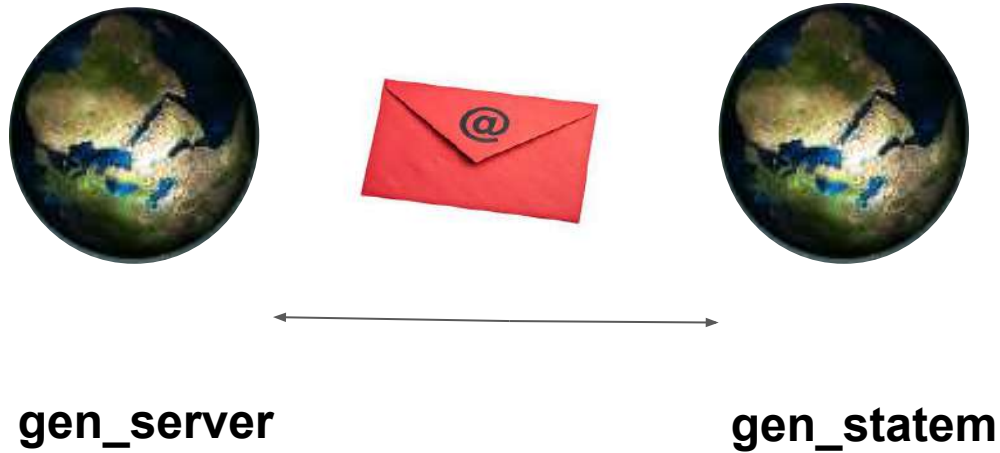
Applications

Modules

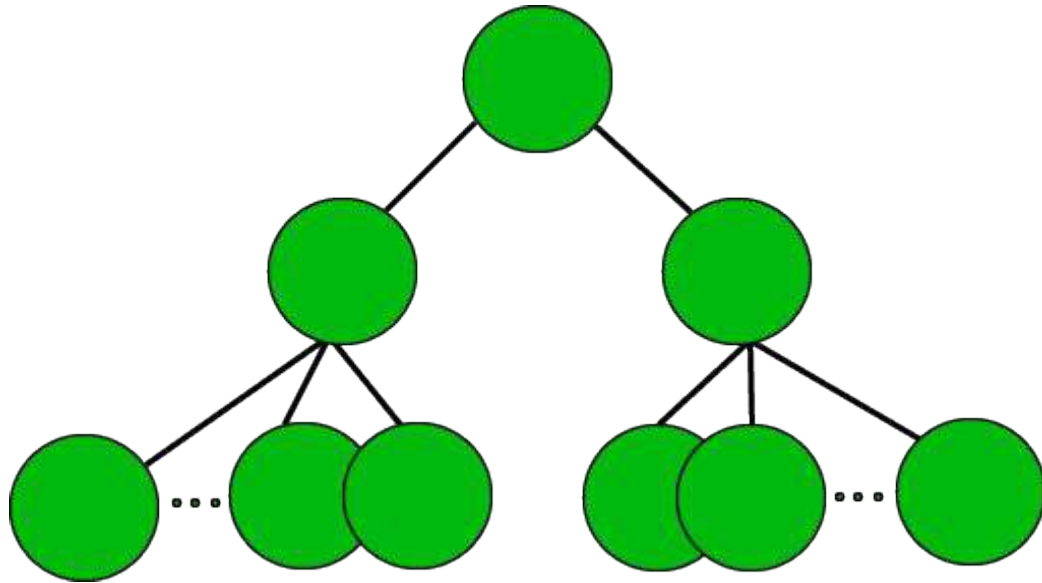
Functions



Parallelism



OTP Application Structure



Library Application

End-entity Certificate

Owner's name
Owner's public key
Issuer's (CA's) name
Issuer's signature

reference

Intermediate Certificate

Owner's (CA's) name
Owner's public key
Issuer's (root CA's) name
Issuer's signature

sign

reference

sign

self-sign

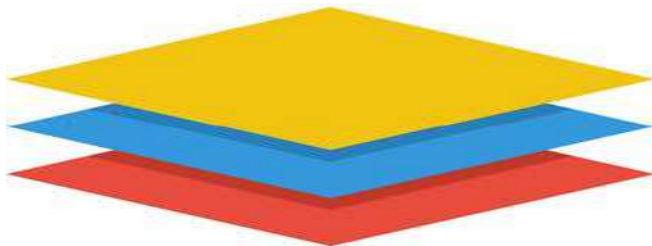
Root CA's name
Root CA's public key
Root CA's signature

Root Certificate



#CodeBEAMSF

Basic Design of TLS (PRE 1.3)



Applications,
HTTPS,
FTPS, ...

Handshake

Alert

Change Cipher Spec

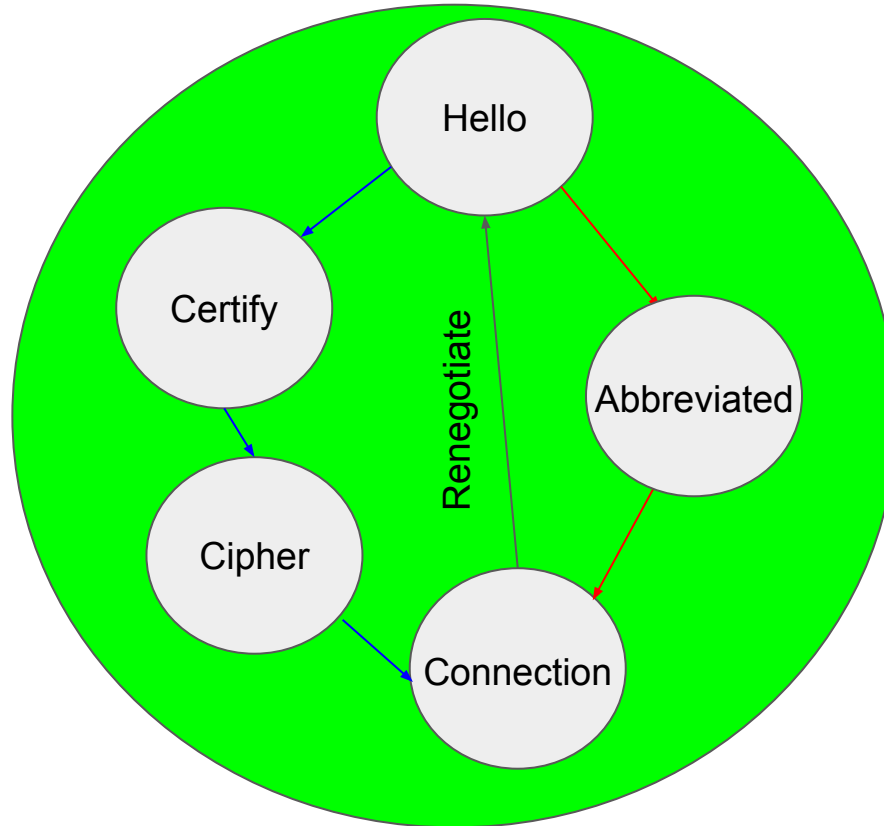
Application Data

TLS Records

<<?BYTE(Type), ?Byte(Major), ?BYTE(Minor), ?UINT16(Length), Data/binary>>

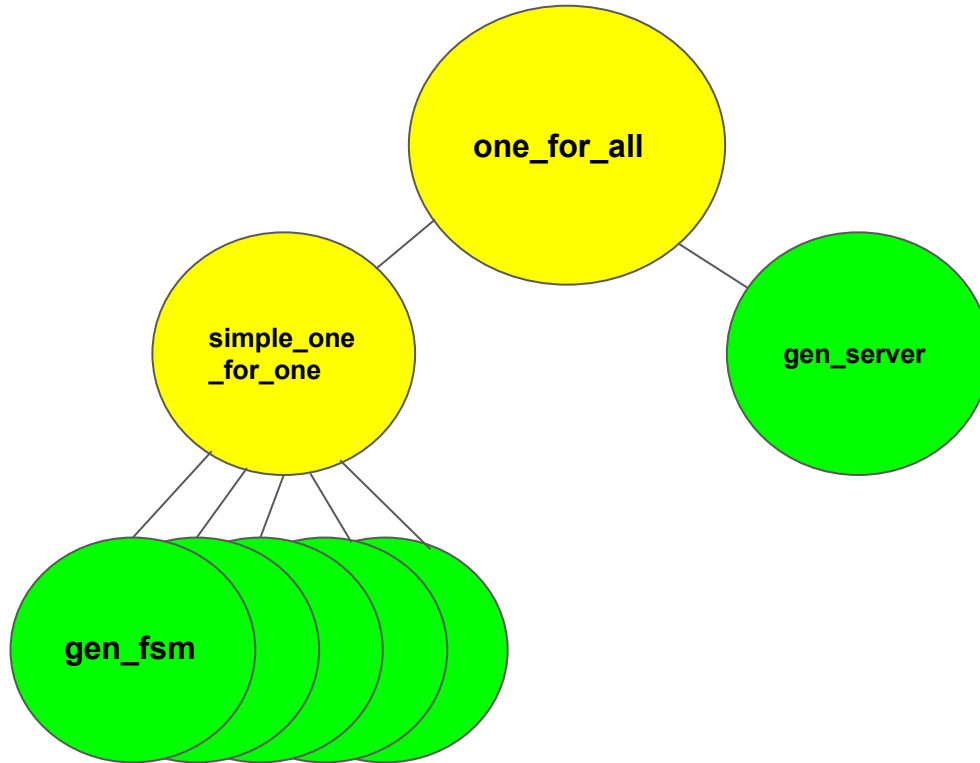
A reliable transport protocol, in practice **TCP/IP**

Basic TLS Finite State Machine PRE 1.3



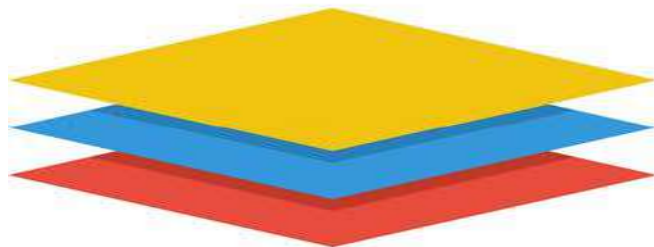
#CodeBEAMSF

Original Design in 2010



#CodeBEAMSF

Basic Design of DTLS



#CodeBEAMSF

Applications,
VPN clients,
WebRTC, ...

Handshake

Alert

Change Cipher Spec

Application Data

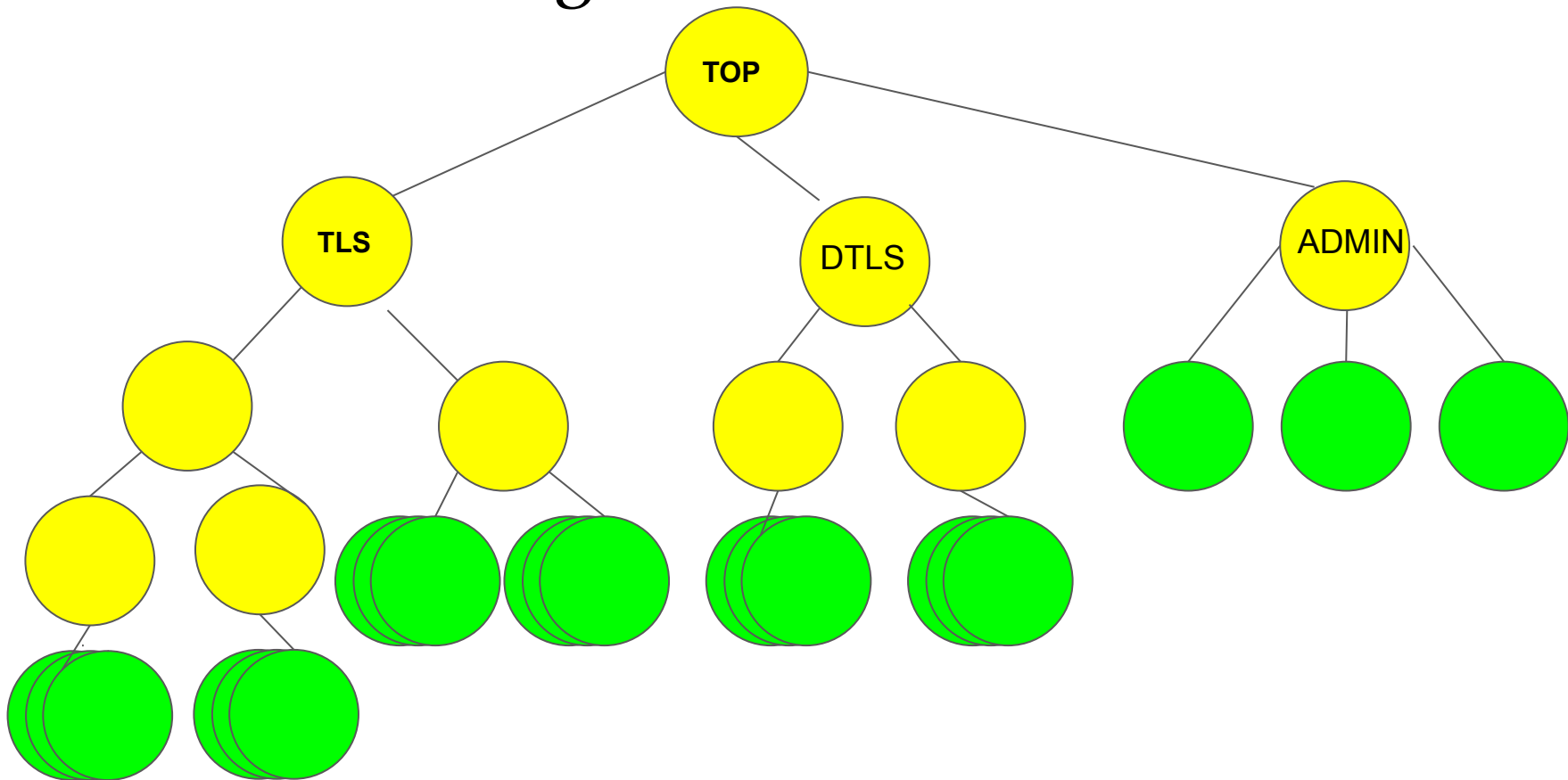
DTLS Records

<<?UINT16(Epoch), ?UINT48(SeqNum), ?BYTE(Type), ?Byte(Major),
?BYTE(Minor), ?UINT16(Length), Data/binary>>

UDP/IP

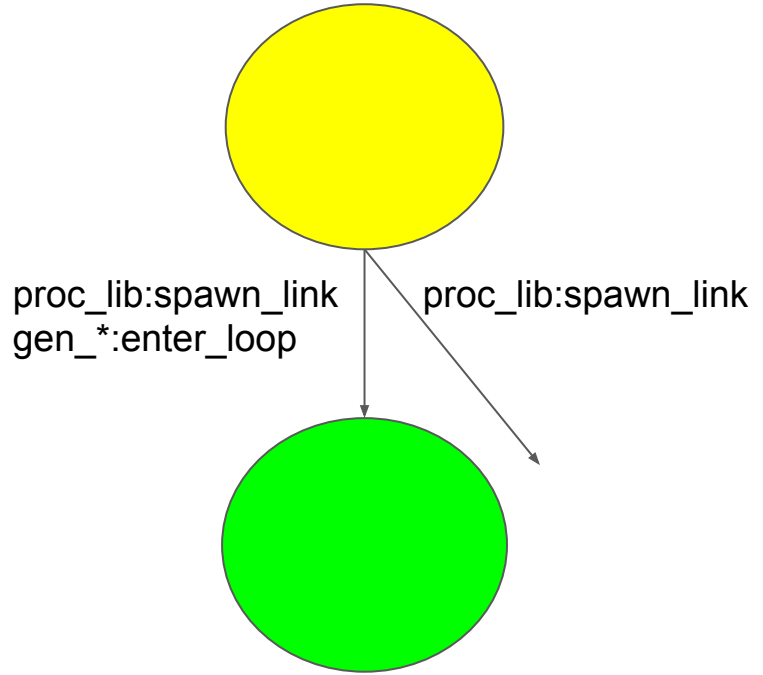
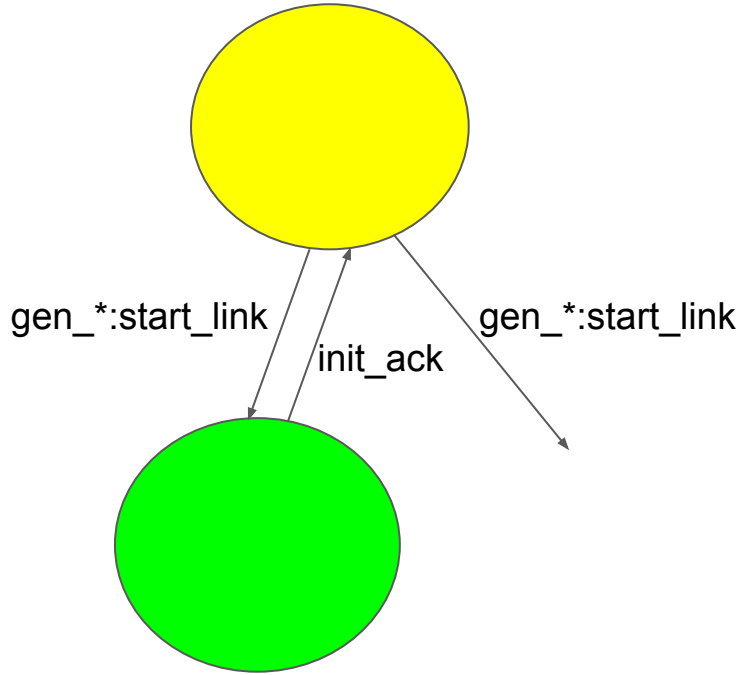
#CodeBEAMSF

Current Design



#CodeBEAMSF

Avoid Ack in Simple One for One Supervisor



#CodeBEAMSF

gen_fsm A



```
Handle = fun({#hello_request{} = Packet, _}, {next_state, connection = SName, State}) ->
  Hs0 = ssl_handshake:init_handshake_history(),
  ?MODULE:SName(Packet, State#state{tls_handshake_history=Hs0,renegotiation = {true, peer}});
({#hello_request{} = Packet, _}, {next_state, SName, State}) ->
  ?MODULE:SName(Packet, State);
({#client_hello{} = Packet, Raw}, {next_state, connection = SName, State}) ->
  Version = Packet#client_hello.client_version,
  Hs0 = ssl_handshake:init_handshake_history(),
  Hs1 = ssl_handshake:update_handshake_history(Hs0, Raw),
  ?MODULE:SName(Packet, State#state{tls_handshake_history=Hs1, renegotiation = {true, peer}});
({Packet, Raw}, {next_state, SName, State = #state{tls_handshake_history=Hs0}}) ->
  Hs1 = ssl_handshake:update_handshake_history(Hs0, Raw),
  ?MODULE:SName(Packet, State#state{tls_handshake_history=Hs1});
(_, StopState) -> StopState
end,
try
  {Packets, Buf} = tls_handshake:get_tls_handshake(Version,Data,Buf0),
  State = State0#state{protocol_buffers =Buffers#protocol_buffers{tls_packets = Packets, tls_handshake_buffer = Buf}},

handle_tls_handshake(Handle, Next, State) ...
```

gen_statem A—————B

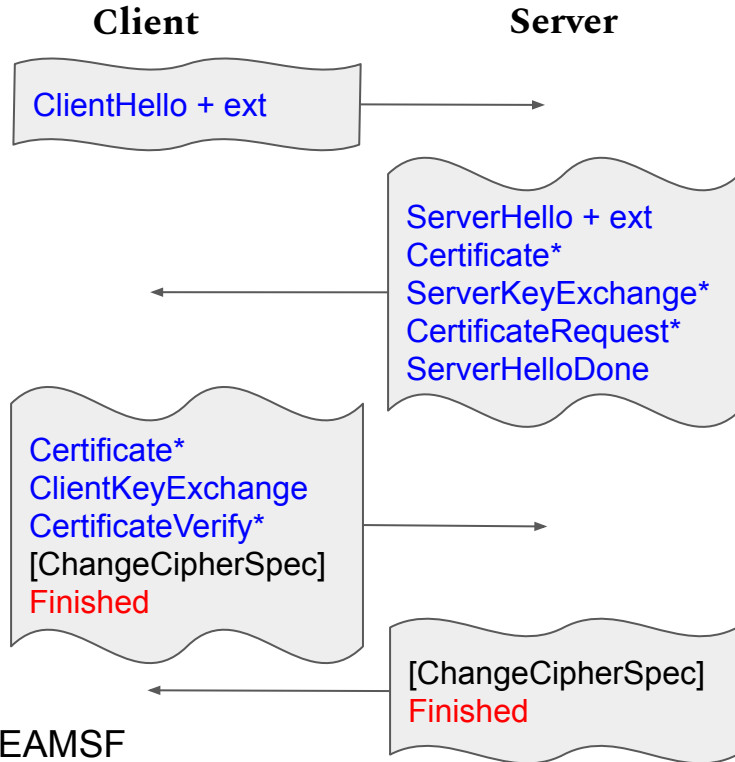
```
{Packets, Buf} = tls_handshake:get_tls_handshake(Version,Data,Buf0,Options),  
Events = tls_handshake_events(Packets),  
{next_state, StateName, State, Events}.
```

```
tls_handshake_events(Packets) ->  
  lists:map(fun(Packet) ->  
    {next_event, internal, {handshake, Packet}}  
  end,  
  Packets).
```

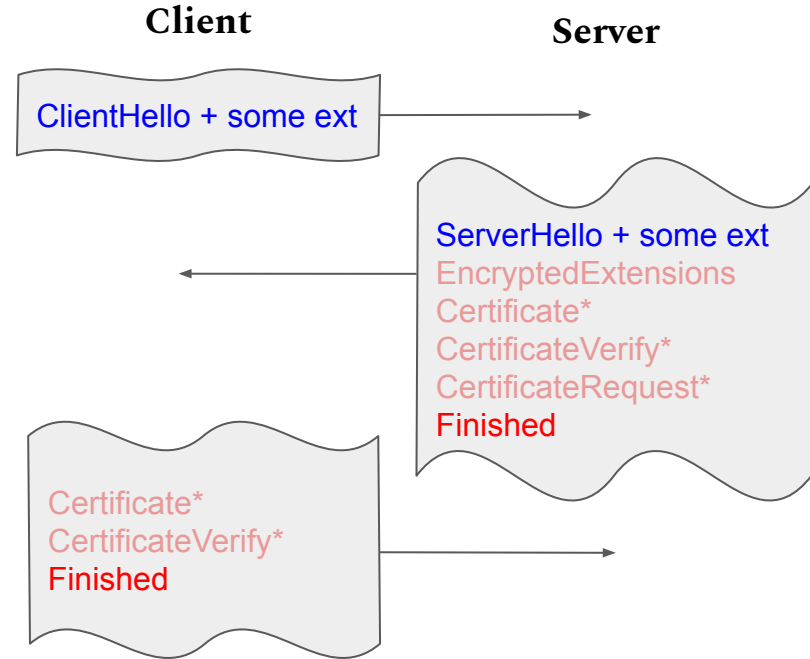
#CodeBEAMSF

Full handshake

TLS-1.0 - TLS-1.2

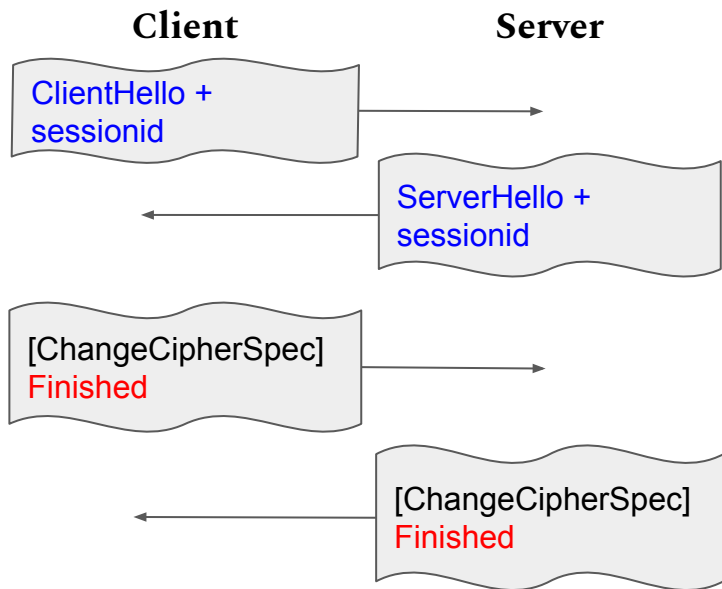


TLS 1.3

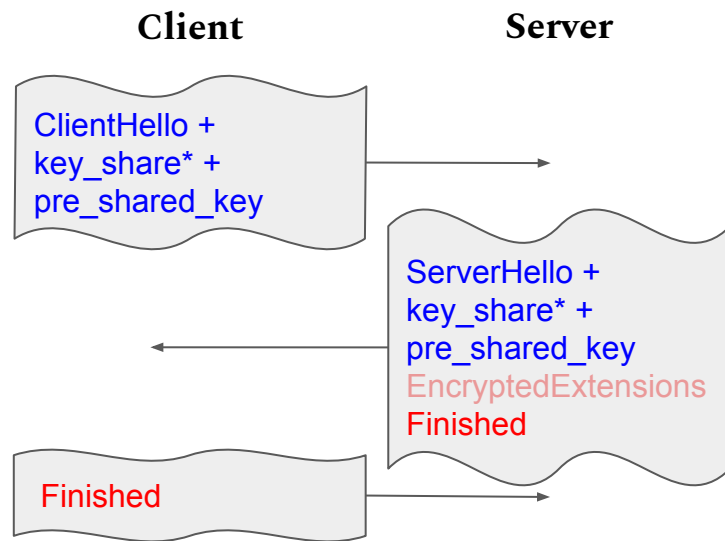


Abbreviated handshake

TLS-1.0 - TLS-1.2



TLS 1.3



Managing Memory Growth

<<Random:32/binary>>

Unique ref => {erlang:monotonic_time,
erlang:unique_integer(monotonic)}

ets



gb_trees



Cipher Suites - Collections of Algorithms

- **SSL 3.0,**
TLS-1.0 and TLS 1.1 e.g. {DH_RSA, 3DES, SHA1}
- **TLS 1.2 e.g. {ECDH_ECDSA, AES_128, SHA256, SHA256}**
- **TLS 1.3 e.g. {AES_256_GCM, SHA384}**



Cipher Suites the OpenSSL way

TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:EC
DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY130
5:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA
256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RS
A-AES128-SHA256:DHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES256-SHA:ECDHE-EC
DSA-AES128-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES128-SHA:RSA-PSK-AES256-GCM-SHA384:DHE-PSK-AES256-GCM-SHA384:RSA-P
SK-CHACHA20-POLY1305:DHE-PSK-CHACHA20-POLY1305:ECDHE-PSK-CHACHA20-POLY1305:AES256-GCM-SHA384:PSK-AES256-GCM-S
HA384:PSK-CHACHA20-POLY1305:RSA-PSK-AES128-GCM-SHA256:DHE-PSK-AES128-GCM-SHA256:AES128-GCM-SHA256:PSK-AES128-G
CM-SHA256:AES256-SHA256:AES128-SHA256:ECDHE-PSK-AES256-CBC-SHA384:ECDHE-PSK-AES256-CBC-SHA:SRP-RSA-AES-256-CBC-S
HA:SRP-AES-256-CBC-SHA:RSA-PSK-AES256-CBC-SHA384:DHE-PSK-AES256-CBC-SHA384:RSA-PSK-AES256-CBC-SHA:DHE-PSK-AES256
-CBC-SHA:AES256-SHA:PSK-AES256-CBC-SHA384:PSK-AES256-CBC-SHA:ECDHE-PSK-AES128-CBC-SHA256:ECDHE-PSK-AES128-CBC-S
HA:SRP-RSA-AES-128-CBC-SHA:SRP-AES-128-CBC-SHA:RSA-PSK-AES128-CBC-SHA256:DHE-PSK-AES128-CBC-SHA256:RSA-PSK-AES12
8-CBC-SHA:DHE-PSK-AES128-CBC-SHA:AES128-SHA:PSK-AES128-CBC-SHA256:PSK-AES128-CBC-SHA

ALL:!EXPORT:!LOW:!aNULL:!eNULL:!SSLv2

Cipher Suites the Erlang way

```
Default = ssl:cipher_suites(default, 'tlsv1.2').
```

```
Suites = ssl:filter_cipher_suites(Default,  
    [{key_exchange,  
        fun(rsa) -> false;  
          (_) -> true  
        end}]])).
```

```
ssl:suite_to_str(ssl:cipher_suites(default, 'tlsv1.3')).
```

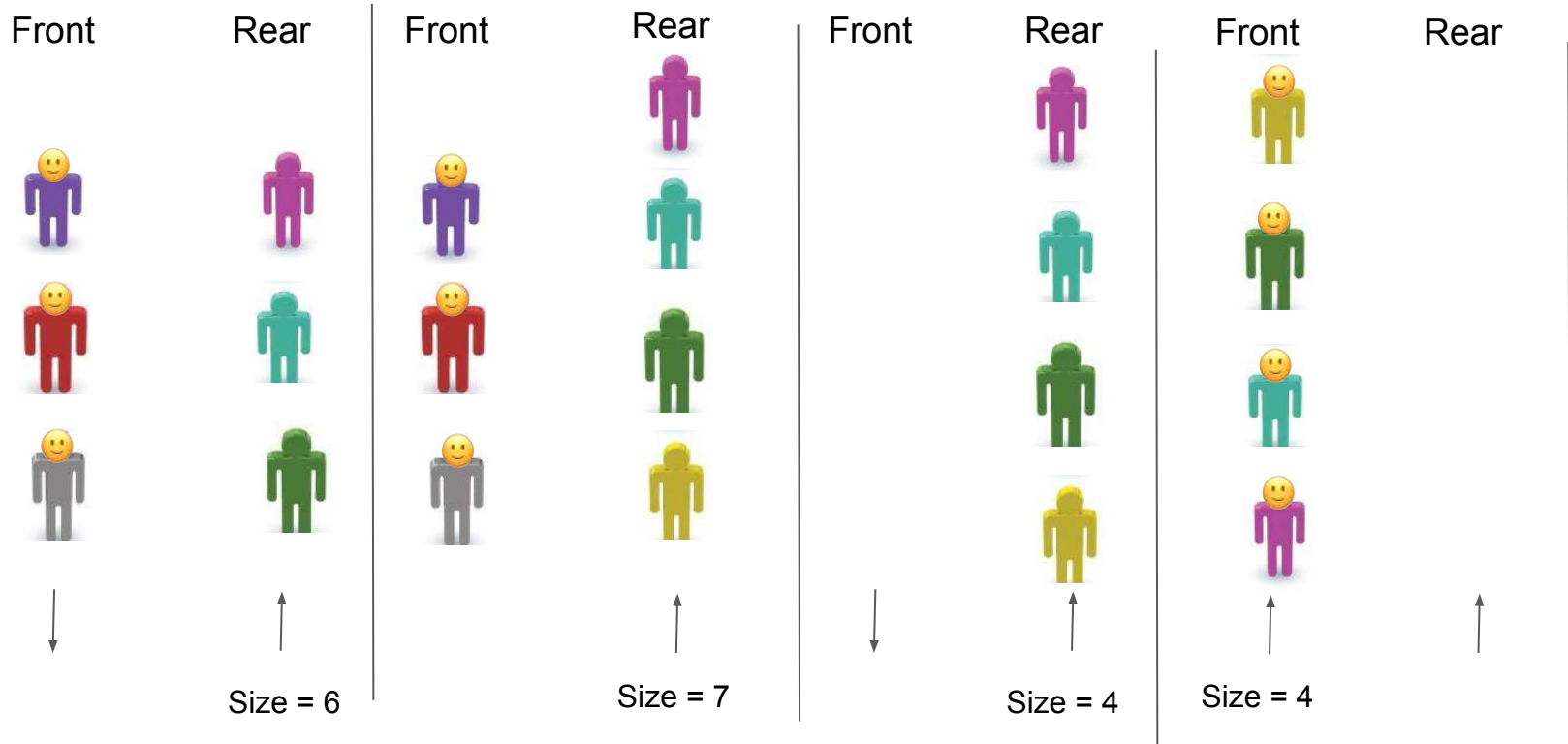
```
["TLS_AES_256_GCM_SHA384", "TLS_AES_128_GCM_SHA256",  
"TLS_CHACHA20_POLY1305_SHA256", "TLS_AES_128_CCM_SHA256"]
```

How to Queue ?!

```
read_application_data(Data, Buffer) ->  
    collect_data(<<Buffer/binary, Data/binary>>).
```



Okasaki Queue



Logging with Logger

=ERROR REPORT===== 8-Jan-2020::16:15:59.050148 ===

Server: httpd_test

Protocol: TLS

Host: 127.0.0.1:33045

Peer: 127.0.0.1:33093

Reason: "TLS server: In state certify received CLIENT ALERT: Fatal - Unknown CA"

#CodeBEAMSF

Logger Single Line Config

```
[{kernel,  
  
  [{logger,  
  
    [{handler, default, logger_std_h,  
  
      #{formatter => {logger_formatter, #{single_line => true}}}]  
  
    ]}]}.  
]
```

2020-01-08T17:24:38.371664+01:00 error: Server: httpd_test, Protocol: TLS, Host: 127.0.0.1:33285, Peer: 127.0.0.1:51055, Reason: "TLS server: In state certify received CLIENT ALERT: Fatal - Unknown CA"

ErrorReport vs Format

```
#{server => httpd_test,  
  protocol => `TLS`,  
  host => {127.0.0.1, 33045}  
  peer => {127.0.0.1, 33093}  
  Reason => "TLS server: In state certify received CLIENT ALERT: Fatal -  
Unknown CA"} %% Report
```

```
format("#{protocol := `TLS' }) -> %% Format callback  
  #{reason := Desc, peer := {PeerPort, Peer}, host := {HostPort, Host},  
  server_name := ServerName} = Report,  
  {"~10s ~s~n ~10s ~s~n"  
   "~10s ~s::~p~n ~10s ~s::~p~n"  
   "~10s ~p~n ~n",  
  ["Server:", "ServerName", "Protocol:", atom_to_list(Protocol)"  
   "Host:", Host, HostPort, "Peer:", Peer, PeerPort "Reason:", Desc]};
```

....

Logger Macros

```
decode_certs(Ref, Cert) ->
```

```
  try ErlCert = public_key:pkix_decode_cert(Cert, otp), ...
```

```
  catch
```

```
  error:_ ->
```

```
    ?LOG_NOTICE("SSL WARNING: Ignoring a CA cert as it could not be"  
                "correctly decoded.~n"),
```

```
  undefined
```

```
end.
```

Logger Main Function

=NOTICE REPORT==== 8-Jan-2020::16:15:59.041496 ===

TLS client: In state certify at ssl_handshake.erl:1766 generated CLIENT ALERT:
Fatal - Unknown CA

```
Meta = ?LOCATION
```

```
...
```

```
logger:log(Level, ReportMap,  
           Meta#{depth => ?DEPTH,  
                report_cb => fun ?MODULE:format/1}).
```

Logger Domains

...

```
logger:log(Level, Report,  
           Metadata#{domain =>  
                 [otp,inets, httpd, Domain, Level],  
           report_cb => fun ?MODULE:format/1})
```

...

OTP 23



SSL-3.0



TLS-1.3



ERICSSON

#CodeBEAMSF