Analysis of DTLS Implementations Using Protocol State Fuzzing

Paul Fiterau-Brostean
Uppsala University

Bengt Jonsson
Uppsala University

Robert Merget
Ruhr-University Bochum

Joeri de Ruiter
SIDN Labs

Konstantinos Sagonas
Uppsala University

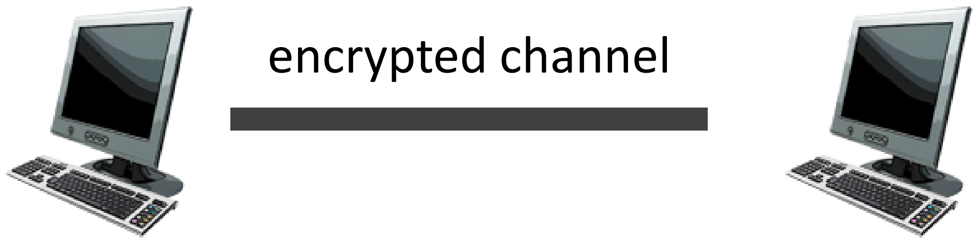
Juraj Somorovsky
Paderborn University

DTLS (TLS over UDP)

IoT

encrypted channel

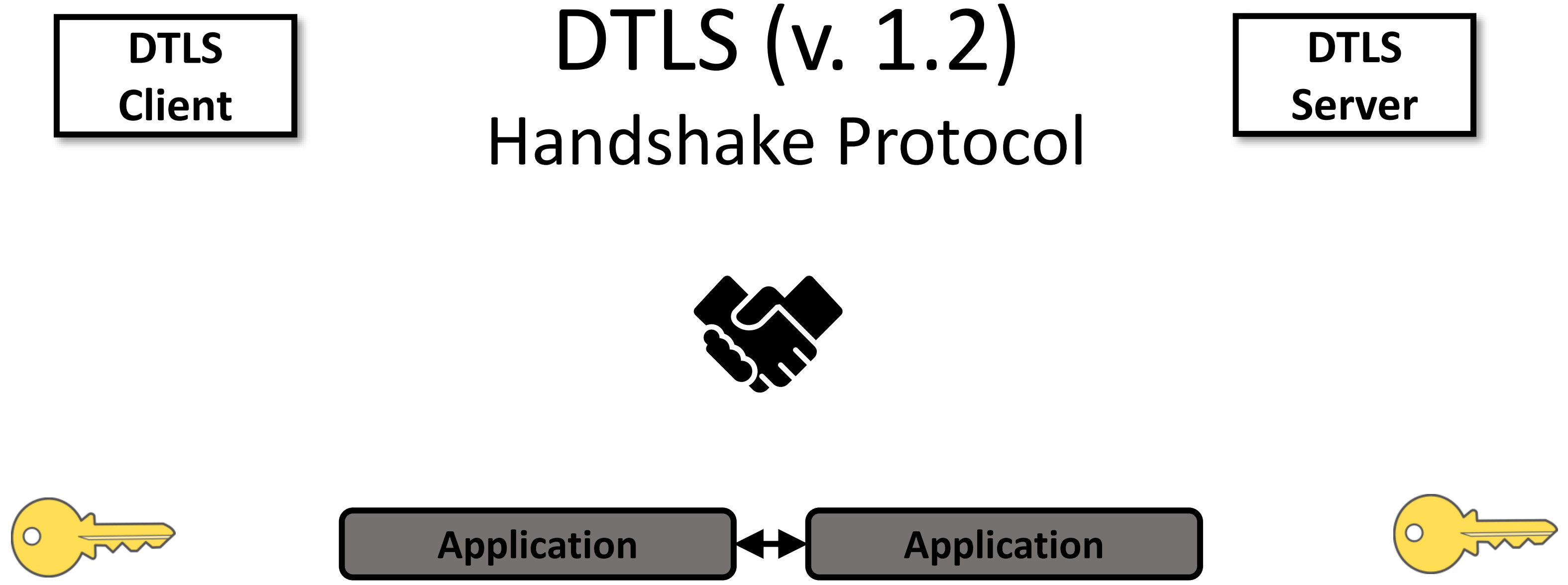
WebRTC



testing not easy
since protocol is **complex**

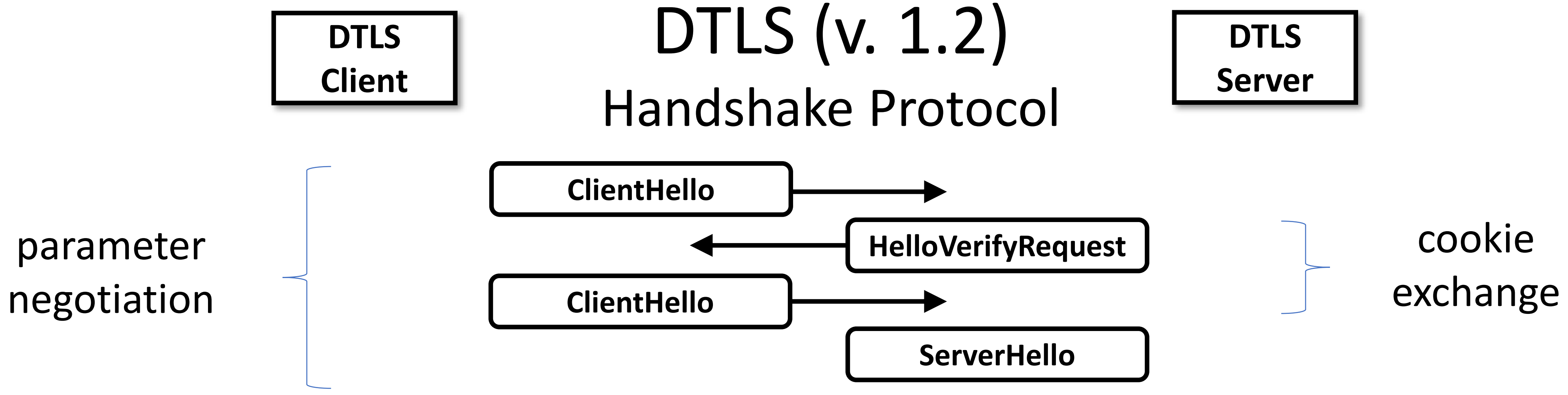
DTLS (v. 1.2)

Handshake Protocol

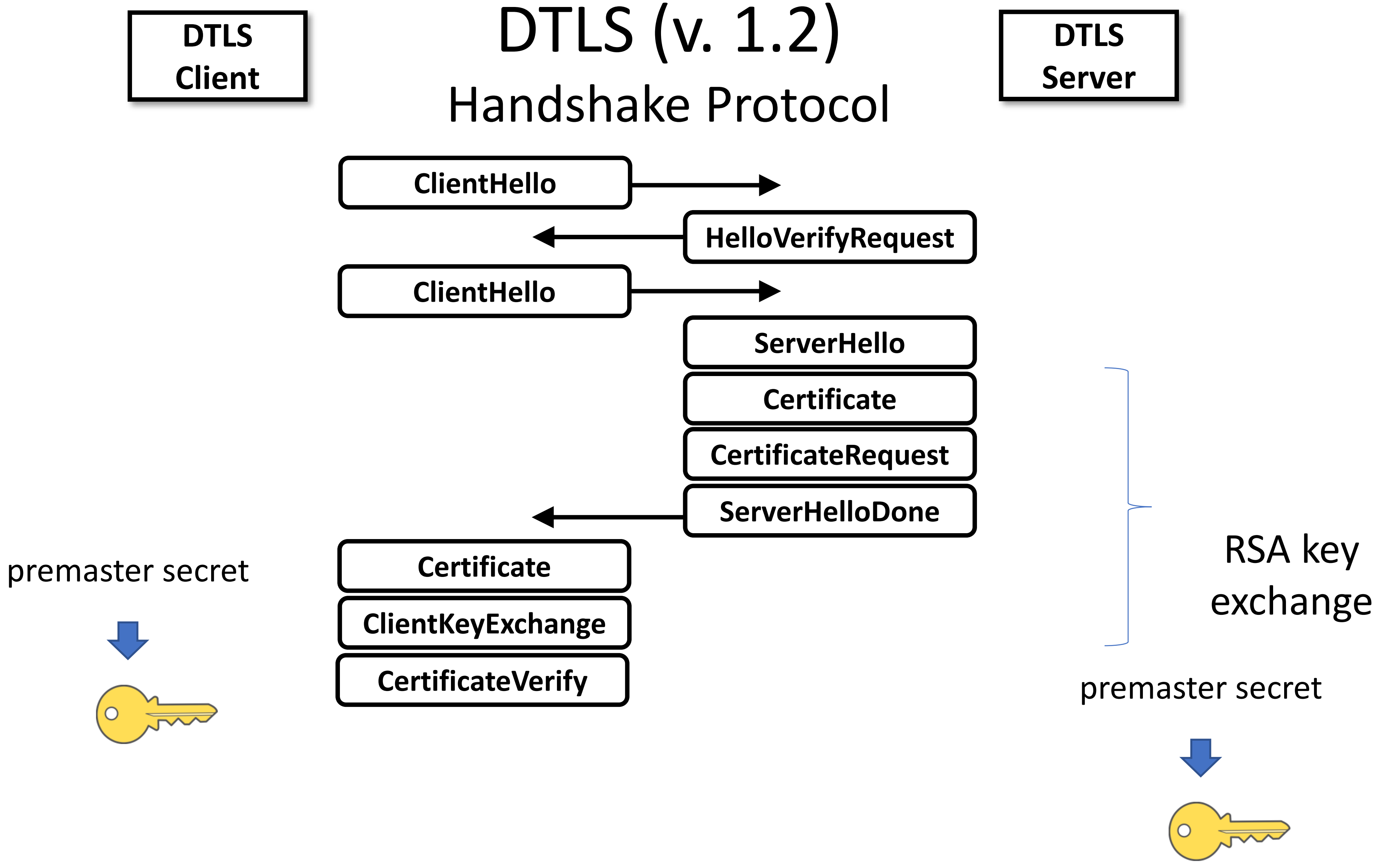


DTLS (v. 1.2)

Handshake Protocol

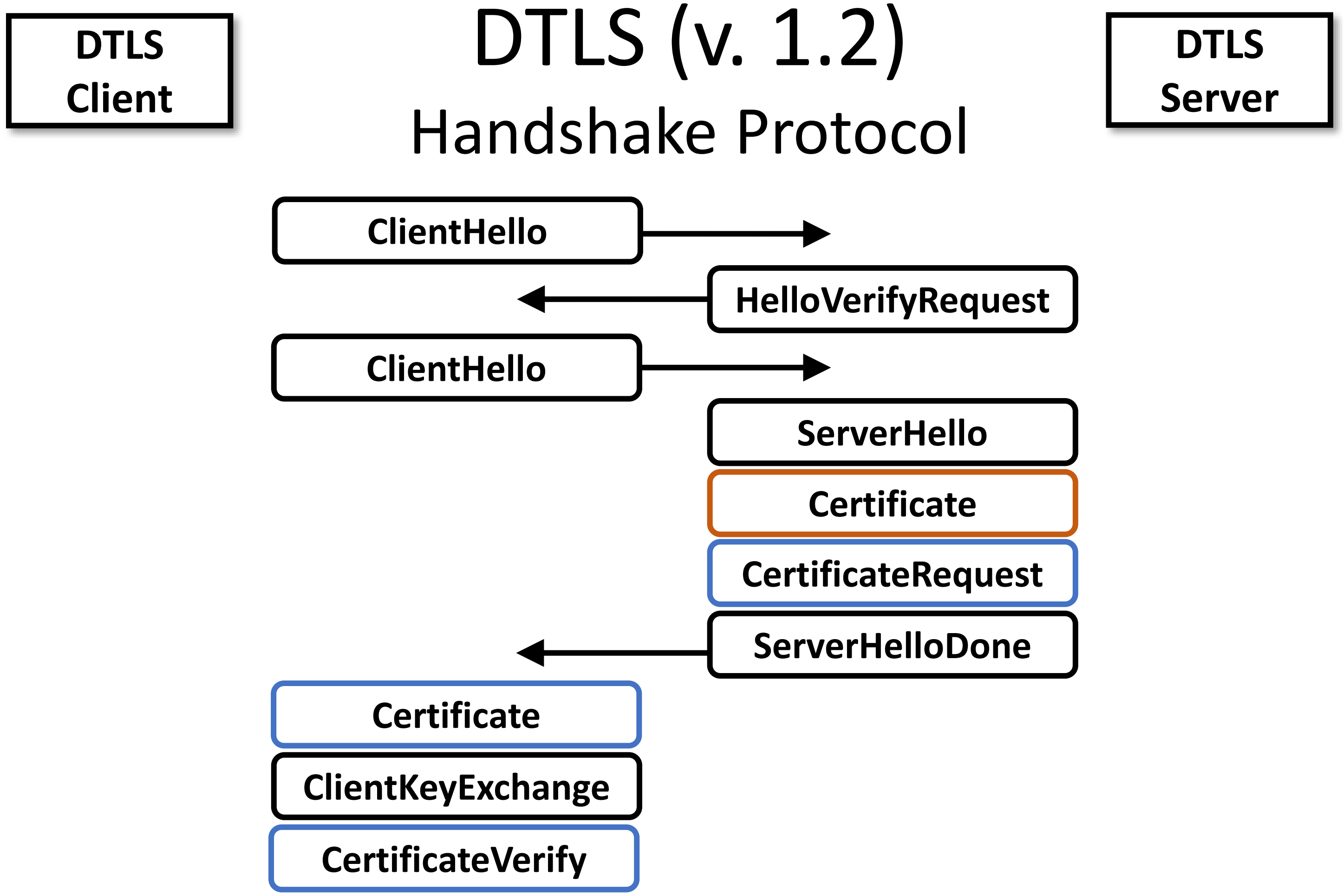


DTLS (v. 1.2) Handshake Protocol

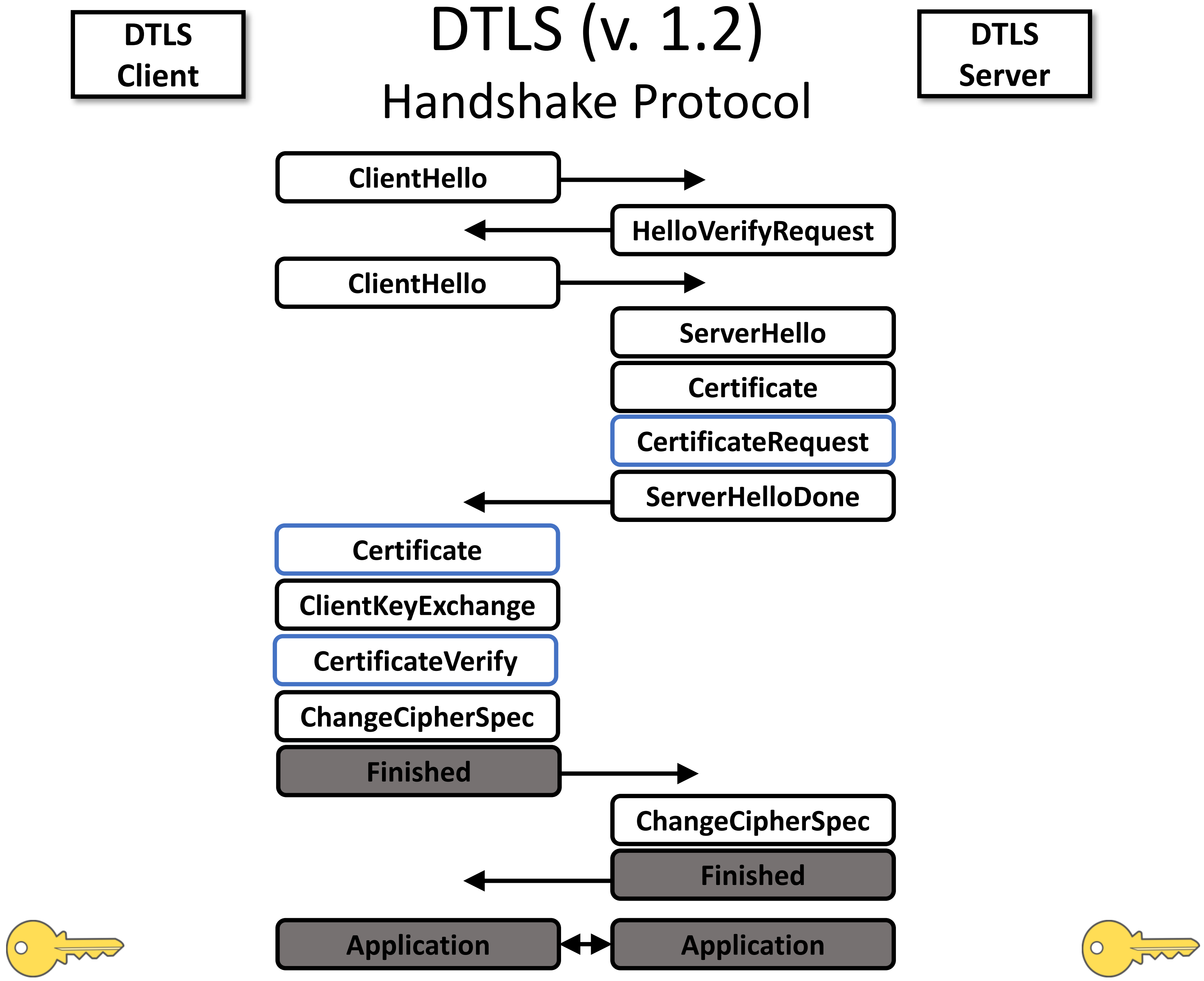


DTLS (v. 1.2)

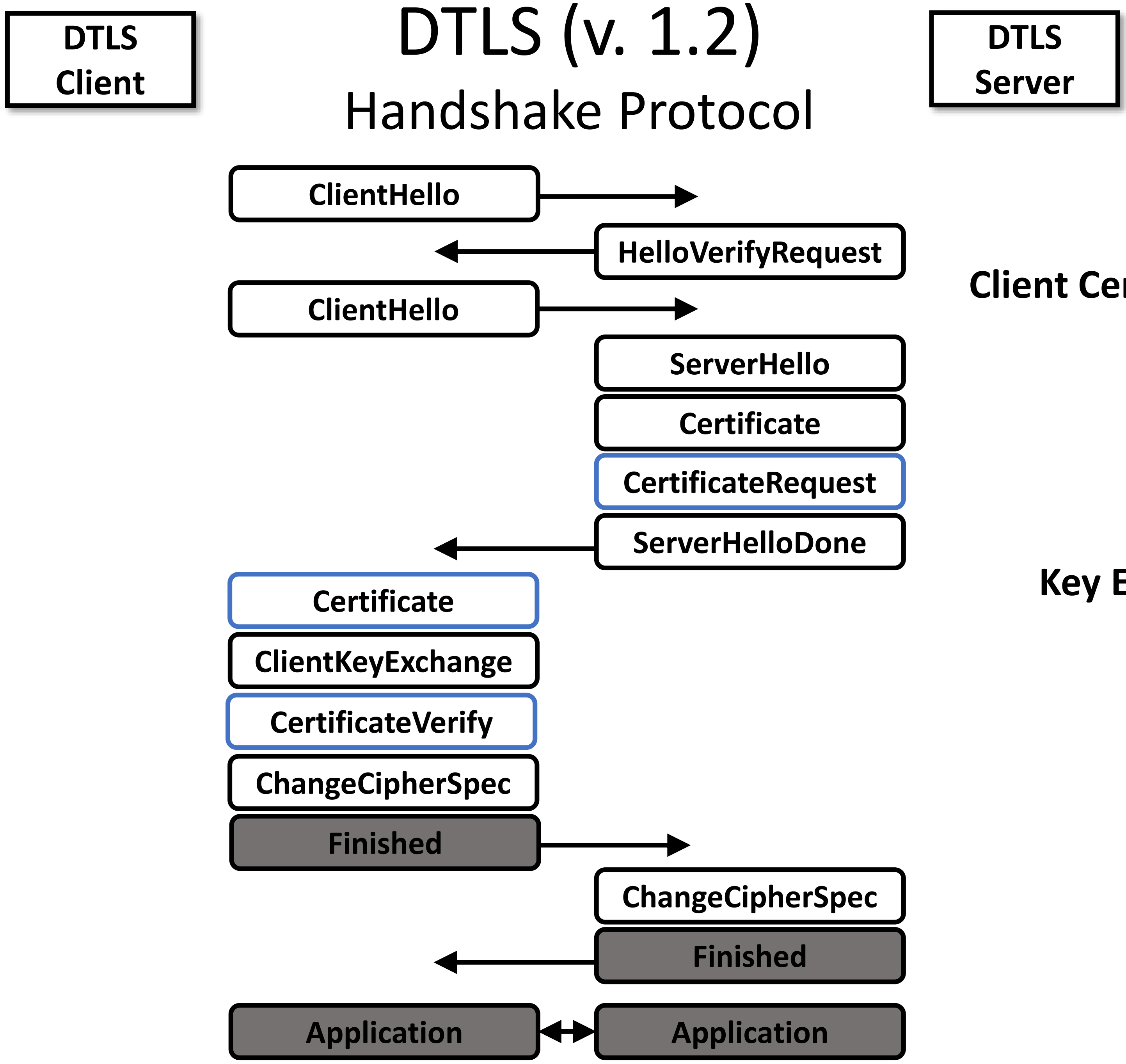
Handshake Protocol



DTLS (v. 1.2)
Handshake Protocol



DTLS (v. 1.2)
Handshake Protocol

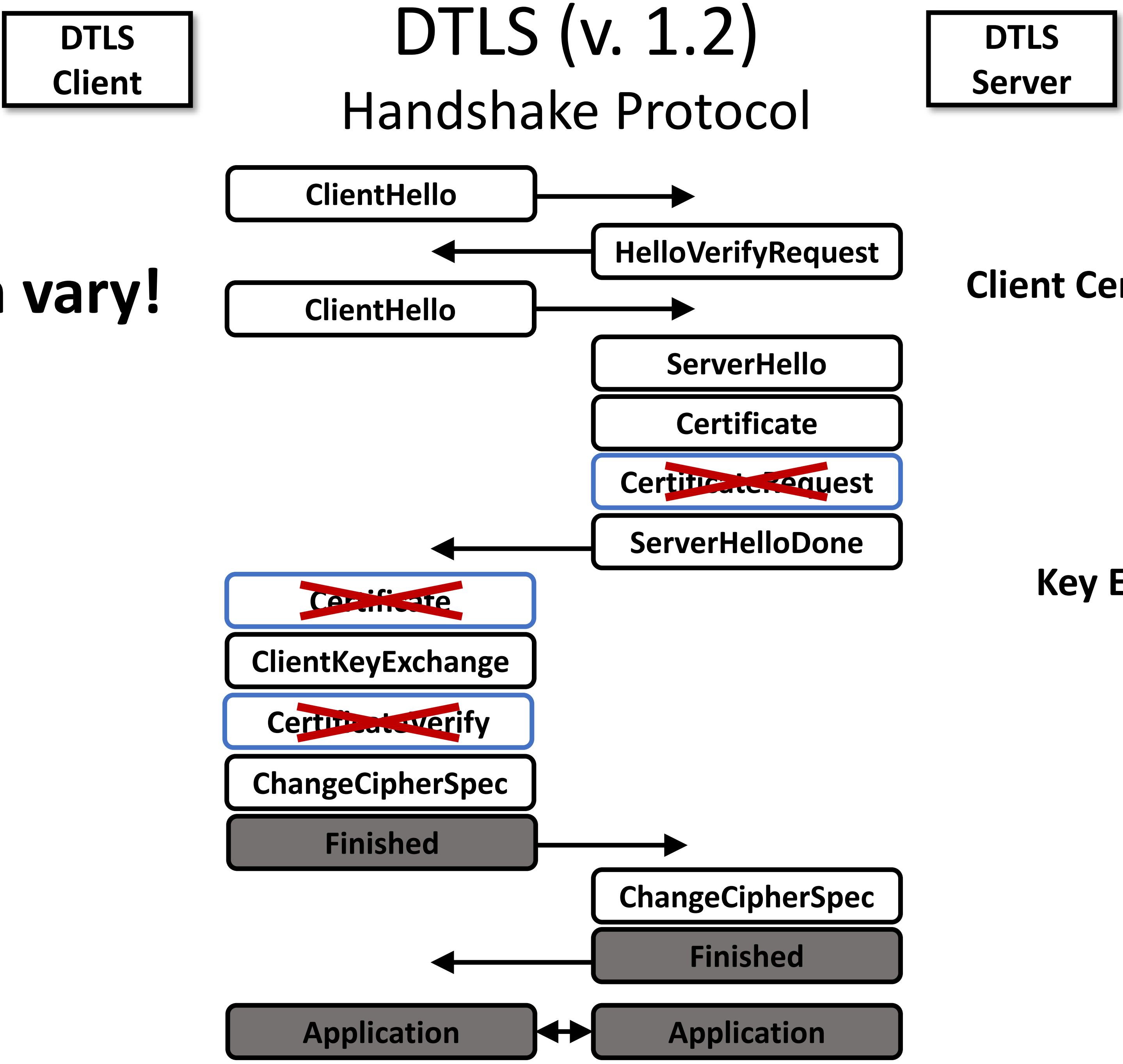


Client Certificate Authentication
required

Key Exchange Algorithm
RSA

DTLS (v. 1.2)
Handshake Protocol

Handshake can vary!

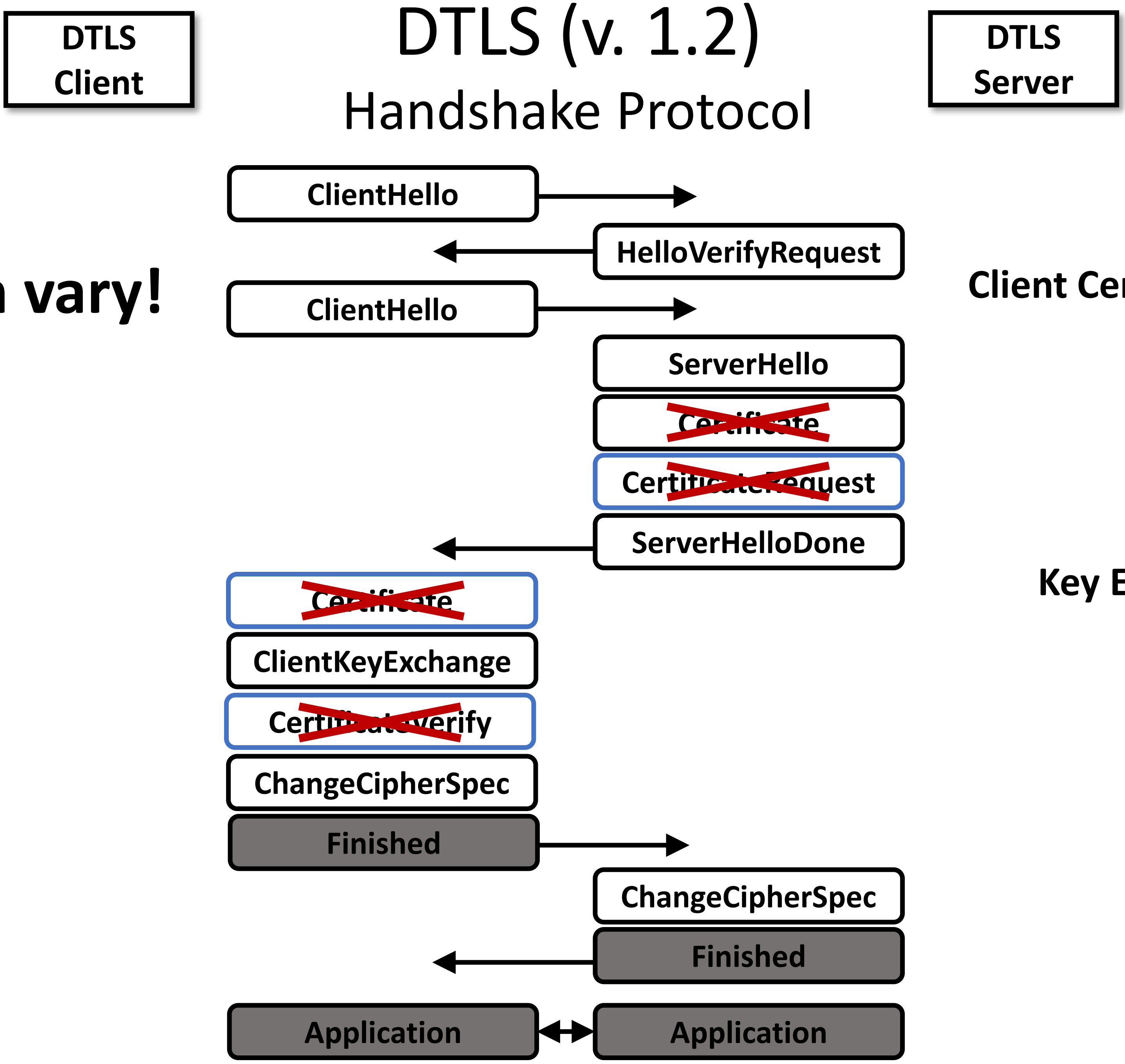


Client Certificate Authentication
disabled

Key Exchange Algorithm
RSA

DTLS (v. 1.2) Handshake Protocol

Handshake can vary!



Client Certificate Authentication
disabled

Key Exchange Algorithm
PSK

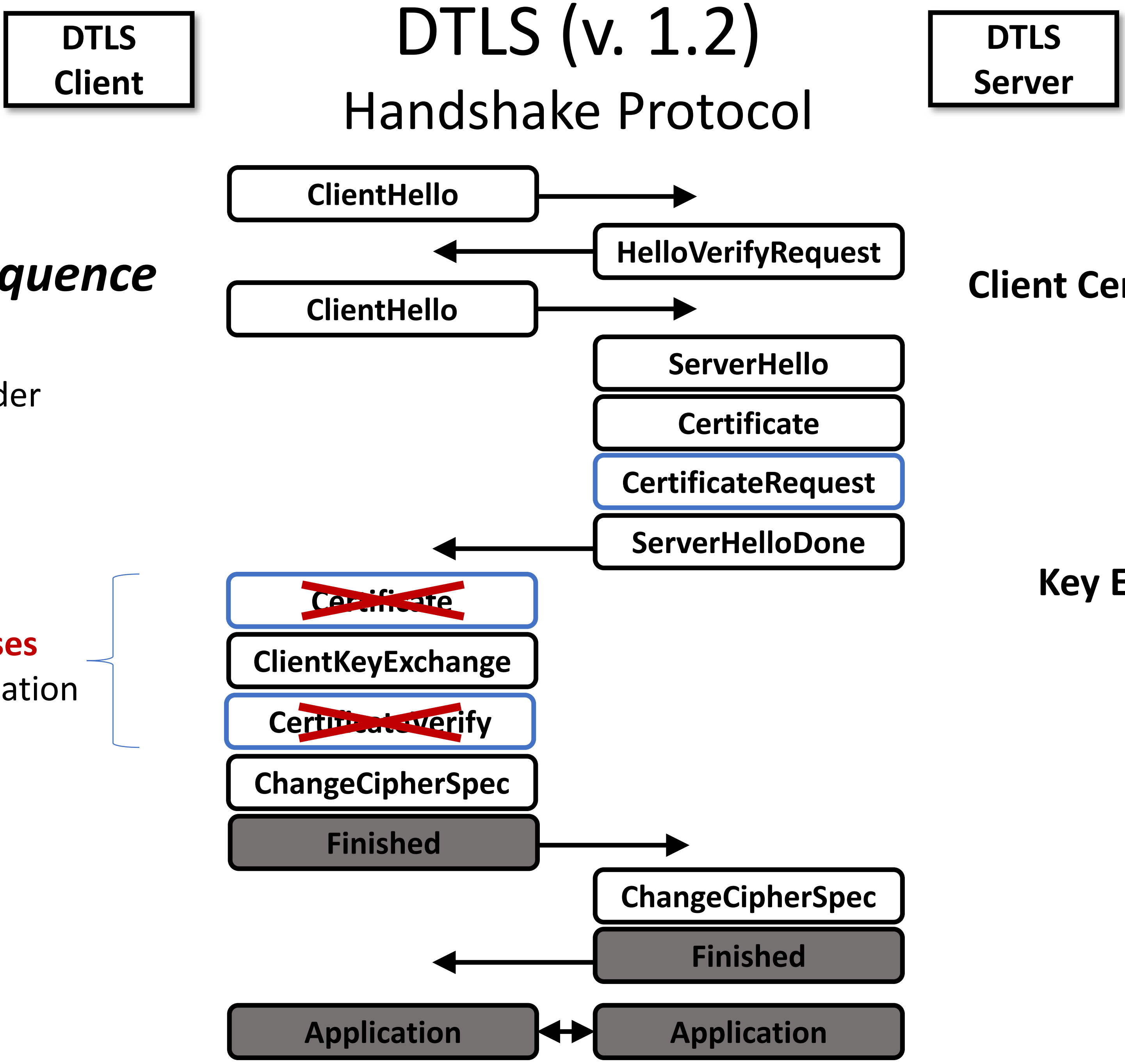
DTLS (v. 1.2)

Handshake Protocol

Control Flow Bug:

accept invalid sequence

i.e. messages
(1) in wrong order
(2) missing



bypasses authentication

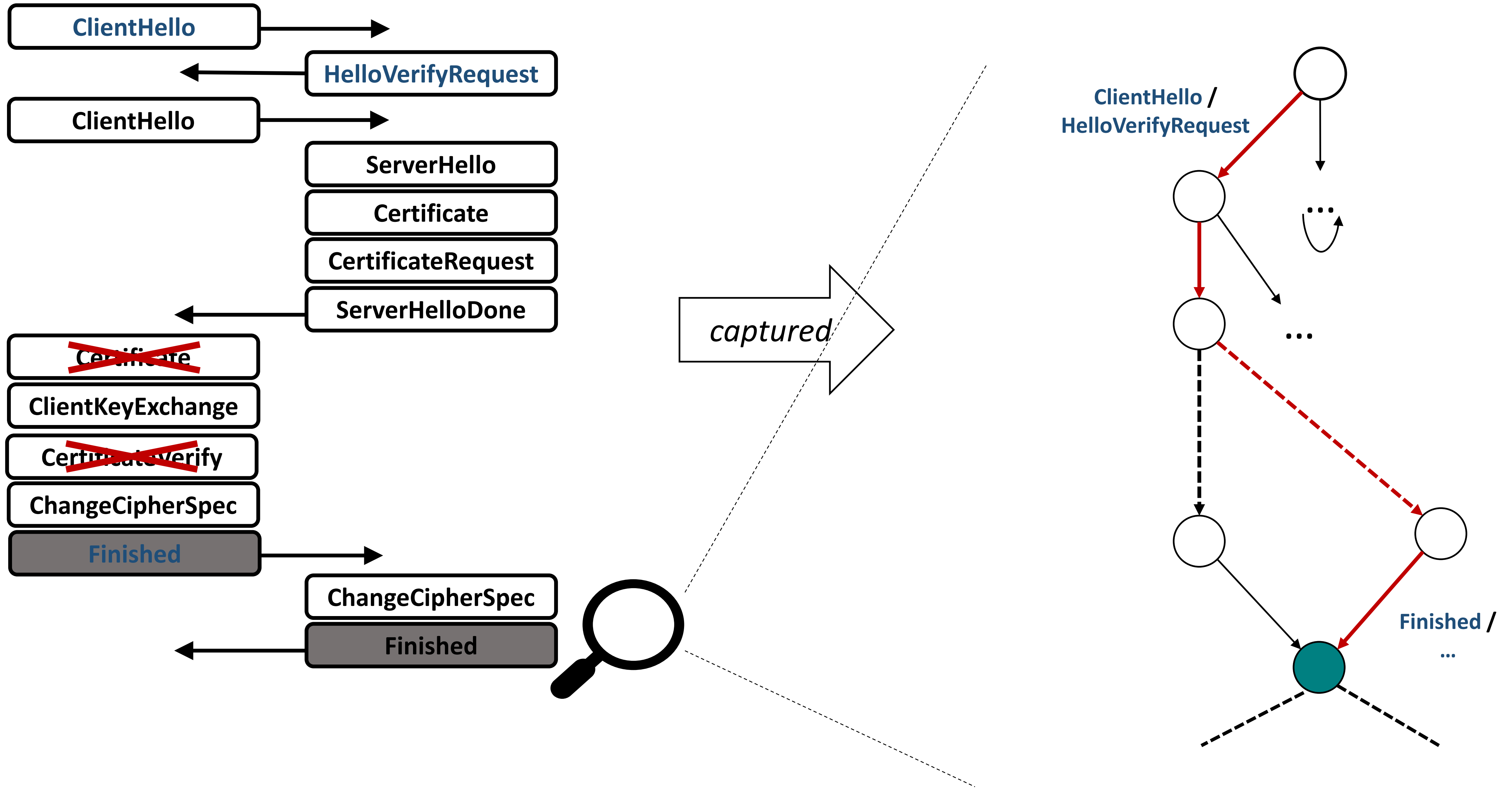
Client Certificate Authentication **required**

Key Exchange Algorithm RSA

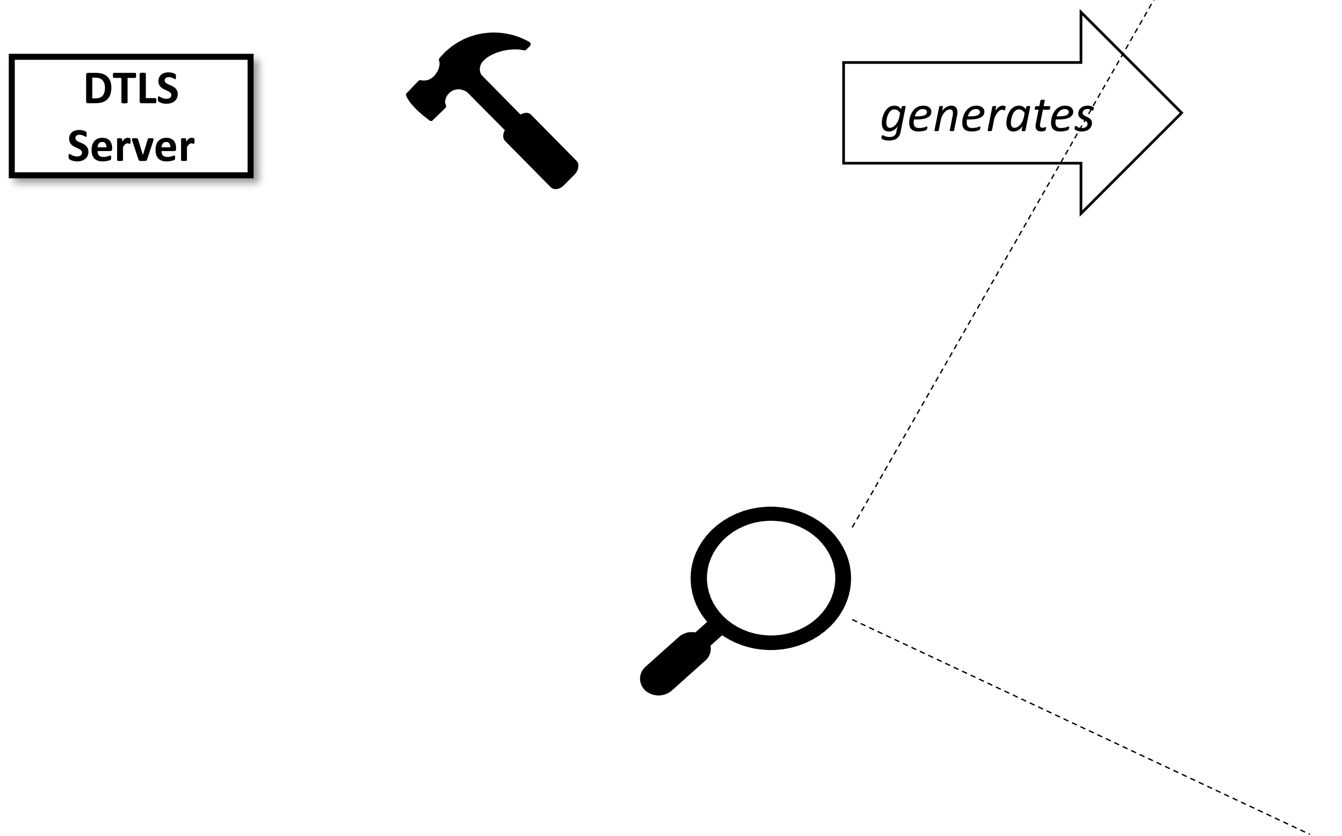
Culprit: JSSE

DTLS Server State Machine

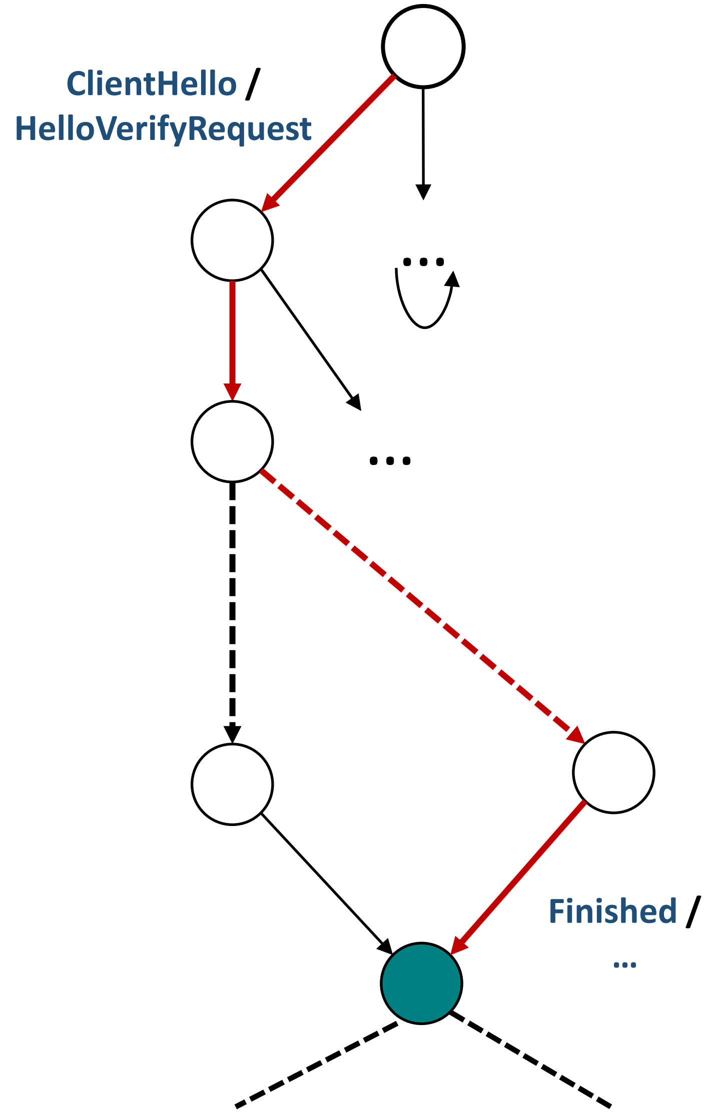
infinitely-many sequences to test



State Fuzzing infers state machine **automatically**



DTLS Server State Machine



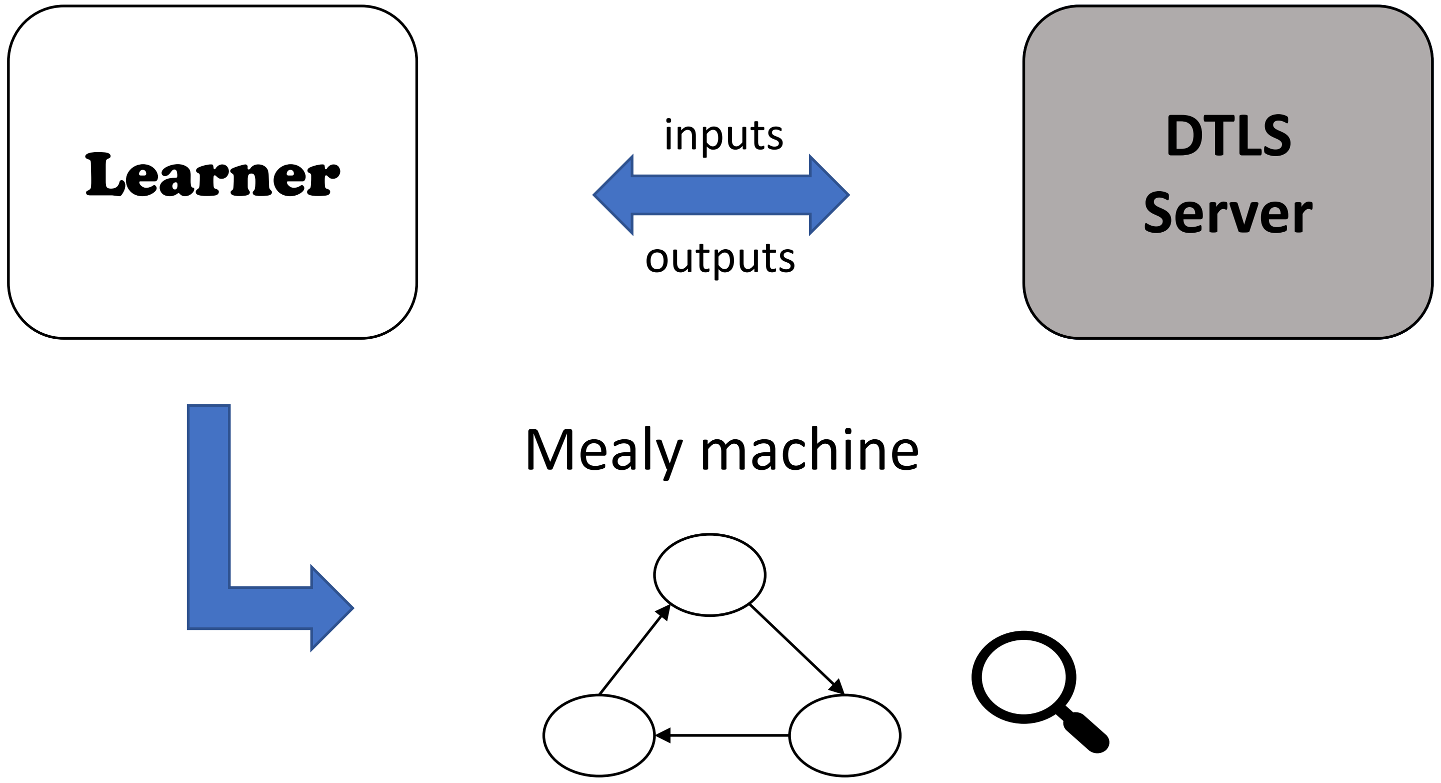
Challenges

- implementing the state fuzzing setup
- apply to many implementations + configurations
- inspect models for security issues

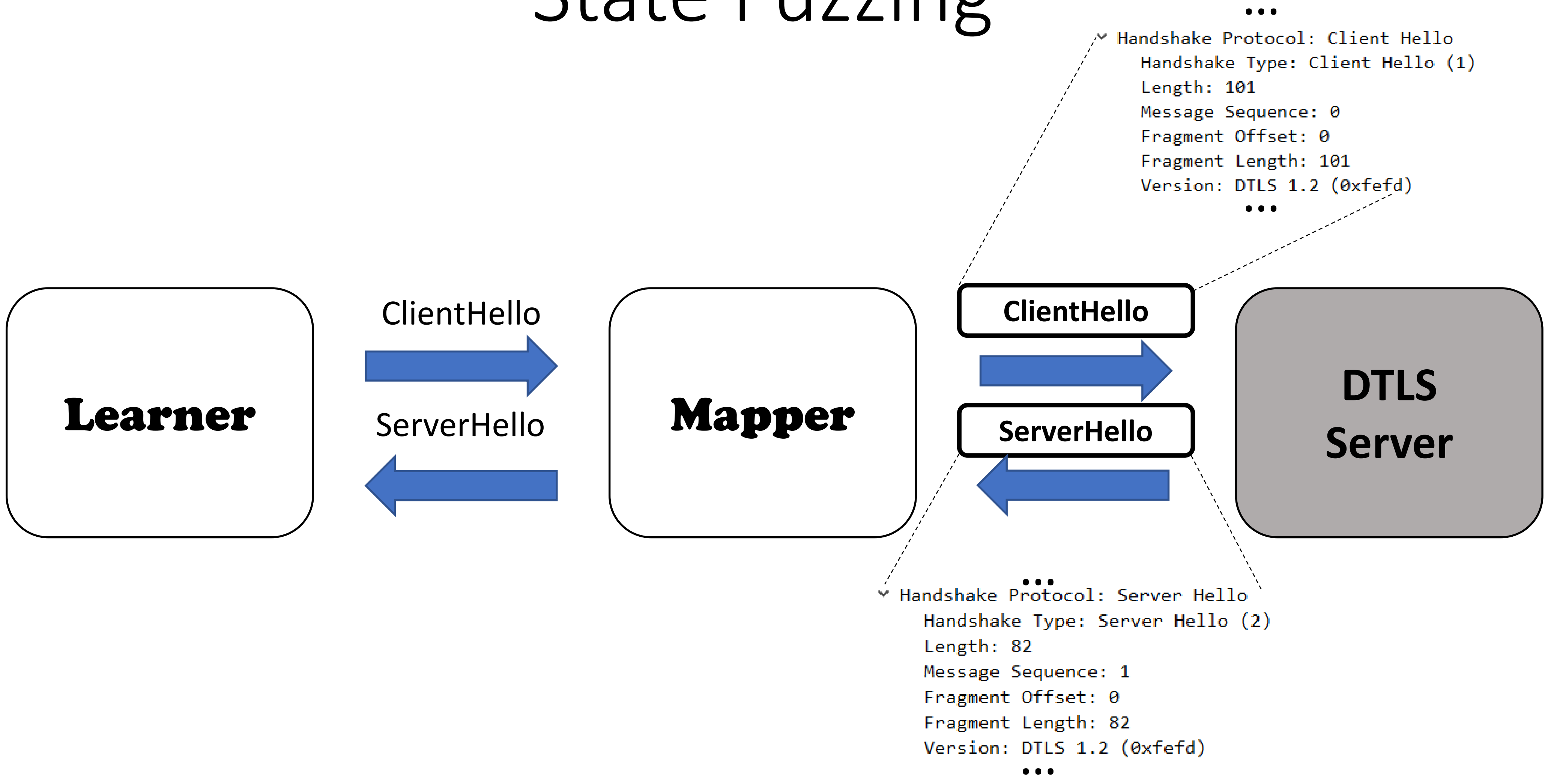
Contributions

- ➢state fuzzing framework for testing DTLS servers
- ➢analysis of 11 server implementations
    - ➢RSA, PSK, DH, ECDH
    - ➢disabled/enabled/required cl. cert. auth.
- ➢>10 bugs ➔ fixes in 5 implementations

State Fuzzing



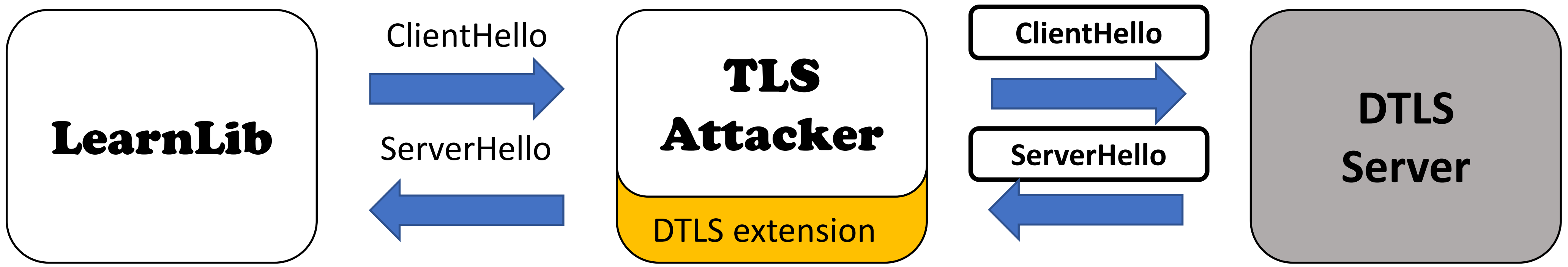
State Fuzzing



```
…
Handshake Protocol: Client Hello
   Handshake Type: Client Hello (1)
   Length: 101
   Message Sequence: 0
   Fragment Offset: 0
   Fragment Length: 101
   Version: DTLS 1.2 (0xfefd)
…
```

```
…
Handshake Protocol: Server Hello
   Handshake Type: Server Hello (2)
   Length: 82
   Message Sequence: 1
   Fragment Offset: 0
   Fragment Length: 82
   Version: DTLS 1.2 (0xfefd)
…
```

Implementing Components



In our setup:

- learner - LearnLib
- mapper - TLS-Attacker extended with DTLS support

Devising I/O Alphabet

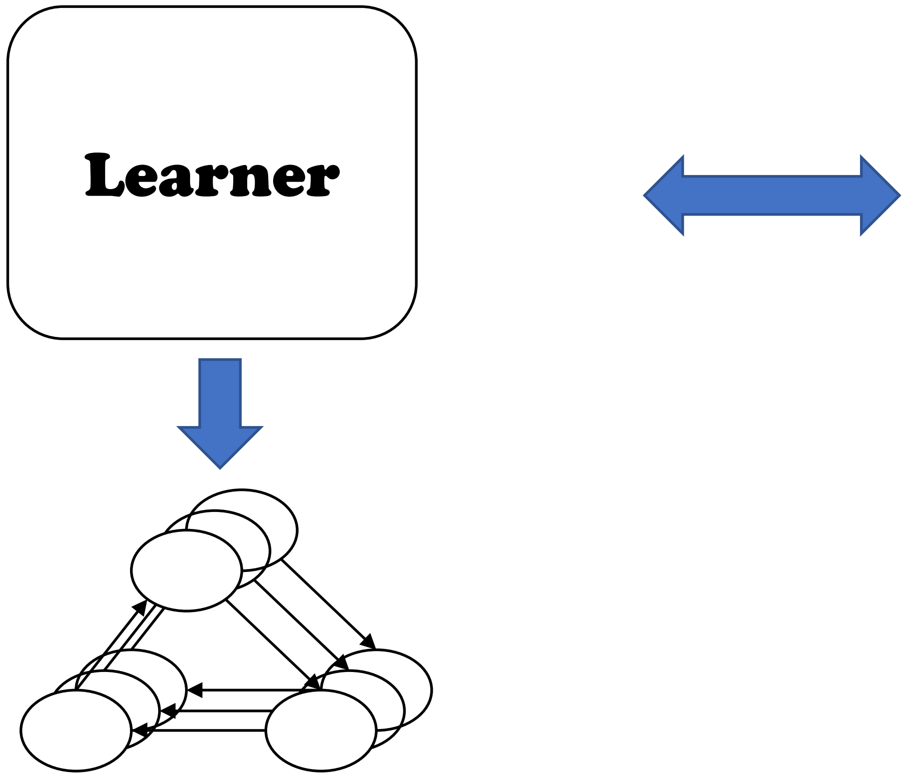
input alphabet

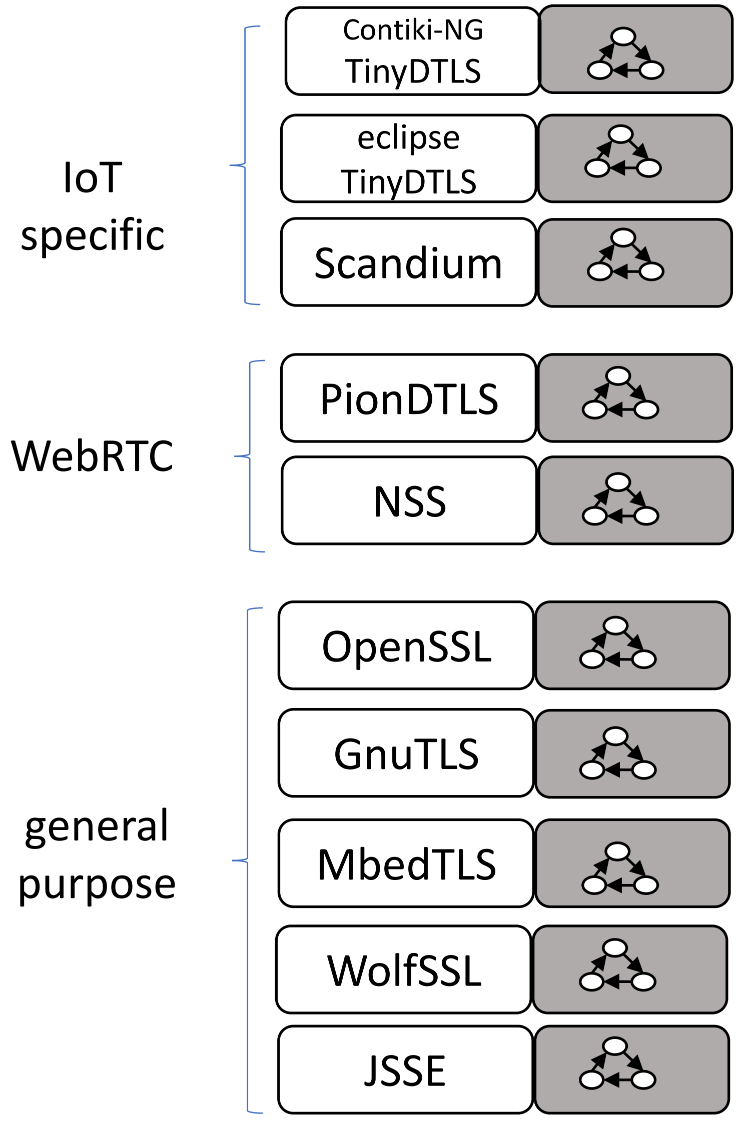
- ClientHello(*kex*) $kex \in \{\text{RSA, DH, PSK, ECDH}\}$
- ClientKeyExchange(*kex*)
- Certificate
- EmptyCertificate
- CertificateVerify
- ChangeCipherSpec
- Finished
- Application
- Alert(*type*) $type \in \{\text{CloseWait, UnexpectedMessage}\}$

output alphabet

- HelloVerifyRequest
- ServerHello
- Certificate
- CertificateRequest
- ServerKeyExchange
- ServerHelloDone
- ChangeCipherSpec
- Finished
- Application
- Alert(*type*) $type \in \{\text{CloseWait,...}\}$
- UnknownMessage (i.e. unable to decrypt)
- - (no response)

Generating Models





for all main key exchange algorithms

RSA, DH, ECDH, PSK

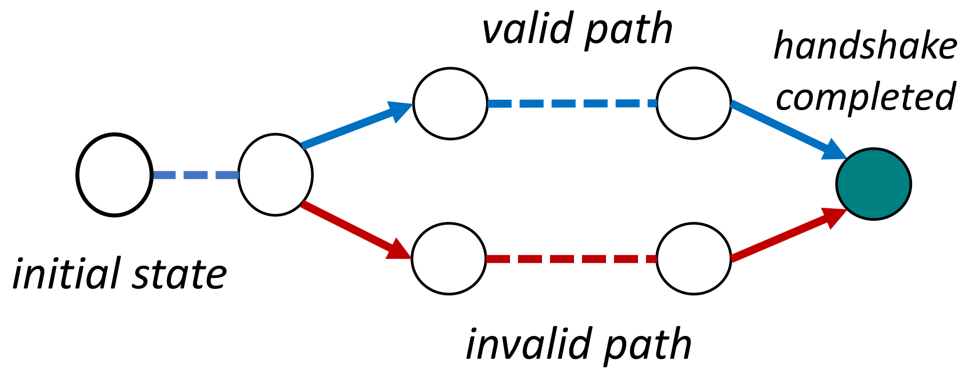
and client cert. authentication settings

NONE, REQ, OPT

Analyzing Results

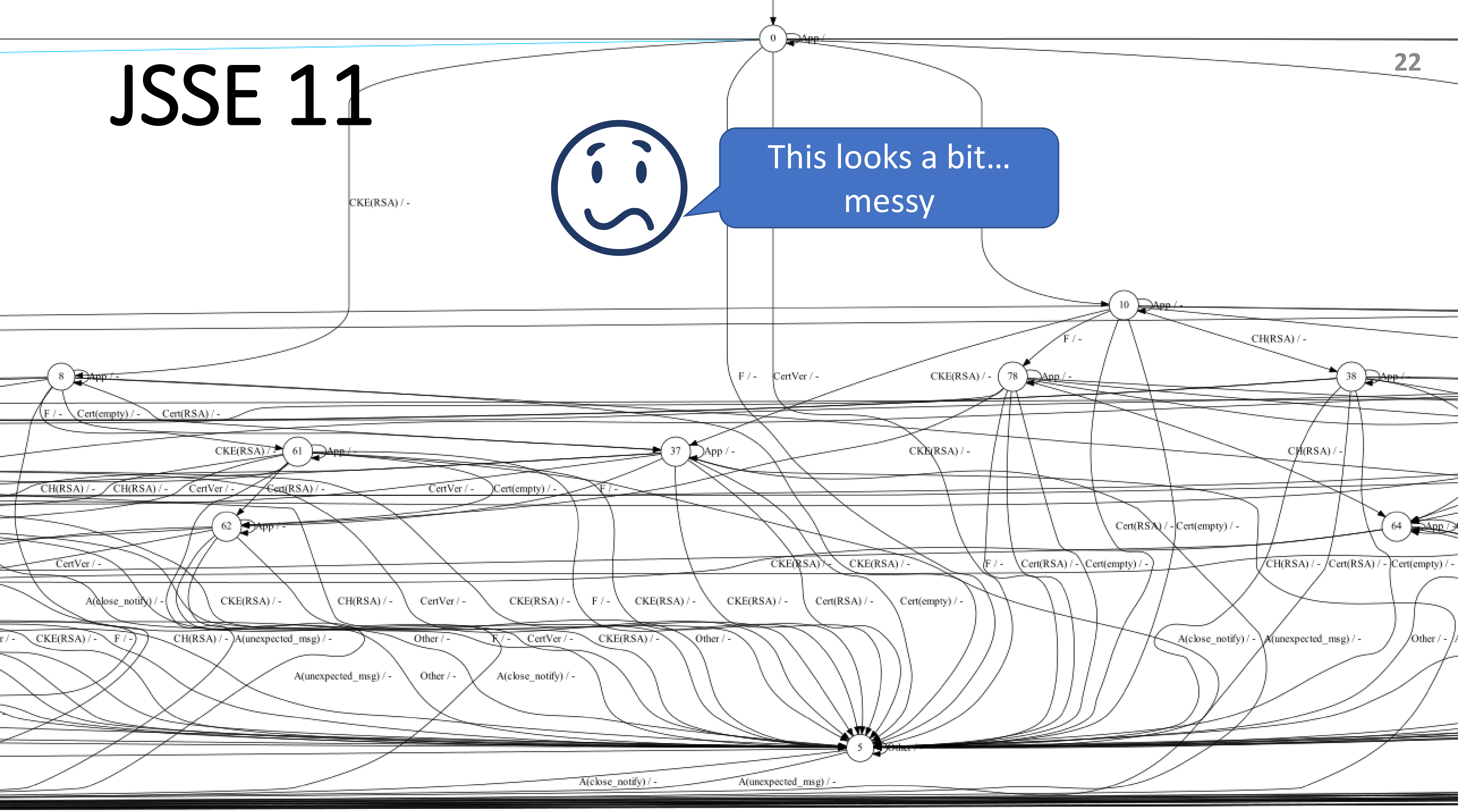
IoT specific	Contiki-NG TinyDTLS		
	eclipse TinyDTLS		
	Scandium	X	🔧
WebRTC	PionDTLS	X	🔧
	NSS		
general purpose	OpenSSL		
	GnuTLS		🔧
	MbedTLS		
	WolfSSL		🔧
	JSSE	X	🔧

- ➢ found RFC violations (i.e. bug) in **all** models
- ➢ prompted fixes (🔧)
- ➢ most severe: handshake bugs (X)



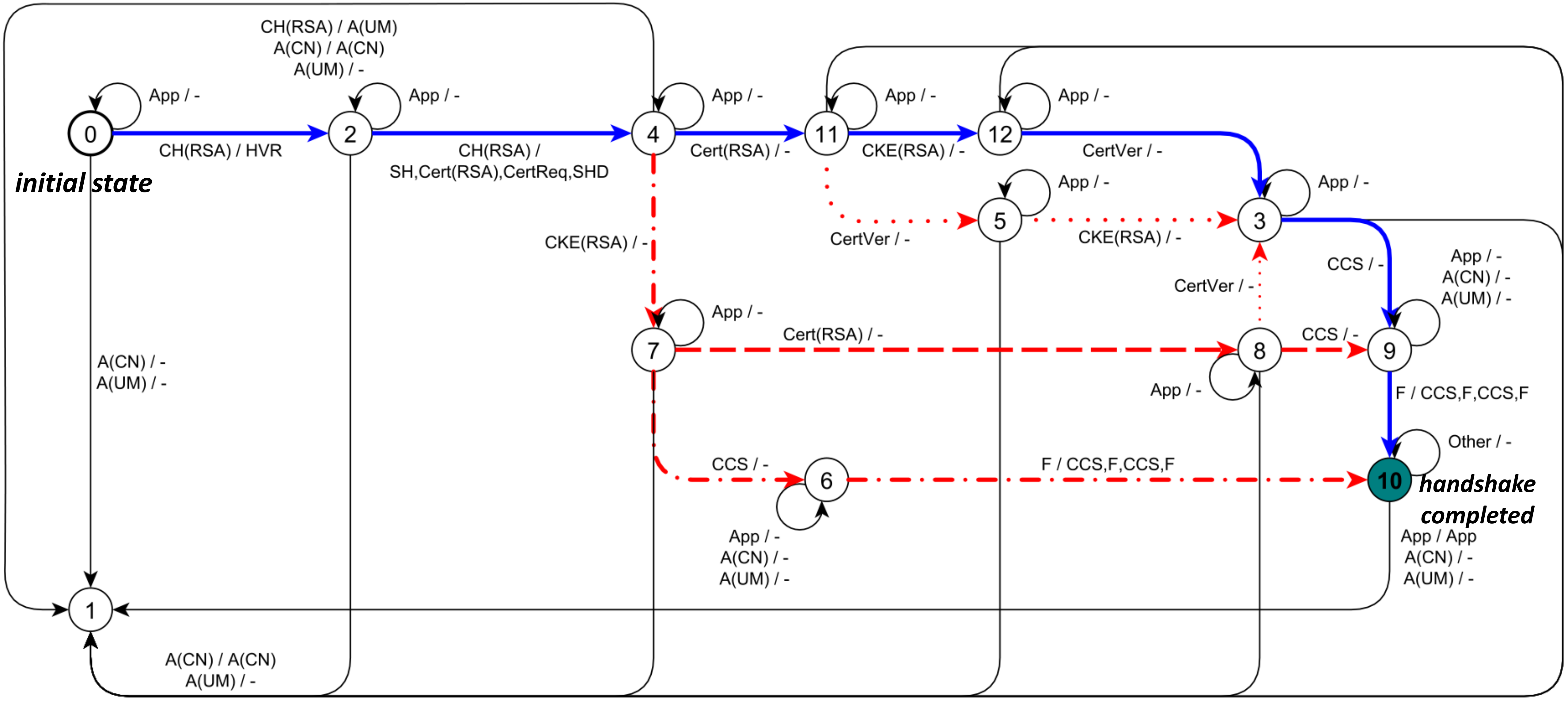
JSSE 11

This looks a bit… messy



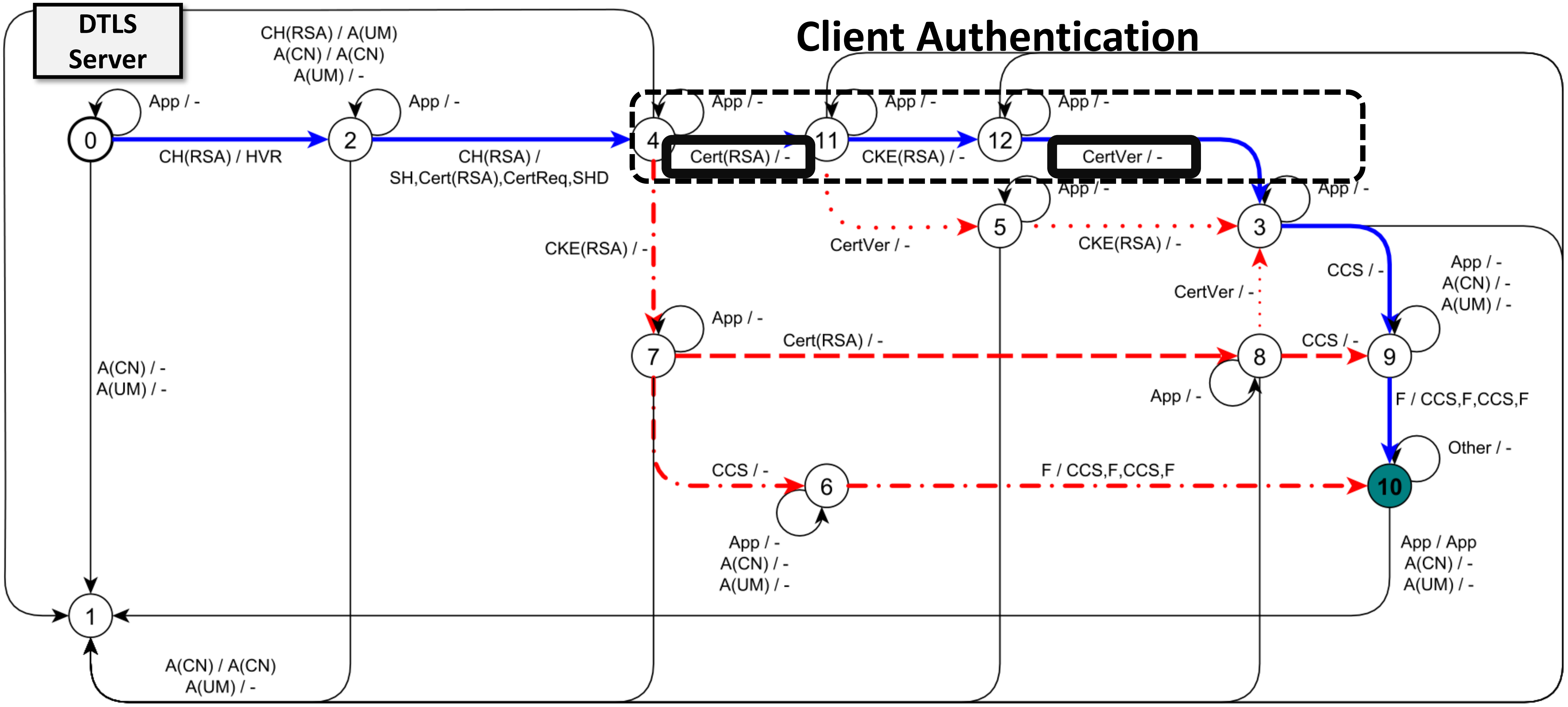
JSSE 11

➔ server configured to require cert. authentication



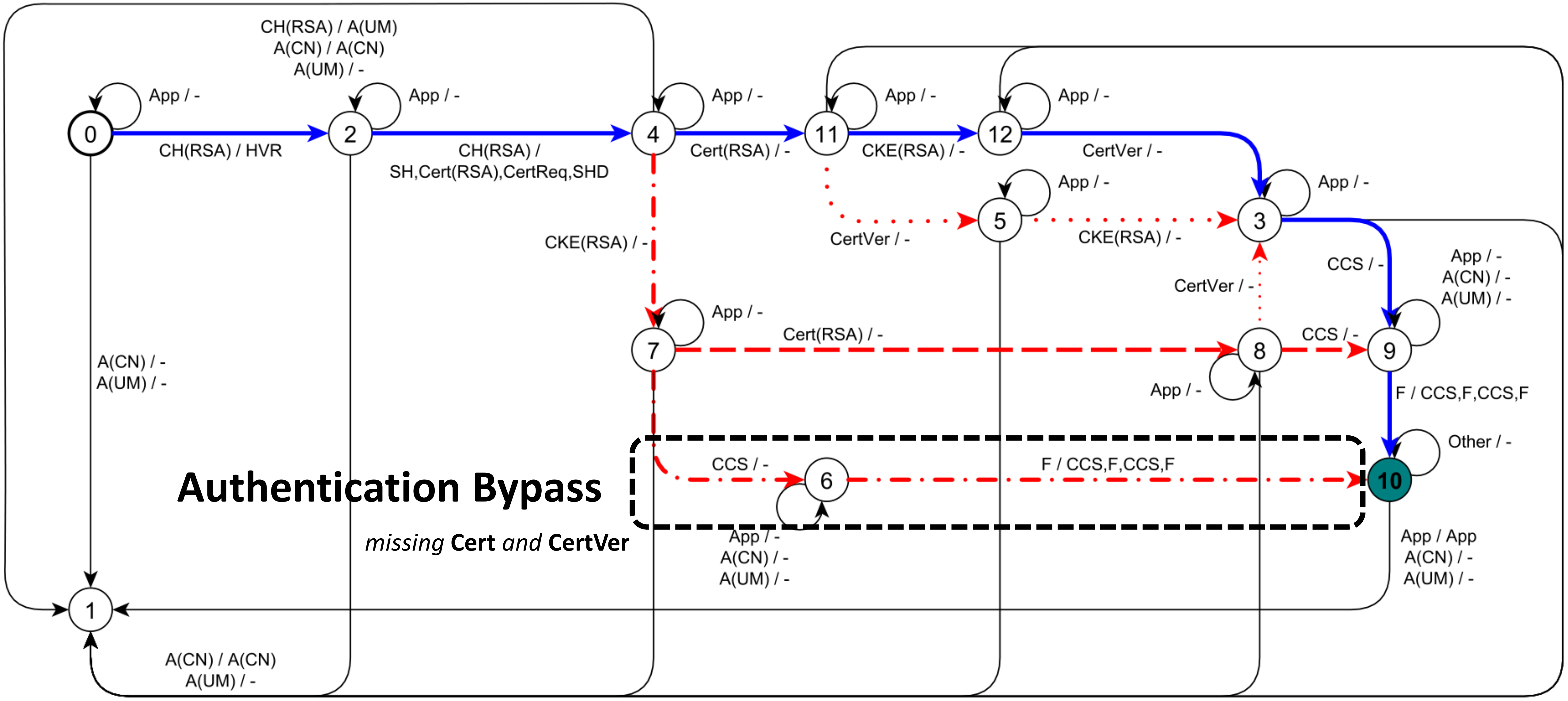
JSSE 11

➔ server configured to require cert. authentication



JSSE 11

➔ bypass authentication

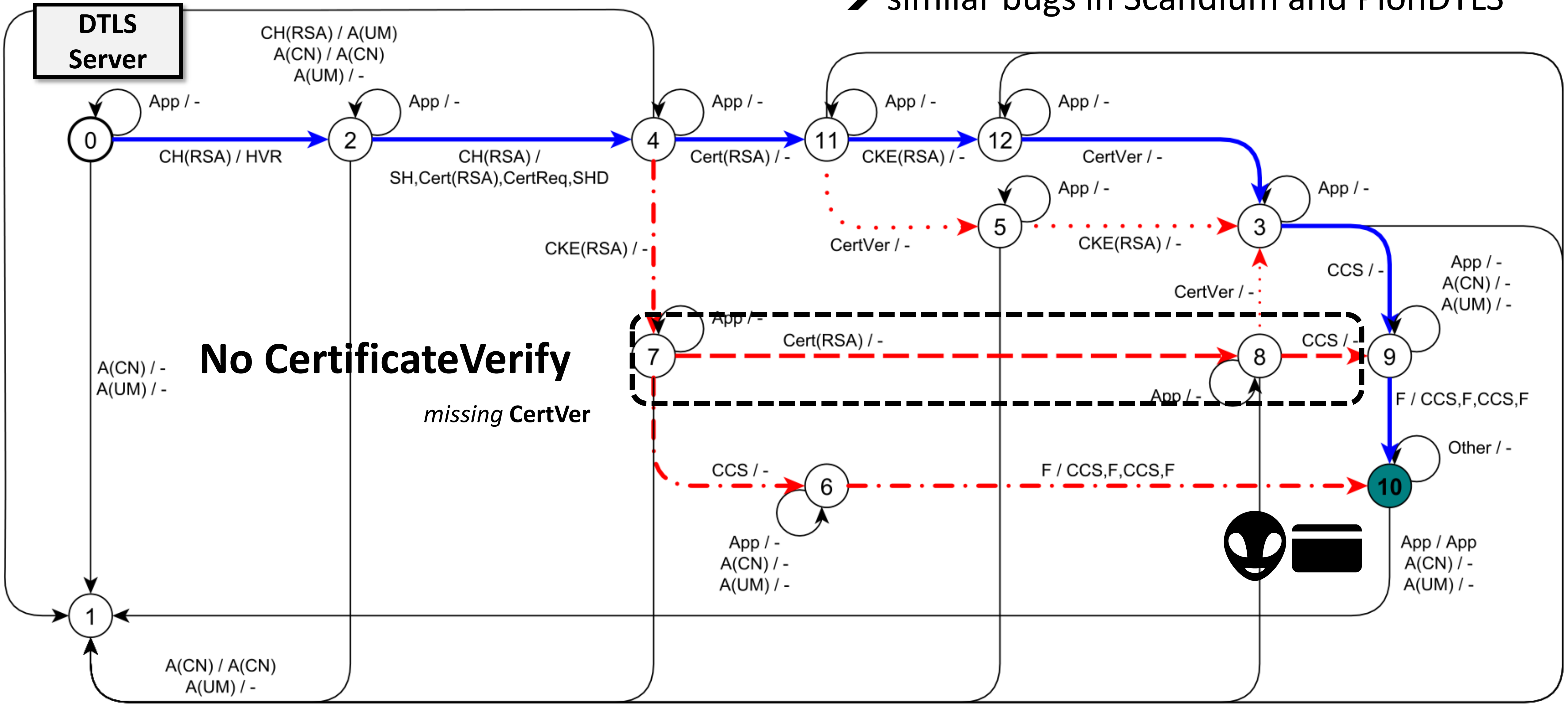


Authentication Bypass

missing **Cert** *and* **CertVer**

JSSE 11

➔ authenticate with **stolen** Certificate
➔ similar bugs in Scandium and PionDTLS



Conclusions

- ➢implemented a state fuzzing framework for DTLS
- ➢extended **TLS-Attacker** with support for DTLS
- ➢analyzed 11 implementations
  - ➢RSA, PSK, DH, ECDH
  - ➢disabled/enabled/required cl. cert. auth.
- ➢found >10 bugs ➔ fixes in 5 implementations (so far…)
- ➢constructed a platform for future work on testing DTLS

Conclusions

- ➢implemented a state fuzzing framework for DTLS
- ➢extended **TLS-Attacker** with support for DTLS
- ➢analyzed 11 implementations for diff. key exchange/cl. cert. auth.
- ➢found >10 bugs ➔ fixes in 5 implementations (so far…)
- ➢constructed a platform for future work on testing DTLS
  - ❑ automatic detection of bugs in models
  - ❑ state fuzzing DTLS clients
  - ❑ testing unexplored functionality (e.g. fragmentation)

guided by the learned models…

State Fuzzing Framework: https://github.com/assist-project/dtls-fuzzer

Conclusions

- ➢implemented a state fuzzing framework for DTLS
- ➢extended **TLS-Attacker** with support for DTLS
- ➢analyzed 11 implementations for diff. key exchange/cl. cert. auth.
- ➢found >10 bugs ➔ fixes in 5 implementations (so far…)
- ➢constructed a platform for future work on testing DTLS
  - ❑ automatic detection of bugs in models
  - ❑ state fuzzing DTLS clients
  - ❑ testing unexplored functionality (e.g. fragmentation)

Thanks for your attention!

State Fuzzing Framework: https://github.com/assist-project/dtls-fuzzer