FORMAL VERIFICATION OF TLS IN THE SECURE SOCKET API

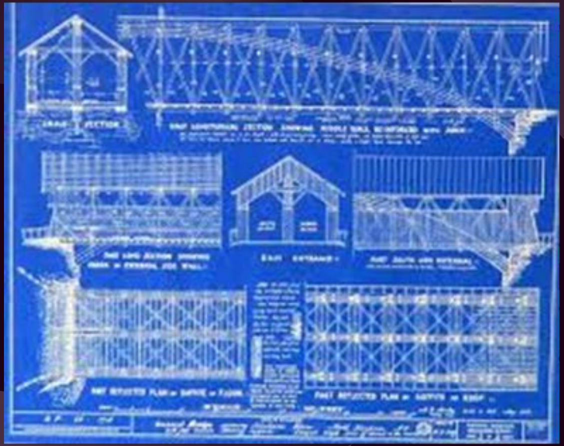
PRESENTED MATTHEW CHRISTENSEN
MARCH 5, 2022

“

FORMAL VERIFICATION

“Program testing can be used to show the presence of bugs, but never to show their **absence**!”

-Edsger W. Dijkstra



THE SECURE SOCKET API

Internet Research Lab

- Using TLS is hard
- Symbols in libssl: 504
- Lines of code: 317

```
int socket = socket(PF_INET, SOCK_STREAM, IPPRO
```

↓

```
int socket = socket(PF_INET, SOCK_STREAM,
IPPROTO_TLS);
```

THE PROBLEM

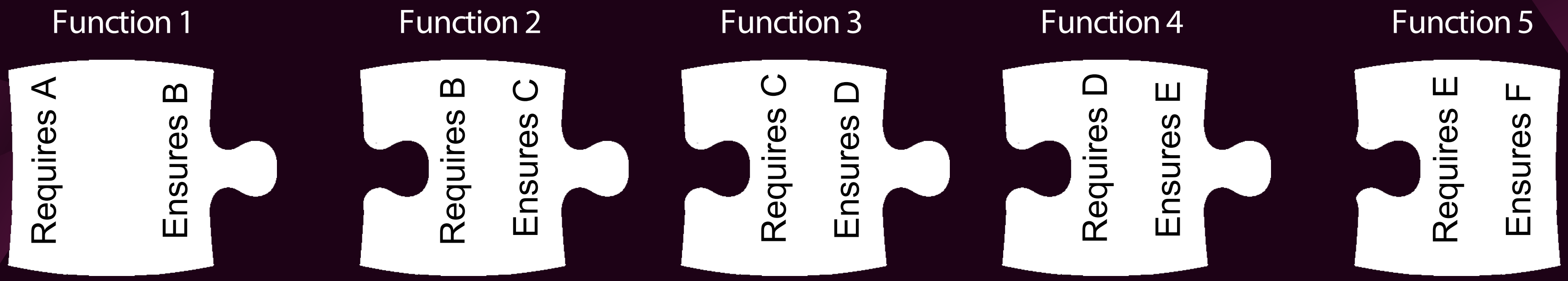
How do we know the Secure Socket API actually makes your socket secure?

FORMAL VERIFICATION PROCESS

Properties	Contracts	Model	Verification
Determine what properties are required for a secure connection.	Determine what is required to guarantee secure properties.	Model codebase in Dafny and overlay contracts onto model.	Verify that the model represents the codebase accurately.

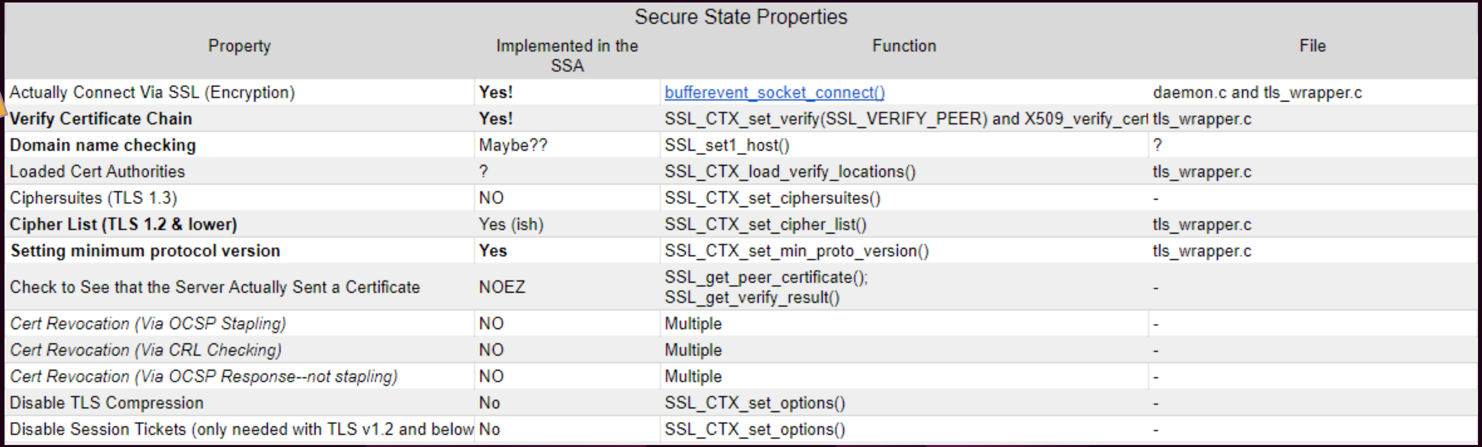
WEAKEST PRECONDITION CALCULUS

- Problem: how to prove quality F?



Thus, given A we can prove F

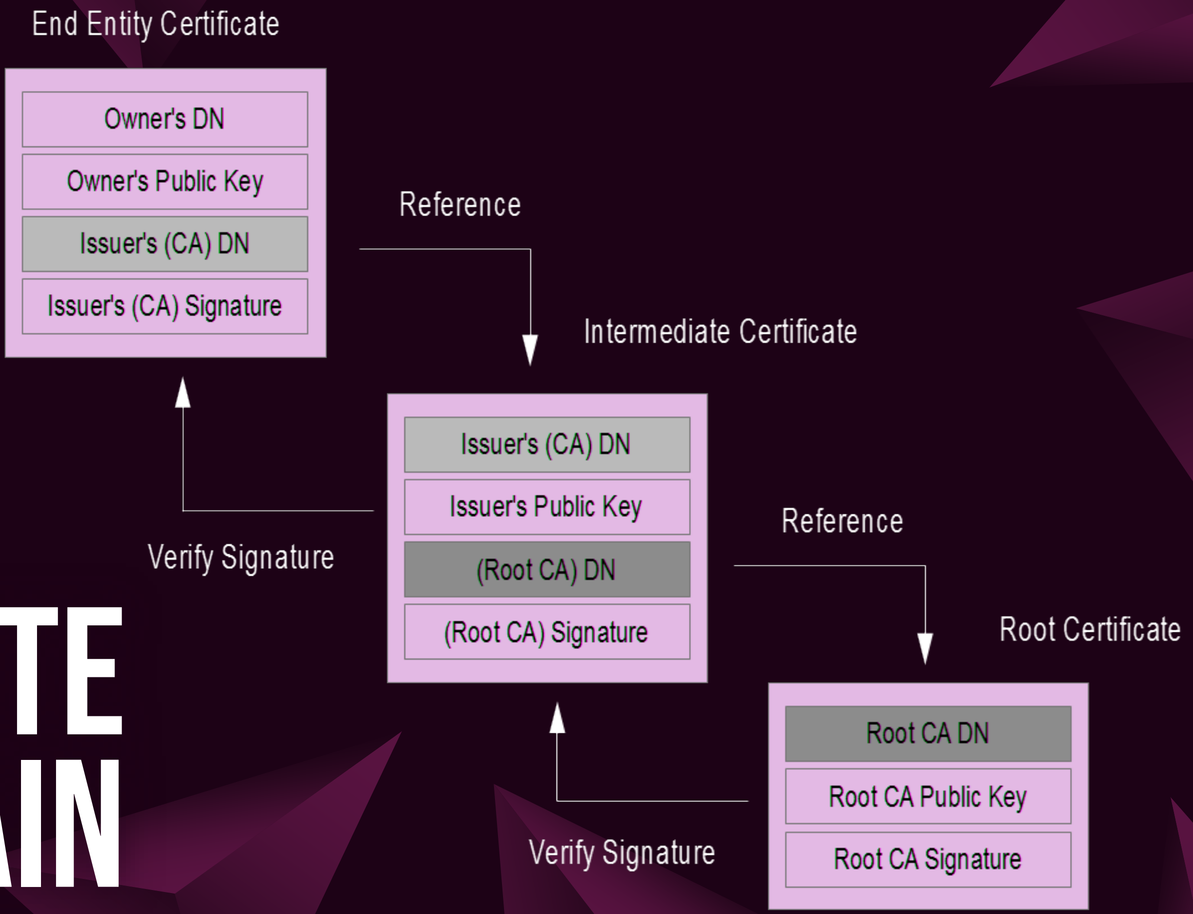
CONCRETE EXAMPLE

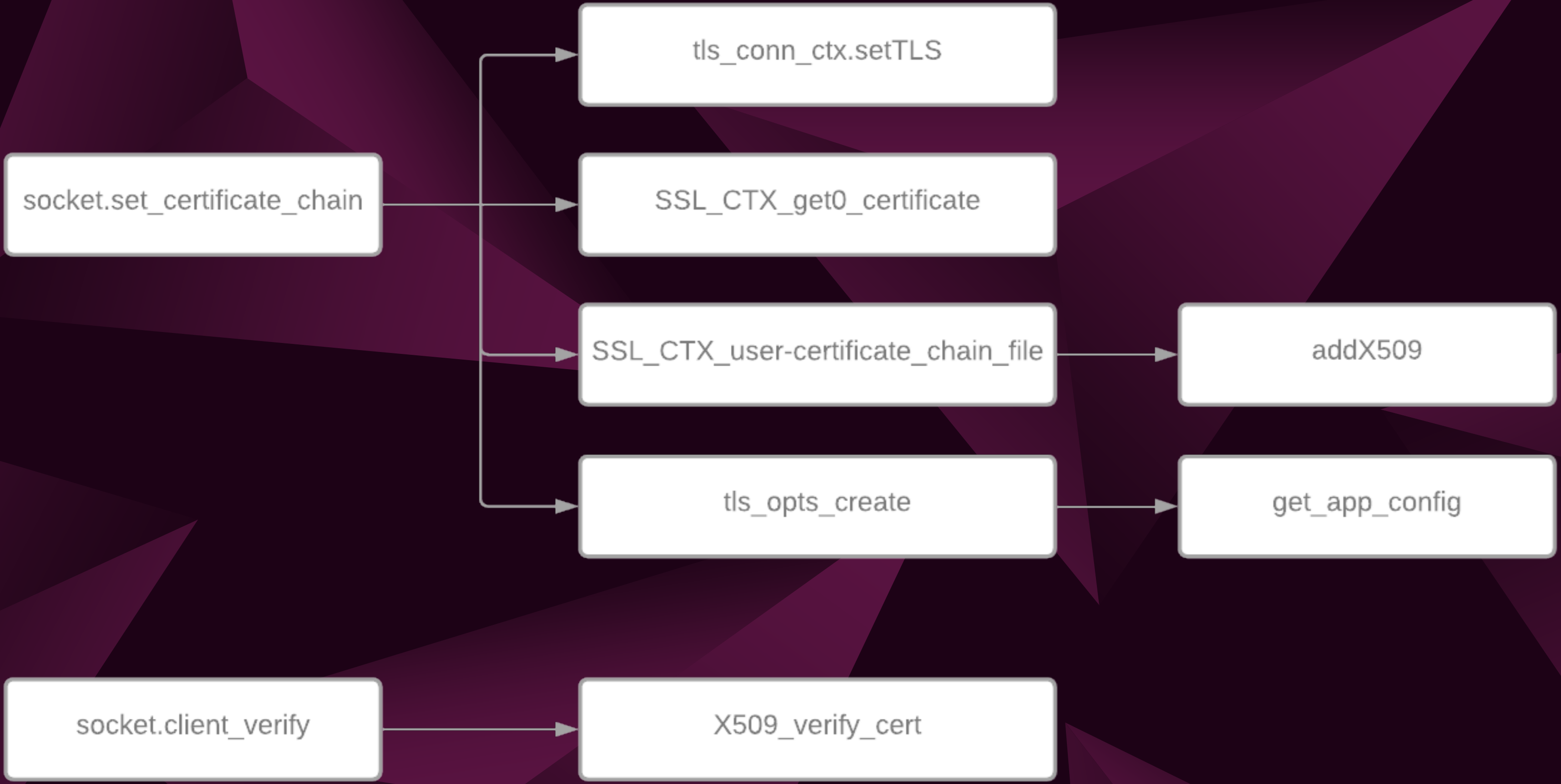


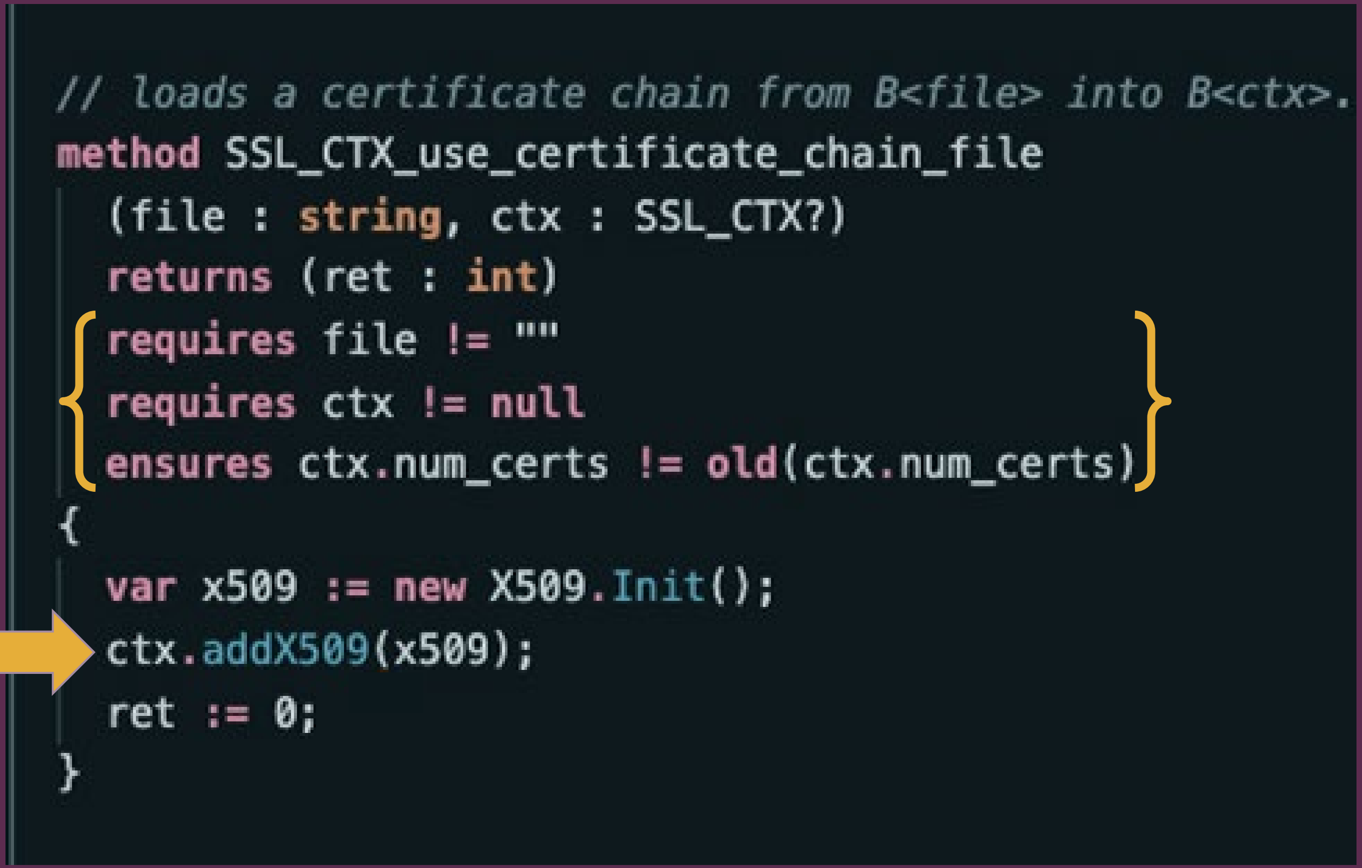
Secure State Properties

Property	Implemented in the SSA	Function	File
Actually Connect Via SSL (Encryption)	**Yes!**	bufferevent_socket_connect()	daemon.c and tls_wrapper.c
Verify Certificate Chain	**Yes!**	SSL_CTX_set_verify(SSL_VERIFY_PEER) and X509_verify_cert	tls_wrapper.c
Domain name checking	Maybe??	SSL_set1_host()	?
Loaded Cert Authorities	?	SSL_CTX_load_verify_locations()	tls_wrapper.c
Ciphersuites (TLS 1.3)	NO	SSL_CTX_set_ciphersuites()	-
Cipher List (TLS 1.2 & lower)	Yes (ish)	SSL_CTX_set_cipher_list()	tls_wrapper.c
Setting minimum protocol version	**Yes**	SSL_CTX_set_min_proto_version()	tls_wrapper.c
Check to See that the Server Actually Sent a Certificate	NOEZ	SSL_get_peer_certificate(); SSL_get_verify_result()	-
Cert Revocation (Via OCSP Stapling)	NO	Multiple	-
Cert Revocation (Via CRL Checking)	NO	Multiple	-
Cert Revocation (Via OCSP Response--not stapling)	NO	Multiple	-
Disable TLS Compression	No	SSL_CTX_set_options()	-
Disable Session Tickets (only needed with TLS v1.2 and below	No	SSL_CTX_set_options()	-

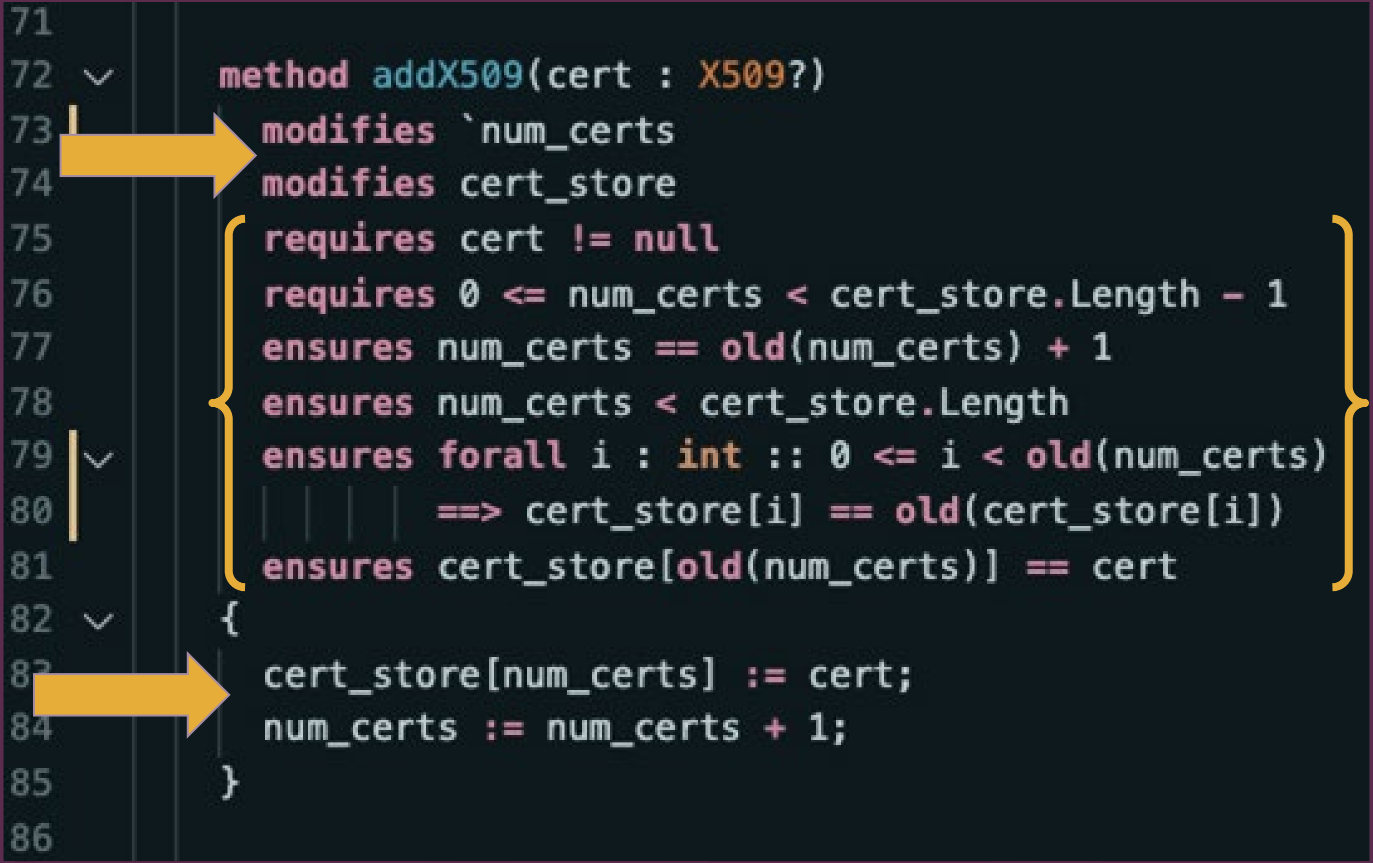
CERTIFICATE CHAIN




socket.set_certificate_chain
tls_conn_ctx.setTLS
SSL_CTX_get0_certificate
SSL_CTX_user-certificate_chain_file
addX509
tls_opts_create
get_app_config
socket.client_verify
X509_verify_cert



```
// loads a certificate chain from B<file> into B<ctx>.
method SSL_CTX_use_certificate_chain_file
  (file : string, ctx : SSL_CTX?)
  returns (ret : int)
  requires file != ""
  requires ctx != null
  ensures ctx.num_certs != old(ctx.num_certs)
{
  var x509 := new X509.Init();
  ctx.addX509(x509);
  ret := 0;
}
```



```
method addX509(cert : X509?)
  modifies `num_certs
  modifies cert_store
  requires cert != null
  requires 0 <= num_certs < cert_store.Length - 1
  ensures num_certs == old(num_certs) + 1
  ensures num_certs < cert_store.Length
  ensures forall i : int :: 0 <= i < old(num_certs)
          ==> cert_store[i] == old(cert_store[i])
  ensures cert_store[old(num_certs)] == cert
{
  cert_store[num_certs] := cert;
  num_certs := num_certs + 1;
}
```

CONCLUSION AND WHAT IS NEXT?

- The Secure Socket API is an effective way of guaranteeing a secure TLS connection (as far as it has been implemented validated)

- Formal verification of meaningful (non-trivial) code is hard

- We lack formal verification that our model represents the codebase
  - Solution: Integrate proof into the codebase

- We need more general tools for formal verification

THANKS!

Any questions?