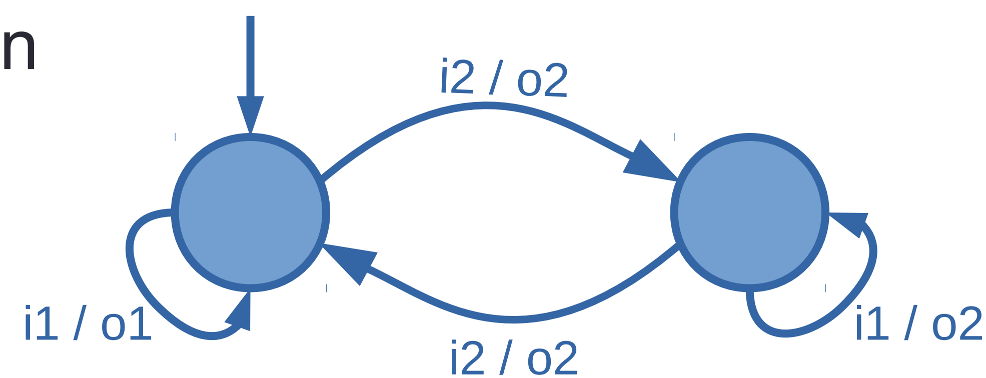
PROTOCOL STATE FUZZING OF TLS IMPLEMENTATIONS

Joeri de Ruiter

Radboud University Nijmegen

State machines

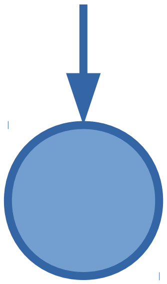
- Every application that implements a protocol has to implement the corresponding state machine
- Mealy machines as formal representation
  - Set of states
  - Input alphabet
  - Output alphabet
- Specify in all states for each input
  - Returned output
  - Next state
- Unambiguous representation



State machine inference

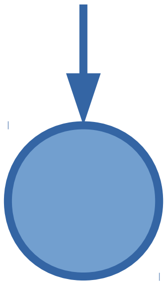
- Extract state machines from implementations by communicating with them
- Fuzzing of message order
- Security analysis
  - Discover bugs
  - Provides interesting insights in the code
  - Will not find carefully hidden backdoors
- Manual analysis by looking for unexpected transitions and other strange behaviour

State machine inference



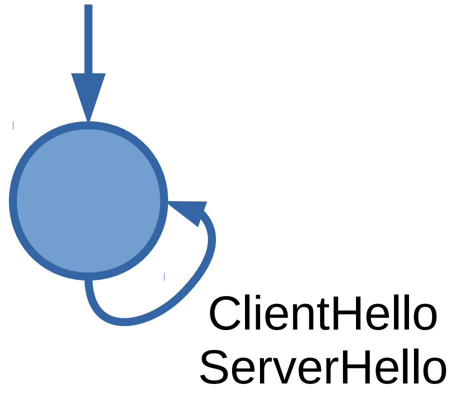
State machine inference

→ ClientHello
← ServerHello



State machine inference

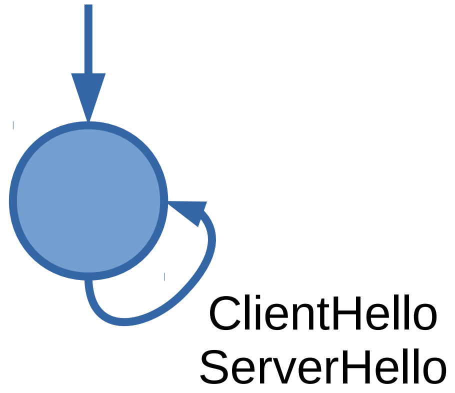
→ ClientHello
← ServerHello



State machine inference

→ ClientHello
← ServerHello

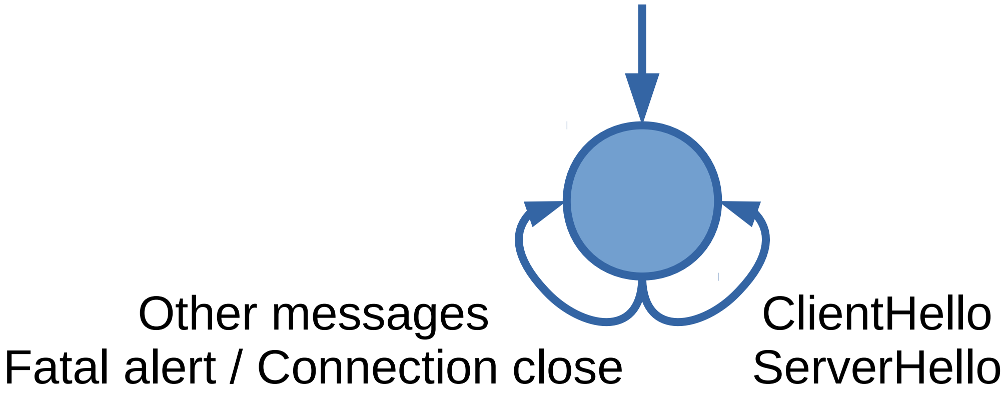
→ Other messages
← Fatal alert / Connection close



State machine inference

→ ClientHello
← ServerHello

→ Other messages
← Fatal alert / Connection close

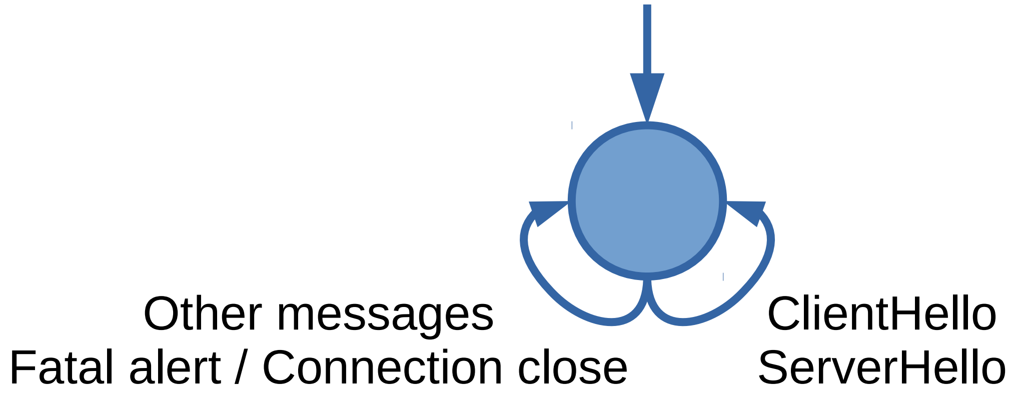


State machine inference

→ ClientHello
← ServerHello

→ Other messages
← Fatal alert / Connection close

→ ClientHello, ClientHello
← Fatal alert / Connection close

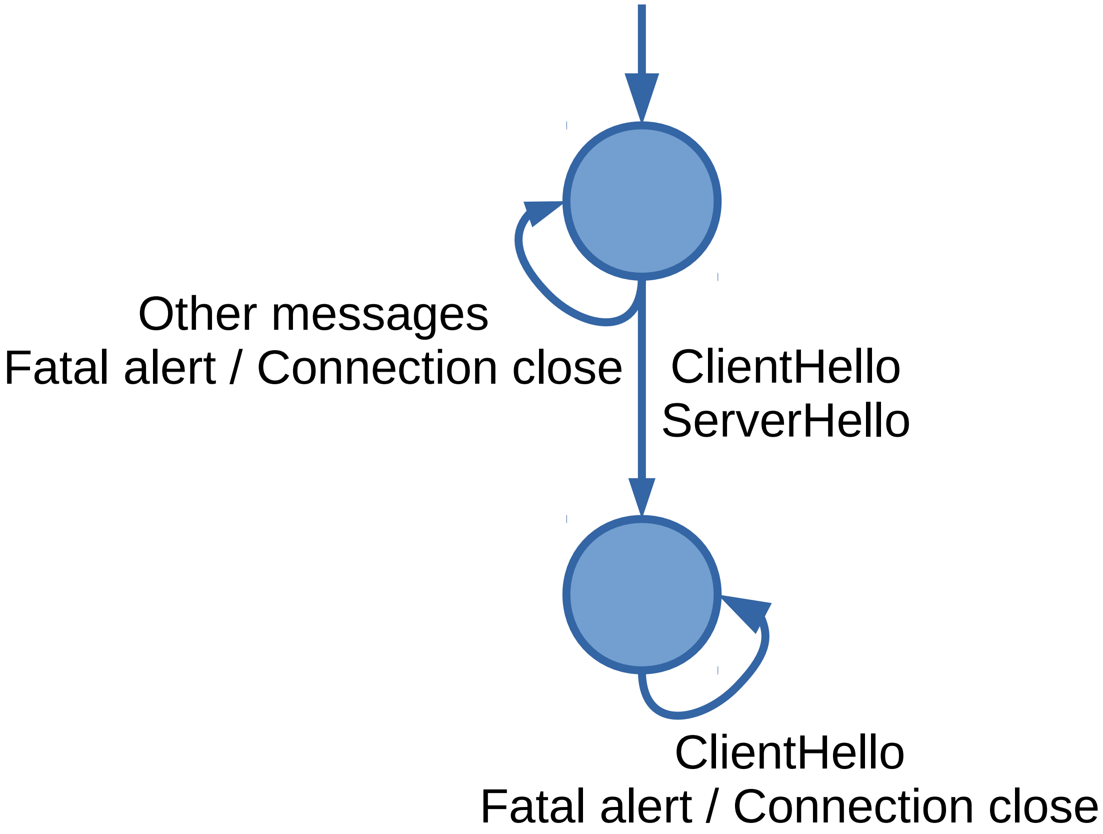


State machine inference

→ ClientHello
← ServerHello

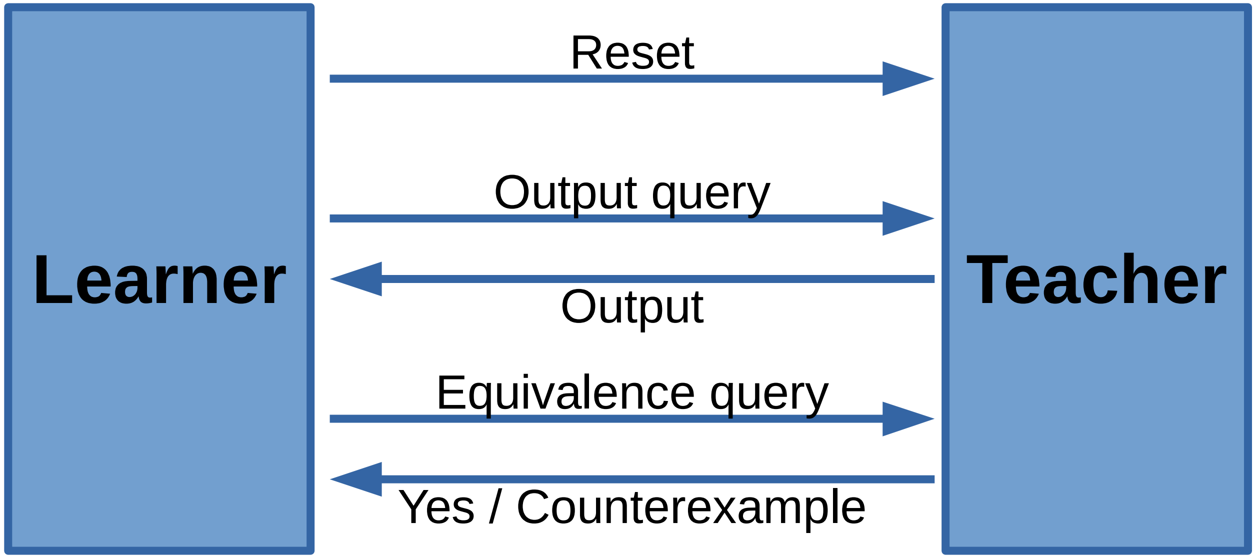
→ Other messages
← Fatal alert / Connection close

→ ClientHello, ClientHello
← Fatal alert / Connection close



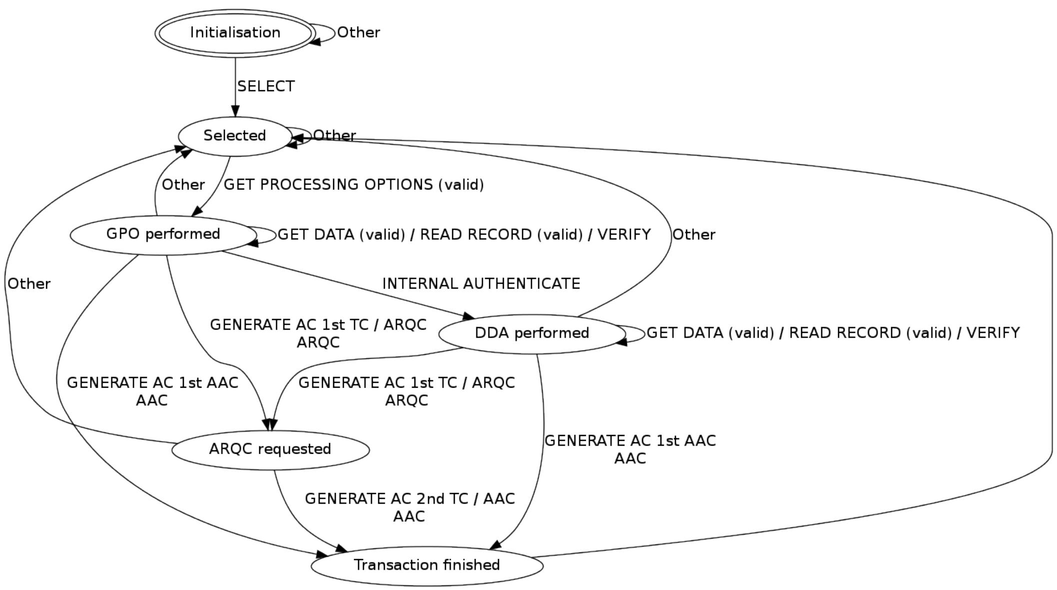
Automated learning

- Deterministic Mealy machine
- Learner
  - Adapted L* algorithm by Niese
- Teacher
  - Equivalence queries approximated
    - Random traces
    - Chow's W-method

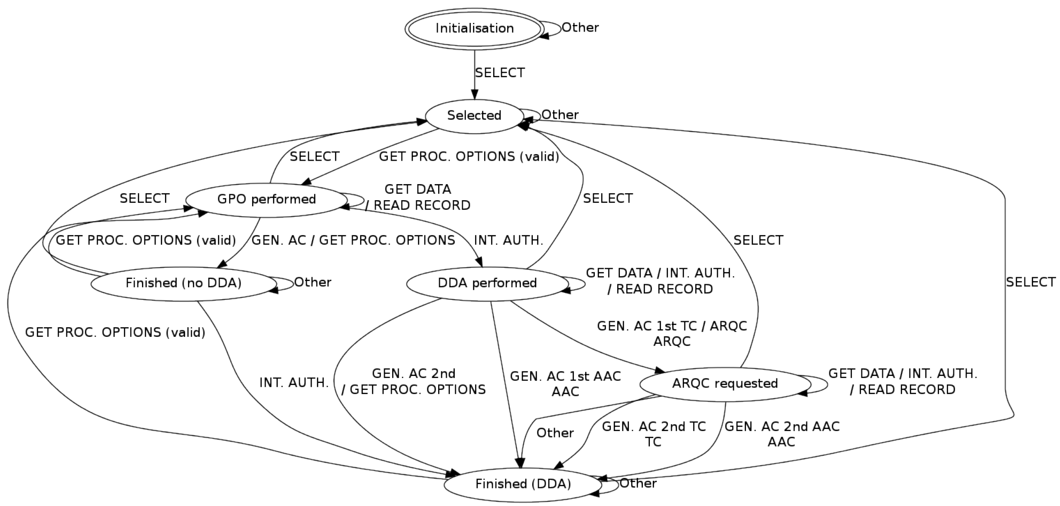


State machine inference for security

- Previously used on bankcards and the e.dentifier2



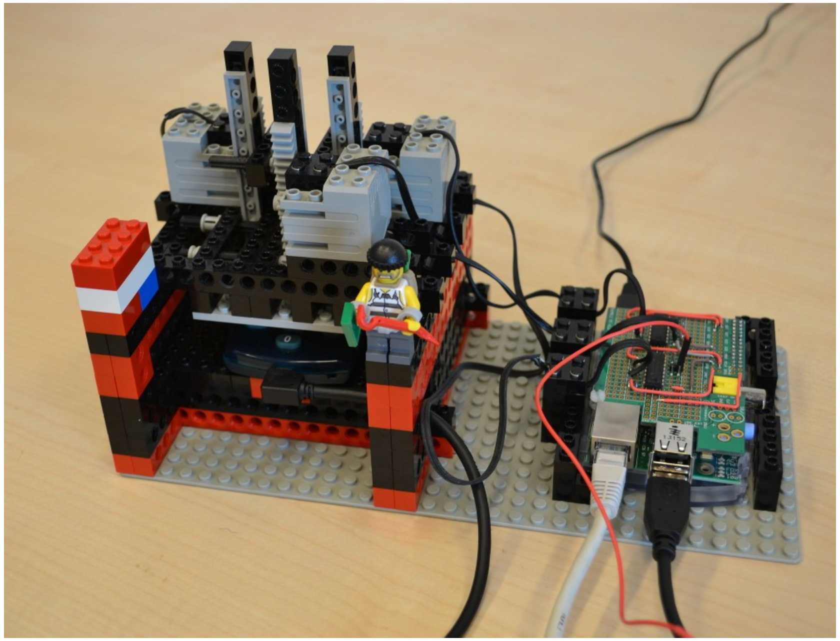
Rabobank Maestro

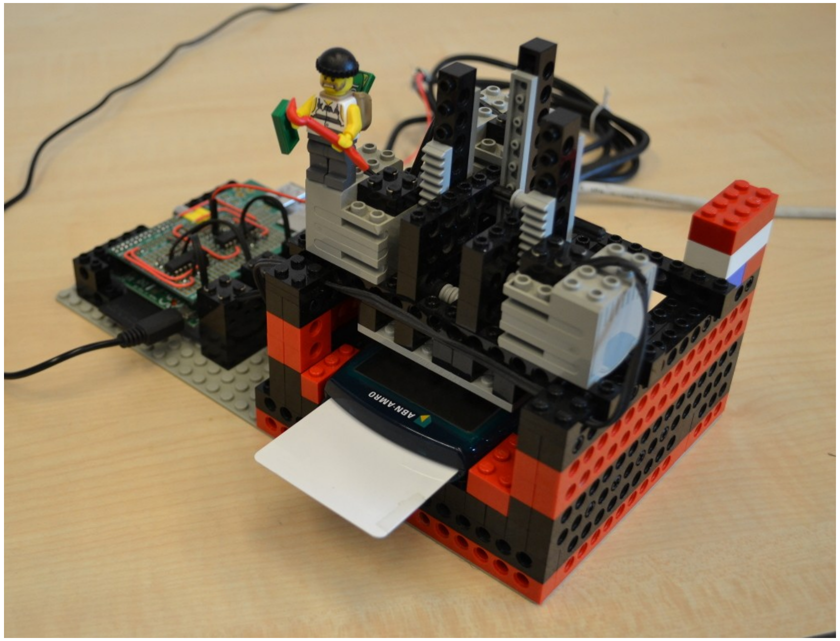


Volksbank Maestro

State machine inference for security

- Previously used on bankcards and the e.dentifier2





Learning of TLS implementations

- LearnLib by TU Dortmund
  - Implementation of adapted L* and equivalence algorithms
- Equivalence checking using modified W-method
  - Given an upper bound it is guaranteed to find the correct state machine
  - Depth specified to search for counter-examples
  - After a socket is closed no data will be received
- Custom test harness for TLS
- Manual analysis if we see unexpected behavior

Short introduction to TLS

Client		**Server**
ClientHello	⟶	
		ServerHello
		Certificate
	⟵	ServerHelloDone
ClientKeyExchange		
ChangeCipherSpec		
Finished	⟶	
		ChangeCipherSpec
	⟵	Finished
ApplicationData	⟷	ApplicationData

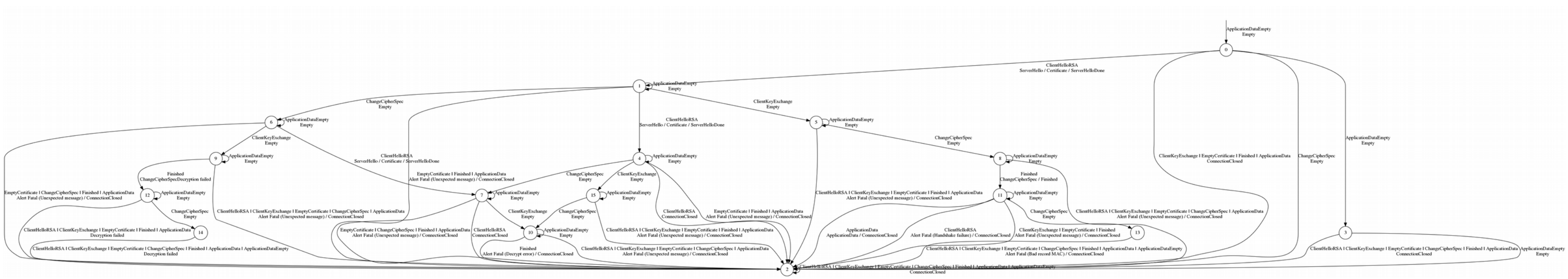
Test harness

- (Almost) stateless TLS implementation
- Minimal state in test harness to handle encryption
- Support to test clients and servers
- All regular TLS messages and Heartbeat extensions
  - RSA and DH key exchange
  - Client authentication
  - Some special symbols that correspond to exceptions in the test harness

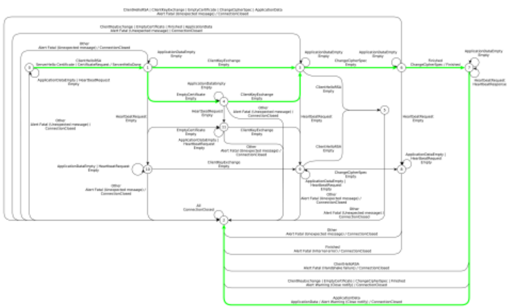
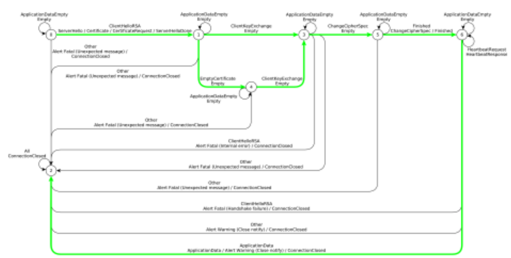
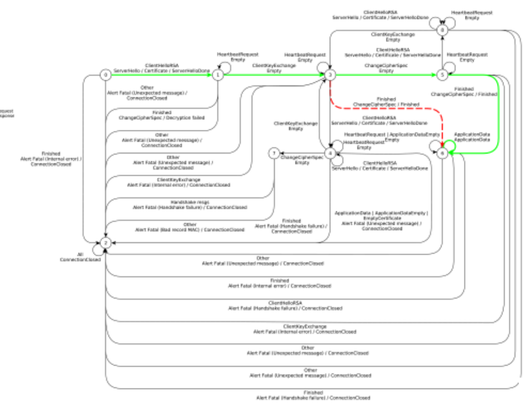
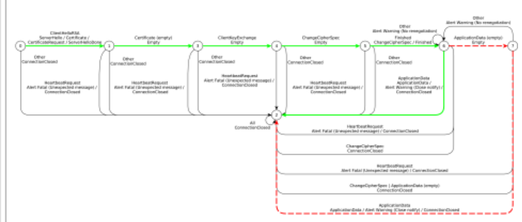
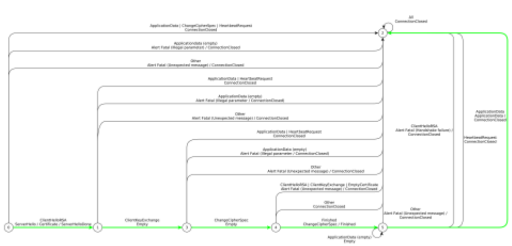
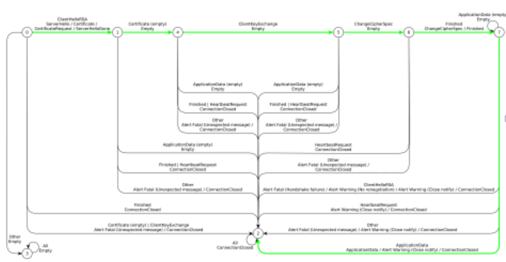
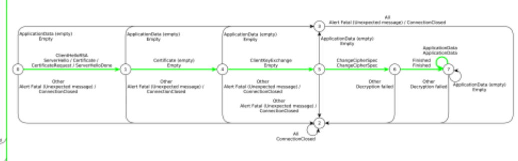
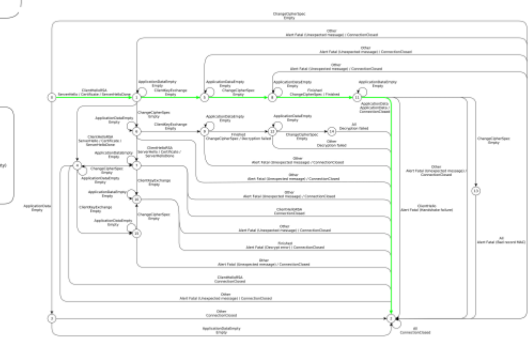
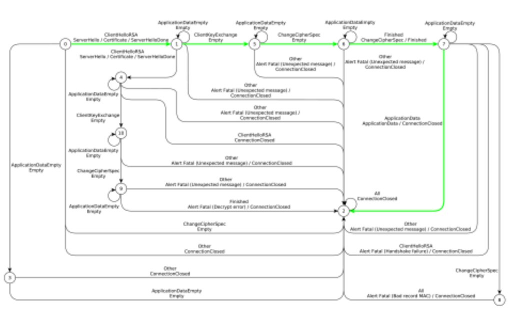
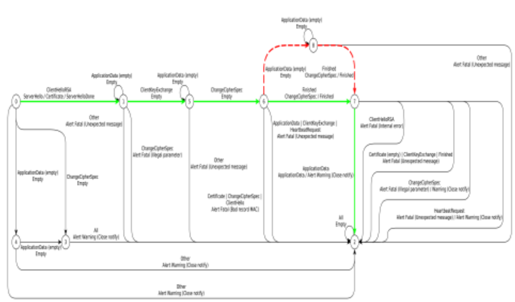
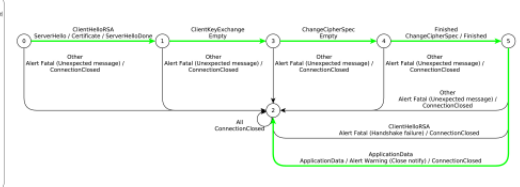
Analysis of TLS servers

- 9 TLS implementations
  - OpenSSL / BoringSSL / LibreSSL
  - GnuTLS
  - Java Secure Socket Extension
  - mbed TLS (previously PolarSSL)
  - NSS
  - RSA BSAFE for C
  - RSA BSAFE for Java
  - miTLS
  - nqsb-TLS
- Every learned model different

Learned models



Learned models

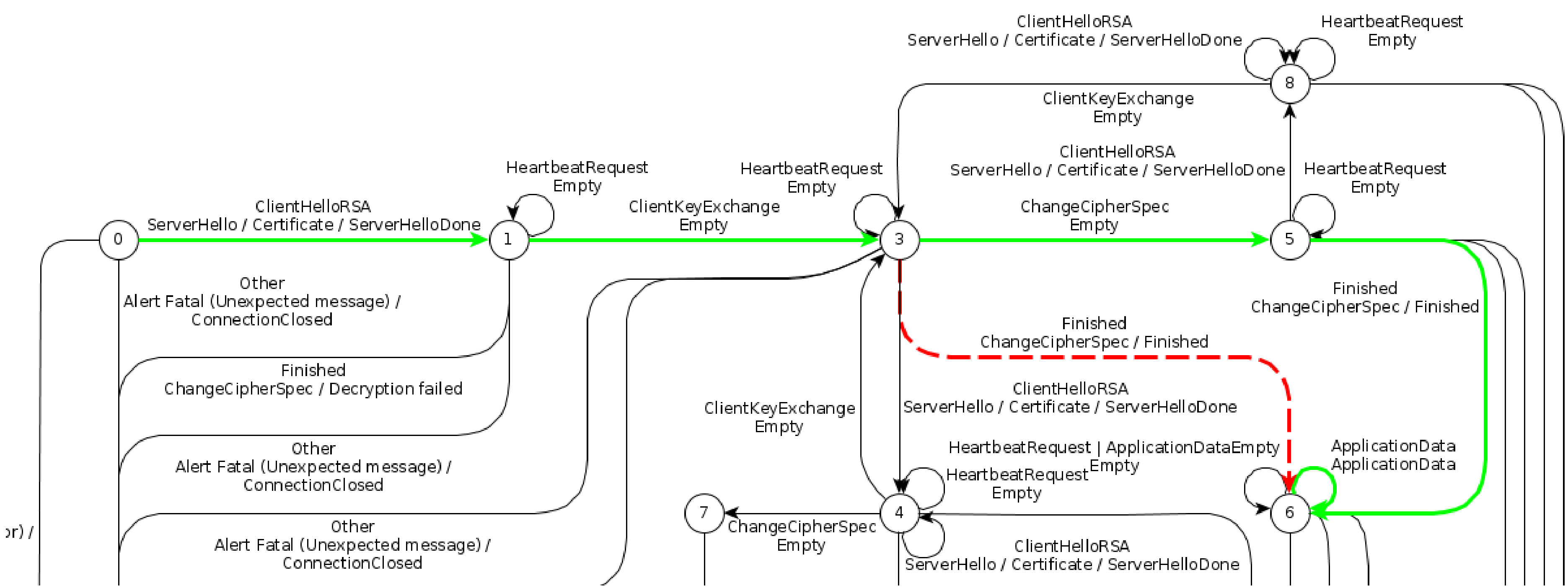












Results

- Used demo applications when provided
- 6 to 16 states
- 6 minutes to over 8 hours
  - Under 1 hour if connections are properly closed
  - Dependent on implementation specific time-outs (100ms to 1,5s)
- Several new flaws in different implementations

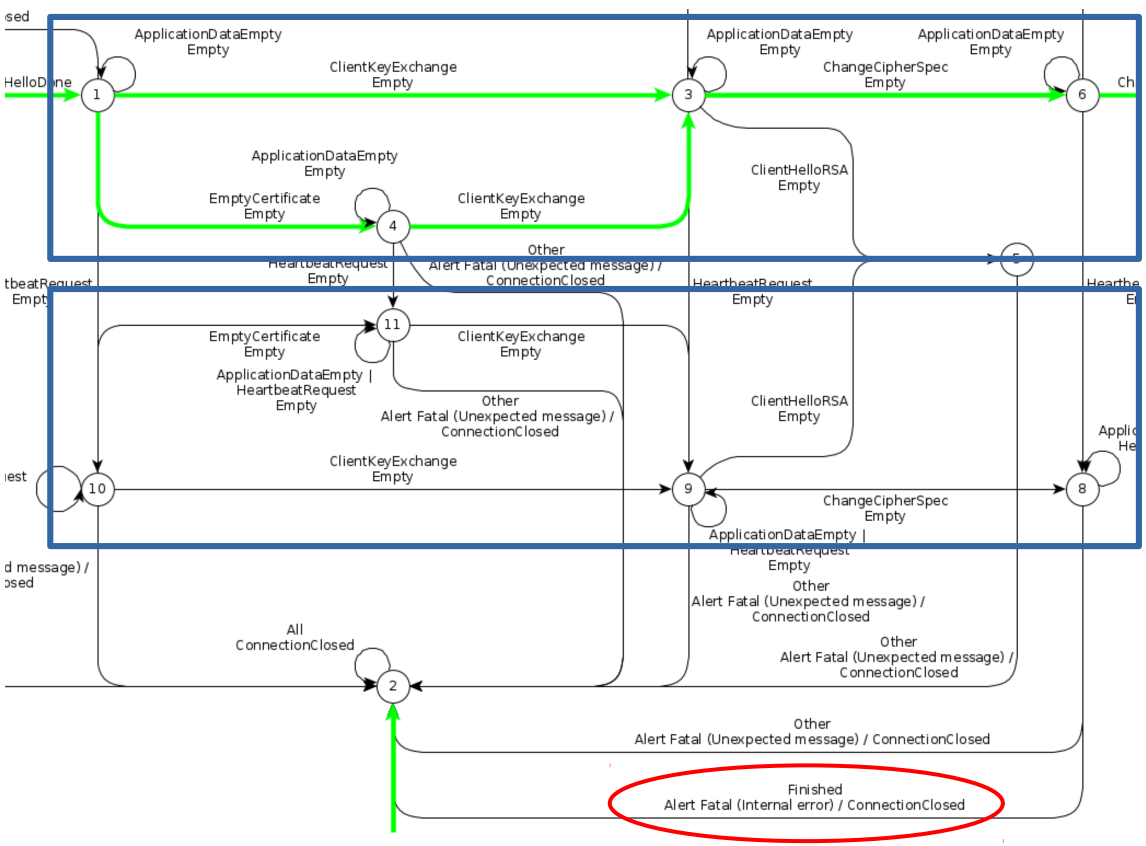
Java Secure Socket Extension

- Possible to skip ChangeCipherSpec message
- Server will accept plaintext data
- Problem also present in client
- Also found by the Prosecco group at INRIA
- Fixed in January 2015



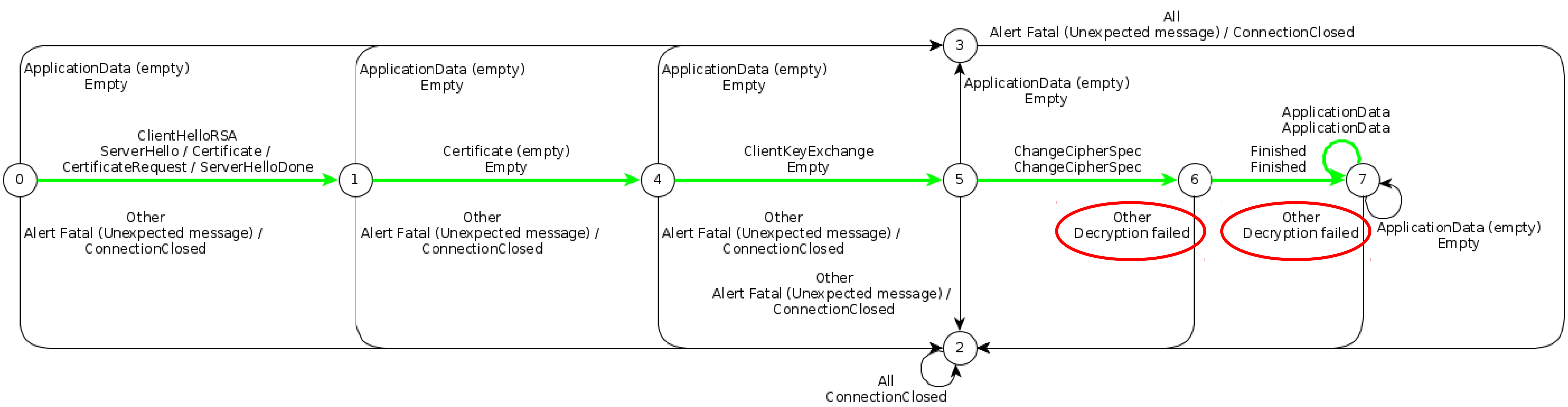
GnuTLS

- Shadow path after sending HeartbeatRequest during handshake
- Buffer handshake messages for hash in Finished reset
- Same problem present in the client



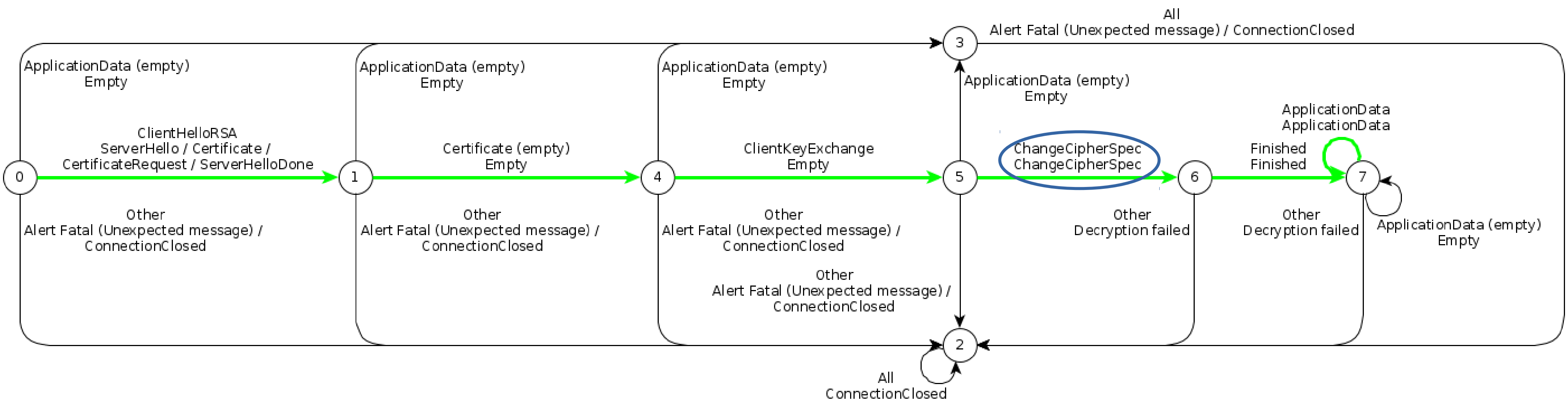
nqsb-TLS

- Plaintext alerts returned after ChangeCipherSpec
- No security flaw
- Quickly fixed
- Shows it is a useful technique during development
- Different interpretation of the specification



nqsb-TLS

- Plaintext alerts returned after ChangeCipherSpec
- No security flaw
- Quickly fixed
- Shows it is a useful technique during development
- Different interpretation of the specification



Conclusions

- State machine inference is a useful technique to find security flaws and other bugs
- Everybody interprets specifications differently and makes different design decisions
  - Can be used for fingerprinting!
- It would be good to include state machines in specifications

Conclusions

- State machine inference is a useful technique to find security flaws and other bugs
- Everybody interprets specifications differently and makes different design decisions
  - Can be used for fingerprinting!
- It would be good to include state machines in specifications

Thank you for your attention!