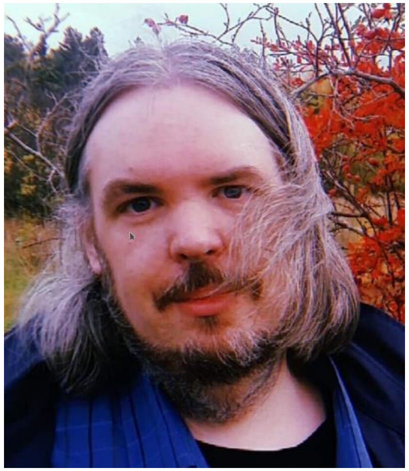
Automata-Based Automated Detection of State Machine Bugs in Protocol Implementations



Paul Fiterău-Broştean[*], Bengt Jonsson,[*] Konstantinos Sagonas[*†] and Fredrik Tåquist[*]
[*]Department of Information Technology, Uppsala University, Uppsala, Sweden
[†]School of Electrical and Computer Engineering, National Technical Univesity of Athens, Athens, Greece

Introduction

- Work done as part of the SSF aSSIsT project
  - **Goal**: Develop techniques to automatically detect bugs and vulnerabilities in network protocol implementations.
  - **Method**: *Protocol State Fuzzing* (a.k.a. Model Learning).

This paper:

- Presents a **general**, **automated, black-box** technique to detect *state machine bugs* in protocols starting from:
  1. a state machine model of the implementation;
  2. a catalogue of bug patterns for the protocol.
- Evaluates it on SSH servers and DTLS servers and clients.

Datagram TLS (DTLS)

The Design and Implementation of Datagram TLS

Test of Time
Award Winner
NDSS 2020

Nagendra Modadugu
Stanford University
nagendra@cs.stanford.edu

Eric Rescorla
RTFM, Inc.
ekr@rtfm.com

Abstract

A number of applications have emerged over recent years that use datagram transport. These applications include real time video conferencing, Internet telephony, and online games such as Quake and StarCraft. These applications are all delay sensitive and use unreliable datagram transport. Applications that are based on reliable transport can be secured using TLS, but no compelling alternative exists for securing datagram based applications. In this paper we present DTLS, a datagram capable version of TLS. DTLS is extremely similar to TLS and therefore allows reuse of pre-existing protocol infrastructure. Our experimental results show that DTLS adds minimal overhead to a previously non-DTLS capable application.

a number of unsatisfactory choices for providing security. First, they can use IPsec [18]. However, IPsec is not well suited for client-server application models and is difficult to package with applications since it runs in the kernel. Section 2.1 has a detailed discussion of why IPsec has been found to be a less than satisfactory option. Second, they can design a custom application layer security protocol. SIP, for instance, uses a variant of S/MIME [2] to secure its traffic. Grafting S/MIME into SIP took vastly more effort than did running the TCP variant of SIP over TLS. Third, one can rehost the application on TCP and use TLS. Unfortunately many such applications depend on datagram semantics and have unacceptable performance when run over a stream protocol such as TCP.

The obvious alternative is to design a generic channel security protocol that will do for long lived applications

DTLS Handshake

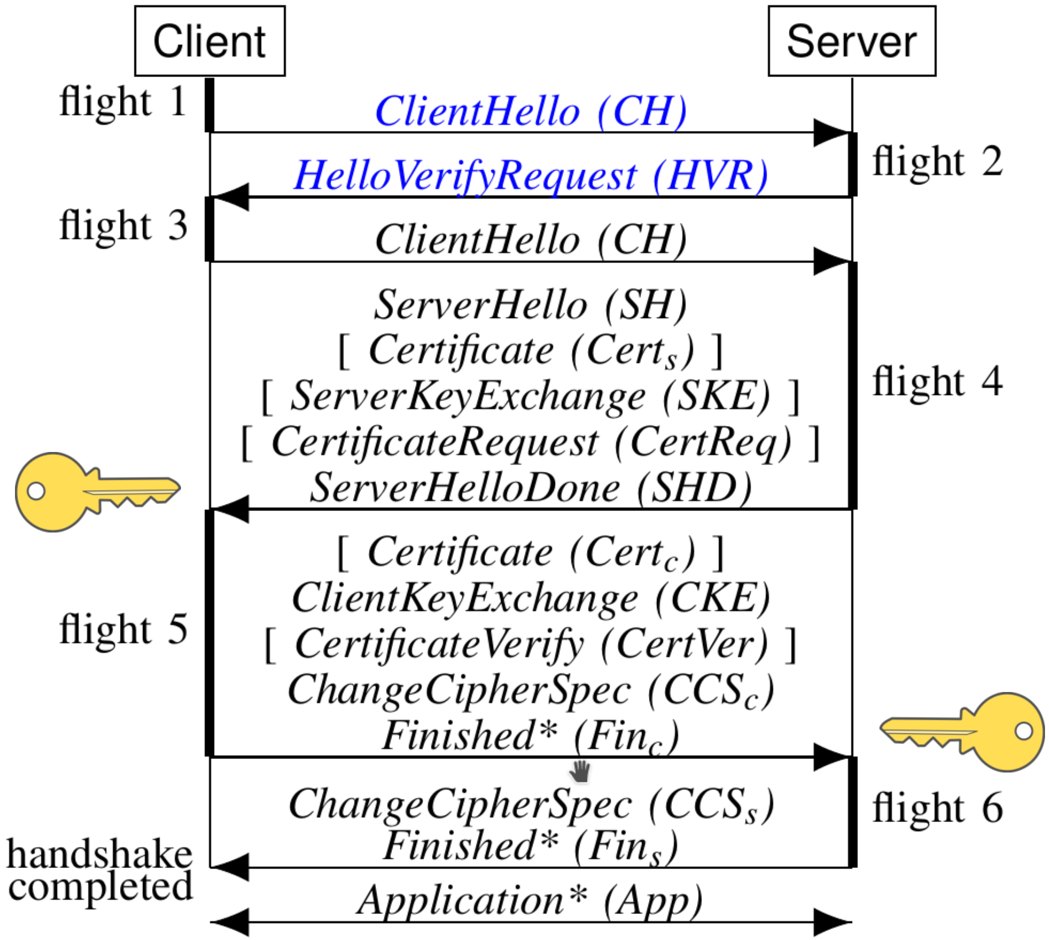
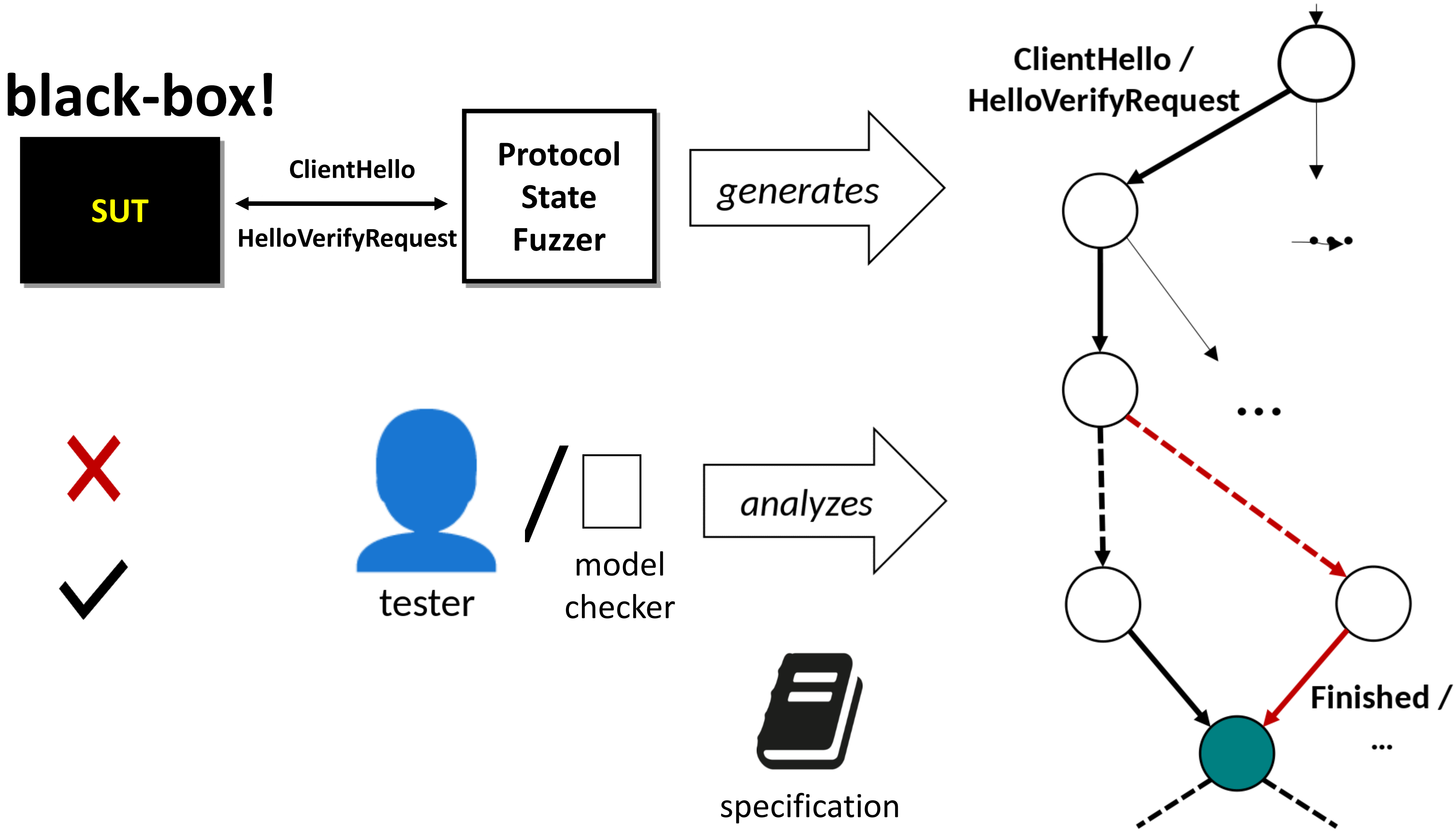


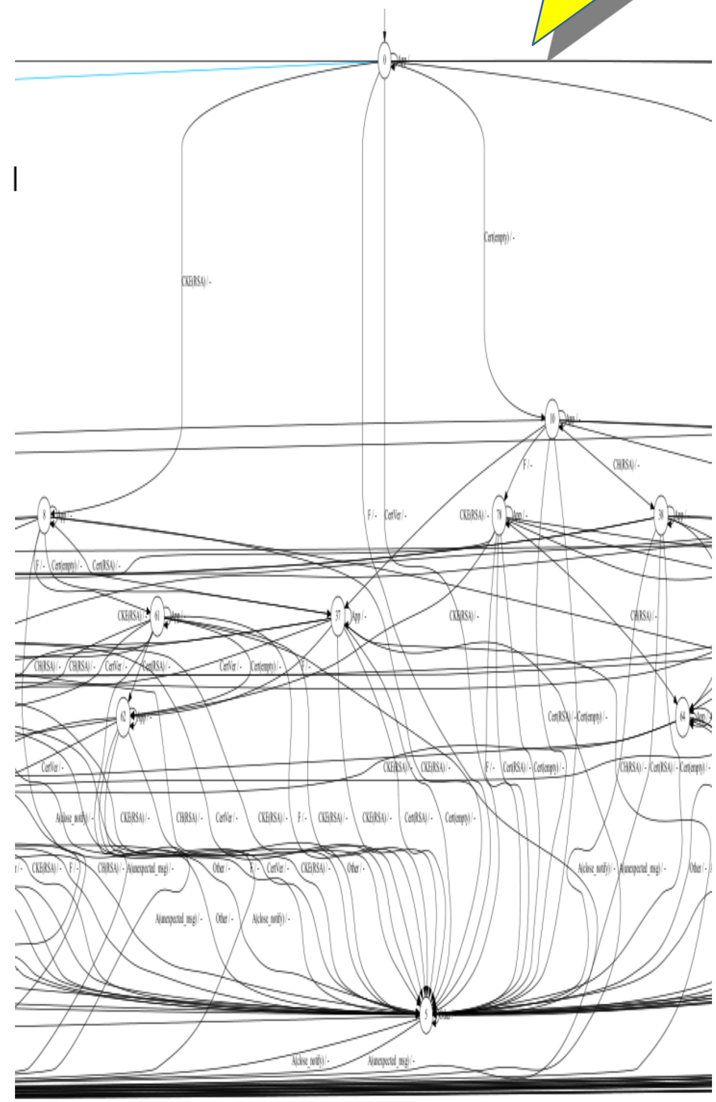
Fig. 2: TLS/DTLS handshake. Messages unique to DTLS are colored blue, optional messages are in [square brackets] and encrypted messages are marked by an asterisk*. Inside parentheses, we show the abbreviations we will use.

Model Learning 🔨) infers state machine **automatically**



Model Learning infers state machine **automatically**

JSSE Server
- 124 states!
- approximate!



black-box!

SUT ⟷ (ClientHello / HelloVerifyRequest) ⟷ Protocol State Fuzzer

generates

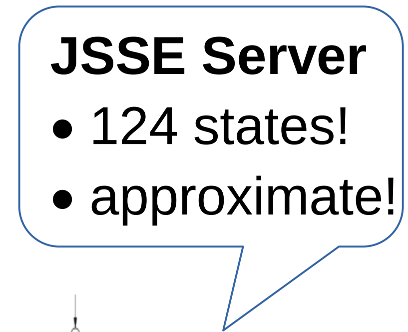
tester

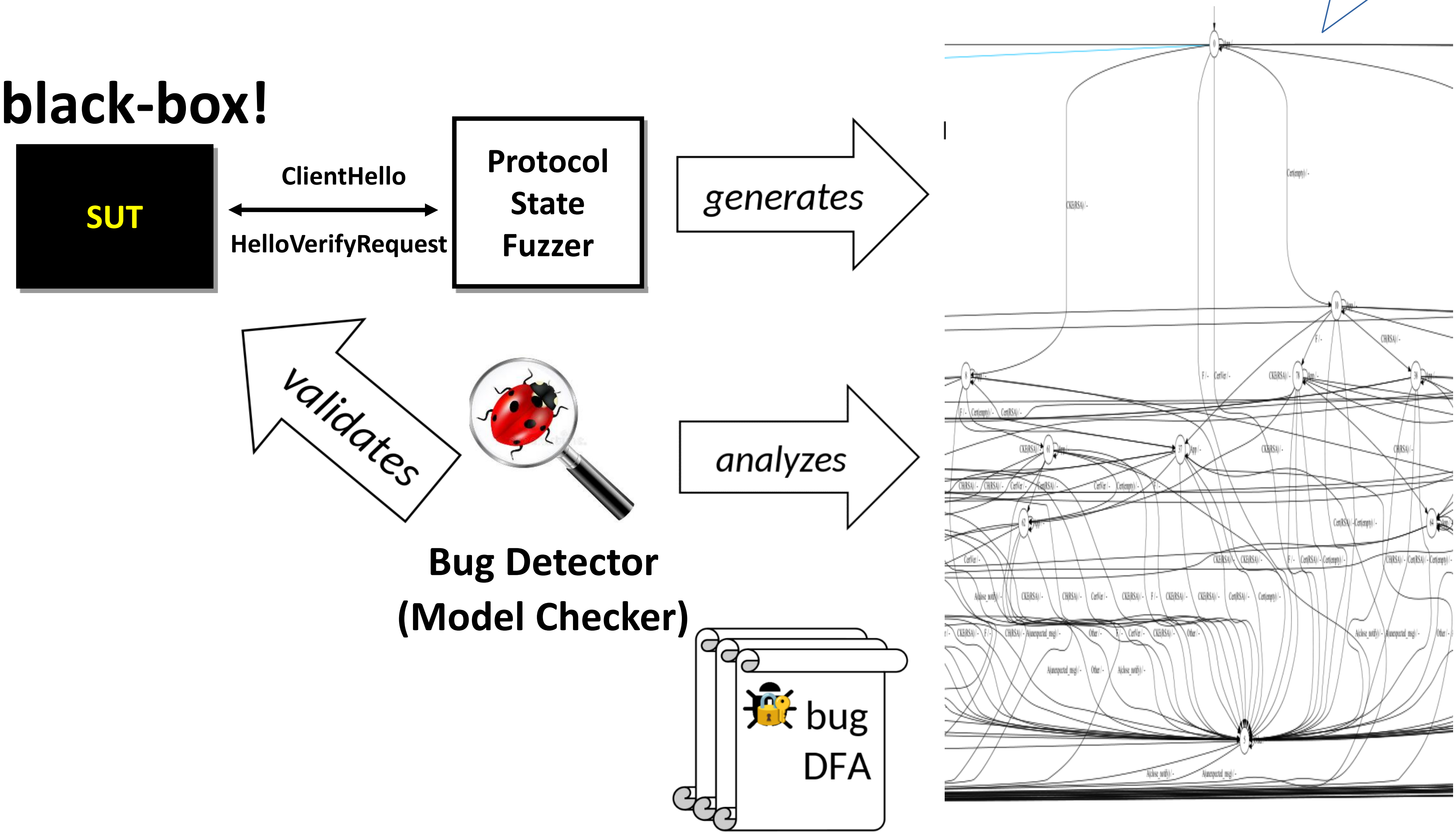
analyzes

specification

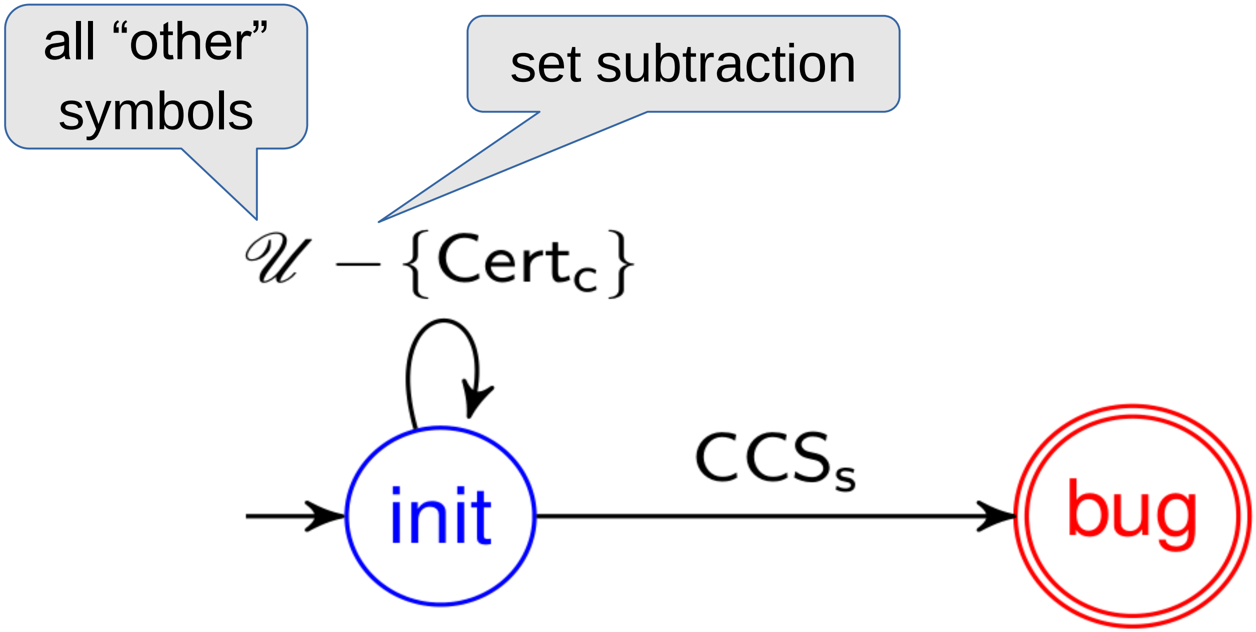
Manual Analysis
- Is time-consuming
- Requires expertise
- Can miss bugs

Automatic Analysis



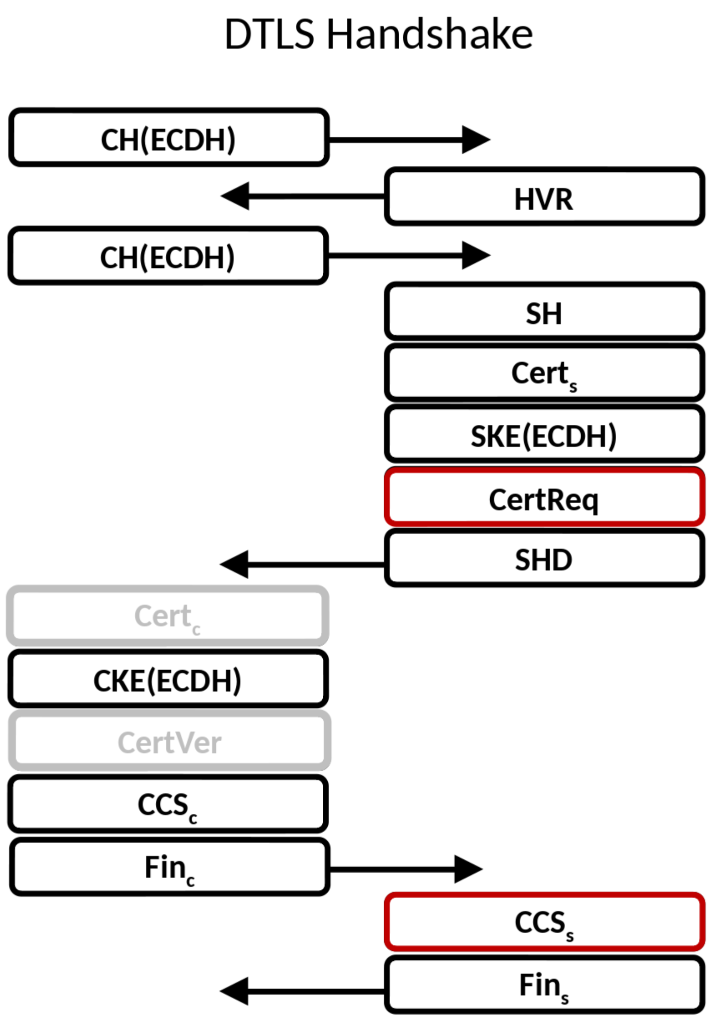


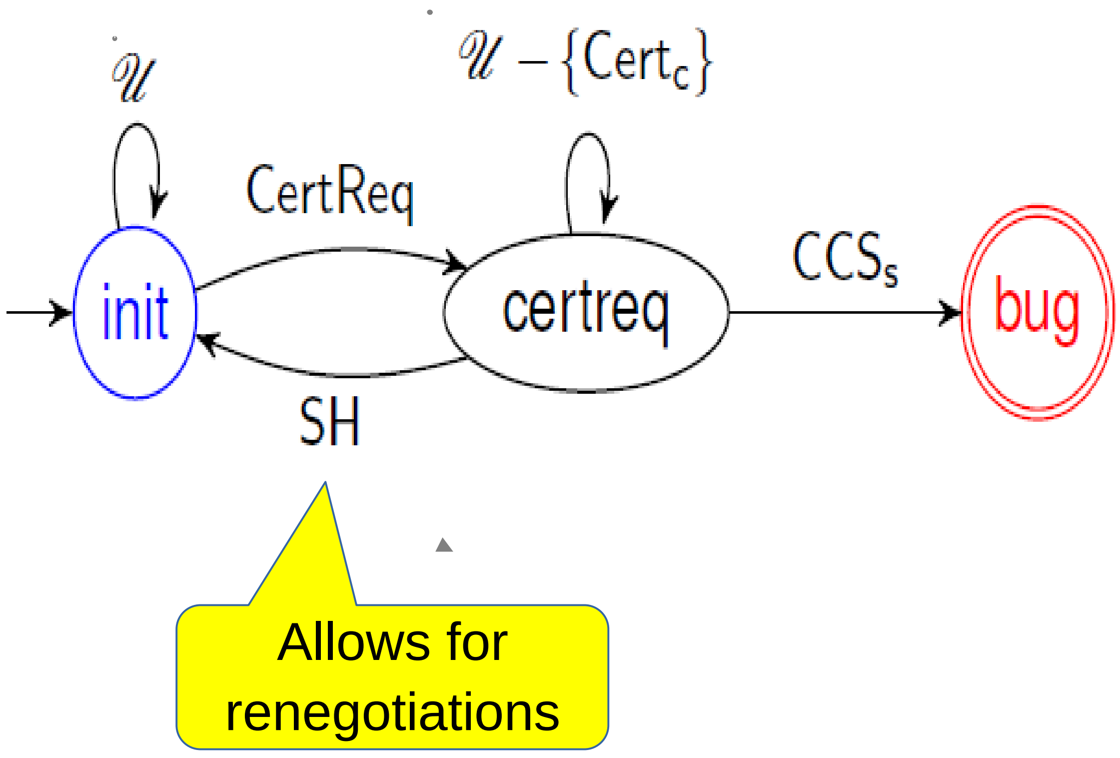
Bug Patterns Encoded as DFAs



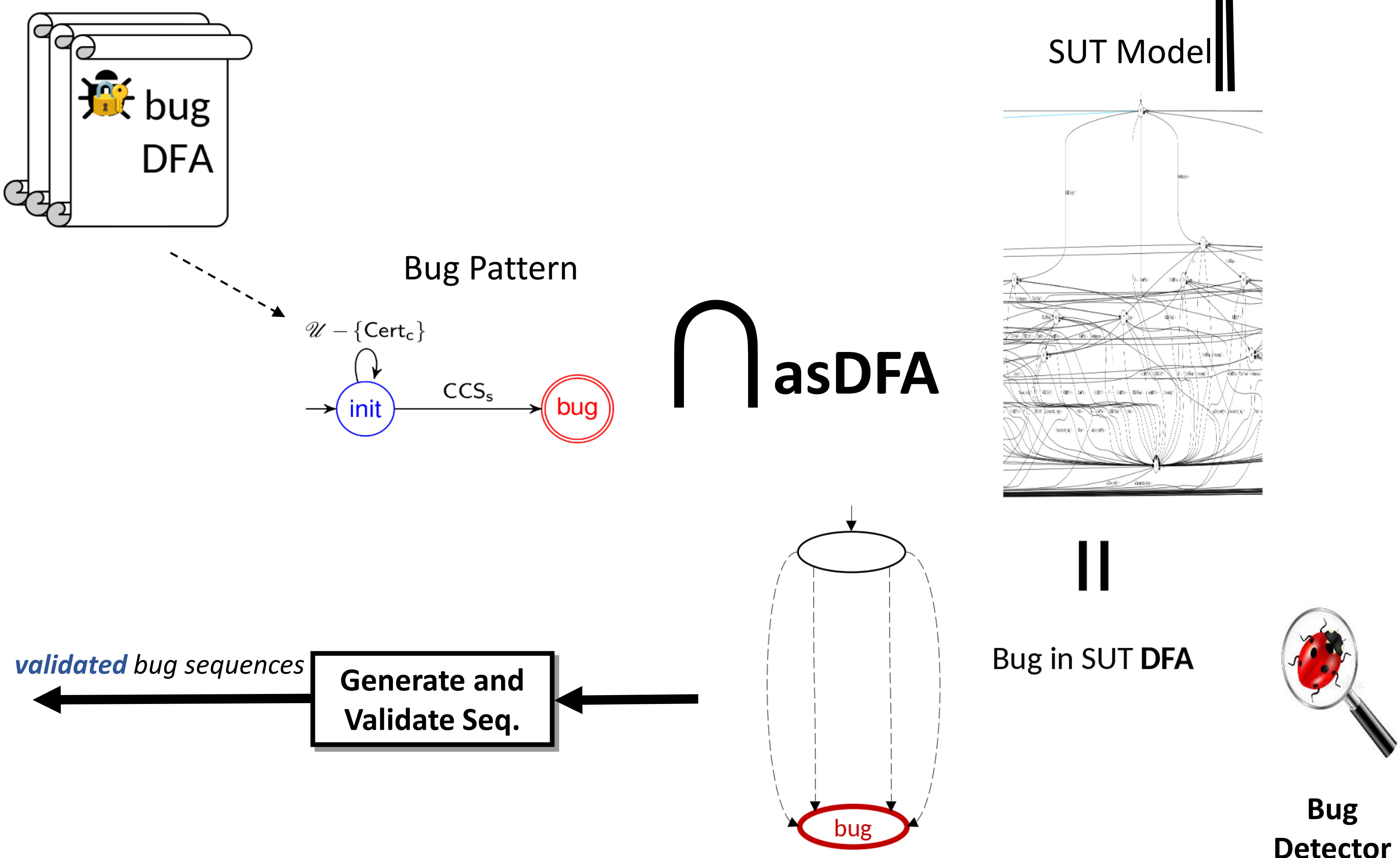
captures sequences without
client **Certificate** ending in a server **ChangeCipherSpec**

The Missing $Cert_c$ Bug Pattern

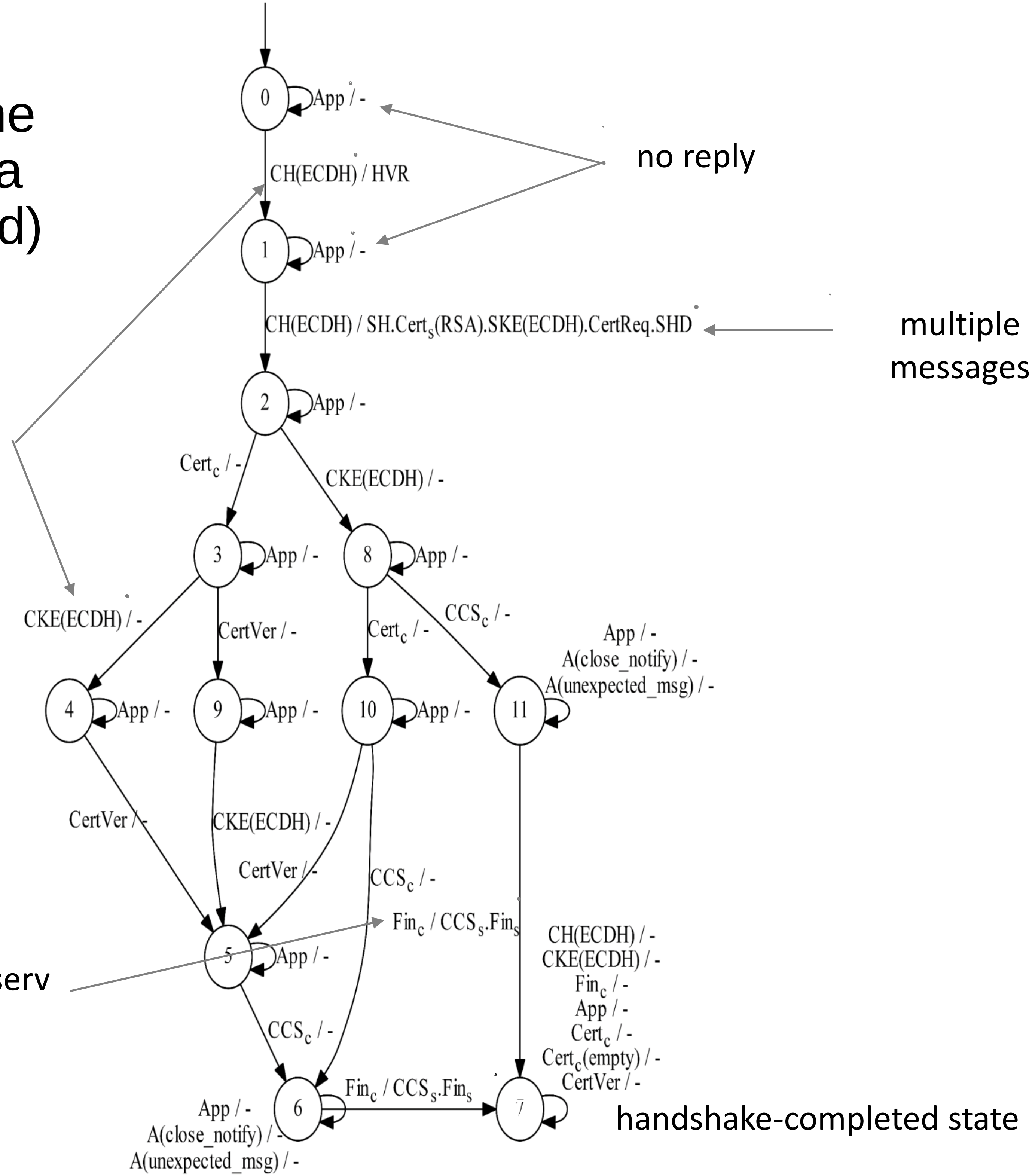




Technique in a Nutshell

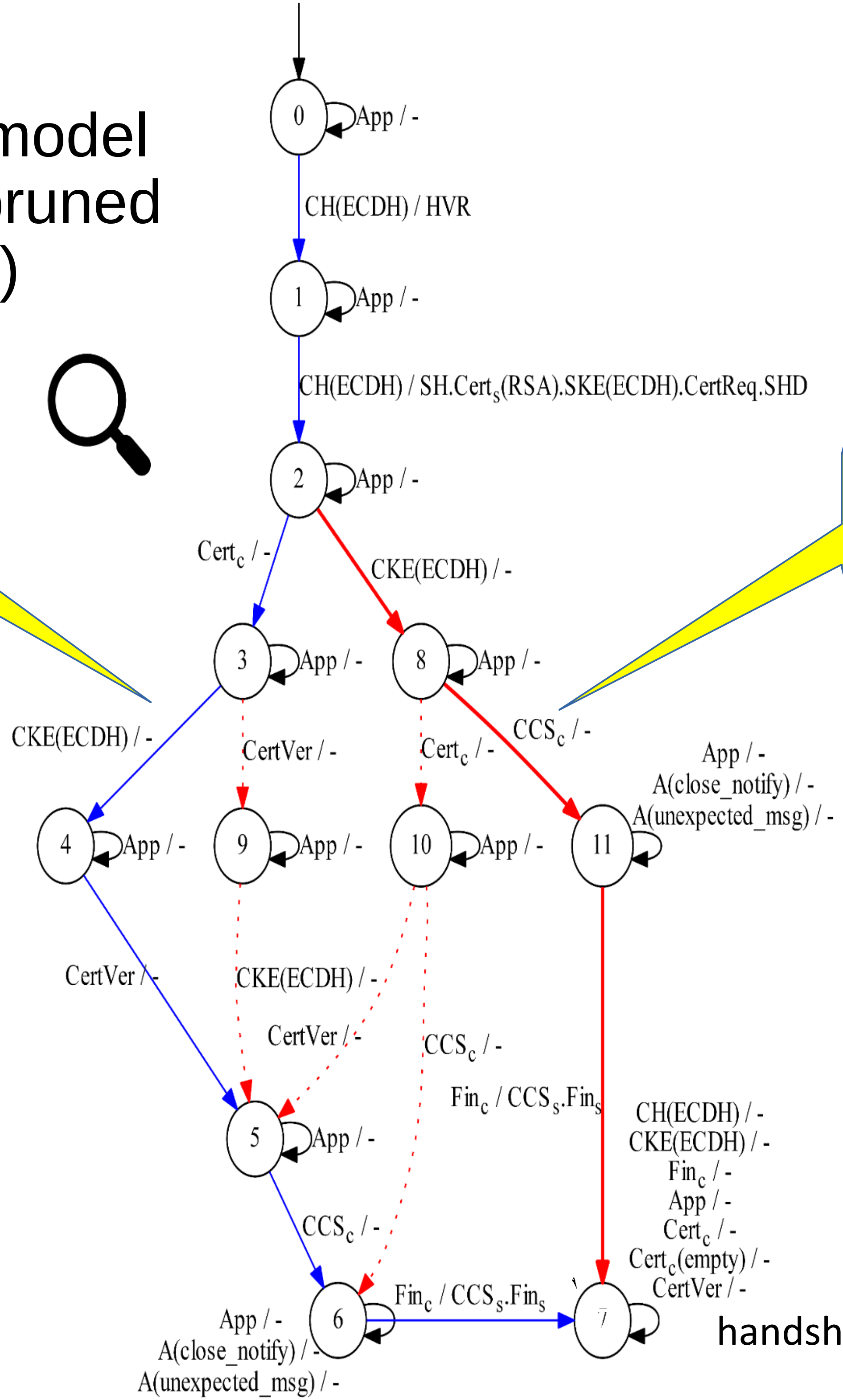


Mealy machine model of Java Server (pruned)



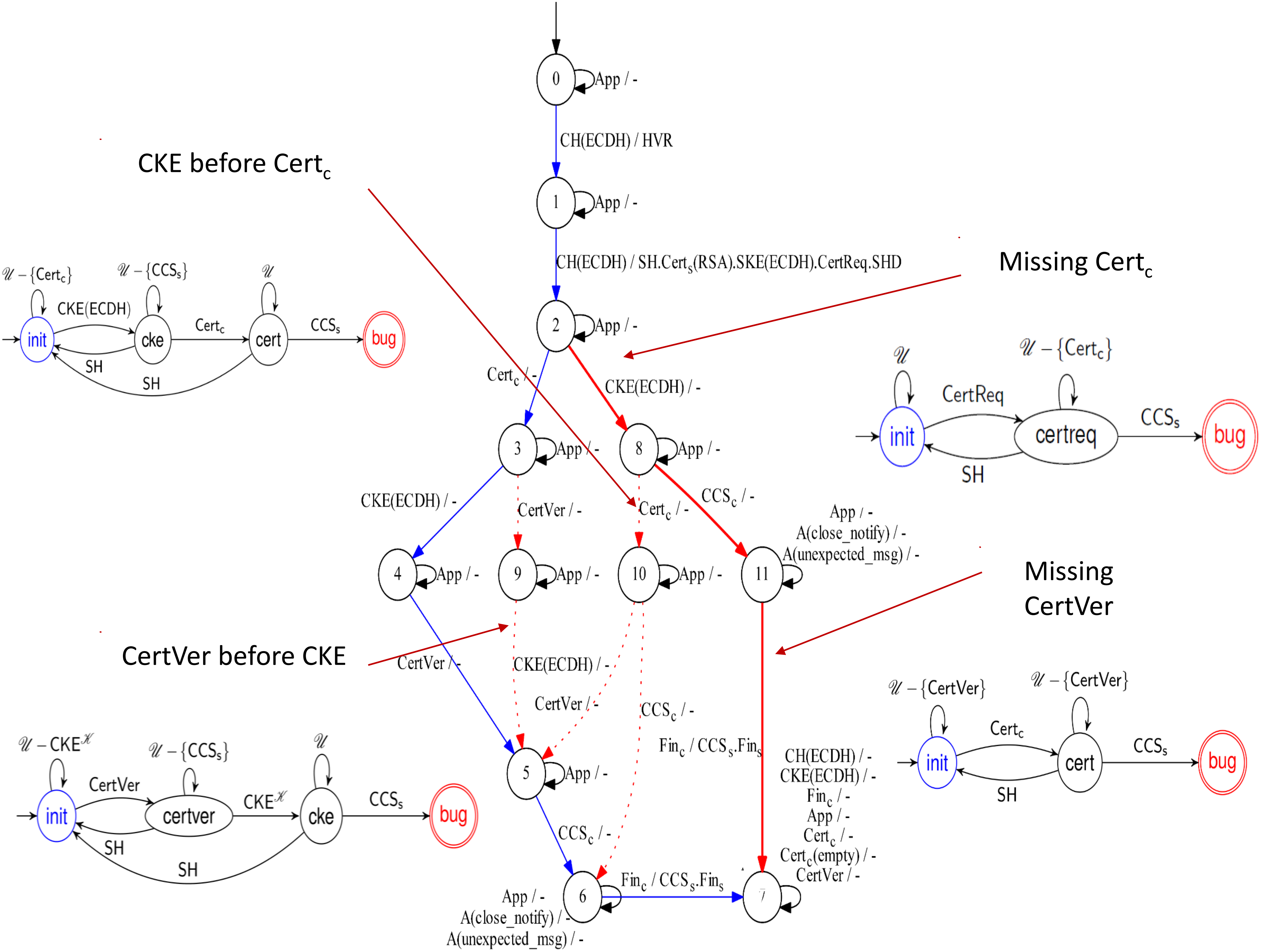
Mealy machine model of Java Server (pruned and colored)

Valid handshake path in **blue**



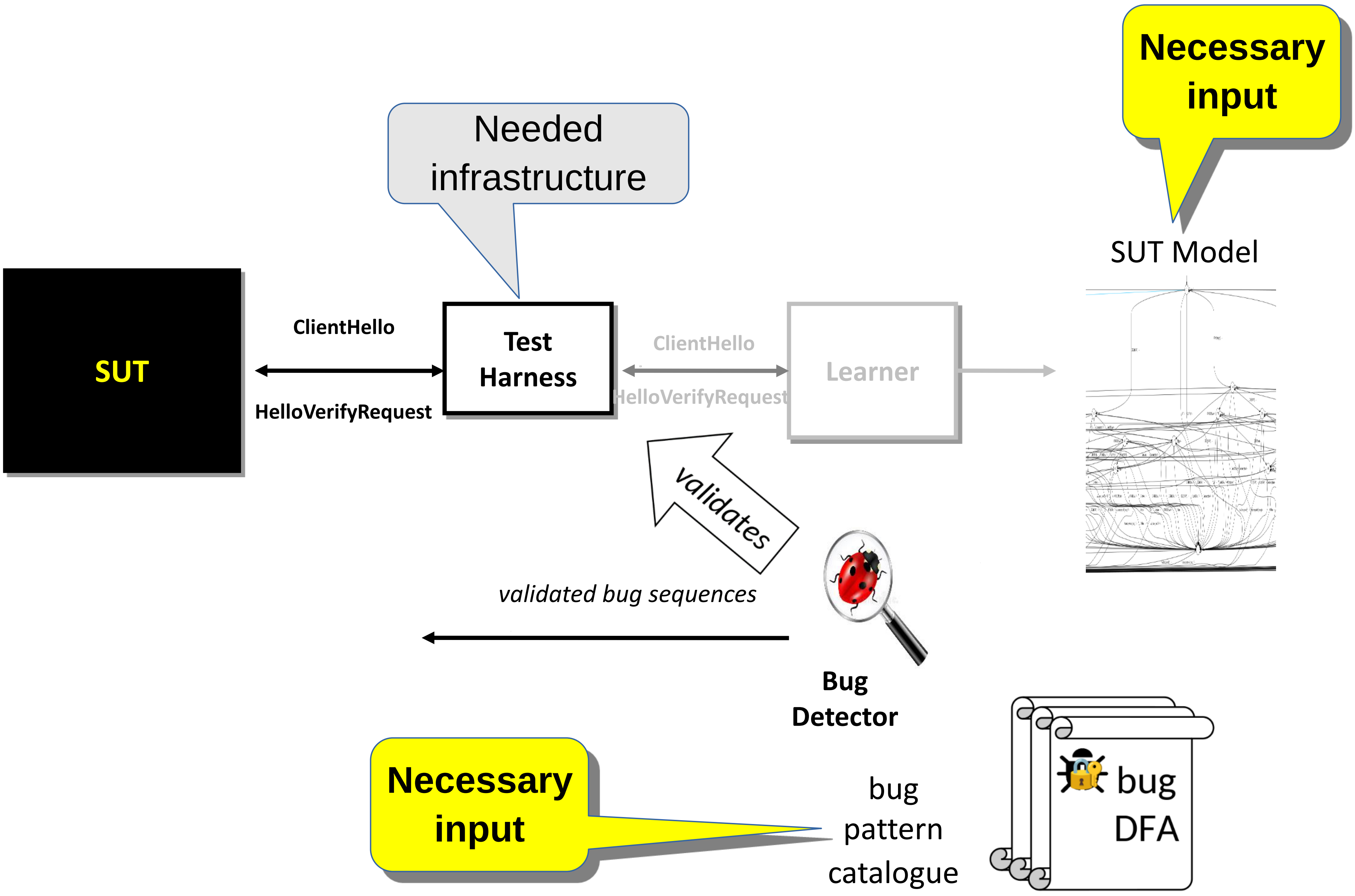
Vulnerabilities in **red**

handshake-completed state



CKE before Cert$_c$
Missing Cert$_c$
CertVer before CKE
Missing CertVer
App / -
CH(ECDH) / HVR
CH(ECDH) / SH.Cert$_s$(RSA).SKE(ECDH).CertReq.SHD
Cert$_c$ / -
CKE(ECDH) / -
CertVer / -
CCS$_c$ / -
A(close_notify) / -
A(unexpected_msg) / -
Fin$_c$ / CCS$_s$.Fin$_s$
Cert$_c$(empty) / -
init
cke
cert
bug
certreq
certver
CertReq
SH
CCS$_s$
Cert$_c$
CertVer
CKE$^{\mathcal{K}}$
$\mathcal{U} - \{Cert_c\}$
$\mathcal{U} - \{CCS_s\}$
$\mathcal{U}$
$\mathcal{U} - \{CertVer\}$
$\mathcal{U} - CKE^{\mathcal{K}}$

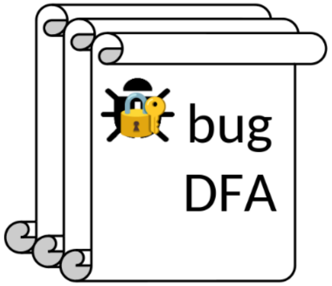
Bug Detection Framework

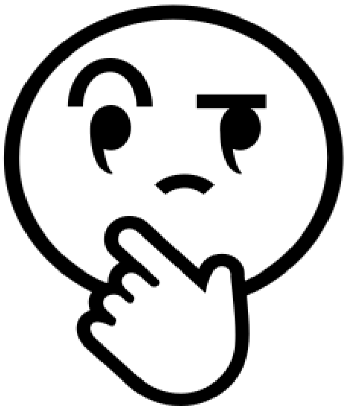


Assembling a Bug Catalogue

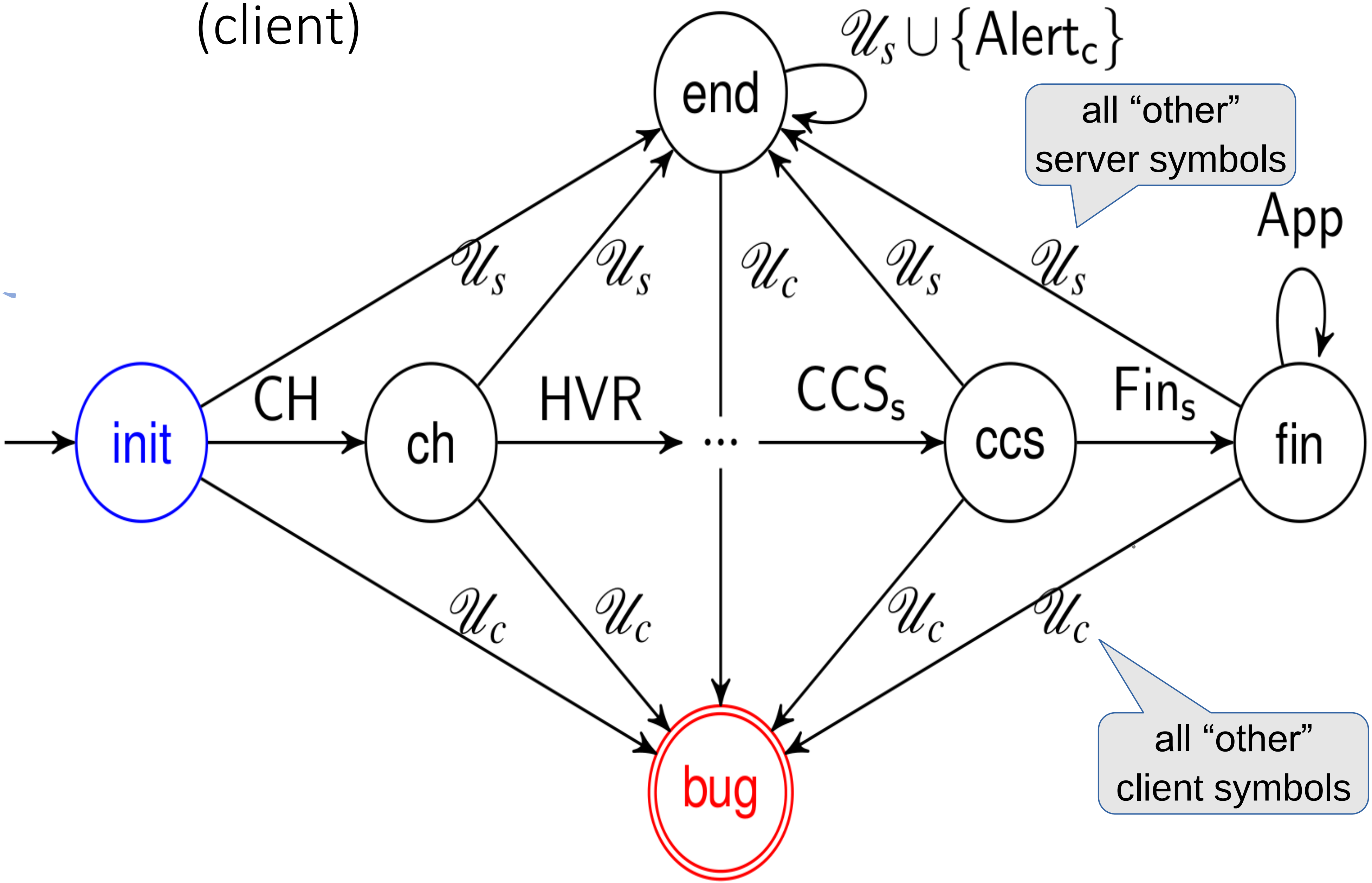
Missing Cert_c

CVE-2020-2655





Invalid Handshake (client)



Evaluation Setup

- DTLS servers (18 bug patterns)
- DTLS clients (16 bug patterns)
- SSH servers (16 bug patterns)
- 24 **new** bug patterns, largely owing to three general bug patterns
- Specific bug patterns are small (all less than 10 nodes)
- Models generated for all SUTs (two day learning time bound)
- Test harness/learning tools: DTLS-Fuzzer[1] and SSH-Mapper[2]
- SUTs: test programs for nine DTLS and three SSH libraries, new versions and (for DTLS) versions used in prior work.

[1]: P. Fiterău-Broştean, B. Jonsson, K. Sagonas and F. Tåquist, "DTLS-Fuzzer: A DTLS Protocol State Fuzzer," IEEE Conference on Software Testing, Verification and Validation (ICST 2022), pp. 456-458, doi: 10.1109/ICST53961.2022.00051.

[2]: https://hdl.handle.net/2066/184275

Evaluation Results (Nutshell)

- Detected and validated automatically **all** bugs found in prior work on DTLS [1] and SSH [2].
- Detected new bugs (incl. those prior work **missed**) and new **vulnerabilities** in Java clients.
- All but one bug were validated successfully.
- All bugs were reported to developers
  → fixes in **four** libraries.

[1] Fiterau-Brostean, P., Jonsson, B., Merget, R., De Ruiter, J., Sagonas, K., and Somorovsky, J. (2020). Analysis of DTLS implementations using protocol state fuzzing. In 29th USENIX Security Symposium (USENIX Security 20) (pp. 2523-2540).

[2] Fiterău-Broştean, P., Lenaerts, T., Poll, E., de Ruiter, J., Vaandrager, F., and Verleg, P. (2017). Model learning and model checking of SSH implementations. In Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software (pp. 142-151).

Results on DTLS Servers

VULNERABILITIES (✔ red), KNOWN BUGS (✔ grey), AND NEW BUGS (✔ blue) DETECTED IN VARIOUS DTLS SERVER IMPLEMENTATIONS.

	GnuTLS			JSSE		MbedTLS		OpenSSL		PionDTLS		Scandium		TinyDTLS^C	TinyDTLS^E	WolfSSL	
Bug Pattern	3.5.19	3.6.7	3.7.1	12.0.2	16.0.1	2.16.1	2.26.0	1.1.1b	1.1.1k	e4481fc	2.0.9	2.0.0-M16	2.6.2	53a0d97	8414f8a	4.0.0	4.7.1r
Certificate-less Client Authentication	-	-	-	✔ (red)	-	-	-	-	-	-	-	-	-	-	-	-	-
CertificateVerify-less Client Authentication	-	-	-	✔ (red)	-	-	-	-	-	-	-	-	-	-	-	-	-
ChangeCipherSpec before CertificateVerify	-	-	-	✔ (grey)	-	-	-	-	-	✔ (grey)	-	-	-	-	-	-	-
ClientKeyExchange before Certificate	-	-	-	✔ (grey)	-	-	-	-	-	-	-	-	-	-	-	-	-
🔧 Continue After Closure Alert	-	-	-	✔ (blue)	-	-	-	✔ (blue)	✔ (blue)	-	-	-	-	-	-	-	-
🔧 Continue After Fatal Error Alert	-	-	-	✔ (blue)	-	-	-	✔ (blue)	✔ (blue)	-	-	-	-	-	-	-	-
Early Finished	-	-	-	-	-	-	-	-	-	✔ (red)	-	✔ (red)	-	-	-	-	-
Finished before ChangeCipherSpec	-	-	-	-	-	-	-	-	-	✔ (grey)	-	-	-	-	-	-	-
HelloVerifyRequest Retransmission	-	-	-	-	-	-	-	-	-	✔ (grey)	-	-	-	-	-	-	-
Insecure Renegotiation	-	-	-	-	-	-	-	-	-	-	-	-	-	✔ (grey)	✔ (grey)	-	-
Internal Error on Finished	-	-	-	-	-	-	-	✔ (grey)	✔ (blue)	-	-	✔ (blue)	✔ (blue)	-	-	-	-
Invalid DecryptError Alert	-	-	-	-	-	-	-	-	-	-	-	-	-	✔ (grey)	✔ (grey)	-	-
Invalid Finished as Retransmission	-	-	-	-	-	-	-	✔ (grey)	✔ (blue)	-	-	-	-	-	-	-	-
Invalid HelloVerifyRequest	✔ (grey)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
🔧 Multiple Certificate	-	-	-	✔ (blue)	-	-	-	-	-	-	-	✖ (blue)	-	-	-	-	-
Multiple ChangeCipherSpec	-	-	-	✔ (blue)	-	-	-	-	-	-	-	✔ (grey)	-	-	-	-	-
Non-conforming Cookie	✔ (grey)	✔ (grey)	✔ (blue)	-	-	✔ (grey)	✔ (blue)	✔ (grey)	✔ (blue)	-	-	-	-	-	-	-	-
Unauthenticated ClientKeyExchange	-	-	-	✔ (grey)	-	-	-	-	-	-	-	-	-	-	-	-	-

🔧 Bug patterns found with our systematic approach.

Blue bug patterns are for new (types of) bugs.

Results on DTLS Clients

		GnuTLS		JSSE		MbedTLS		OpenSSL		PionDTLS		Scandium		TinyDTLS^C	TinyDTLS^E	WolfSSL	
	Bug Pattern	3.6.7	3.7.1	12.0.2	16.0.1	2.16.1	2.26.0	1.1.1b	1.1.1k	e4481fc	2.0.9	2.0.0-M16	2.6.2	53a0d97	8414f8a	4.0.0	4.7.1r
	CertificateRequest before Certificate	-	-	✓	✓	-	-	-	-	✓	-	✓	-	-	-	-	-
🔧	Continue After Closure Alert	✓	✓	✓	✓	-	-	✓	✓	-	-	-	-	-	-	-	-
🔧	Continue After Fatal Error Alert	✓	✓	✓	✓	-	-	✓	✓	✓	-	-	-	-	-	-	-
🔧	Early Finished	-	-	-	-	-	-	-	-	✓	-	✓	-	-	-	-	-
🔧	Finished Before ChangeCipherSpec	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-
🔧	Invalid DecryptError Alert	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-
🔧	Multiple Certificate	-	-	✓	✓	-	-	✓	✓	✓	-	✓	-	-	-	-	-
🔧	Multiple CertificateRequest	-	-	✓	✓	-	-	✓	✓	✓	-	✓	-	✓	✓	-	-
🔧	Multiple ChangeCipherSpec	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-
🔧	Multiple ServerKeyExchange	-	-	✓	✓	-	-	✓	✓	✓	-	✓	-	-	-	-	-
🔧	Premature HelloRequest	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-
🔧	ServerHello Flight Restart	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-
🔧	Switching Cipher Suite	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-
	Unexpected ClientHello	-	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	✓	✓
🔧	Unrequested Certificate	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-
🔧	Wrong Certificate Type	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-	-

🔧 Bug patterns found with our systematic approach.

Blue bug patterns are for new (types of) bugs.

Quantitative Measurements for DTLS Client Experiments

Our technique:

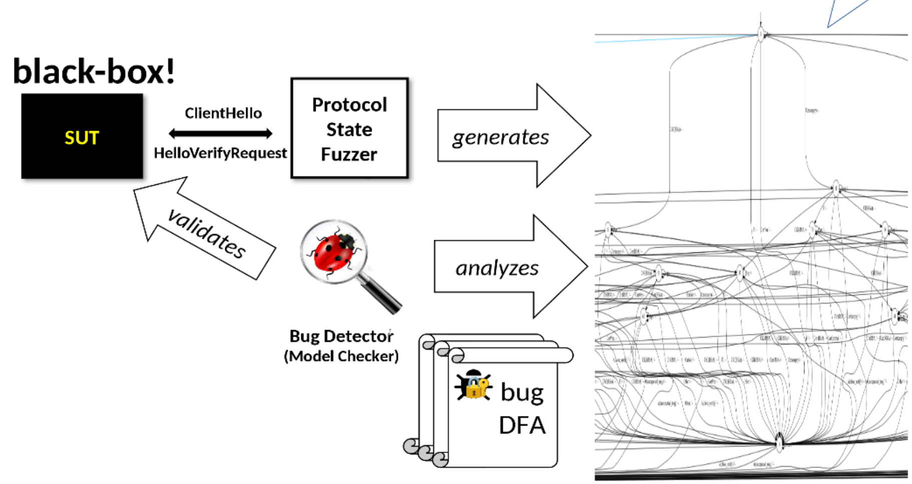
- Handles large models (e.g., 387 states for OpenSSL)
- Given reasonably accurate models (after two days)
  - Successfully validates all bugs (one test/bug)
  - Finishes in under one minute (even when 10 bugs are detected)
- Works reasonably well even with inaccurate models
  - Detects fewer bugs
  - Needs more time/tests to validate them

Read the Paper for

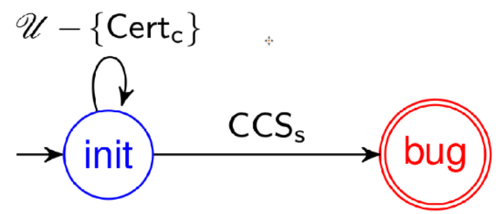
- More tables and experiments.
- How to systematically assemble a bug pattern catalogue.
- Application and evaluation on SSH Servers.
- Related work.
- Information about the paper's artifact.

In Summary

Automatic Analysis

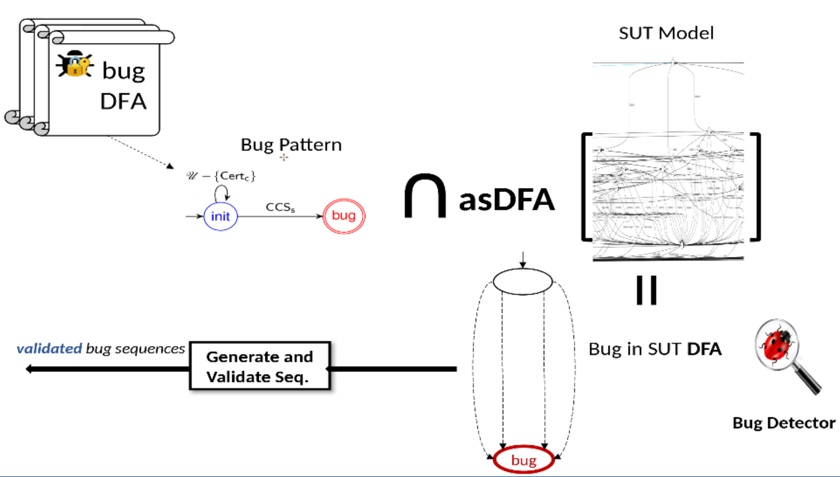


Bug Patterns Encoded as DFAs



captures sequences without
client **Certificate** ending in a server **ChangeCipherSpec**

Our Technique in a Nutshell



Results on DTLS Servers

VULNERABILITIES (✓ red), KNOWN BUGS (✓ gray), AND NEW BUGS (✓ blue) DETECTED IN VARIOUS DTLS SERVER IMPLEMENTATIONS.

	GnuTLS			JSSE		MbedTLS		OpenSSL		PionDTLS		Scandium		TinyDTLS^C	TinyDTLS^E	WolfSSL	
Bug Pattern	3.5.19	3.6.7	3.7.1	12.0.2	16.0.1	2.16.1	2.26.0	1.1.1b	1.1.1k	e4481fc	2.0.9	2.0.0-M16	2.6.2	53a0d97	8414f8a	4.0.0	4.7.1r
Certificate-less Client Authentication	-	-	-	✓ (red)	-	-	-	-	-	-	-	-	-	-	-	-	-
CertificateVerify-less Client Authentication	-	-	-	✓ (red)	-	-	-	-	-	-	-	-	-	-	-	-	-
ChangeCipherSpec before CertificateVerify	-	-	-	✓ (gray)	-	-	-	-	-	✓ (gray)	-	-	-	-	-	-	-
ClientKeyExchange before Certificate	-	-	-	✓ (gray)	-	-	-	-	-	-	-	-	-	-	-	-	-
🔧 Continue After Closure Alert	-	-	-	✓ (blue)	-	-	-	✓ (blue)	✓ (blue)	-	-	-	-	-	-	-	-
🔧 Continue After Fatal Error Alert	-	-	-	✓ (blue)	-	-	-	✓ (blue)	✓ (blue)	-	-	-	-	-	-	-	-
Early Finished	-	-	-	-	-	-	-	-	-	✓ (red)	-	✓ (red)	-	-	-	-	-
Finished before ChangeCipherSpec	-	-	-	-	-	-	-	-	-	✓ (gray)	-	-	-	-	-	-	-
HelloVerifyRequest Retransmission	-	-	-	-	-	-	-	-	-	✓ (gray)	-	-	-	-	-	-	-
Insecure Renegotiation	-	-	-	-	-	-	-	-	-	-	-	-	-	✓ (gray)	✓ (gray)	-	-
Internal Error on Finished	-	-	-	-	-	-	-	✓ (gray)	✓ (blue)	-	-	✓ (blue)	✓ (blue)	-	-	-	-
Invalid DecryptError Alert	-	-	-	-	-	-	-	-	-	-	-	-	-	✓ (gray)	✓ (gray)	-	-
Invalid Finished as Retransmission	-	-	-	-	-	-	-	✓ (gray)	✓ (blue)	-	-	-	-	-	-	-	-
Invalid HelloVerifyRequest	✓ (gray)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
🔧 Multiple Certificate	-	-	-	✓ (blue)	-	-	-	-	-	-	-	✖	-	-	-	-	-
Multiple ChangeCipherSpec	-	-	-	✓ (blue)	-	-	-	-	-	-	-	✓ (gray)	-	-	-	-	-
Non-conforming Cookie	✓ (gray)	✓ (gray)	✓ (blue)	-	-	✓ (gray)	✓ (blue)	✓ (gray)	✓ (blue)	-	-	-	-	-	-	-	-
Unauthenticated ClientKeyExchange	-	-	-	✓ (gray)	-	-	-	-	-	-	-	-	-	-	-	-	-

🔧 Bug patterns found with our systematic approach.
Blue bug patterns are for new (types of) bugs.