Automated Analysis of TLS 1.3

0-RTT, Resumption and Delayed Authentication

TRON, 21 February 2016

Cas Cremers

Marko Horvat

Sam Scott

Thyla van der Merwe

UNIVERSITY OF OXFORD

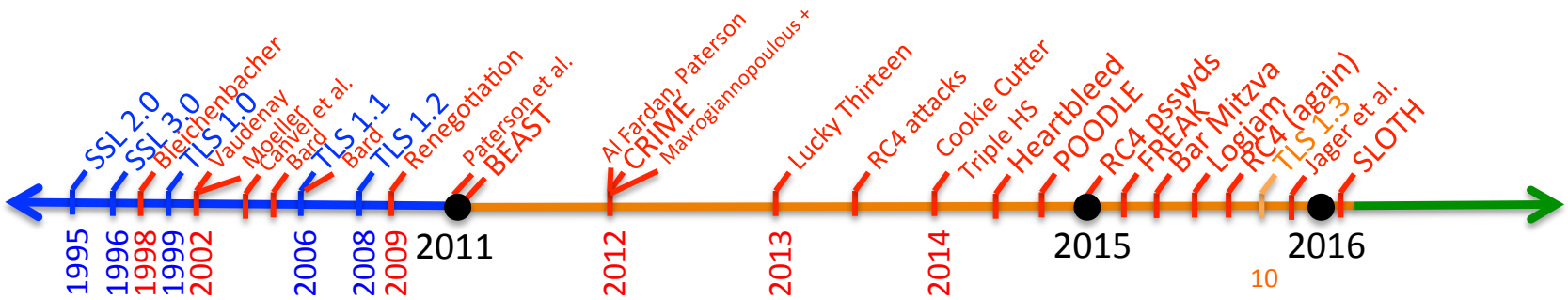
ROYAL HOLLOWAY UNIVERSITY OF LONDON

mozilla

What we did (nutshell)

- We built a symbolic model of the TLS 1.3 specification - draft 10
- We wanted to verify the **main** properties of TLS 1.3 as an authenticated key exchange protocol
  - secrecy of session keys
  - unilateral (mutual) authentication
- We extended our draft 10 model to include the delayed authentication mechanism
- We found a potential attack - disclosed this to the IETF TLS WG
- We have updated our model to draft 11 of the specification and are 'half-way' through proving

Where our work fits in...



Program verification

Ongoing work – later talks

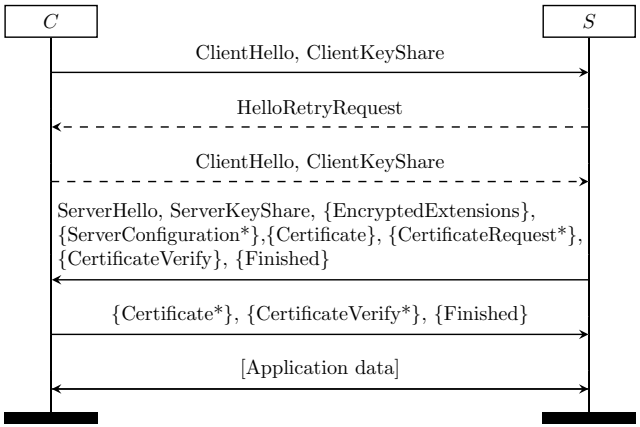
Provable security

Dowling et al. [draft-05]
Kohlweiss et al. [draft-05]
Krawczyk and Wee [OPTLS]
Dowling et al. [draft-10]

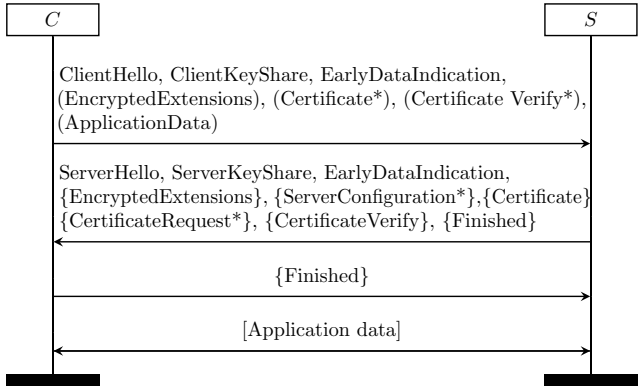
Formal methods

We are here!

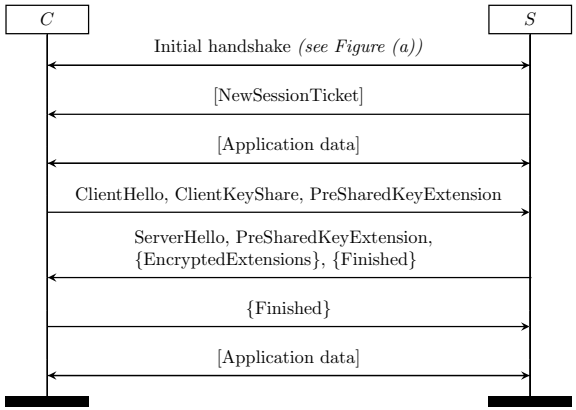
TLS 1.3



(a) Initial (EC)DHE handshake

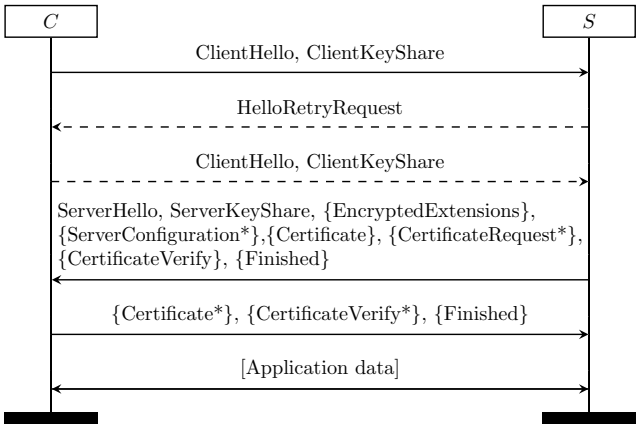


(b) 0-RTT handshake

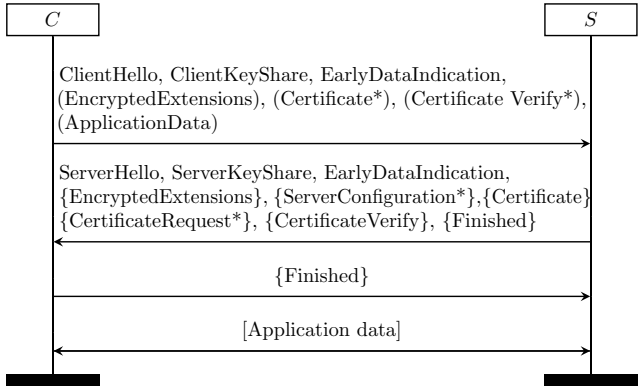


(c) PSK-resumption handshake (+ PSK-DHE)

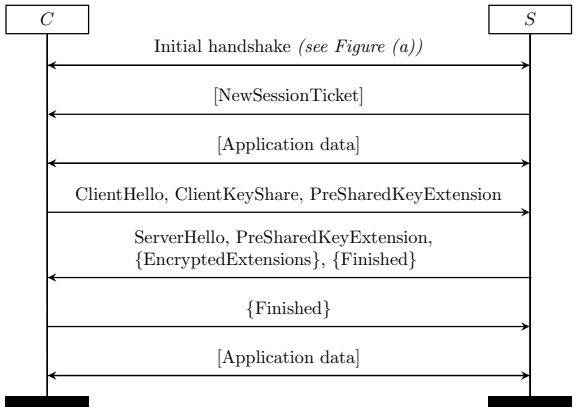
TLS 1.3



(a) Initial (EC)DHE handshake



(b) 0-RTT handshake



(c) PSK-resumption handshake (+ PSK-DHE)

Look at the full interaction of all these components!

Symbolic analysis

- A symbolic model allows us to examine the logical soundness of the protocol
- We systematically specify and hope to exclude a class of attacks - 'logical' attacks (think Triple Handshake for TLS 1.2)
- We assume black-box cryptography
- We analyse the specification only

Using Tamarin

Tool details available at:
http://www.infsec.ethz.ch/research/software/tamarin.html

Using Tamarin

We built our model for use in the Tamarin prover

- Automated tool for protocol analysis
- Good symbolic Diffie-Hellman support
- Considers an unbounded number of parties/handshakes

Using Tamarin

We built our model for use in the Tamarin prover

- Automated tool for protocol analysis
- Good symbolic Diffie-Hellman support
- Considers an unbounded number of parties/handshakes

How does it work?

- For simple models/properties, can prove automatically
- Complex models require more user interaction
- A proof shows that a property holds in all possible combinations of client, server, and adversary behaviours

Using Tamarin

We built our model for use in the Tamarin prover

- Automated tool for protocol analysis
- Good symbolic Diffie-Hellman support
- Considers an unbounded number of parties/handshakes

How does it work?

- For simple models/properties, can prove automatically
- Complex models require more user interaction
- A proof shows that a property holds in all possible combinations of client, server, and adversary behaviours

Tamarin is good because

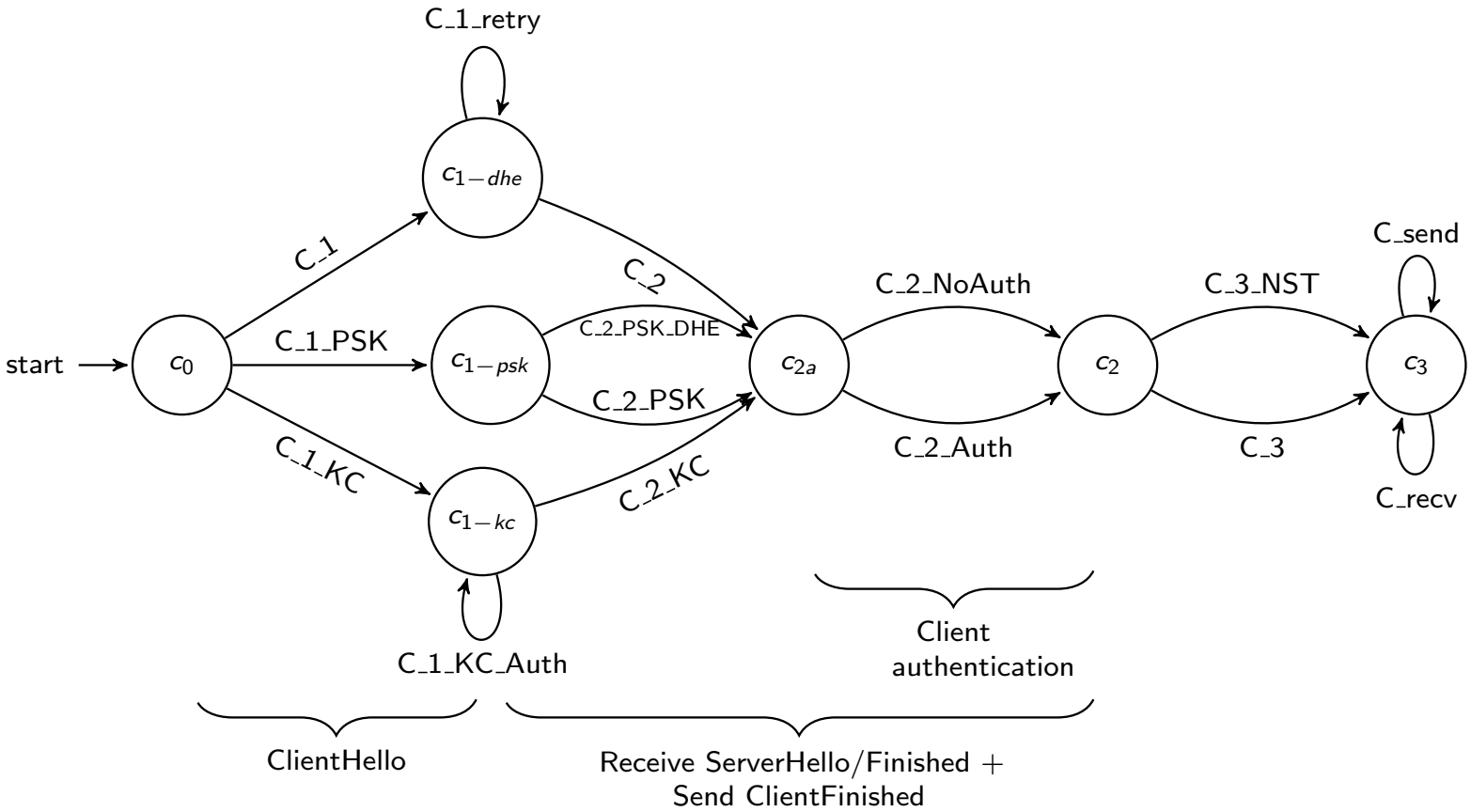
- We can precicely model the TLS state machines
- We can accurately capture security properties

Step 1: Building a model

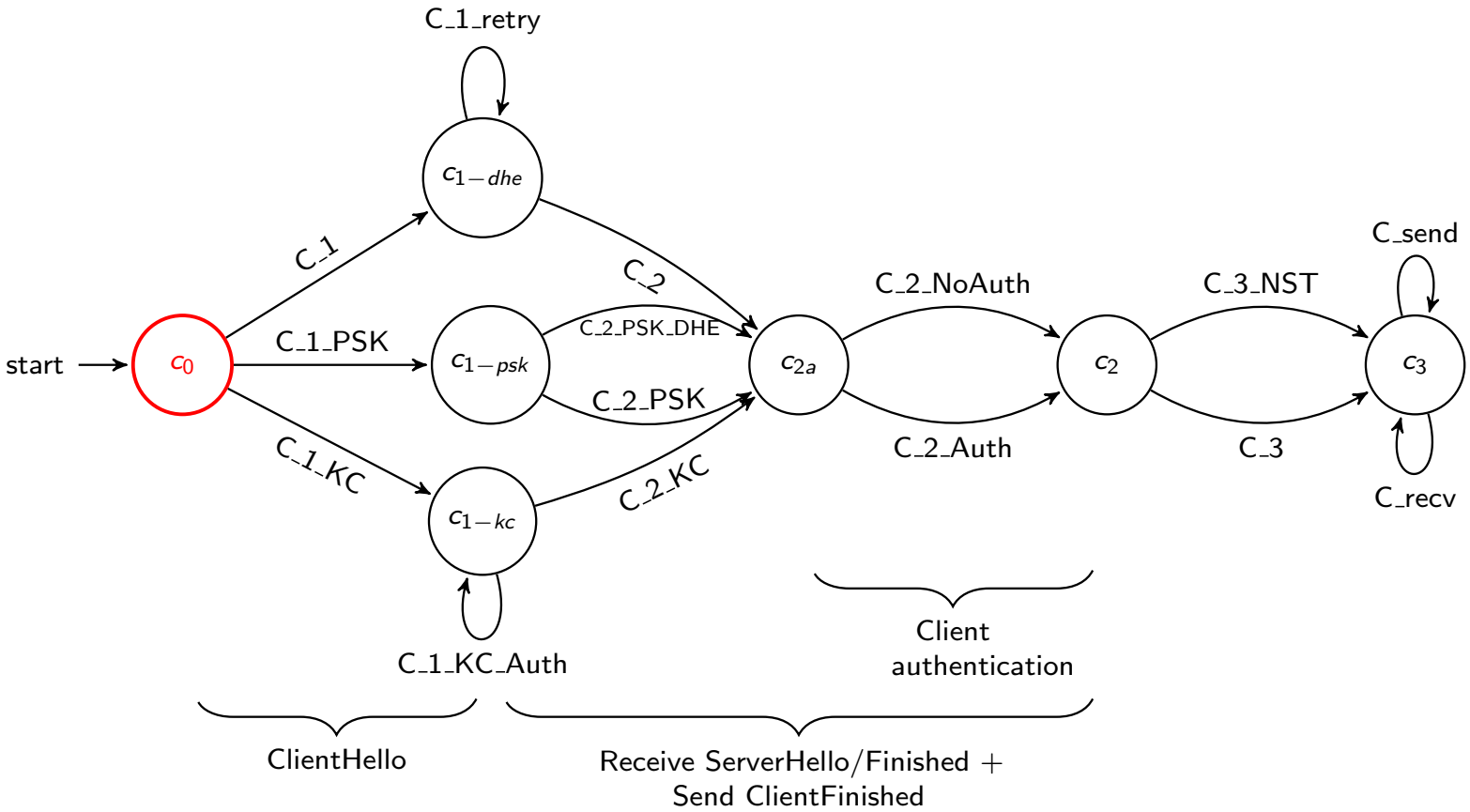
Step 1: Building a model

- Encode honest party and adversary actions as Tamarin 'rules'
- For honest clients and servers rules correspond to flights of messages
- Rules transition the protocol from one state to the next

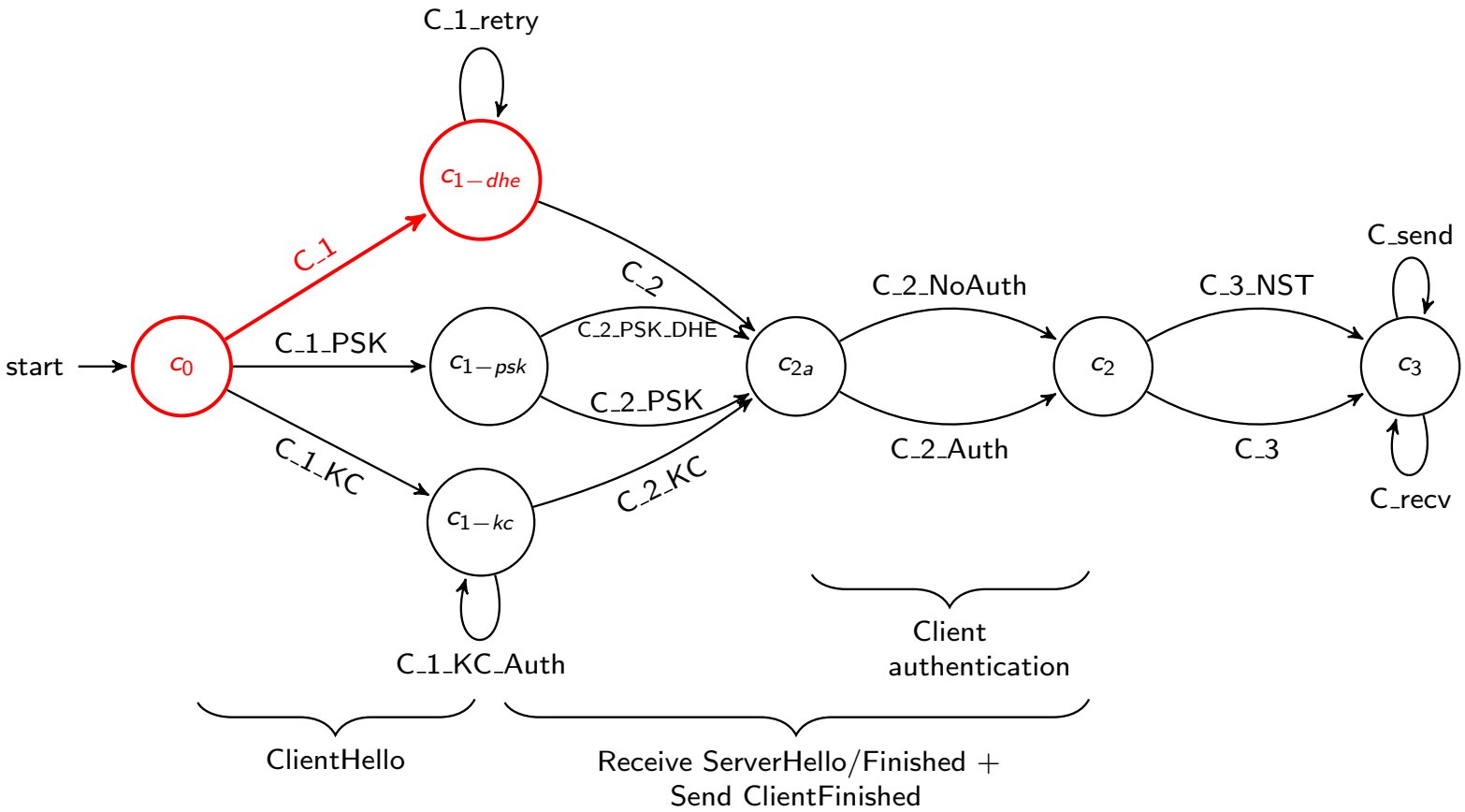
Step 1: Building a model



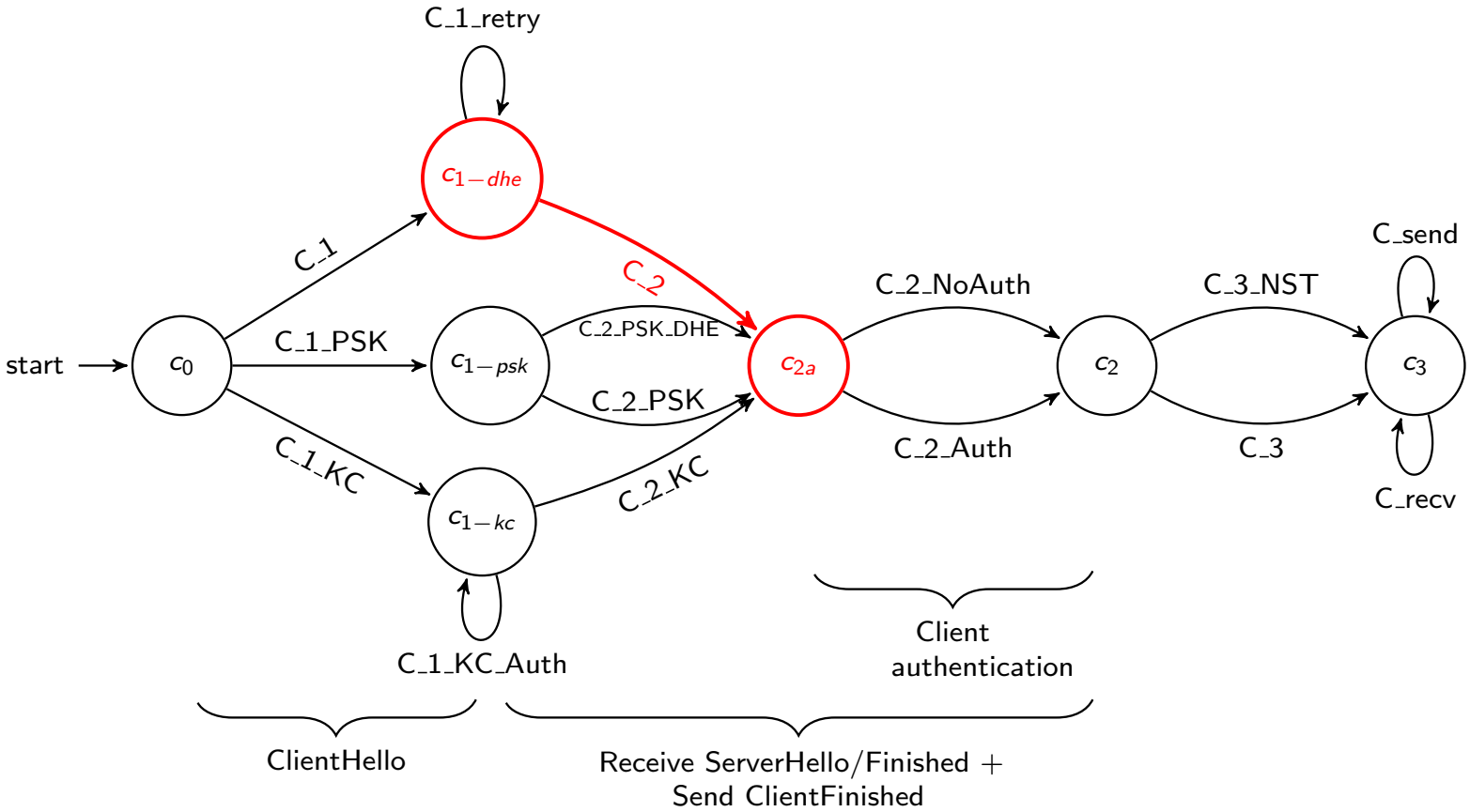
Step 1: Building a model



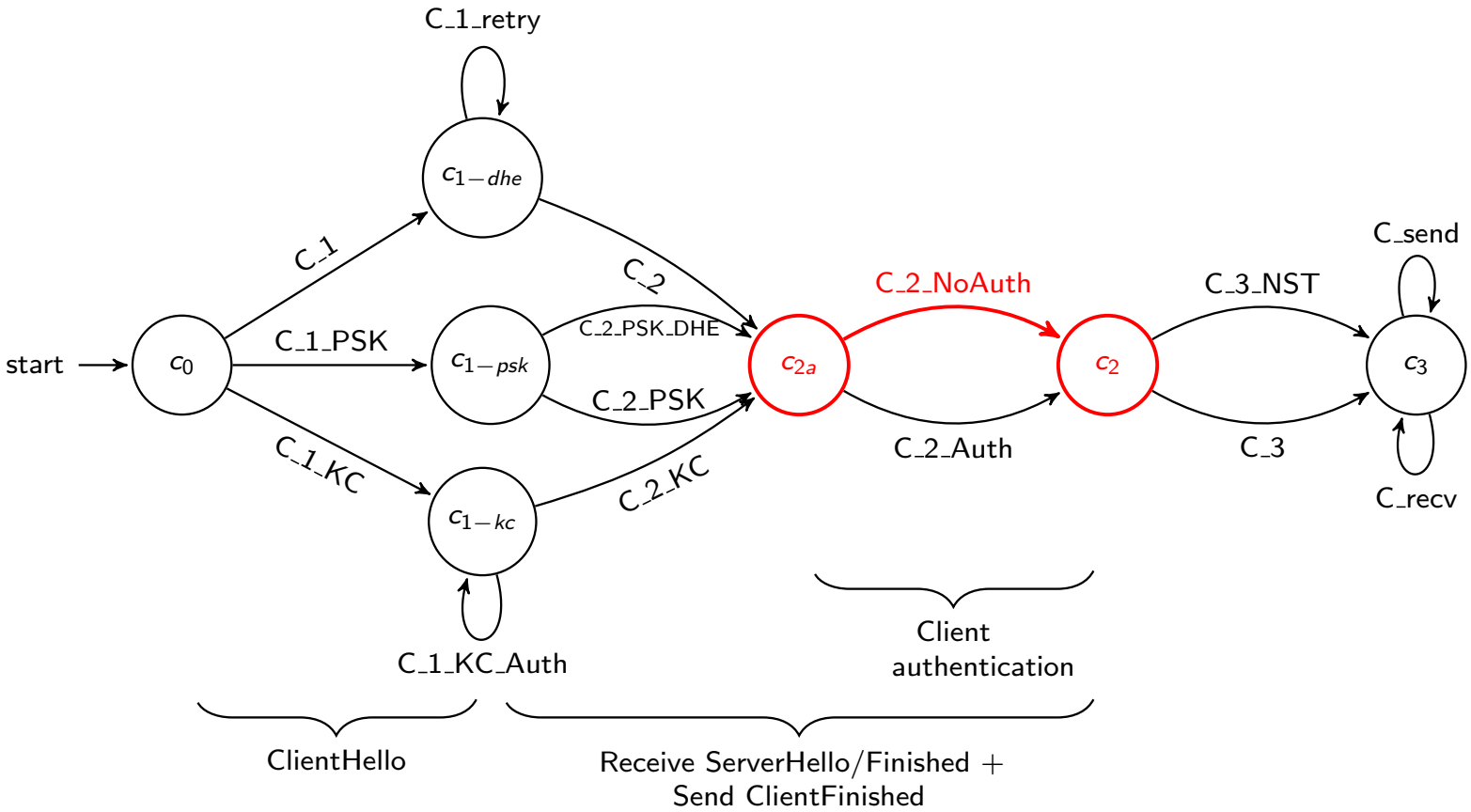
Step 1: Building a model



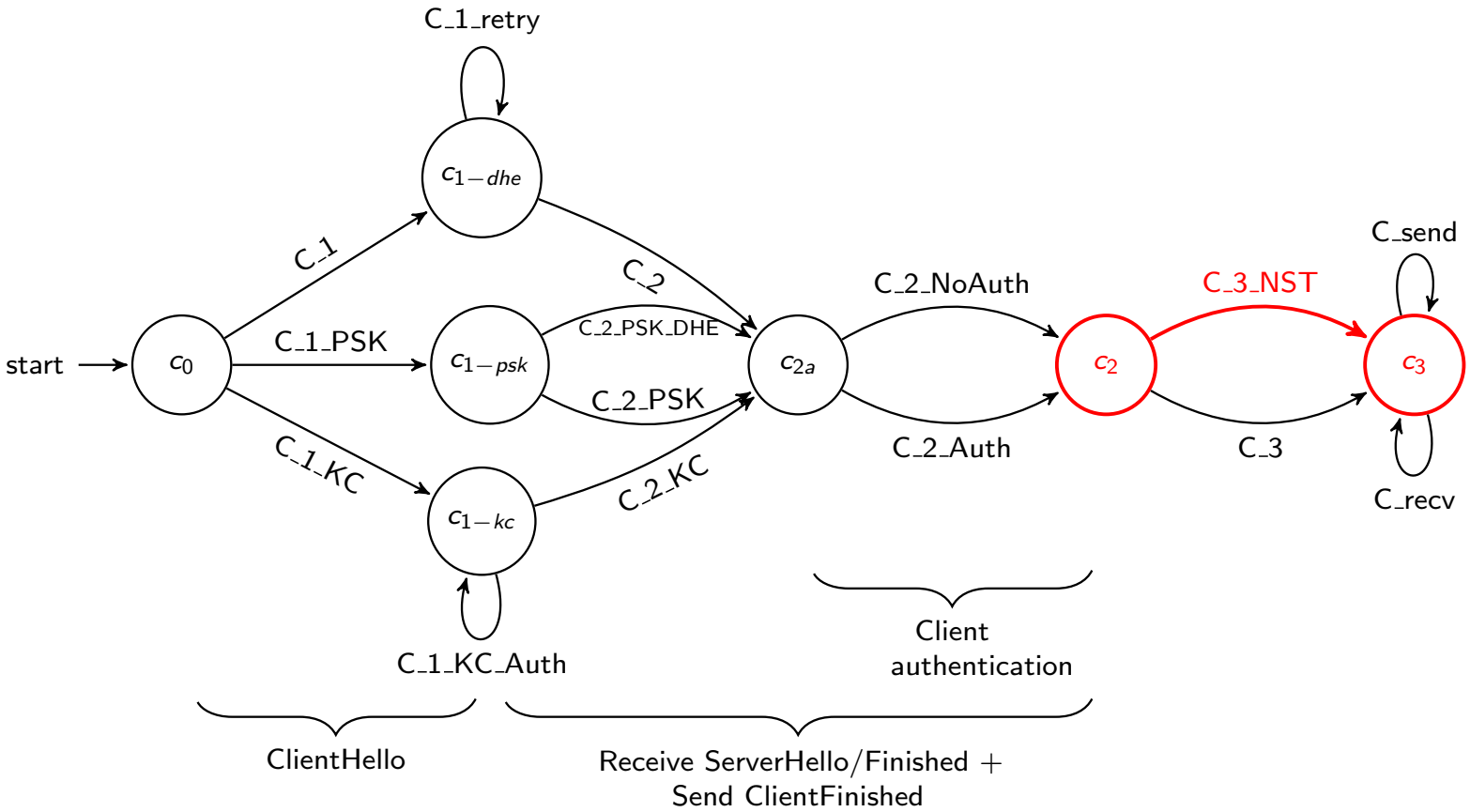
Step 1: Building a model



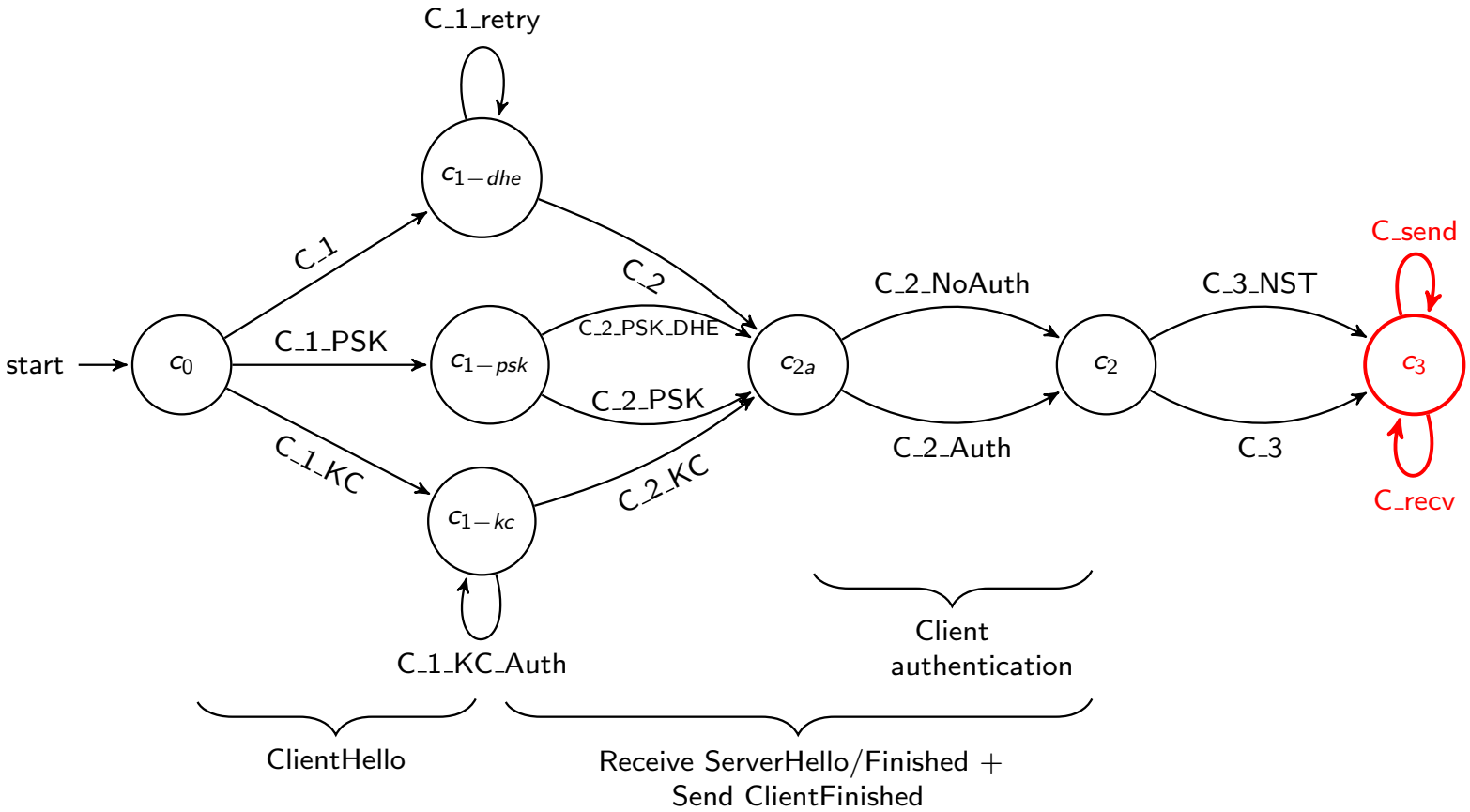
Step 1: Building a model



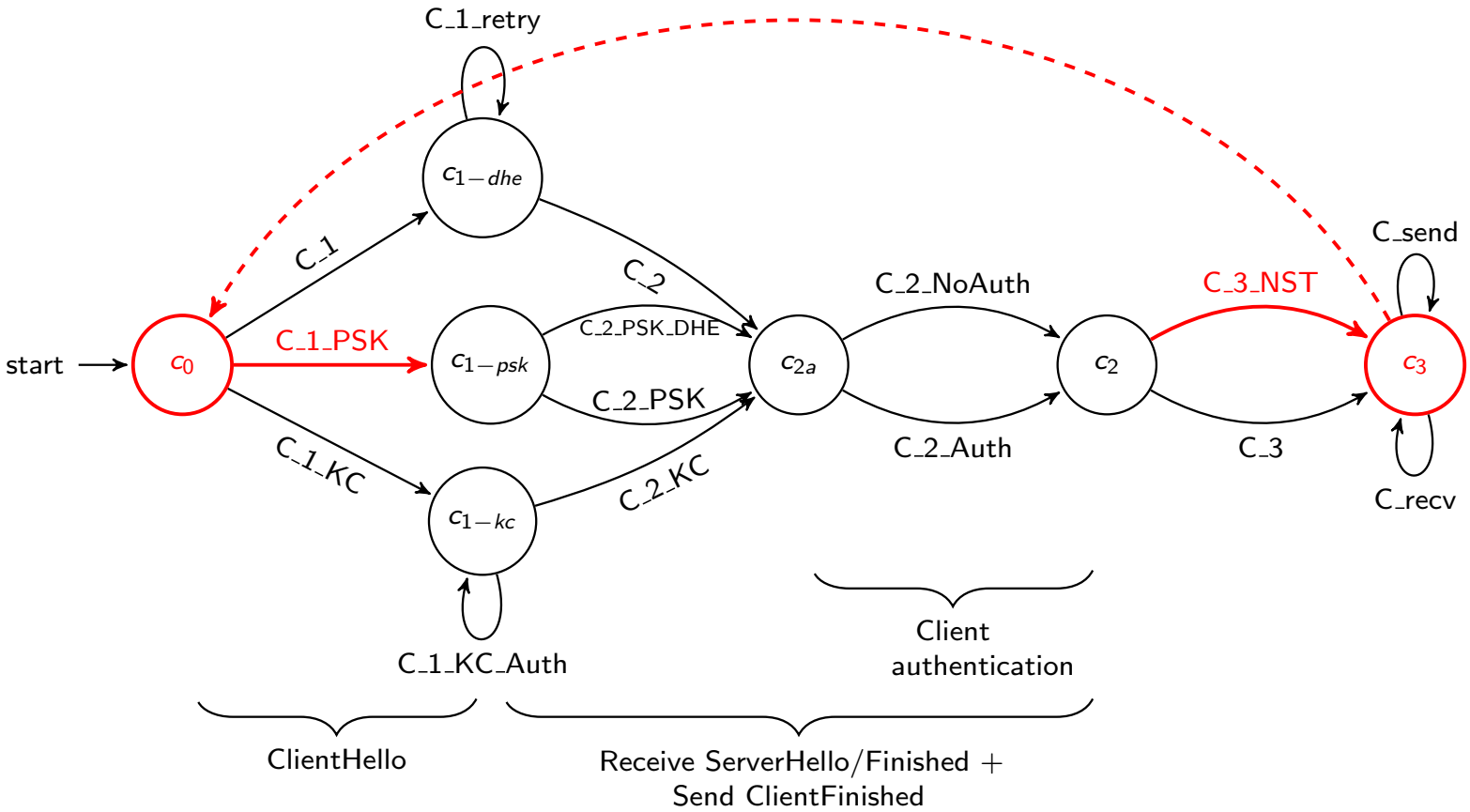
Step 1: Building a model



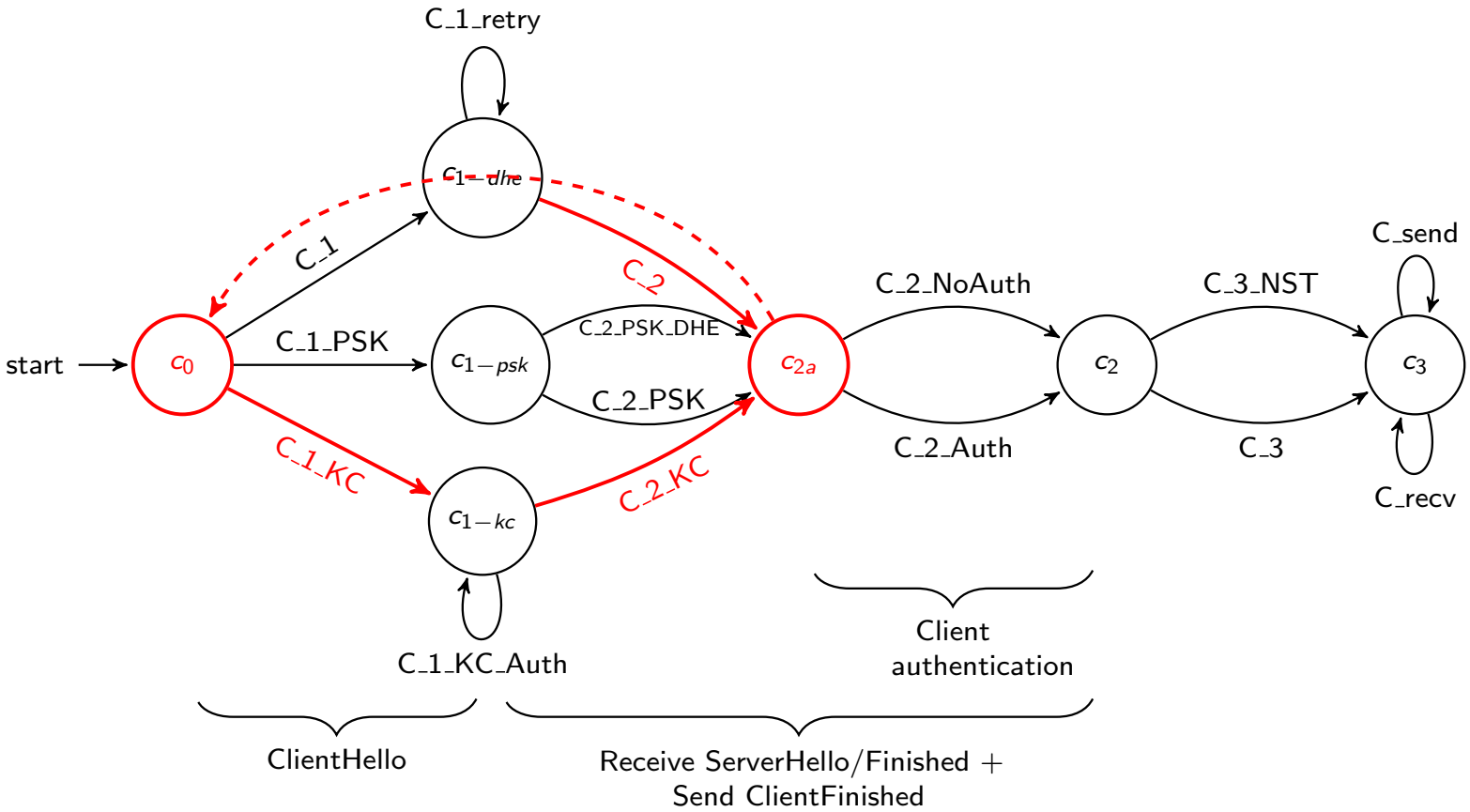
Step 1: Building a model



Step 1: Building a model



Step 1: Building a model



Step 1: Building a model

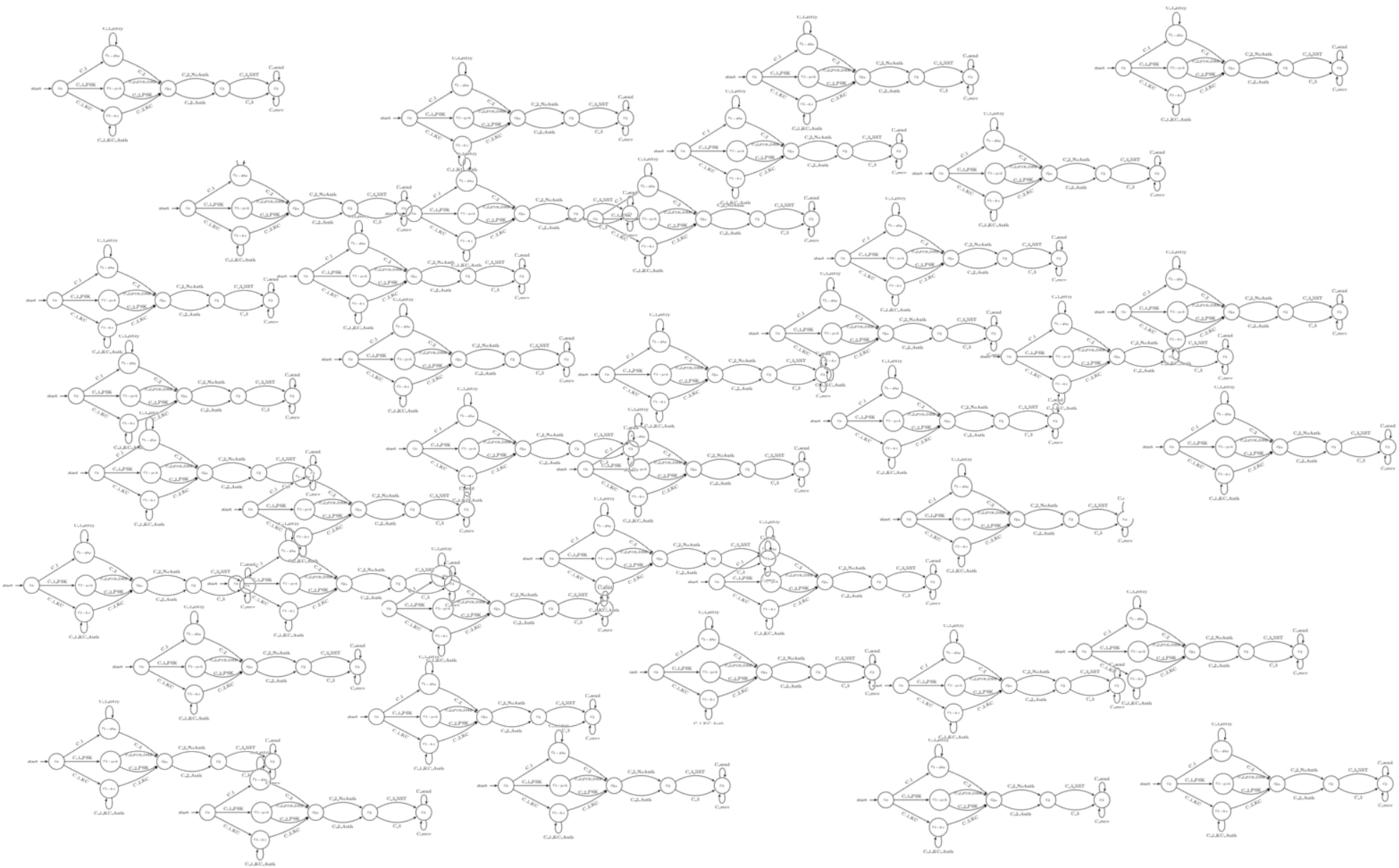
```
rule C_1:
let
    // Default C1 values
    tid =  ~nc

    // Client Hello
    C   = $C
    nc  = ~nc
    pc  = $pc
    S   = $S

    // Client Key Share
    ga   = 'g'^~a

    messages = <nc, pc,ga>
in
    [ Fr(nc)
    , Fr(~a)
    ]
  --[ C1(tid)
    , Start(tid, C, 'client')
    , Running(C, S, 'client', nc)
    , DH(C, ~a)
    ]->
    [ St_C_1_init(tid, C, nc, pc, S, ~a, messages, 'no_auth')
    , Out(<C,nc, pc,ga>)
    ]
```

Step 1: Building a model



Step 2: Encoding security properties

TLS 1.3 goals include:

- unilateral authentication of the server (mandatory)
- mutual authentication (optional)
- confidentiality and perfect forward secrecy of session keys

We have Dolev-Yao style adversary that has the ability to compromise long-term keys

Step 2: Encoding security properties

Security property	**Covered by analysis**	**Source**
Unilateral authentication (server)	Y	D.1.1
Mutual authentication	Y	D.1.1
Confidentiality of ephemeral secret	Y	D.1.1
Confidentiality of static secret	Y	D.1.1
Perfect forward secrecy	Y	D.1.1.1
Integrity of handshake messages	Y	D.1.3
Protection of application data	N	D.2
Denial of service	N	D.3
Version rollback	N	D.1.2

Table : TLS 1.3 rev 10 properties

Step 2: Encoding security properties

```
secret_session_keys:
(1) "All actor peer role k #i.
(2)  SessionKey(actor, peer, role, <k, 'authenticated'>)@i
(3)  & not ((Ex #r. RevLtk(peer)@r  & #r < #i)
           |(Ex #r. RevLtk(actor)@r & #r < #i))
(4)  ==> not Ex #j. KU(k)@j"
```

This says...

- for all possible values of variables on the first line (1),
- if key k is accepted at time point i (2), and
- the adversary has not revealed the long term keys of the actor or the peer before the key is accepted (3),
- then the adversary cannot derive the key (4).

Step 2: Encoding security properties

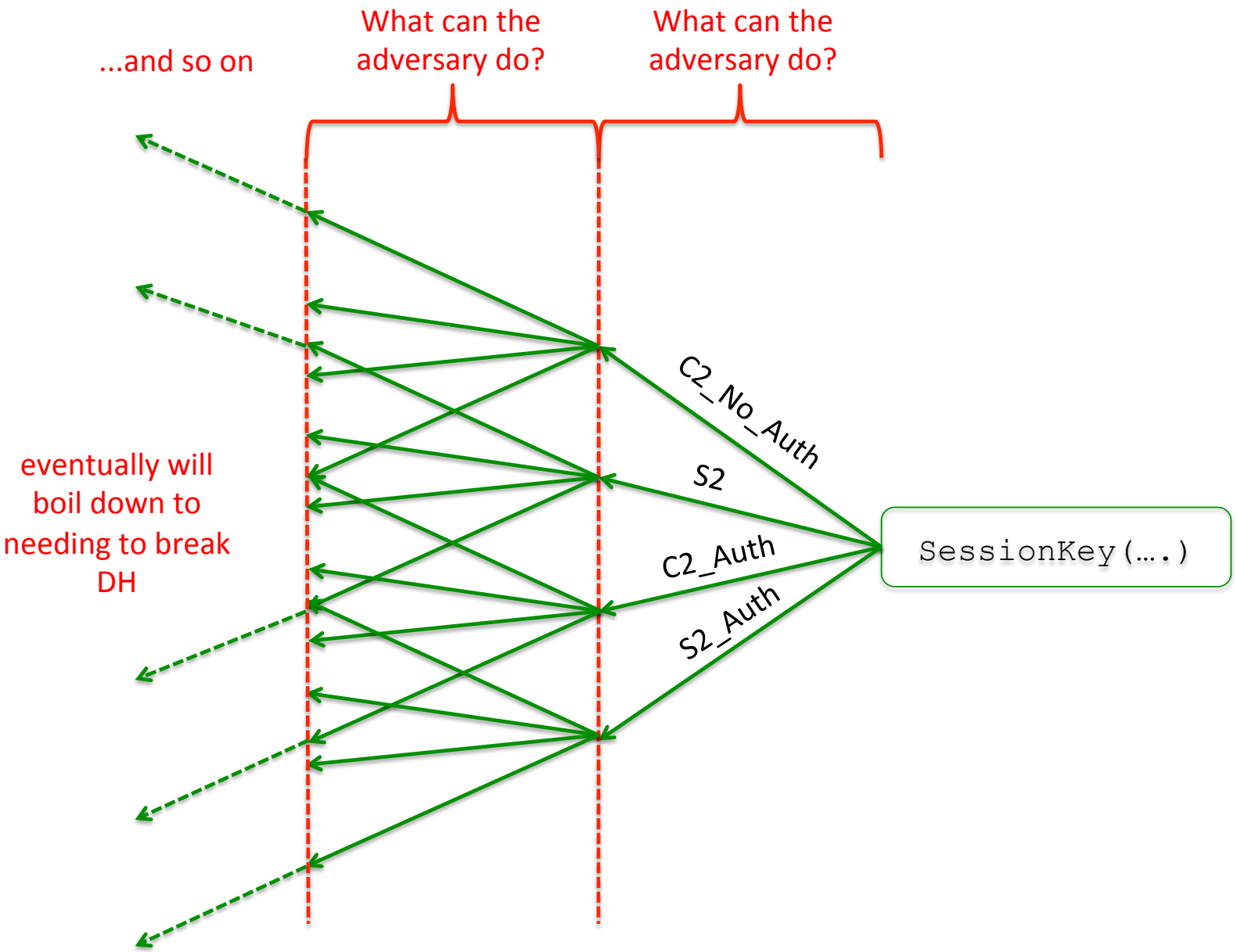
```
secret_session_keys:
(1) "All actor peer role k #i.
(2)  SessionKey(actor, peer, role, <k, 'authenticated'>)@i
(3)  & not ((Ex #r. RevLtk(peer)@r  & #r < #i)
           |(Ex #r. RevLtk(actor)@r & #r < #i))
(4)  ==> not Ex #j. KU(k)@j"
```

This says...

- for all possible values of variables on the first line (1),
- if key k is accepted at time point i (2), and
- the adversary has not revealed the long term keys of the actor or the peer before the key is accepted (3),
- then the adversary cannot derive the key (4).

Want to show that this holds for all combinations of client, server and adversary behaviours - ALL traces

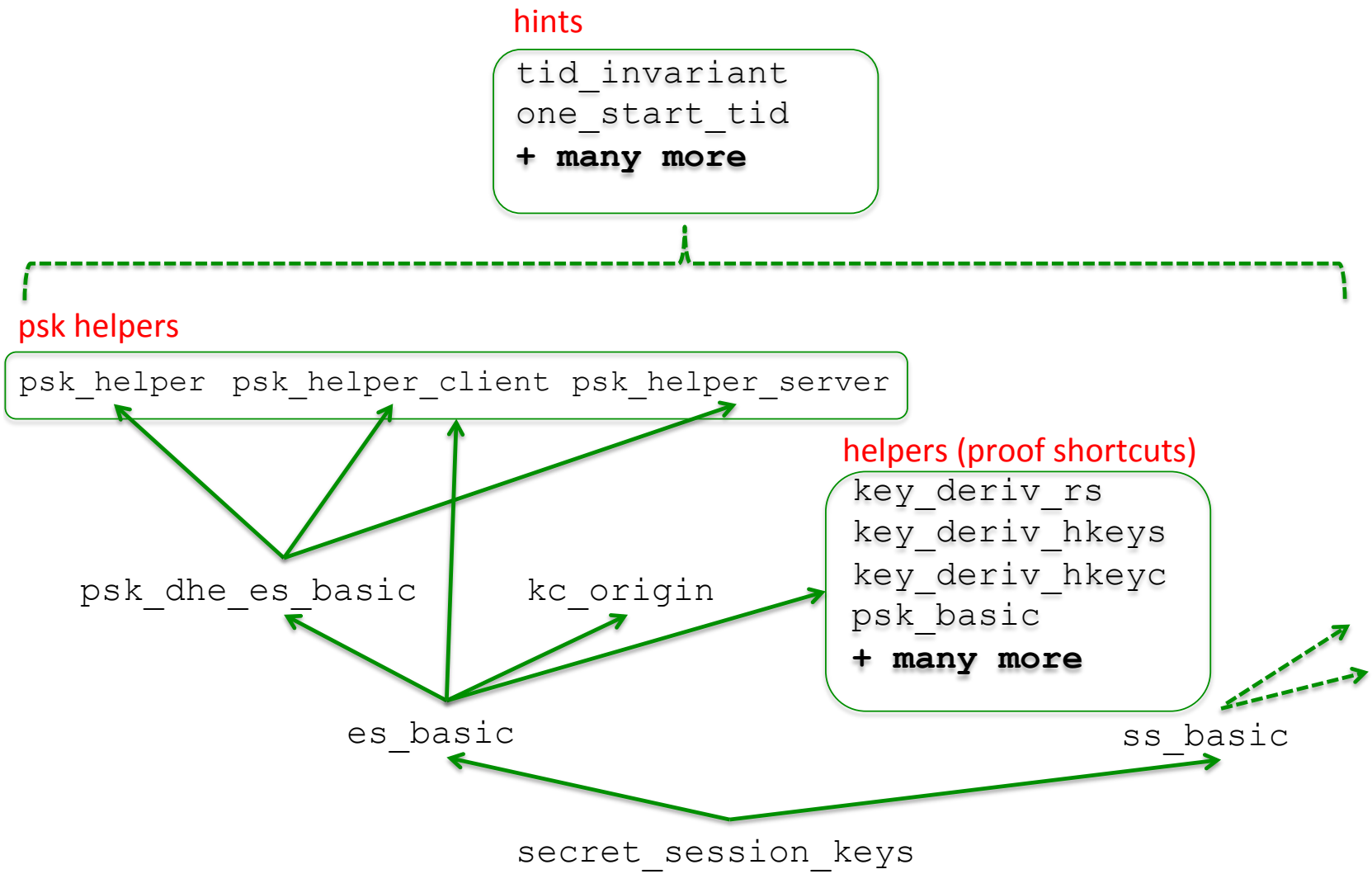
Step 3: Proving security properties



Step 3: Proving security properties

- Not a straightforward application of Tamarin
  - several man-months of work
  - specification a moving target
  - updating takes time, can be error-prone
- Need an intimate knowledge of the protocol - high degree of interaction with the tool in some cases
- Not a push-button analysis
- We have 45 auxiliary lemmas

Step 3: Proving security properties

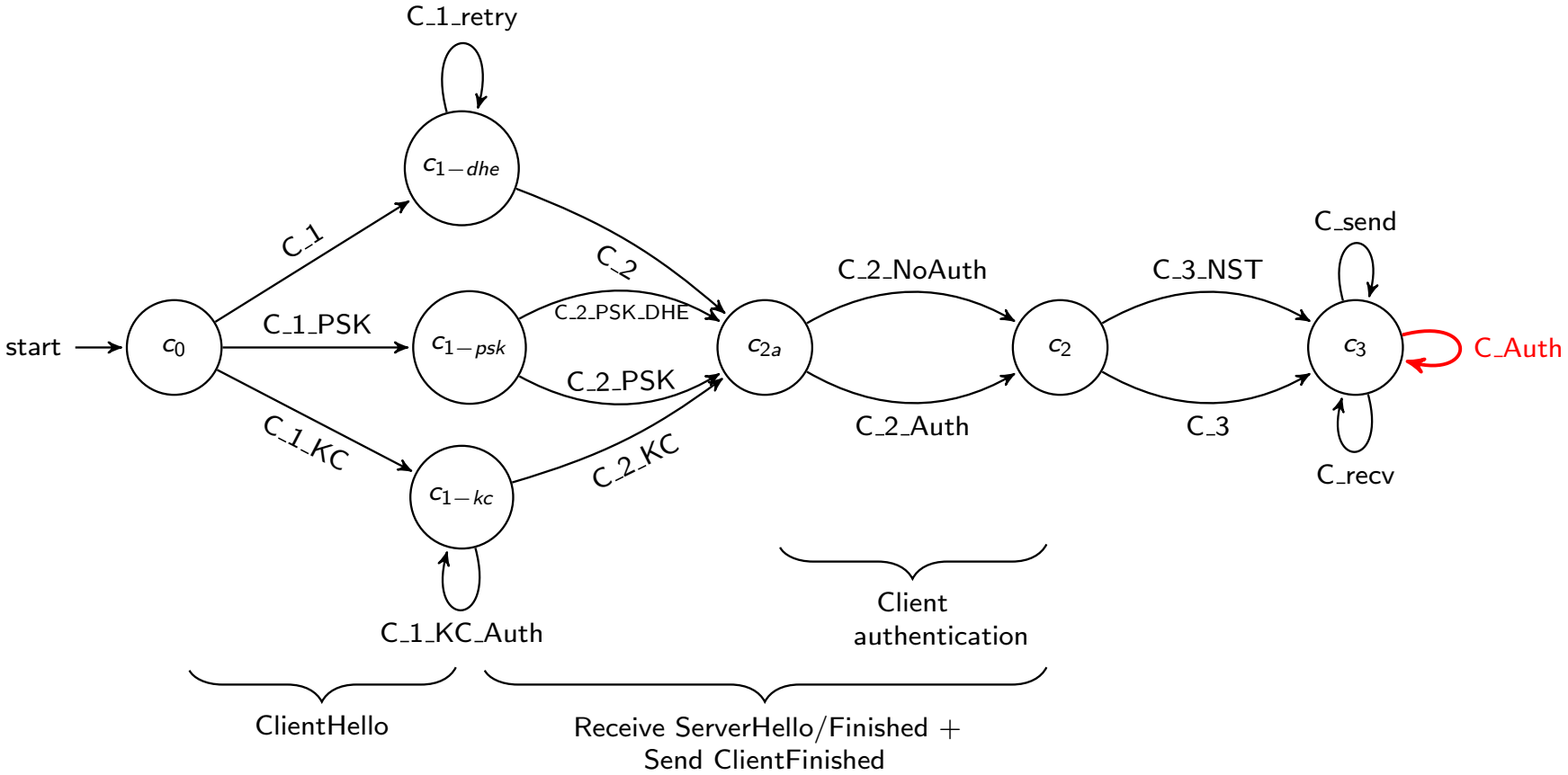


Step 3: Proving security properties

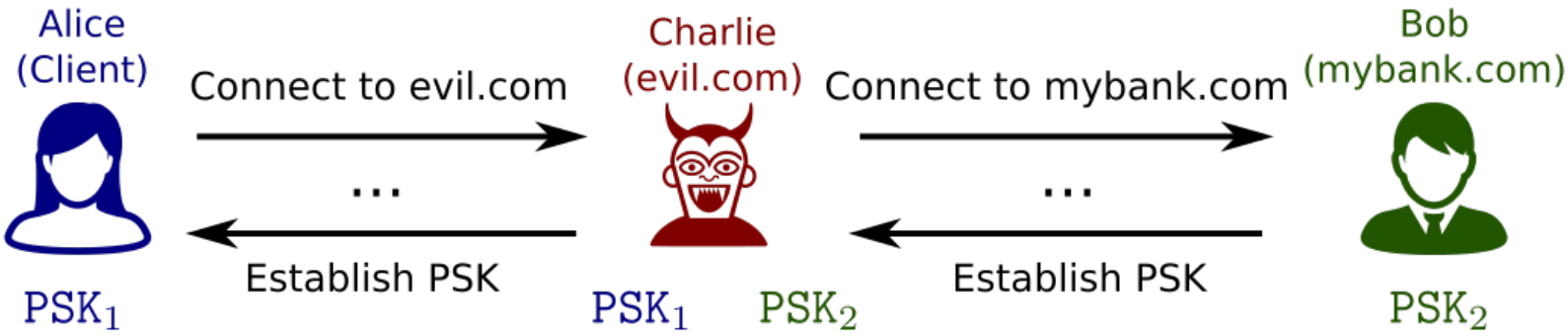
We verified the main properties of TLS 1.3 revision 10 as an authenticated key exchange protocol:

- Secrecy of session keys
  - holds for both client and server
  - forward secrecy
- Mutual authentication

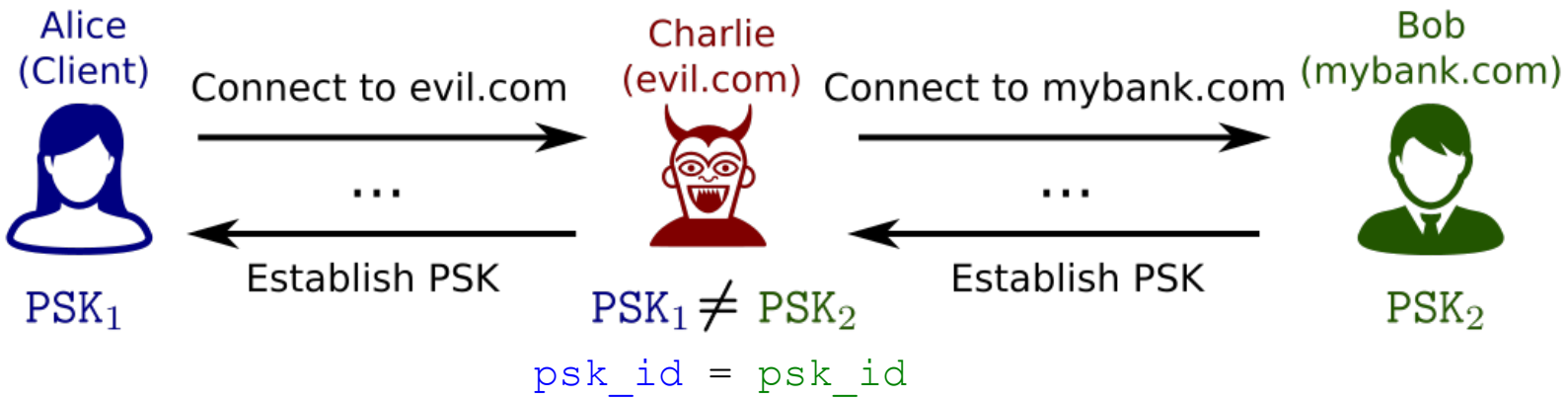
Attacking client authentication (revision 10+)



Attacking client authentication



Attacking client authentication

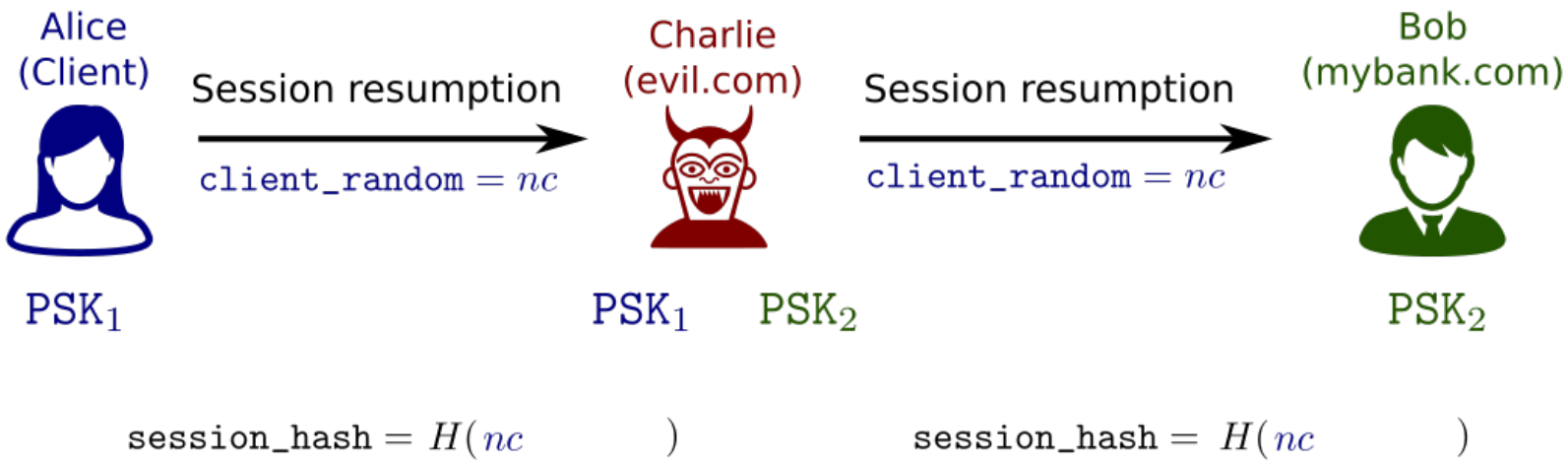


`psk_id` = `psk_id`

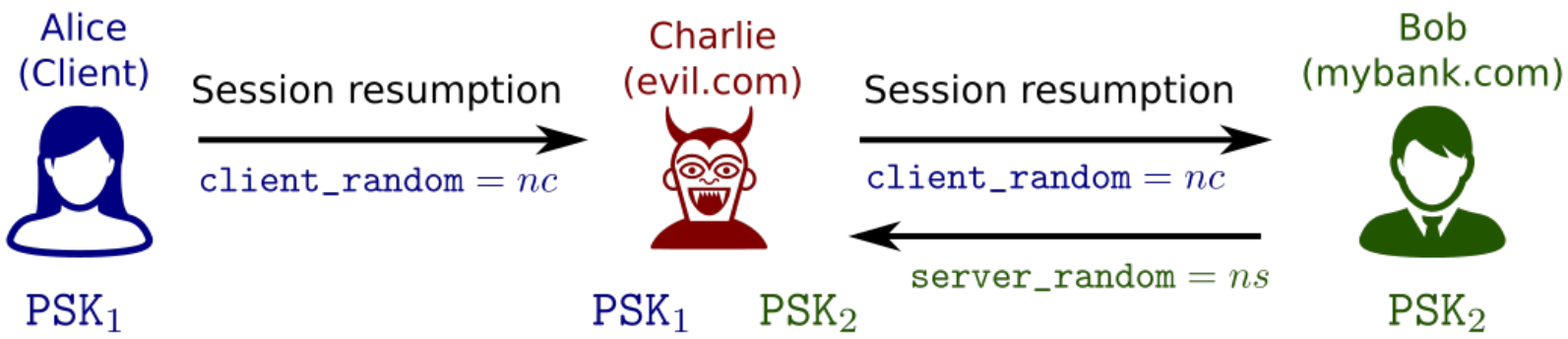
Attacking client authentication

$$\texttt{session\_hash} = H(nc \qquad)$$

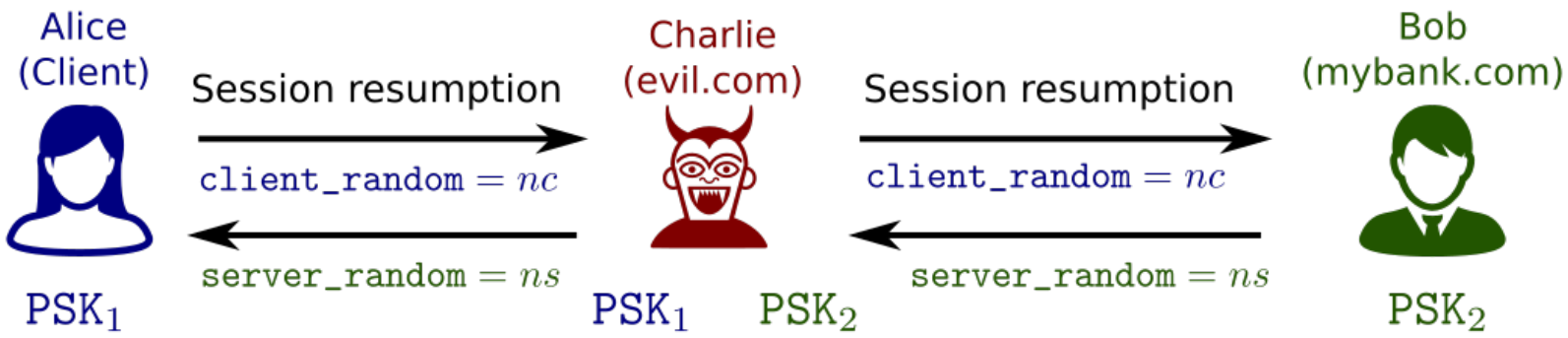
Attacking client authentication



Attacking client authentication



Attacking client authentication

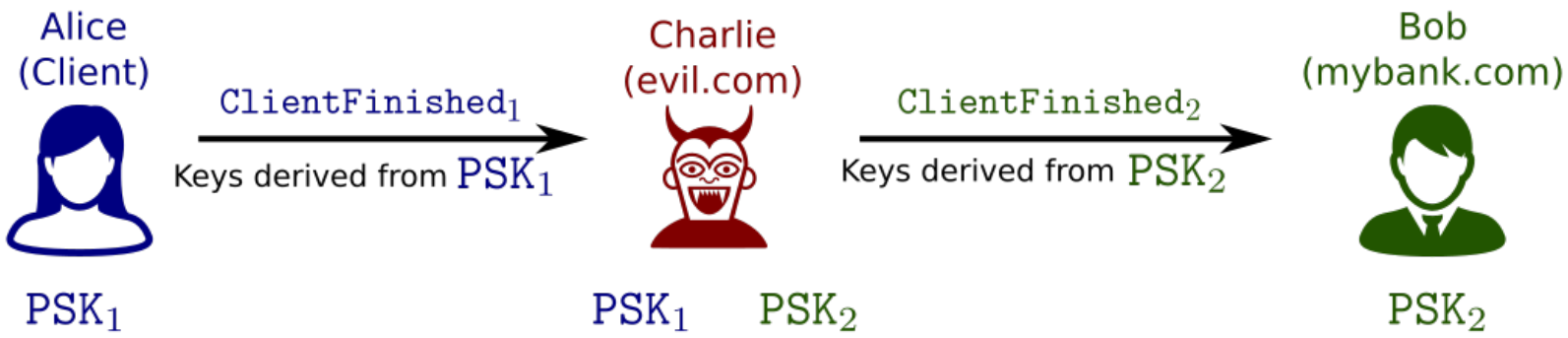


$\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$ $\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$

Attacking client authentication

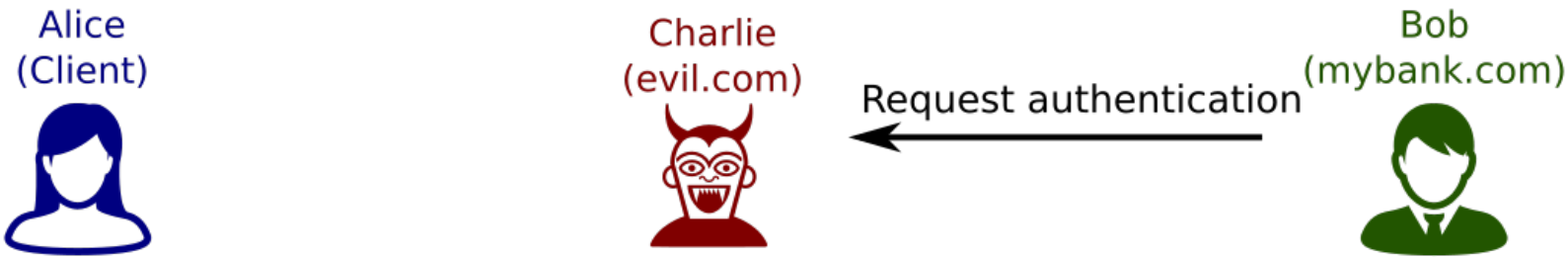
$\texttt{session\_hash} = H(nc\ \ ns\ \ \dots)$ $\texttt{session\_hash} = H(nc\ \ ns\ \ \dots)$

Attacking client authentication



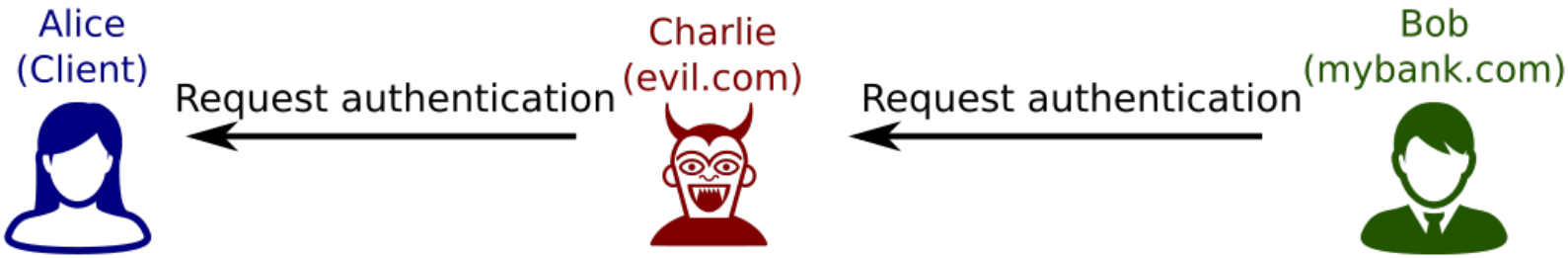
$\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$ $\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$

Attacking client authentication



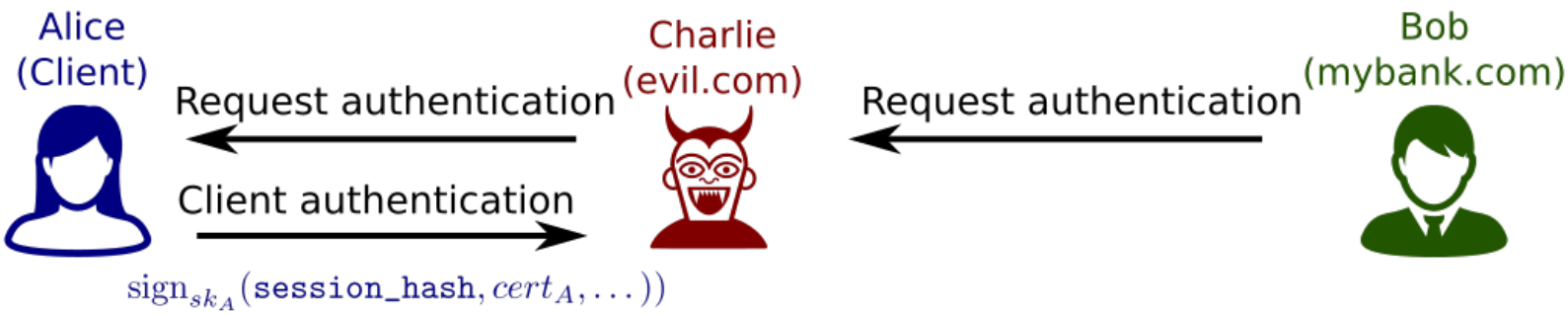
$\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$ $\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$

Attacking client authentication



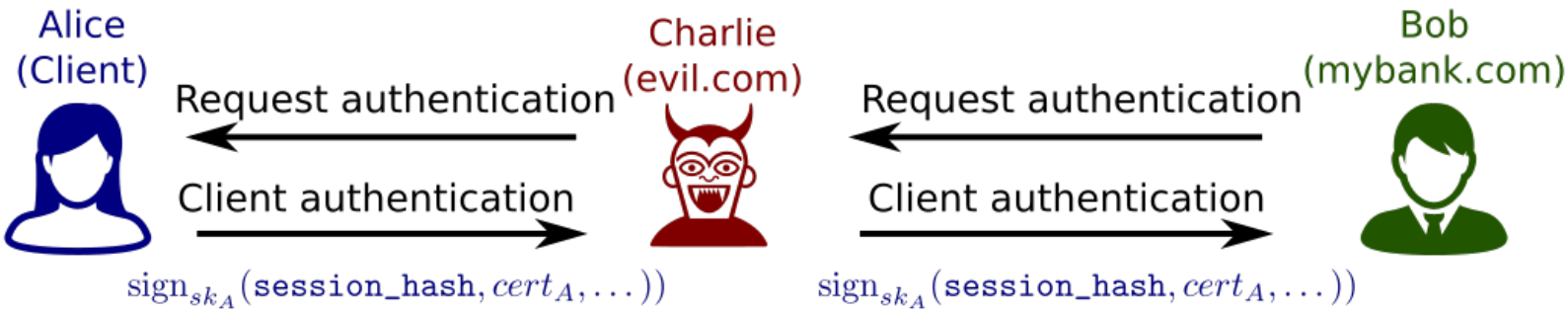
$\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$ $\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$

Attacking client authentication



$\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$ $\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$

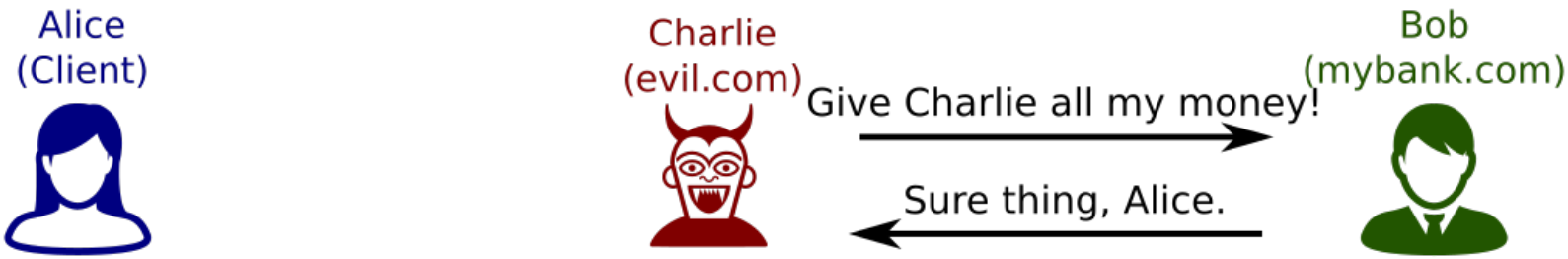
Attacking client authentication



$\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots\ )$

$\texttt{session\_hash} = H(nc\ \ ns\ \ \ldots)$

Attacking client authentication



Cause and mitigation

- Prime example of an attack that can arise because of the interaction of modes
- No binding between the client signature and session for which it is intended
- Complicated to find
- Example of the positive interaction of analysis and design
- Communicated this to the IETF TLS Working Group...

Cause and mitigation

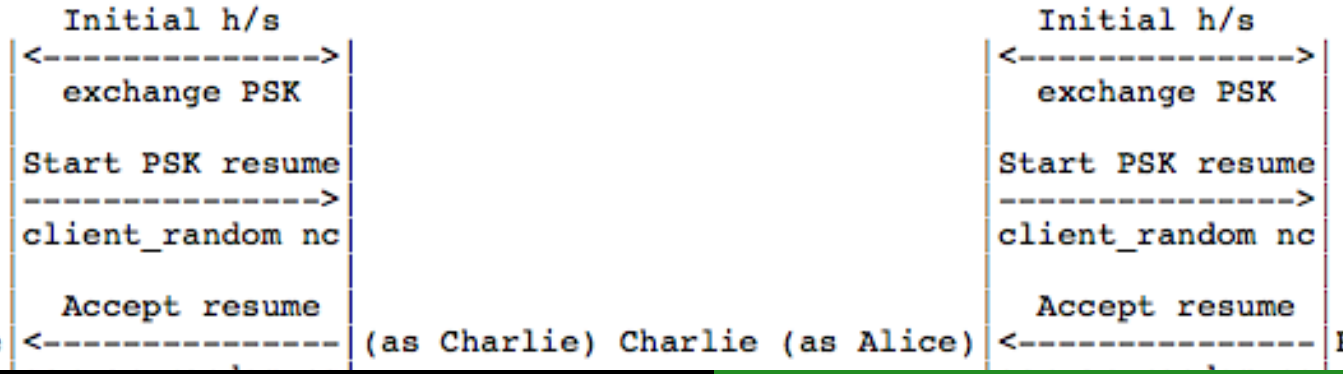
https://www.ietf.org/mail-archive/web/tls/current/msg18215.html

Dear all,

We [1] are in the process of performing an automated symbolic analysis of the TLS 1.3 specification draft (revision 10) using the Tamarin prover [2], which is a tool for automated security protocol analysis.

While revision 10 does not yet appear to permit certificate-based client authentication in PSK (and in particular resumption using PSK), we modelled what we believe is the intended functionality. By enabling client authentication either in the initial handshake, or with a post- handshake signature over the handshake hash, our Tamarin analysis finds an attack. The result is a complete breakage of client authentication, as the attacker can impersonate a client when communicating with a server:

Suppose a client Alice performs an initial handshake with Charlie. Charlie, masquerading as Alice, subsequently performs a handshake with Bob. Following a PSK resumption, Bob requests authentication from Charlie (impersonating Alice). Charlie then requests authentication from Alice, and the returned signature will also be a valid signature for the session with Bob.



Cause and mitigation

"Nice analysis! I think that the composition of different mechanisms in the protocol is likely to be where many subtle issues lie, and analyses like this one support that concern."

Cause and mitigation

"Nice analysis! I think that the composition of different mechanisms in the protocol is likely to be where many subtle issues lie, and analyses like this one support that concern."

"Thanks for posting this. It's great to see people doing real formal analysis of the TLS 1.3 draft; this is really helpful in guiding the design."

Cause and mitigation

"Nice analysis! I think that the composition of different mechanisms in the protocol is likely to be where many subtle issues lie, and analyses like this one support that concern."

"Thanks for posting this. It's great to see people doing real formal analysis of the TLS 1.3 draft; this is really helpful in guiding the design."

"This result motivates and confirms the need to modify the handshake hashes to contain the server Finished when we add post-handshake authentication as is done in PR#316..."

Cause and mitigation

"Nice analysis! I think that the composition of different mechanisms in the protocol is likely to be where many subtle issues lie, and analyses like this one support that concern."

"Thanks for posting this. It's great to see people doing real formal analysis of the TLS 1.3 draft; this is really helpful in guiding the design."

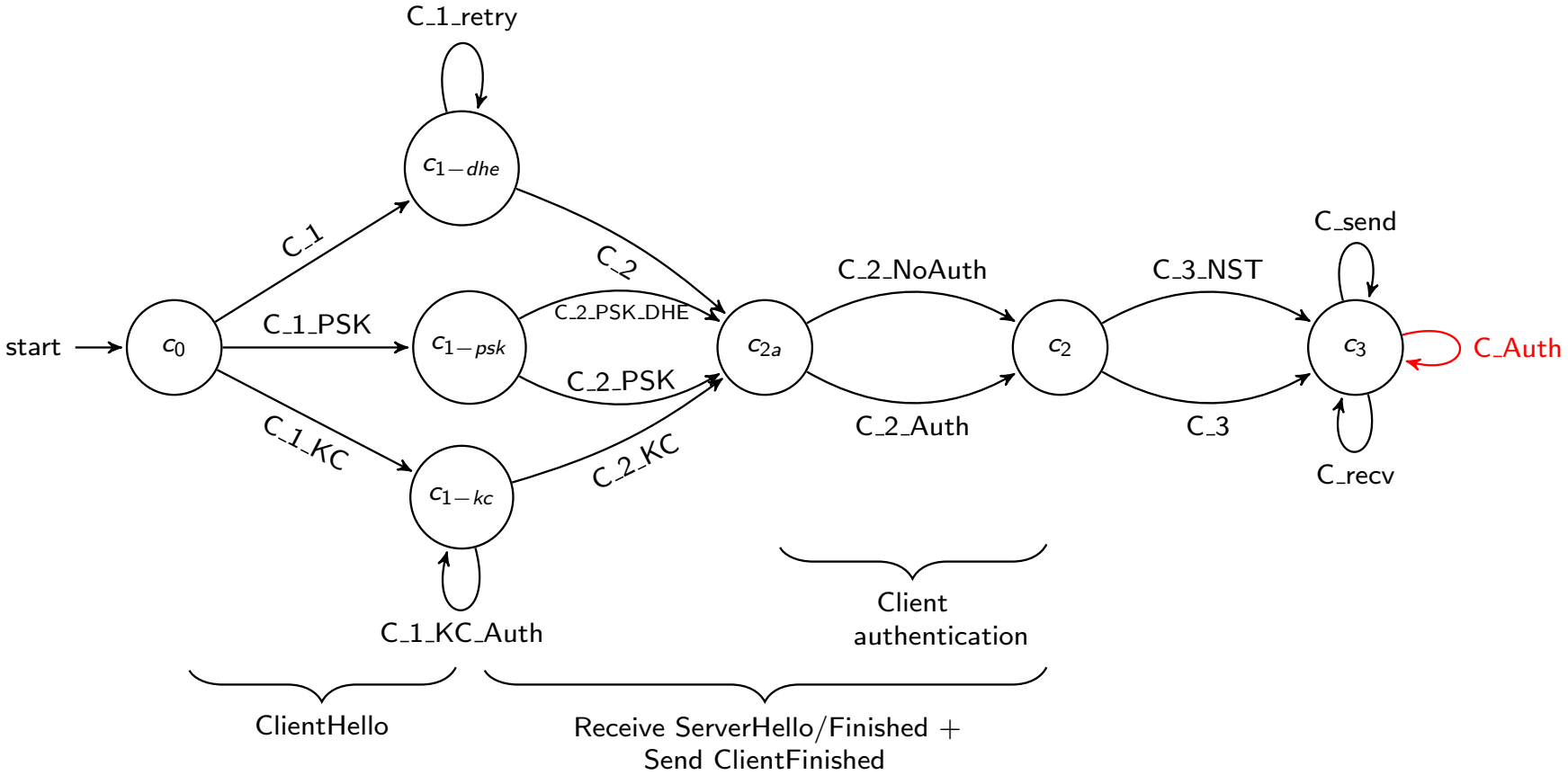
"This result motivates and confirms the need to modify the handshake hashes to contain the server Finished when we add post-handshake authentication as is done in PR#316..."

Although already included in PR316, the attack highlights strict necessity of including the `Finished` message in the session hash and draft 11 does this - creates a binding between signature and the session

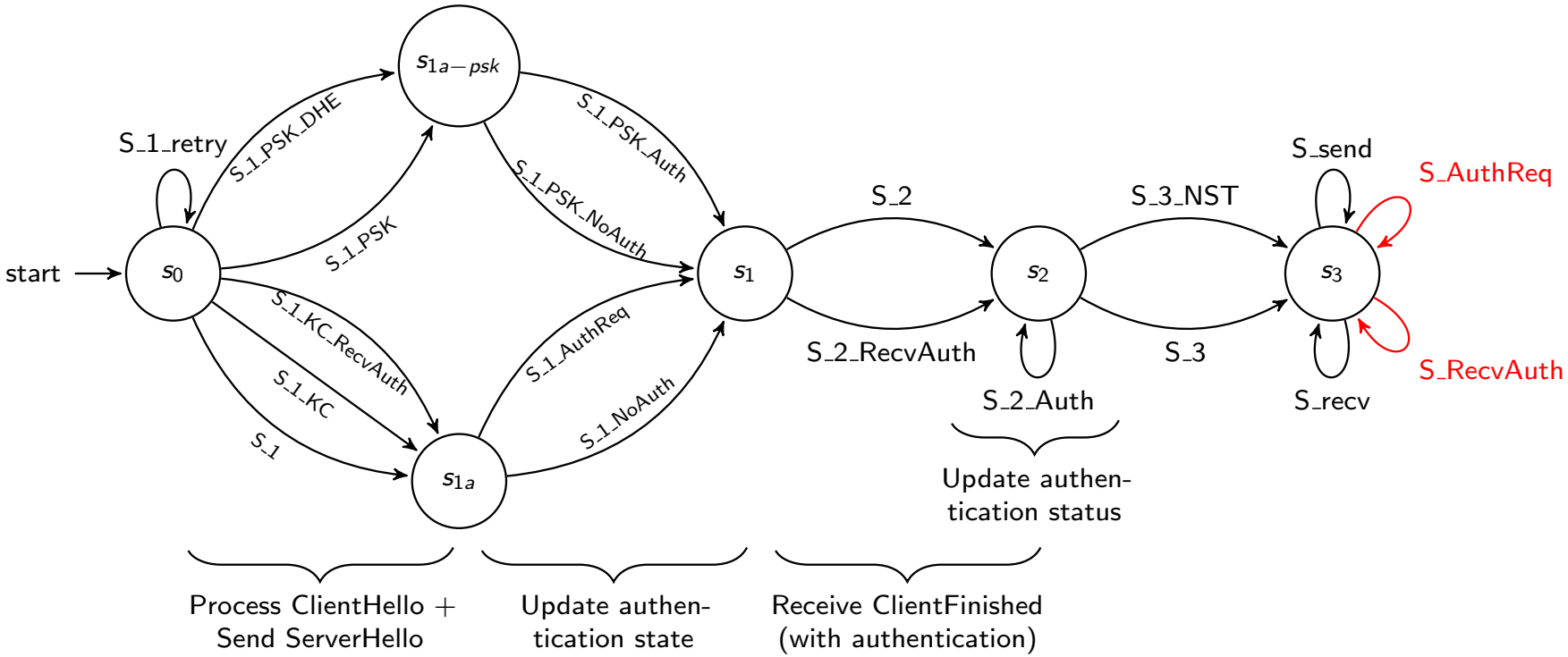
Where we stand

- We show that draft 10 meets the goals of authenticated key exchange, and the changes in draft 11 appear to address the attack
- First comprehensive analysis of the interaction of the new TLS 1.3 modes
  - we confirmed the base design is solid
  - prevented a potential weakness
- Draft 11...

Where we stand



Where we stand



Half-way through...

Where we stand

Making changes

- The model can be updated to accommodate changes
- Additions can complicate the analysis (adding loops)
- Reductions/simplifications are easier to handle
- Changes mean more man-hours

Making changes

- The model can be updated to accommodate changes
- Additions can complicate the analysis (adding loops)
- Reductions/simplifications are easier to handle
- Changes mean more man-hours

http://tls13tamarin.github.io/TLS13Tamarin/

Authors:

Cas Cremers
cas.cremers@cs.ox.ac.uk

Marko Horvat
marko.horvat@cs.ox.ac.uk

Sam Scott
sam.scott.2012@live.rhul.ac.uk

Thyla van der Merwe
thyla.vandermerwe.2012@live.rhul.ac.uk