Mathematical Models, Analysis Tools, and Internet Security

FOSAD 2016

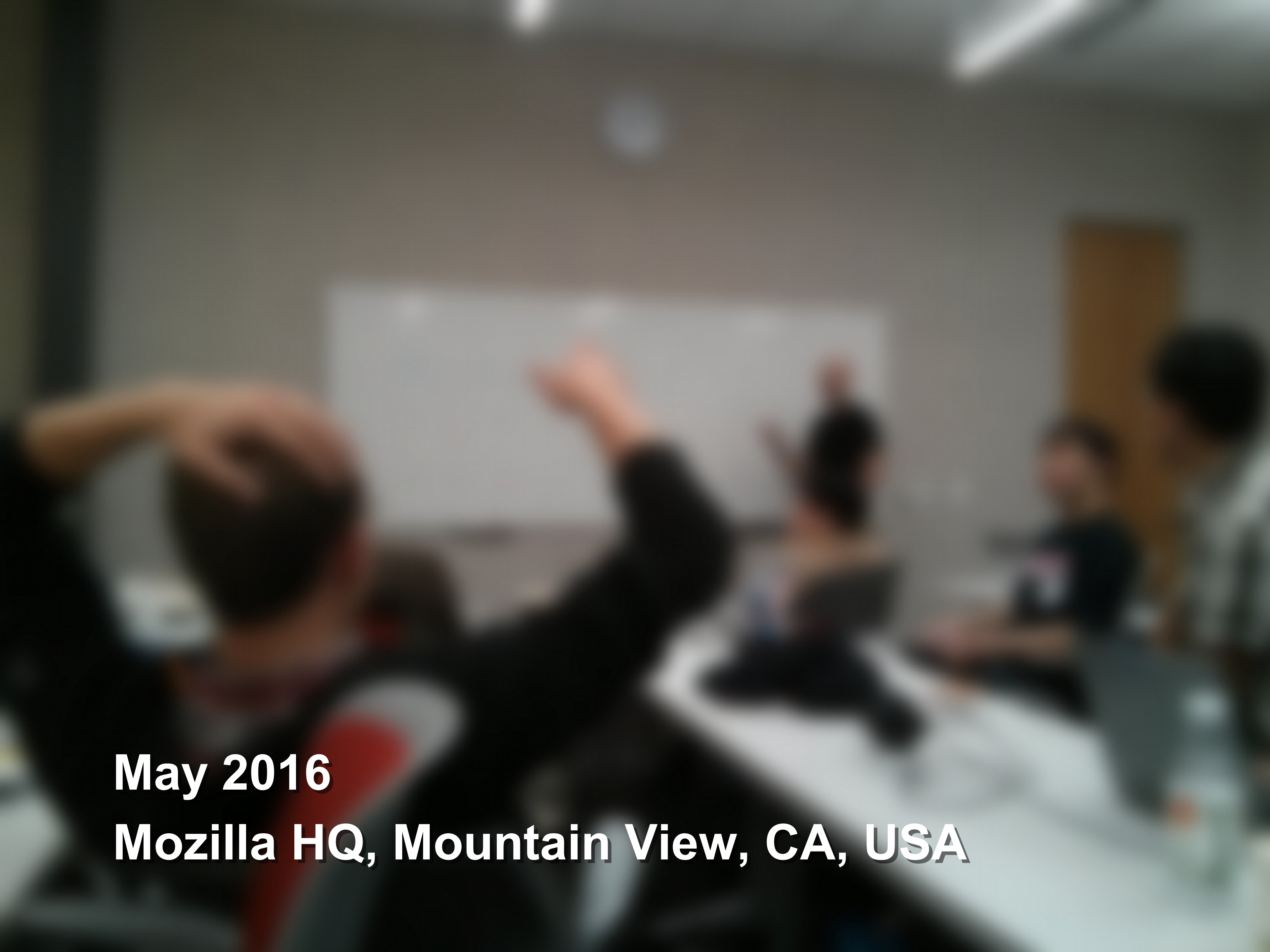
Cas Cremers
University of Oxford

Information Security Group in Oxford

- Upcoming academic year:
  - 6/7 PhD students, perhaps one PostDoc
- **Theory**
  - Mathematical models of what security is
  - Symbolic and computational approaches, as well as bridging work between them
- **Methodology and proofs**
- **Tools**
  - Scyther, Scyther-proof, Tamarin, ...
- **Applications**
  - Not just toy examples!

Today and tomorrow

- **Modeling, Automated tools, and Internet Security**
  - Focus on symbolic methods
  - Historical perspective
  - Why we built some tools and what happened
  - From theory and toy examples to real-world practice over the years



May 2016

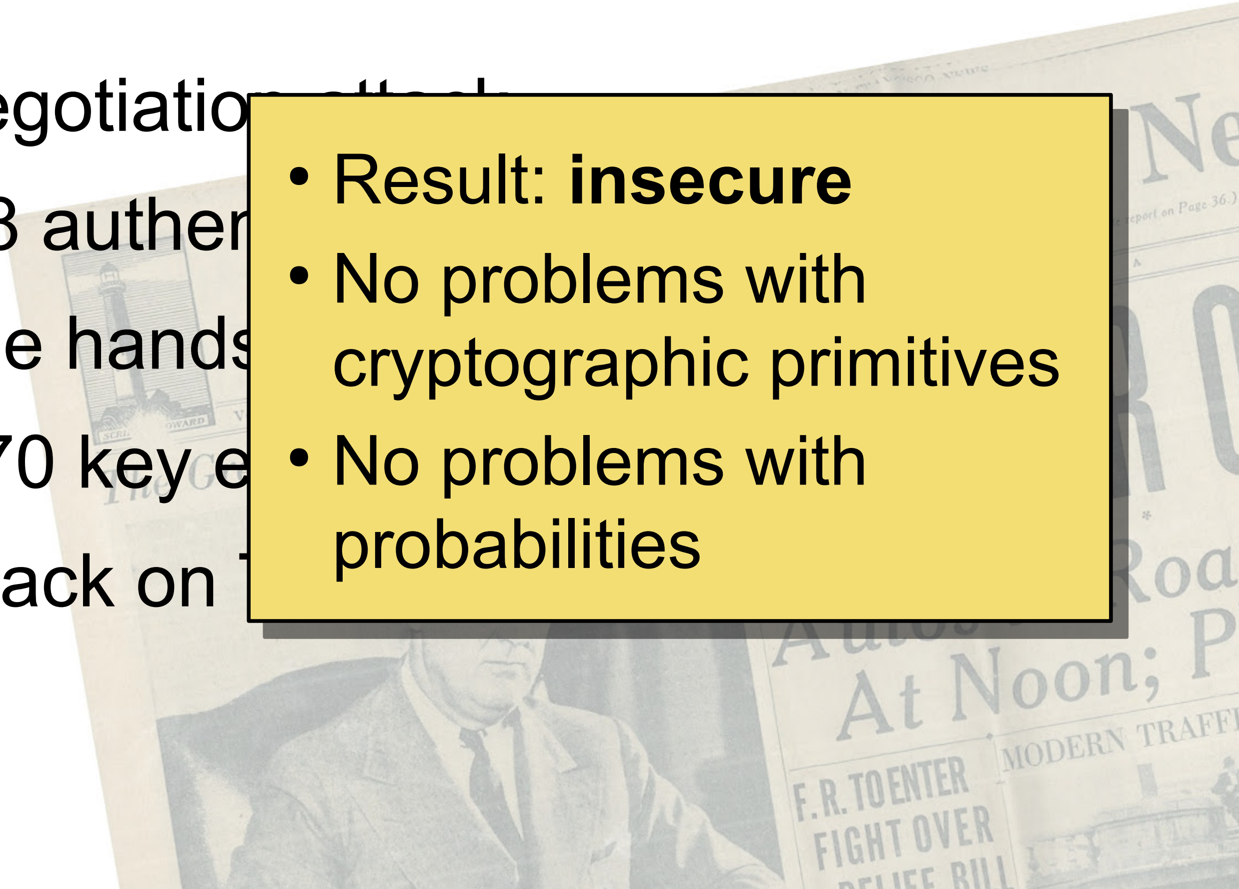
Mozilla HQ, Mountain View, CA, USA

2004

What is the problem?

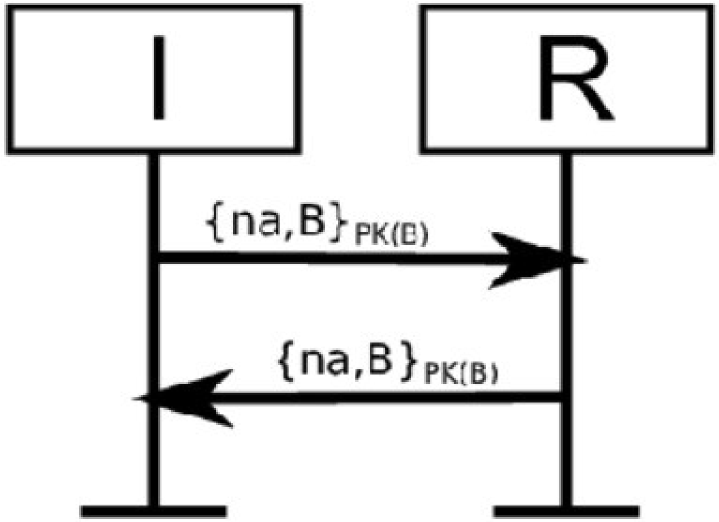
99 problems...

- 2003: PKCS#11 crypto API attacks
- 2008: Google single-sign on protocol (SAML) attack
- 2009: TLS renegotiatio
- 2012: ISO 9798 authen
- 2014: TLS Triple hands
- 2014: ISO 11770 key e
- 2015: Freak attack on T
- Etc etc

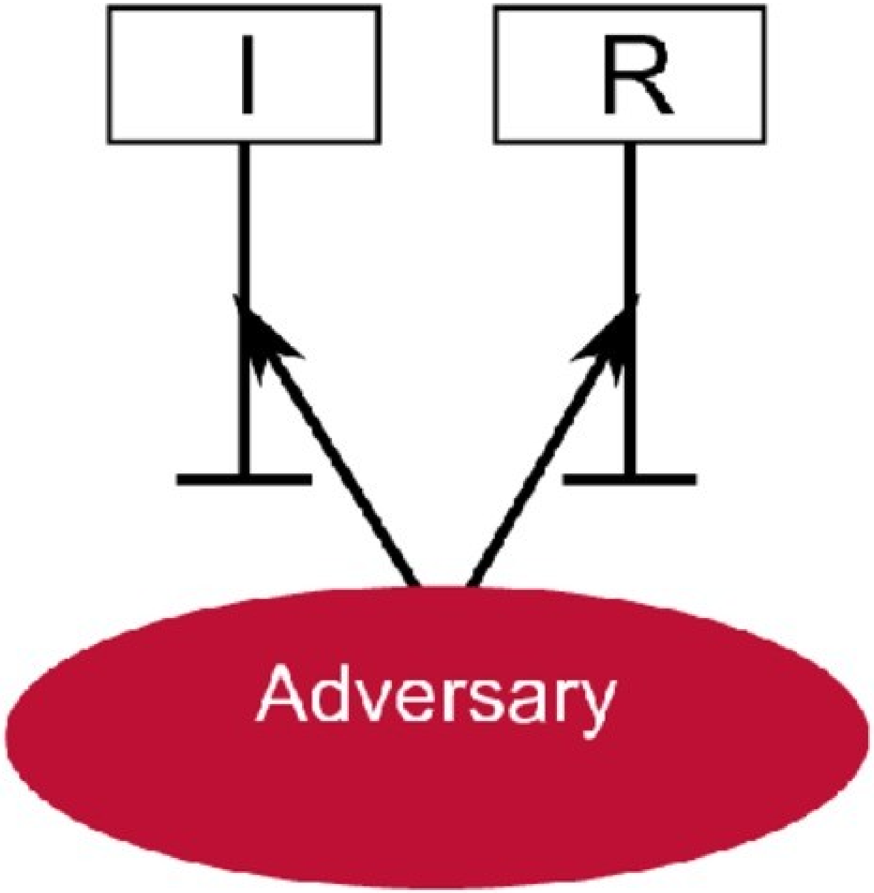


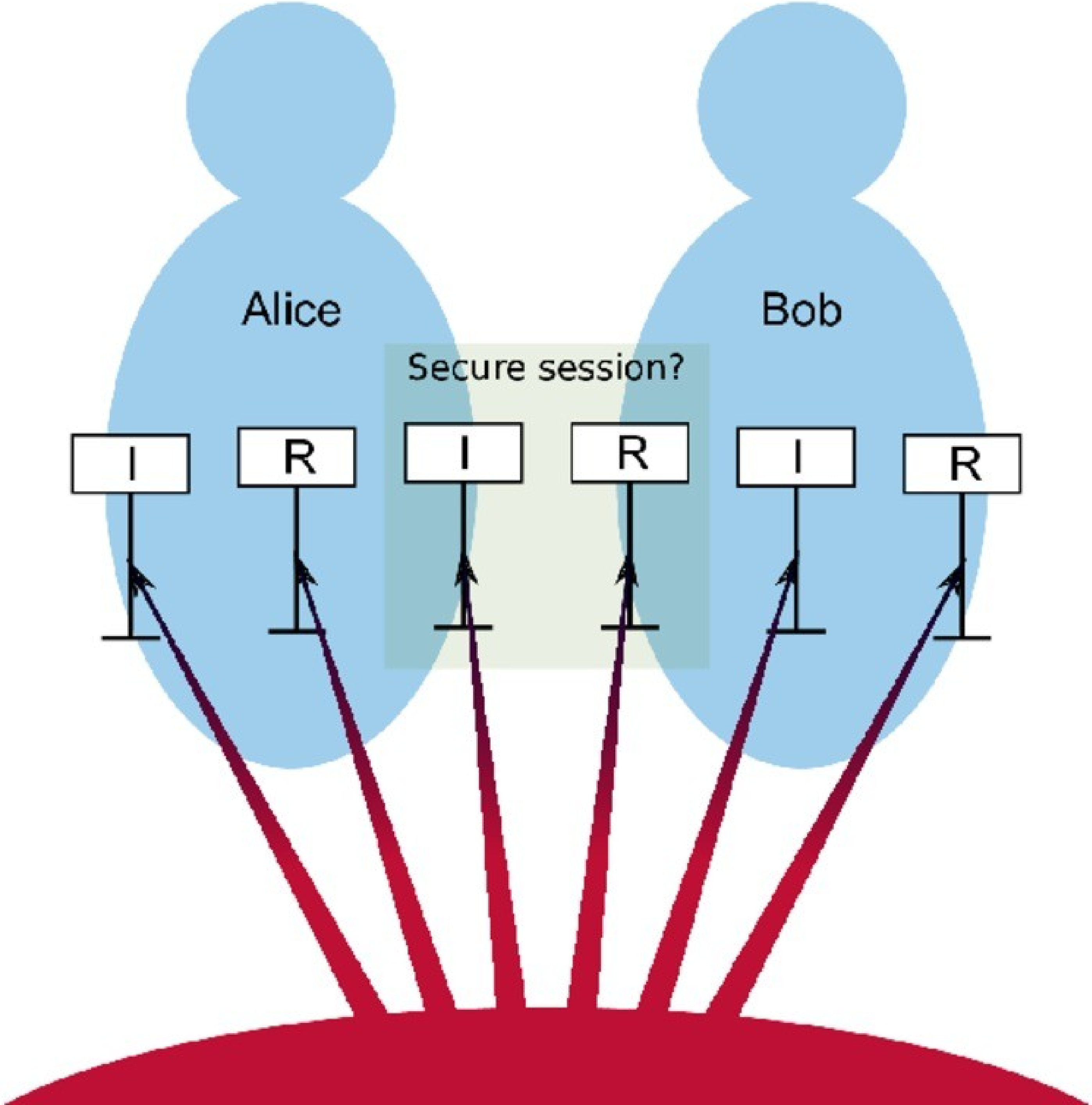
- Result: **insecure**
- No problems with cryptographic primitives
- No problems with probabilities

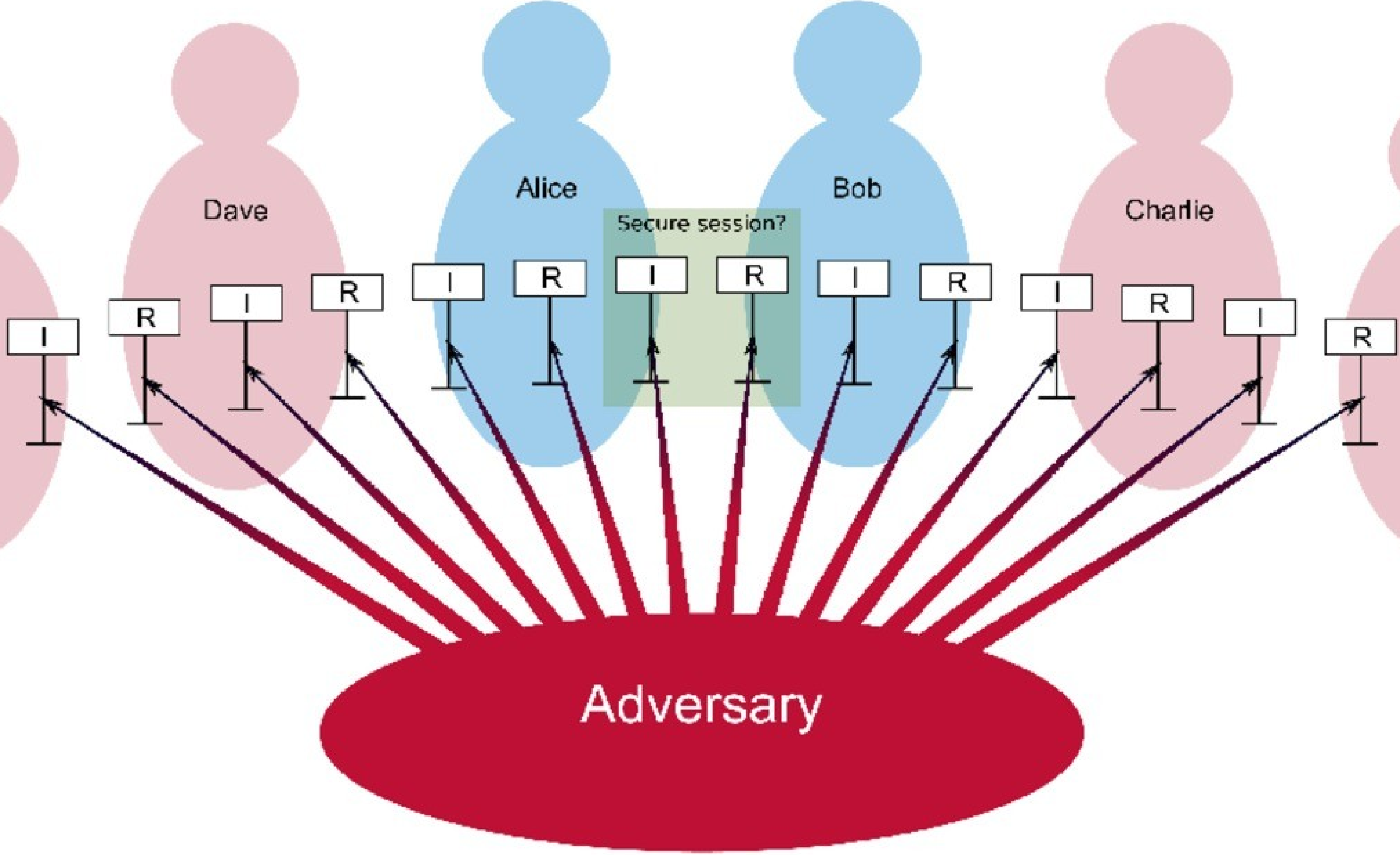
Secure session?



Secure session?




Alice
Bob
Secure session?
I
R
I
R
I
R


Dave
Alice
Secure session?
Bob
Charlie
I
R
I
R
I
R
I
R
I
R
I
R
I
R
I
R
Adversary

Internet Key Exchange (IKE, in IPv6)

“IKE is fairly complicated; to fully understand it, it’s helpful to possess ***multiple advanced degrees in mathematics and cryptography*** and to have ***copious amounts of spare time*** to read many detailed yet highly valuable resources.”

Microsoft TechNet: *How IPsec works*

Source: `http://technet.microsoft.com/en-us/library/cc512617.aspx`

(Retrieved in 2011 and again on August 29, 2016)

Example IKE exchange

$$\begin{array}{llll}
1. & A \rightarrow B: & \mathsf{HDR}_1, & \mathsf{SA}, g^{x_A}, N_A, ID_A \\
2. & B \rightarrow A: & \mathsf{HDR}_2, & \mathsf{SA}', g^{x_B}, N_B, ID_B, \\
 & & & \{\mathsf{prf}_K(g^{x_B}, g^{x_A}, CKY_B, CKY_A, ID_B)\}_{sk(B)} \\
3. & A \rightarrow B: & \mathsf{HDR}_3, & \{\mathsf{prf}_K(g^{x_A}, g^{x_B}, CKY_A, CKY_B, ID_A)\}_{sk(A)}
\end{array}$$

$$\text{where } K = \mathsf{prf}_{(N_A,N_B)}(g^{x_A x_B}).$$

IKEv1 Aggressive Mode with digital signatures

$$
\begin{array}{llll}
1. & A \rightarrow B: & \mathrm{HDR}_1, & SA, g^{x_A}, N_A, ID_A \\
2. & B \rightarrow A: & \mathrm{HDR}_2, & SA, g^{x_B}, N_B, ID_B, \\
 & & & \{prf_K(g^{x_B}, g^{x_A}, CKY_B, CKY_A, ID_B)\}_{sk(B)} \\
3. & A \rightarrow B: & \mathrm{HDR}_3, & \{prf_K(g^{x_A}, g^{x_B}, CKY_A, CKY_B, ID_A)\}_{sk(A)}
\end{array}
$$

IKEv1 Main Mode with digital signatures

IKEv1 Aggressive Mode with digital signatures	**IKEv1 Main Mode with digital signatures**
IKEv1 Aggressive Mode with Pre-shared keys	**IKEv1 Main Mode with Pre-shared keys**
IKEv1 Aggressive Mode with Public keys	**IKEv1 Main Mode with Public keys**
IKEv1 Aggressive Mode with Public keys (2)	**IKEv1 Main Mode with Public keys (2)**

Note: some minor variants omitted!

Phase 1

IKEv1 Aggressive Mode with digital signatures	IKEv1 Main Mode with digital signatures
IKEv1 Aggressive Mode with Pre-shared keys	IKEv1 Main Mode with Pre-shared keys
IKEv1 Aggressive Mode with Public keys	IKEv1 Main Mode with Public keys
IKEv1 Aggressive Mode with Public keys (2)	IKEv1 Main Mode with Public keys (2)

Phase 2

IKEv1 Quick Mode	IKEv1 Quick Mode without PFS
IKEv1 Quick Mode without Identity	

Note: some minor variants omitted!

IKEv1

Phase 1

IKEv1 Aggressive Mode with digital signatures	IKEv1 Main Mode with digital signatures
IKEv1 Aggressive Mode with Pre-shared keys	IKEv1 Main Mode with Pre-shared keys
IKEv1 Aggressive Mode with Public keys	IKEv1 Main Mode with Public keys
IKEv1 Aggressive Mode with Public keys (2)	IKEv1 Main Mode with Public keys (2)

Phase 2

IKEv1 Quick Mode	IKEv1 Quick Mode without PFS
IKEv1 Quick Mode without Identity	

Note: some minor variants omitted!

IKEv2

Phase 1

IKEv2 SIG	IKEv2 SIG noid
IKEv2 MAC	IKEv2 MAC noid
IKEv2 EAP	IKEv2 EAP noid
IKEv2 SIG/MAC asymmetric variants	IKEv2 SIG/MAC asymmetric variants
IKEv2 SIG/MAC asymmetric variants	IKEv2 SIG/MAC asymmetric variants

Phase 2

IKEv2 child mode	IKEv2 child mode without PFS

Modern adversary/threat models

- Adversary can
  - learn **long-term keys**,
  - learn the **randomness** generated in sessions,
  - learn **session keys**
  - learn (part of) the **session state**
- Security guarantee holds for all **clean** sessions
  - A complex condition that involves:
    - All other sessions
    - Checking partial authentication
    - Temporal ordering of events

Can tools help out?

2006

Scyther (Cremers, 2006)

- Focusses on event structures
- Does **not use abstraction**
  - Never finds ``false'' attacks
- Input language: domain-specific language (SPDL)
  - Linear role scripts

Basis: Dolev Yao adversary model

- Models an active intruder with full network control and perfect recall

- Idealized black-box cryptography

$$\frac{t \in M}{M \vdash t} \qquad \frac{M \vdash t_1 \qquad M \vdash t_2}{M \vdash (t_1, t_2)} \qquad \frac{M \vdash (t_1, t_2)}{M \vdash t_1} \qquad \frac{M \vdash (t_1, t_2)}{M \vdash t_2}$$

$$\frac{M \vdash t}{M \vdash hash(t)} \qquad \frac{M \vdash t_1 \qquad M \vdash t_2}{M \vdash \{t_1\}_{t_2}} \qquad \frac{M \vdash \{t_1\}_{t_2} \qquad M \vdash t_2^{-1}}{M \vdash t_1}$$

Successful: interesting theory and powerful tools

Terms, roles, and protocols

- **Terms**: operators for constructing cryptographic messages

$$\mathit{Term} \quad ::= \mathit{Agent} \mid \mathit{Fresh} \mid \mathit{Var} \mid (\mathit{Term}, \mathit{Term}) \mid \{\mathit{Term}\}_{\mathit{Term}} \mid \ldots$$

- **Roles:** sequences of **agent events**

$$\begin{aligned} \mathit{AgentEvent} ::= {} & \mathsf{create}(\mathit{Role}, \mathit{Agent}) \mid \\ & \mathsf{send}(\mathit{Agent}, \mathit{Agent}, \mathit{Term}) \mid \mathsf{recv}(\mathit{Agent}, \mathit{Agent}, \mathit{Term}) \end{aligned}$$

- **Example**

$$\begin{aligned} I \rightarrow R : &\quad \{I, K\}_{K_{IR}} \\ R \rightarrow I : &\quad \{R, M\}_{K} \end{aligned}$$

⇒

$$P(I) = [\quad \mathsf{send}(I, R, \{I, K\}_{k(I,R)}); \; \mathsf{recv}(R, I, \{R, y\}_{K}) \quad]$$

$$P(R) = [\quad \mathsf{recv}(I, R, \{I, x\}_{k(I,R)}); \; \mathsf{send}(R, I, \{R, M\}_{x}) \quad]$$

Threads

- A **thread** is a role instance (local session)
  - No limit to number of threads
  - Each thread assigned a unique identifier from the set TID.
  - We **instantiate names** and **syntactically bind** fresh **values** and **variables** to their owning thread, e.g. K#1, y#1

$$P(I) = [\ \begin{array}{l} \mathsf{generate}(\{K\}); \\ \mathsf{send}(I, R, \{I, K\}_{k(I,R)}); \\ \mathsf{recv}(R, I, \{R, y\}_{K}) \end{array}\ ] \xrightarrow{\begin{array}{c} \text{Thread 1} \\ I \mapsto A \\ R \mapsto B \end{array}} [\ \begin{array}{l} \mathsf{generate}(\{K\sharp 1\}); \\ \mathsf{send}(A, B, \{A, K\sharp 1\}_{k(A,B)}); \\ \mathsf{recv}(B, A, \{B, y\sharp 1\}_{K\sharp 1}) \end{array}\ ]$$

- For currently active threads, we store the remaining sequence of steps in a thread pool *th* : $TID \nrightarrow AgentEvent^*$

Core symbolic model

(slightly simplified)

$$A \rightarrow B \;:\; n$$

- **State** (tr,IK,th)
  - *tr* : trace of events that have occurred
  - *IK* : “intruder knowledge” of adversary, initially IK_0
  - *th* : thread pool, mapping thread identifiers to remaining steps
- **Transition system** modeling agents' threads and adversary

$$\frac{th(tid) = \langle \mathrm{send}(m) \rangle \hat{}\, l}{(tr, IK, th) \longrightarrow (tr \hat{}\, \langle (tid, \mathrm{send}(m)) \rangle, IK \cup \{m\}, th[l \leftarrow\!\shortmid\, tid])}\,[\mathrm{send}]$$

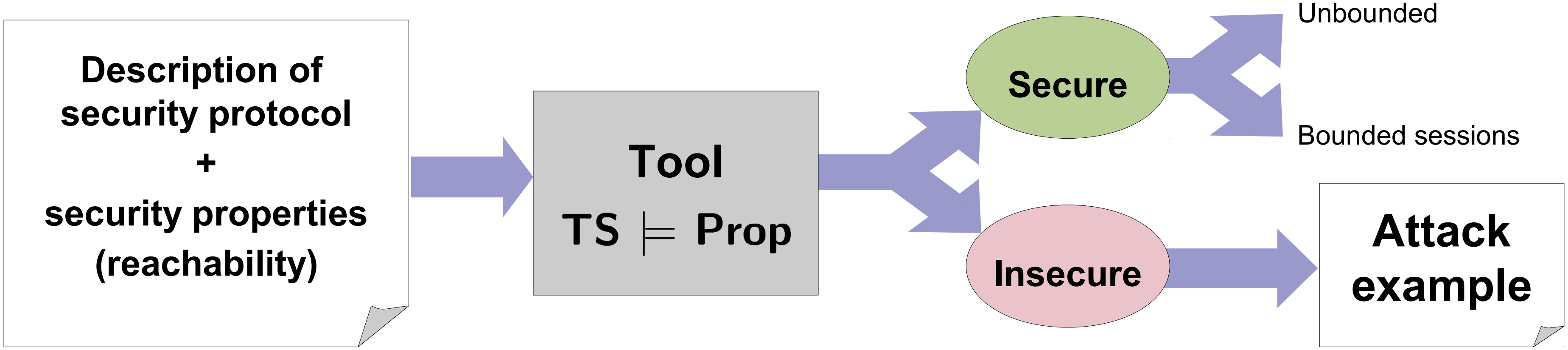
$$\frac{th(tid) = \langle \mathrm{recv}(pt) \rangle \hat{}\, l \qquad IK \vdash \sigma(pt) \qquad dom(\sigma) = FV(pt)}{(tr, IK, th) \longrightarrow (tr \hat{}\, \langle (tid, \mathrm{recv}(\sigma(pt))) \rangle, IK, th[\sigma(l) \leftarrow\!\shortmid\, tid])}\,[\mathrm{recv}]$$

Example of reachable state:

$$(\underbrace{\langle (1, \mathrm{send}(A, B, n\sharp 1)) \rangle}_{tr}, \underbrace{IK_0 \cup \{n\sharp 1\}}_{IK}, \underbrace{\{1 \mapsto \langle\rangle, 2 \mapsto \langle \mathrm{recv}(A, B, X\sharp 2) \rangle\}}_{th})$$

Reasoning about protocol semantics (TS)

- General complexity
  - **Reachability** properties are **undecidable**, e.g. secrecy
    (Durgin, Lincoln, Mitchell, Scedrov 1999)
  - **NP-hard,** even when number of sessions is bounded
    (Rusinowitch, Turuani, 1999)
- **Scyther tool** often successful in protocol analysis



DEMO

2008

Scyther pros and cons

- Pros
  - Unbounded analysis by backwards search
    - no bound on the number of possible threads in attacks
  - Fast, push-button
  - Many case studies
  - Support for different adversary models

- Cons
  - Linear role scripts
    - No if/then
    - No loops within protocol
  - No good support for equational theories
  - No mutable global state
  - Fixed set of security properties

The Tamarin Prover

The Tamarin Prover

Family of small monkeys in South America

Choice: Emperor Tamarin

Important: Not
near extiction

Joint work with:

**Simon
Meier**

**Benedikt
Schmidt**

**David
Basin**

Tamarin prover: History

- Idea: **generalize Scyther**'s approach
  - Better support for Diffie-Hellman
  - Loops, branches
  - Property specification
- From vague idea to theory to tool between 2008 and 2012
  - Simon and Benedikt: vast majority of the development
  - Cedric Staub worked on the GUI
  - Many people involved in models
  - Several person years of work

2011

The ISO/IEC 9798 Standard

- Entity Authentication Mechanisms

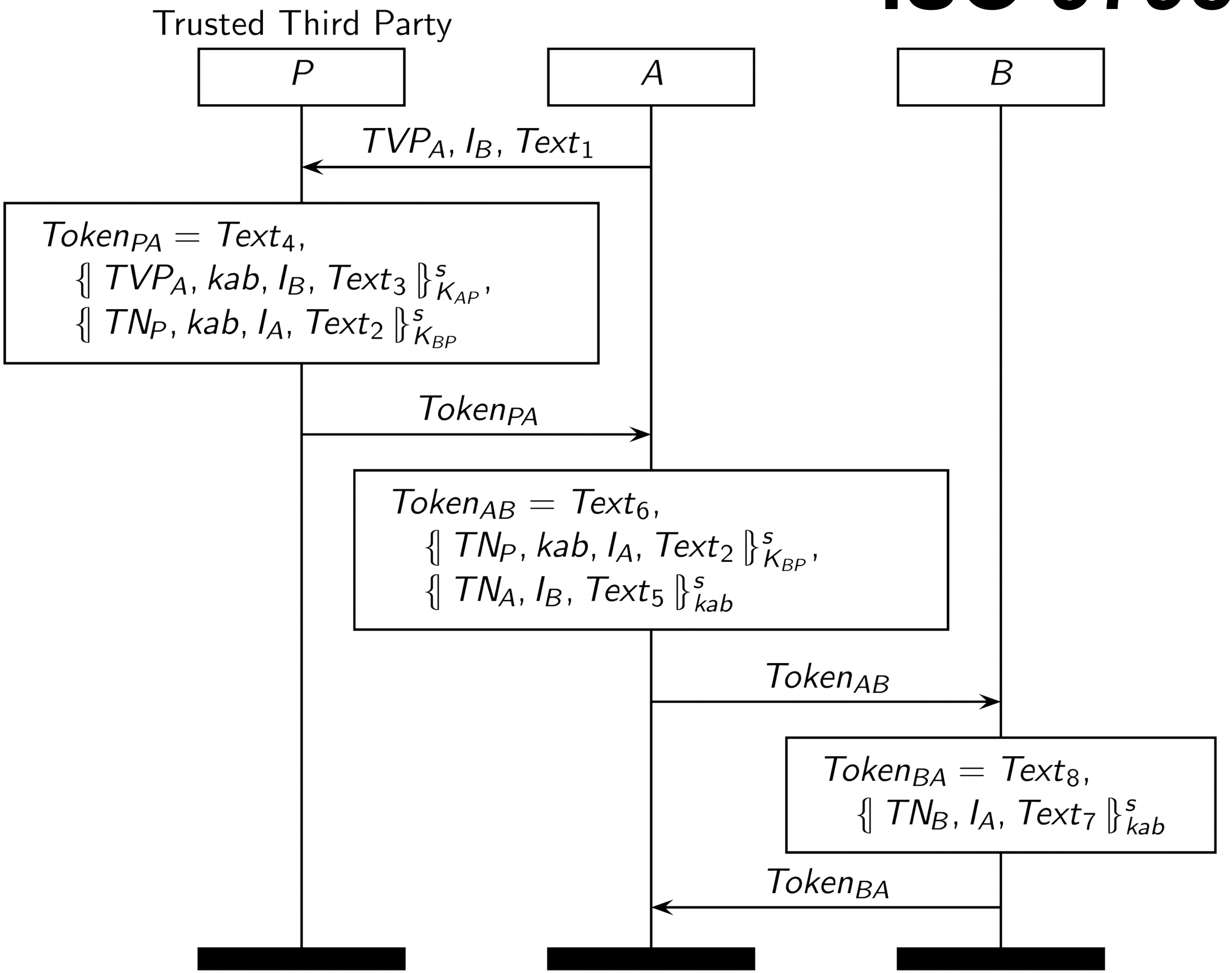
- **18 base protocols**
  - Symmetric-key encryption,
    Digital signatures,
    Cryptographic check functions
  - Unilateral or Mutual authentication
  - Additional protocols with TTP
- **Further variants** from optional fields

The ISO/IEC 9798 Standard

- History
  - Active development and updates since 1991
  - Blueprints for protocol design
  - Basis for ISO 11770 (Key Exchange) and NIST FIPS 196
  - Mandated by other standards
    - e.g. European Banking Commission's smart card standards

- Intended properties
  - Entity authentication?
  - E.g. Resistant to reflection attacks
  - Encrypted/signed payloads?

ISO 9798-2-5



Trusted Third Party

P

A

B

$TVP_A, I_B, Text_1$

$$Token_{PA} = Text_4, \{\!| TVP_A, kab, I_B, Text_3 |\!\}^s_{K_{AP}}, \{\!| TN_P, kab, I_A, Text_2 |\!\}^s_{K_{BP}}$$

$Token_{PA}$

$$Token_{AB} = Text_6, \{\!| TN_P, kab, I_A, Text_2 |\!\}^s_{K_{BP}}, \{\!| TN_A, I_B, Text_5 |\!\}^s_{kab}$$

$Token_{AB}$

$$Token_{BA} = Text_8, \{\!| TN_B, I_A, Text_7 |\!\}^s_{kab}$$

$Token_{BA}$

Analysis

- Request by CryptRec to evaluate standard

  CRYPTREC
  Cryptography Research and Evaluation Committees

  - Cryptography Research and Evaluation Committees
  - Funded by the Japanese government
  - Part of long-running program to evaluate cryptographic mechanisms

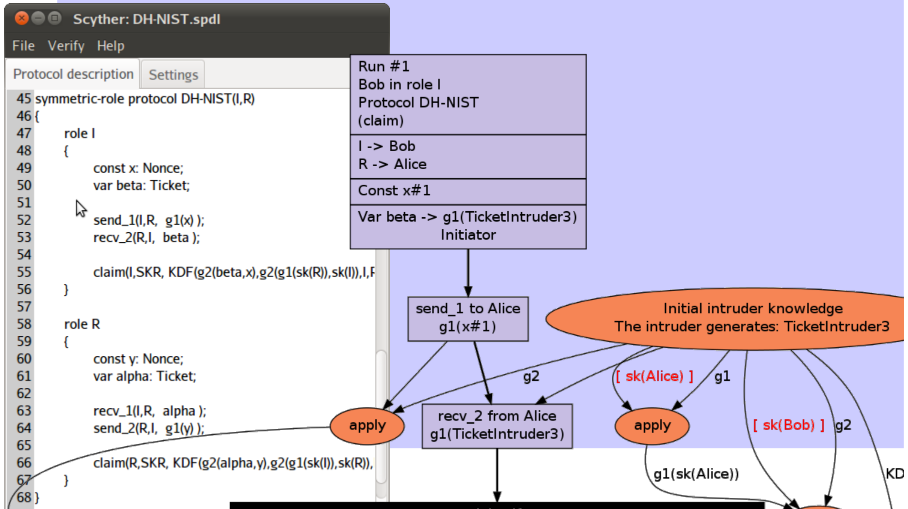
- Confirmation expected
  - Standard has been improved since 1994
  - Multiple previous analysis

Tools used

Scyther

Symbolic analysis of security protocols

- Falsification (attack finding)
- Unbounded verification



```
symmetric-role protocol DH-NIST(I,R)
{
        role I
        {
                const x: Nonce;
                var beta: Ticket;

                send_1(I,R,  g1(x) );
                recv_2(R,I,  beta );

                claim(I,SKR, KDF(g2(beta,x),g2(g1(sk(R)),sk(I)),I,F
        }

        role R
        {
                const y: Nonce;
                var alpha: Ticket;

                recv_1(I,R,  alpha );
                send_2(R,I,  g1(y) );

                claim(R,SKR, KDF(g2(alpha,y),g2(g1(sk(I)),sk(R)),
        }
}
```

Scyther-proof

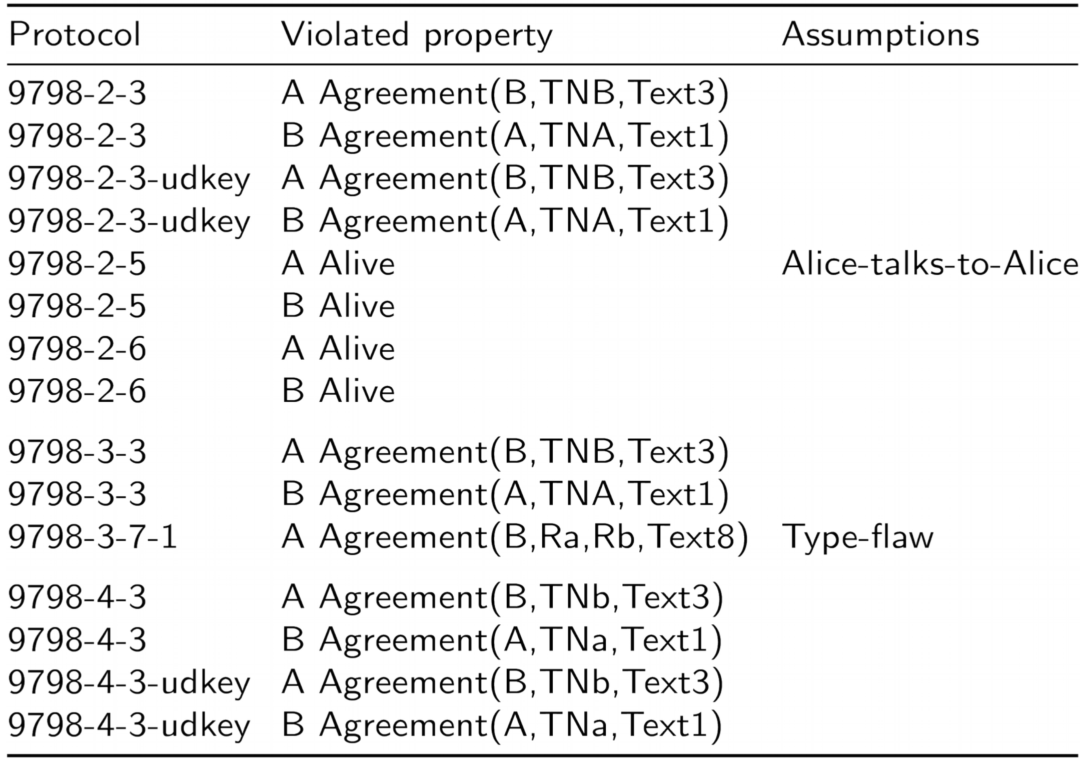
- Embedding of protocol semantics and protocol-independent invariant in the **Isabelle/HOL** theorem prover
- Algorithm similar to Scyther that **outputs proof script** for Isabelle/HOL
- Independent verifiability

Results

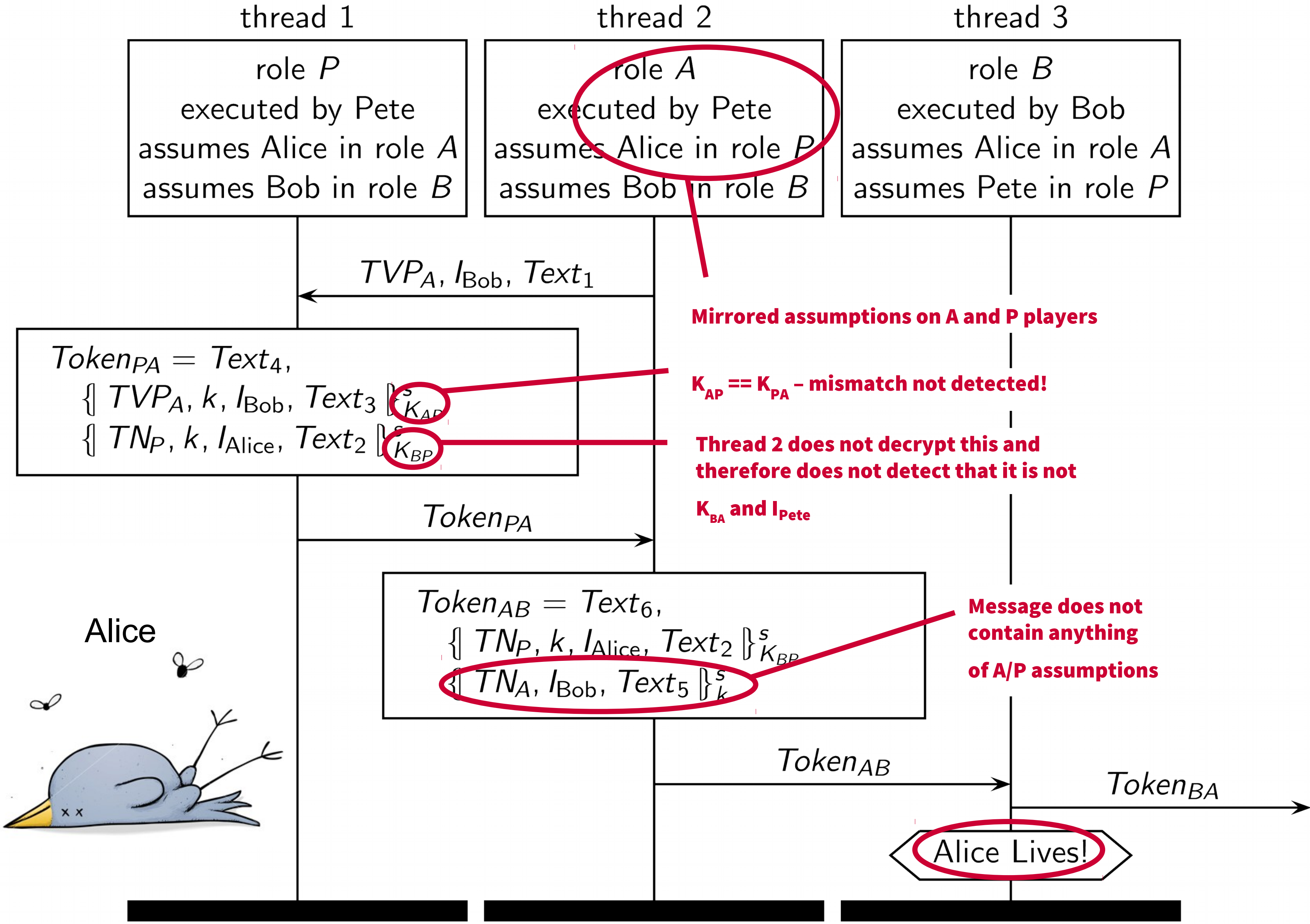
- No strong authentication properties

  Aliveness < Agreement < Synchronisation

- Under some conditions no authentication



Protocol	Violated property	Assumptions
9798-2-3	A Agreement(B,TNB,Text3)	
9798-2-3	B Agreement(A,TNA,Text1)	
9798-2-3-udkey	A Agreement(B,TNB,Text3)	
9798-2-3-udkey	B Agreement(A,TNA,Text1)	
9798-2-5	A Alive	Alice-talks-to-Alice
9798-2-5	B Alive	
9798-2-6	A Alive	
9798-2-6	B Alive	
9798-3-3	A Agreement(B,TNB,Text3)	
9798-3-3	B Agreement(A,TNA,Text1)	
9798-3-7-1	A Agreement(B,Ra,Rb,Text8)	Type-flaw
9798-4-3	A Agreement(B,TNb,Text3)	
9798-4-3	B Agreement(A,TNa,Text1)	
9798-4-3-udkey	A Agreement(B,TNb,Text3)	
9798-4-3-udkey	B Agreement(A,TNa,Text1)	



thread 1

role P
executed by Pete
assumes Alice in role A
assumes Bob in role B

thread 2

role A
executed by Pete
assumes Alice in role P
assumes Bob in role B

thread 3

role B
executed by Bob
assumes Alice in role A
assumes Pete in role P

$TVP_A, I_{\text{Bob}}, Text_1$

$Token_{PA} = Text_4,$
$\{\!| TVP_A, k, I_{\text{Bob}}, Text_3 |\!\}^s_{K_{AP}}$
$\{\!| TN_P, k, I_{\text{Alice}}, Text_2 |\!\}^s_{K_{BP}}$

$Token_{PA}$

$Token_{AB} = Text_6,$
$\{\!| TN_P, k, I_{\text{Alice}}, Text_2 |\!\}^s_{K_{BP}}$
$\{\!| TN_A, I_{\text{Bob}}, Text_5 |\!\}^s_k$

$Token_{AB}$

$Token_{BA}$

Alice Lives!

Alice

Mirrored assumptions on A and P players

$K_{AP} == K_{PA}$ – mismatch not detected!

Thread 2 does not decrypt this and therefore does not detect that it is not

K_{BA} and I_{Pete}

Message does not contain anything

of A/P assumptions

Root Causes of the Problems

- Message format is **consistent** and minimal
  - Good design individually, but leads to possible confusion between different messages
- **No type information** for fields
  - Combined with above, can lead to type flaw attacks
- Identity of **one agent** always included to break symmetry of shared keys
  - Great but doesn't work for three parties

Repairing ISO/IEC 9798

- We proposed **fixes** and **machine-checked correctness proofs**
  - Fixes do not require additional cryptography

- **Scyther-proof** generates proof scripts for Isabelle-HOL
  - Minor extension over original [CSF2011] developed for bidirectional keys

- Proofs even guarantee correctness when executing all ISO 9798 protocols in parallel
  - Exclude multi-protocol attacks

Effort

- Modeling effort: a couple of weeks
  - Abstraction level of standard close to formal models
  - Some iteration inevitable after initial analysis with scyther
- Generating proof scripts using Scyther-proof
  - 20 seconds
- Checking correctness in Isabelle/HOL
  - 3 hours (correctness for all protocols in parallel)

ISO/IEC 9798: Conclusions

- Improving the standard
  - Old version: **only weak authentication**, sometimes none
  - Succesful interaction between researchers and standardization committee:
  - **New version of the standard** has been released which guarantees **strong authentication** (synchronisation)
  - Machine-checked symbolic proofs of standard
- We later similarly tackled ISO/IEC 11770

2012

Tamarin: model

- **Term algebra**
  - enc(_,_), dec(_,_), h(_,_), _^_, $\_^{-1}$, _*_, 1, …
- **Equational theory**
  - $dec(enc(m,k),k) =_E m$,
  - $(x\text{^}y)\text{^}z =_E x\text{^}(y*z)$,
  - $(x^{-1})^{-1} =_E x$, …
- **Facts**
  - F(t1,…,tn)

- **Transition system**
  - State: multiset of facts
  - Rules: l –[a]→ r
- **Tamarin-specific**
  - Built-in Dolev-Yao attacker rules
    - In( ), Out( ), K( )
  - Special **Fresh** rule:
    - [] --[]--> [Fr(**x**)]
      - With additional constraints on systems such that **x** unique

Semantics

- **Transition relation**

  $S \xrightarrow{[a]} ((S \setminus^{\#} l) \cup^{\#} r)$

  where $l \xrightarrow{[a]} r$ is a ground instance of a rule and $l \subseteq^{\#} S$

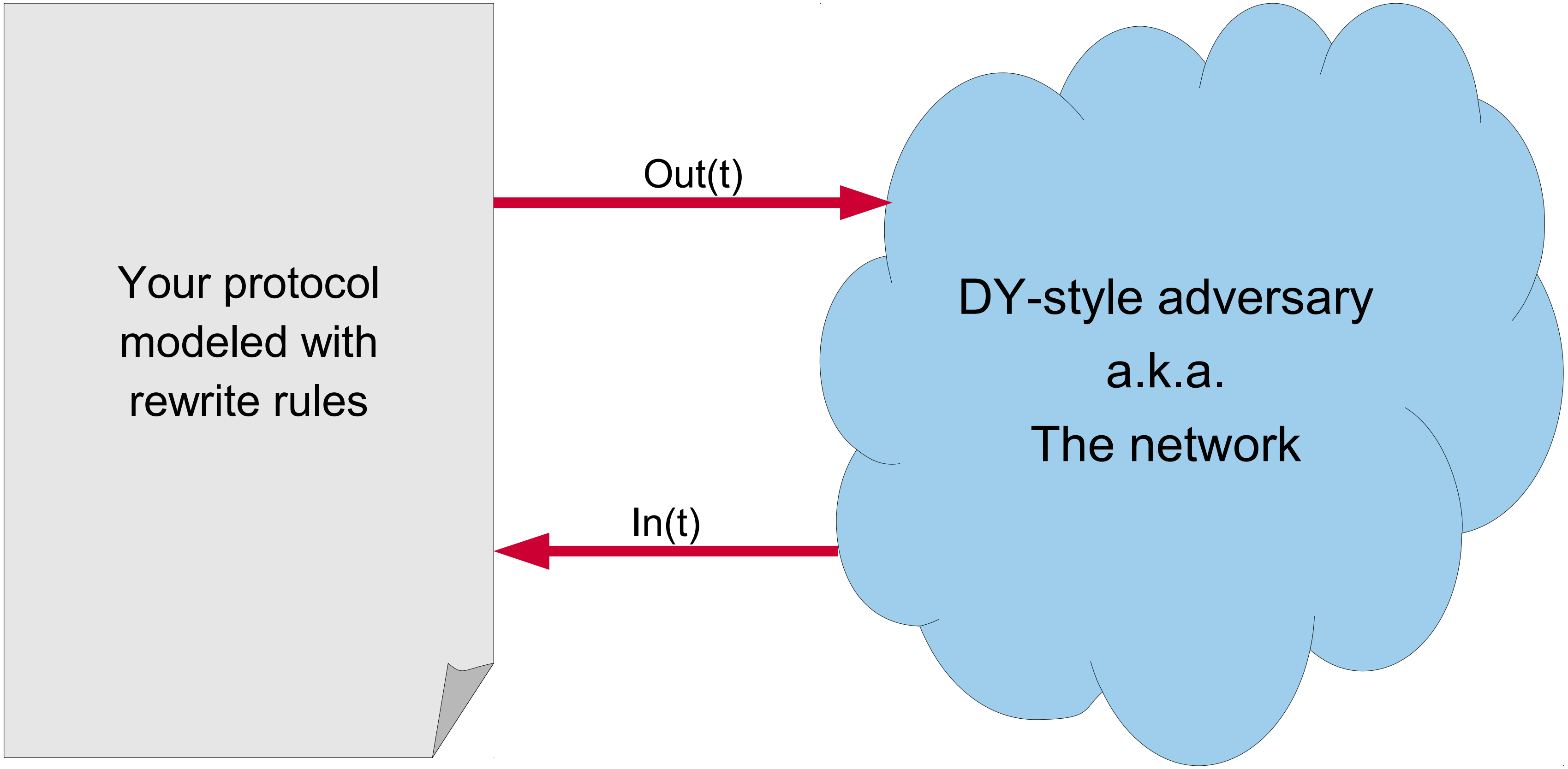
- **Executions**

  $\mathrm{Exec}(R) = \{ \emptyset \xrightarrow{[a_1]} \ldots \xrightarrow{[a_n]} S_n$
  $\mid \forall n\,.\,\mathrm{Fr}(n)$ appears only once on rhs $\}$

- **Traces**

  $\mathrm{Traces}(R) = \{ [a_1,\ldots,a_n]$
  $\mid \emptyset \xrightarrow{[a_1]} \ldots \xrightarrow{[a_n]} S_n \in \mathrm{Exec}(R) \}$

Tamarin tackles complex interaction with adversary



Your protocol modeled with rewrite rules

Out(t)

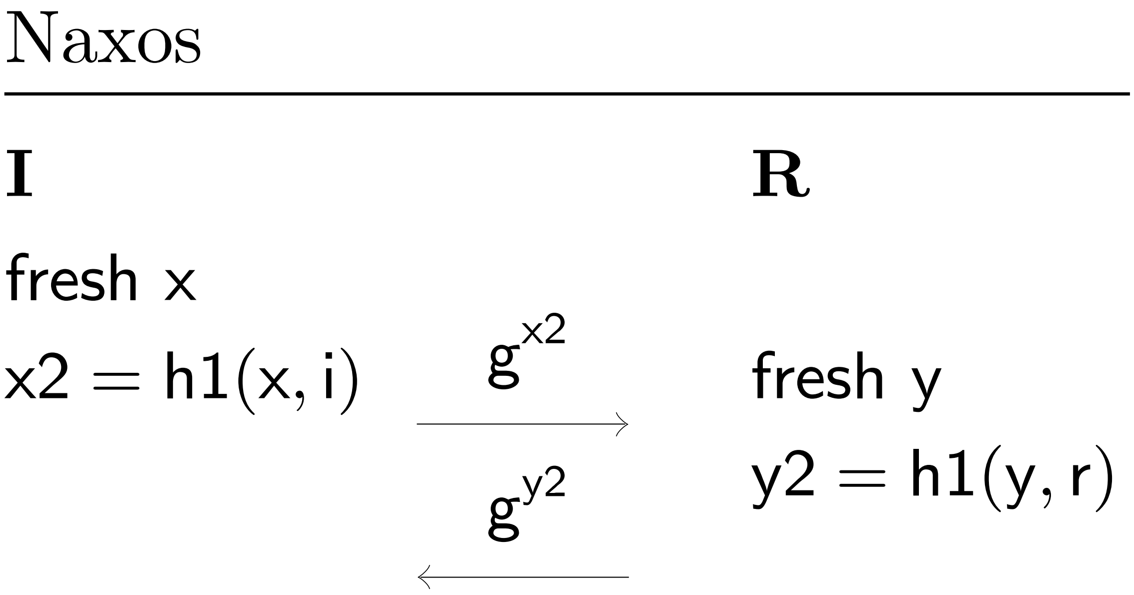
In(t)

DY-style adversary

a.k.a.

The network

The Naxos protocol



Naxos

I

fresh x

$\mathsf{x2} = \mathsf{h1}(\mathsf{x}, \mathsf{i})$

$\xrightarrow{g^{x2}}$

$\xleftarrow{g^{y2}}$

R

fresh y

$\mathsf{y2} = \mathsf{h1}(\mathsf{y}, \mathsf{r})$

private keys: i, r
public keys: g^i, g^r

$$K = h2(\ g^{i \cdot y2},\ \ g^{r \cdot x2},\ \ g^{x2 \cdot y2},\ \ I, R)$$

Naxos I

I

fresh x

$x2 = h1(x, i)$ $\xrightarrow{g^{x2}}$

$\xleftarrow{g^{y2}}$

```
rule generate_ltk:
   let pkI = 'g'^~i
   in
   [ Fr(~i) ]
   -->
   [ Ltk( $I, ~i ) ]

rule Init_1:
  let x2 = h1(<~x, ~i >)
      m1 = 'g'^x2
  in
   [ Fr( ~x ), Ltk( $I, ~i ) ]
   -->
   [ Init_1( ~x, $I, $R, ~i, m1 ) , Out( m1 ) ]

rule Init_2:
   [ Init_1( ~x, $I, $R, ~i, m1), In( m2 ) ]
   -->
   []
```

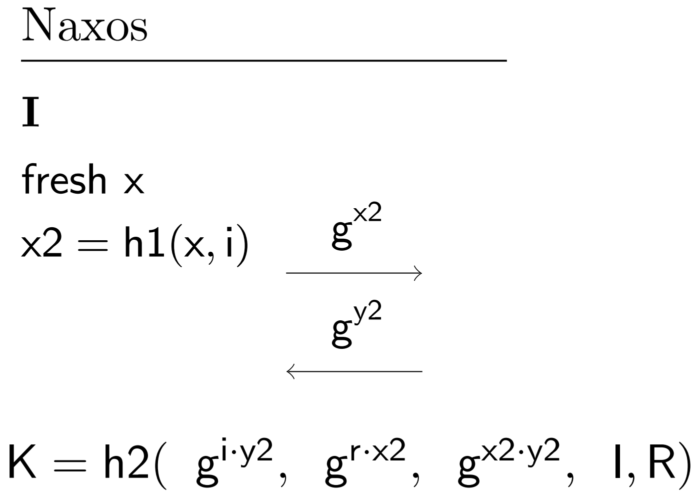
Property specification

Property specification

- 2-sorted (temp,msg) first order logic interpreted over a trace

– False	False
– Equality	$m_1 =_E m_2$
– Timepoint ordering	#t1 < #t2
– Timepoint equality	#t1 = #t2
– Action at timepoint #t	A@#t

Property specification



Naxos

I

fresh x

$x2 = h1(x, i)$ $\xrightarrow{g^{x2}}$

$\xleftarrow{g^{y2}}$

$K = h2(\ g^{i \cdot y2},\ \ g^{r \cdot x2},\ \ g^{x2 \cdot y2},\ \ I, R)$

- Rules:
  - l –[a]→ r
  - Instantiated actions stored as (action) trace
    - Additionally: adversary knows facts: K()

```
rule Init_2:
  let pkR = 'g'^~r,
      x2  = h1(< ~x, ~i >),
      kI  = h2(< m2^~i, pkR^x2, m2^x2, $I, $R >)
  in
   [ Init_1( ~x, $I, $R, ~i , m1), In( m2 ) ]
  --[ Accept(~x, $I, $R, kI) ]-->
   []


Lemma key_secret:
  ''(All #t Test A B k. Accept(Test,A,B,k)@t => Not (Ex #t2. K(k)@t2 ))''
```

Advanced property specification

eCK security model for key exchange

- Adversary can
  - learn **long-term keys**,
  - learn the **randomness** generated in sessions,
  - learn **session keys**
- But only as long as the Test session is *clean*:
  - **No reveal of session key of** Test session or its **matching session**, and
  - No reveal of randomness of Test session as well as the long-term key of the actor, and
  - If there exists a matching session, then *something* is disallowed...
  - If there is no matching session, then *something else*...

```
Lemma eCK_key_secrecy:
  "(All #t1 #t2 Test A B k. Accept(Test, A, B, k) @ t1
                             & K( k ) @ t2 ==>
  (
      (Ex #t3. SesskRev( Test ) @ t3 )
    | (Ex MatchingSession #t3 #t4 ms.
           ( Sid ( MatchingSession, ms ) @ t3
           & Match( Test, ms ) @ t4)
           & (Ex #t5. SesskRev( MatchingSession ) @ t5 ))
    | […]
  )"
end
```

Demo

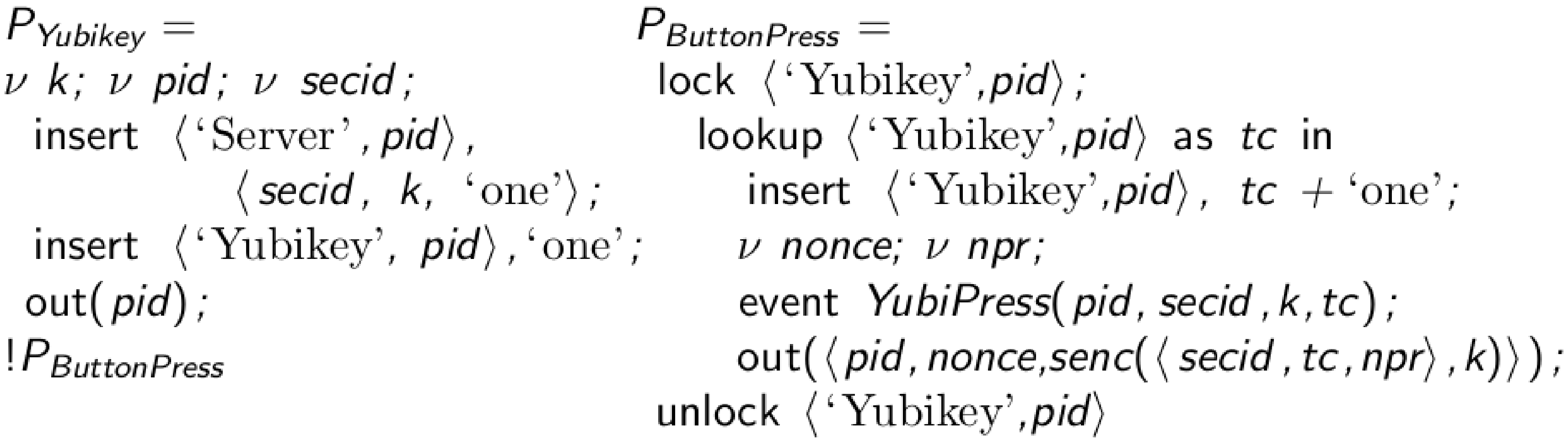
2014

Tamarin: Selected case studies

- Key exchange protocols
  - Naxos
  - Signed DH
  - KEA+
  - UM
  - Tsx
  - TLS handshake
- Group protocols
  - GDH
  - TAK
  - (Sig)Joux
  - STR
- ID-based AKE
  - RYY
  - Scott
  - Chen-Kudla
- Protocols with loops
  - TESLA1
  - TESLA2
- Non-monotonic global state
  - Keyserver
  - Envelope
  - Exclusive secrets
  - Contract signing
  - Security device
  - YubiKey
  - YubiHSM
- PKI with strong guarantees
  - ARPKI (also global state)
- Transparency
  - KUD/DECIM (also global state)

SAPIC

- Stateful applied Pi calculus + tool
  - Steve Kremer & Robert Künnemann
- Compiles to Tamarin input



```
P_Yubikey =
ν k; ν pid; ν secid;
  insert ⟨'Server', pid⟩,
            ⟨secid, k, 'one'⟩;
  insert ⟨'Yubikey', pid⟩, 'one';
  out(pid);
!P_ButtonPress
```

```
P_ButtonPress =
 lock ⟨'Yubikey',pid⟩;
   lookup ⟨'Yubikey',pid⟩ as tc in
     insert ⟨'Yubikey',pid⟩, tc + 'one';
     ν nonce; ν npr;
     event YubiPress(pid, secid, k, tc);
     out(⟨pid, nonce, senc(⟨secid, tc, npr⟩, k)⟩);
 unlock ⟨'Yubikey',pid⟩
```

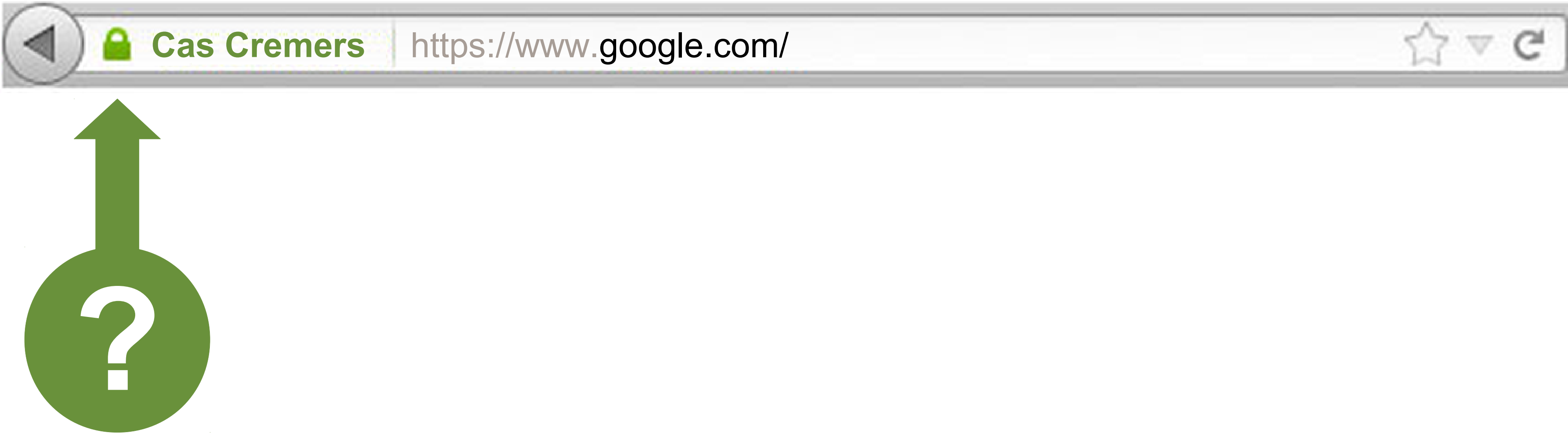
Tamarin summary

- We can now deal with:
  - Any number of instances, even with loops and mutable global state
  - Complex protocol details and property specifications
  - Some support for **observational (trace) equivalence** (2016)
  - But still much left to be handled and automated

- The **Tamarin prover** is freely available
  - Theses Simon Meier & Benedikt Schmidt
  - Papers: CSF 2012, CAV 2013,
    IEEE S&P 2014, …
  - Manual (PDF and website)
  - Development on github

2015

Internet Security



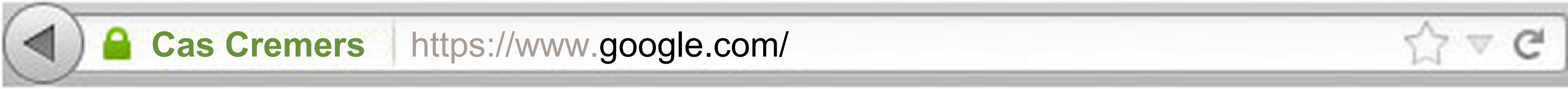
Overview

- Case study: TLS 1.3
  - What is it?
  - Our analysis approach
  - Some details
  - Results
- Wrap up

These all implement the **TLS** protocol: **T**ransport **L**ayer **S**ecurity

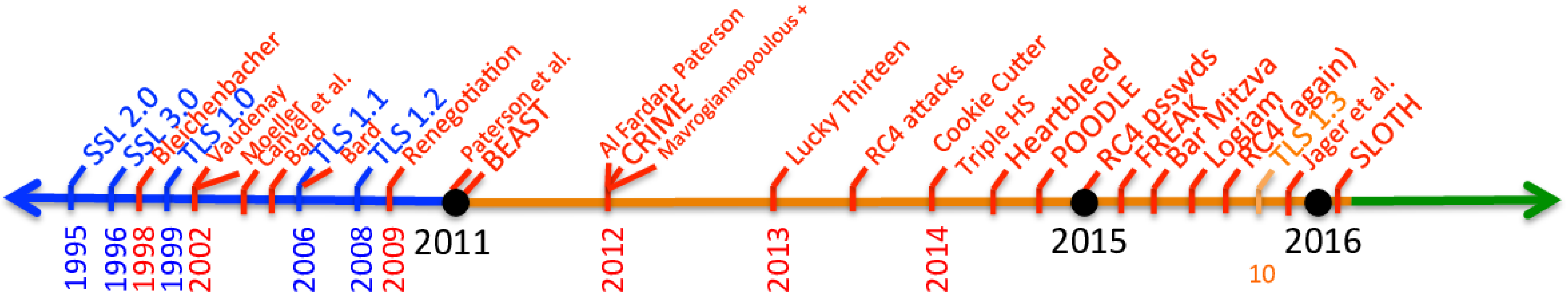
previously known as SSL;
also the 'S' in 'https';
a.k.a. the green lock

The *purpose* of TLS:
To provide a secure channel to transfer messages



Cas Cremers https://www.google.com/

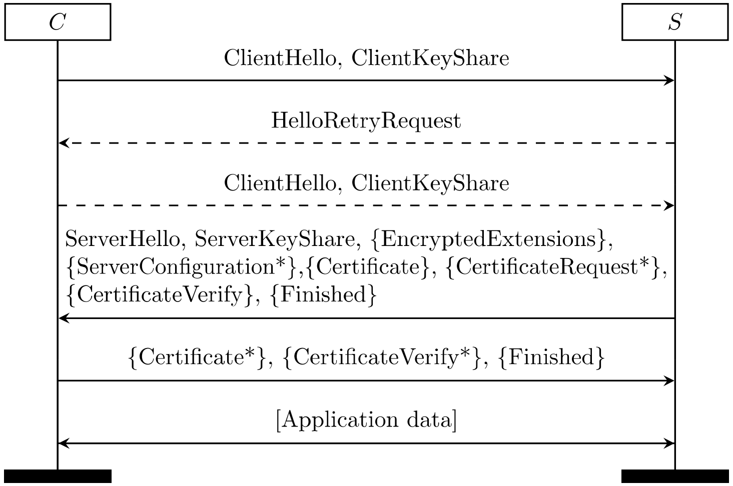
Security of TLS over time



TLS development

- Currently under development: **TLS 1.3**
  - Led by the **Internet Engineering Task Force (IETF)**
  - Public mailing list discussions
  - Long, complex process

TLS 1.3



C

S

ClientHello, ClientKeyShare

HelloRetryRequest

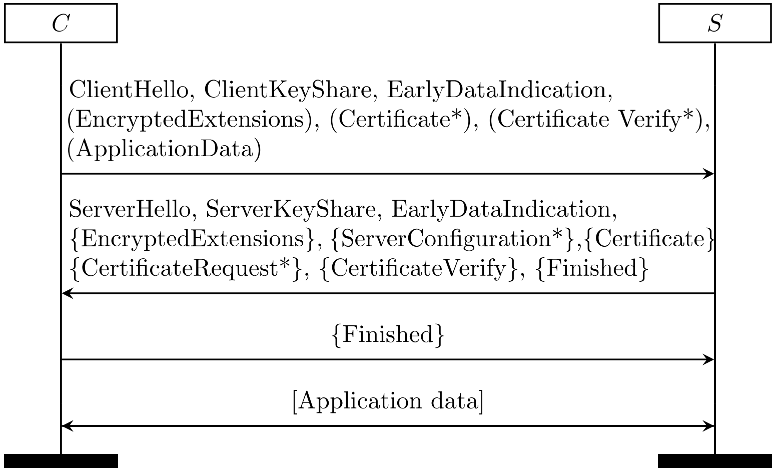
ClientHello, ClientKeyShare

ServerHello, ServerKeyShare, {EncryptedExtensions}, {ServerConfiguration*},{Certificate}, {CertificateRequest*}, {CertificateVerify}, {Finished}

{Certificate*}, {CertificateVerify*}, {Finished}

[Application data]

(a) Initial (EC)DHE handshake



C

S

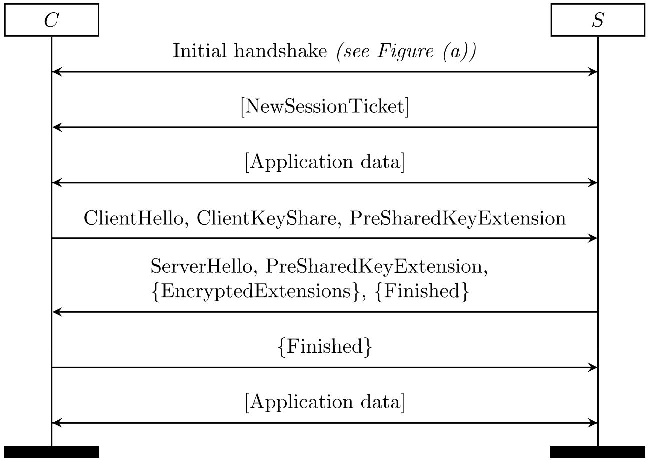
ClientHello, ClientKeyShare, EarlyDataIndication, (EncryptedExtensions), (Certificate*), (Certificate Verify*), (ApplicationData)

ServerHello, ServerKeyShare, EarlyDataIndication, {EncryptedExtensions}, {ServerConfiguration*},{Certificate} {CertificateRequest*}, {CertificateVerify}, {Finished}

{Finished}

[Application data]

(b) 0-RTT handshake



C

S

Initial handshake *(see Figure (a))*

[NewSessionTicket]

[Application data]

ClientHello, ClientKeyShare, PreSharedKeyExtension

ServerHello, PreSharedKeyExtension, {EncryptedExtensions}, {Finished}

{Finished}

[Application data]

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

What we did (nutshell)

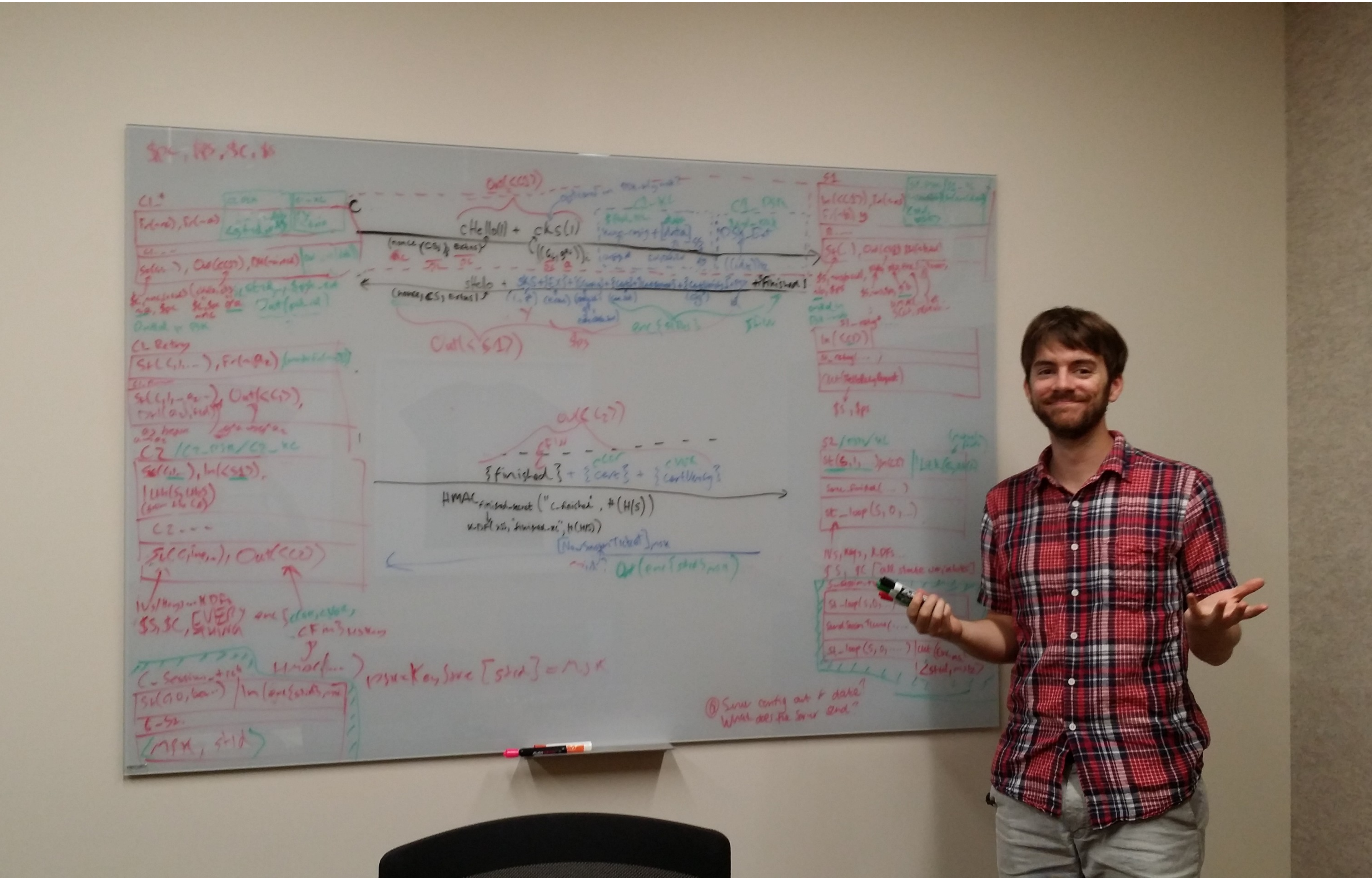
- Collaboration with Royal Holloway
  - Cas with Marko Horvat, Sam Scott, and Thyla van der Merwe

- We built a symbolic model of the TLS 1.3 specification currently under development (draft 10)
- We wanted to verify the **core** properties of TLS 1.3 as an authenticated key exchange protocol
  - secrecy of session keys
  - unilateral (mutual) authentication
- We found a potential attack – disclosed this to the IETF TLS WG

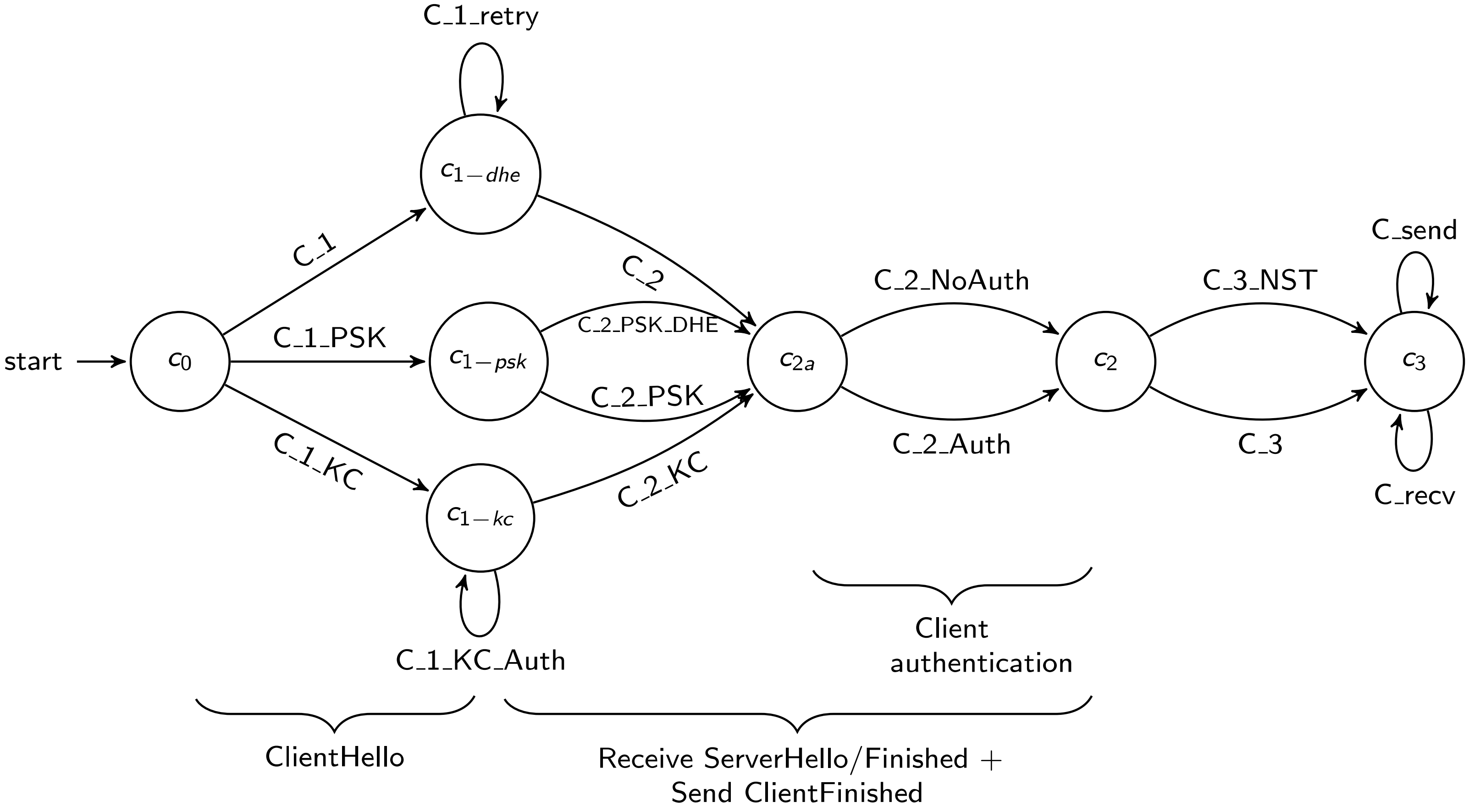
TLS 1.3 and Tamarin

- We built our model for use in the Tamarin prover
  - Reasons:
    - Supports loops and branches well
    - Good symbolic Diffie-Hellman support

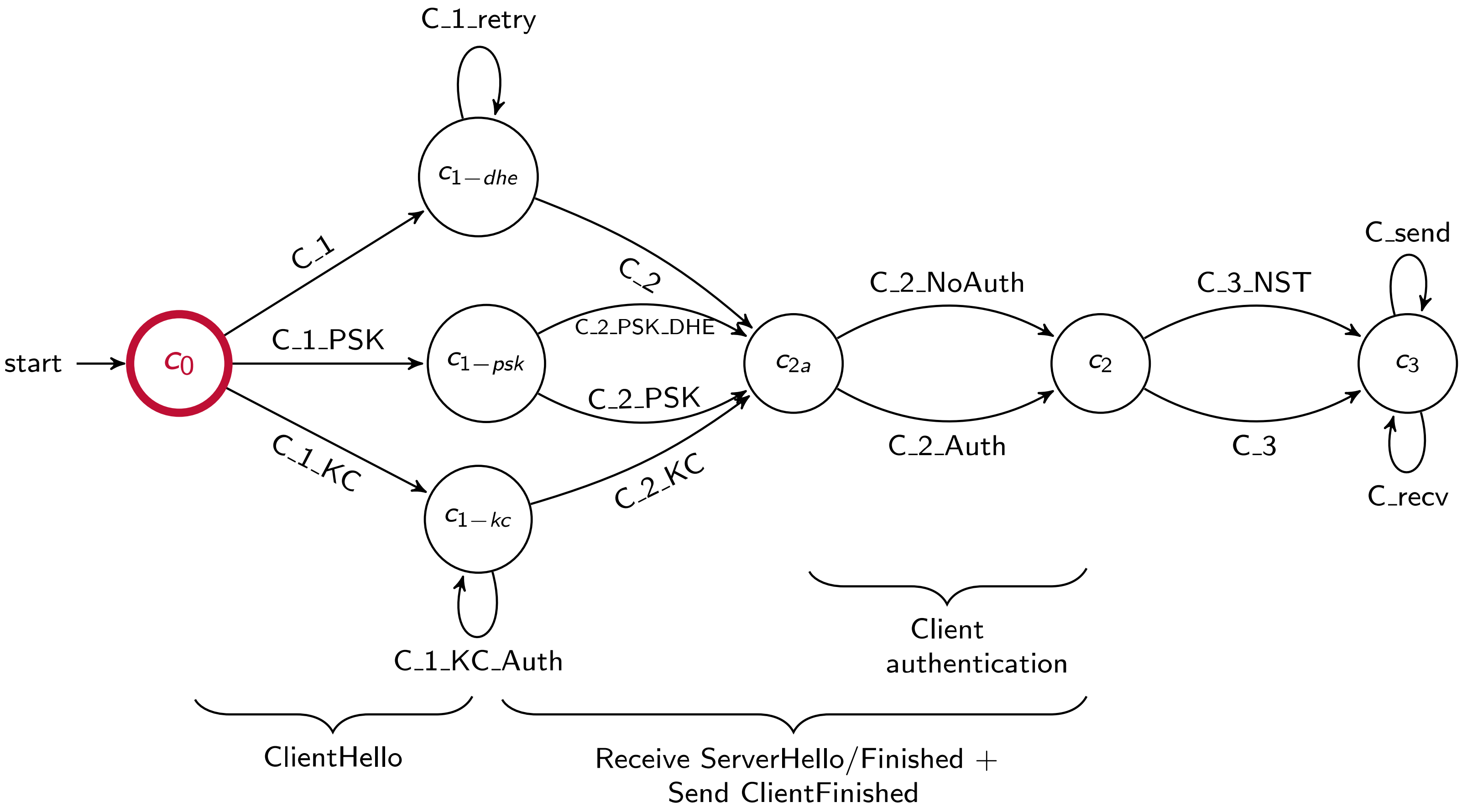
Step 1: Building a model



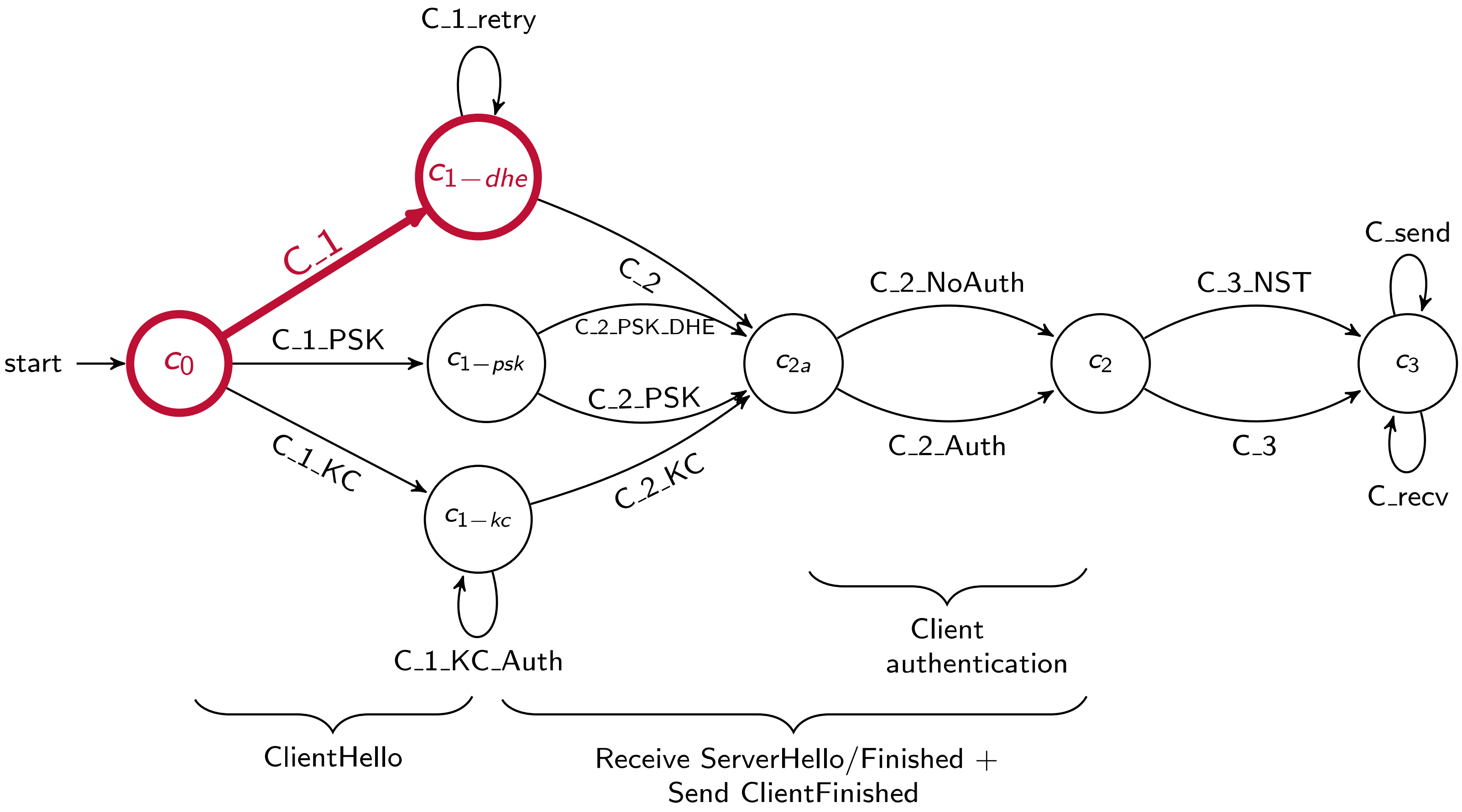
Step 1: Building a model



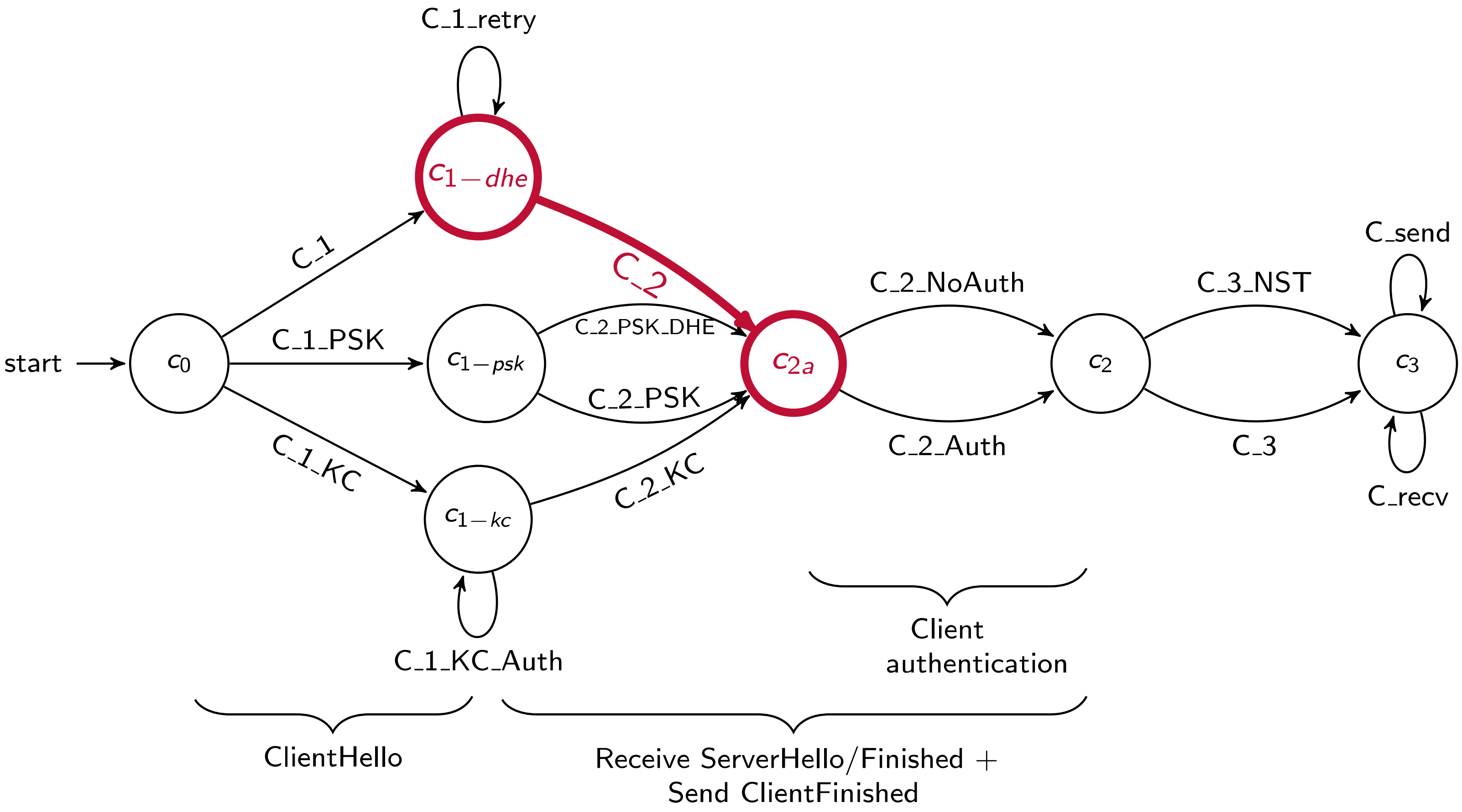
Step 1: Building a model



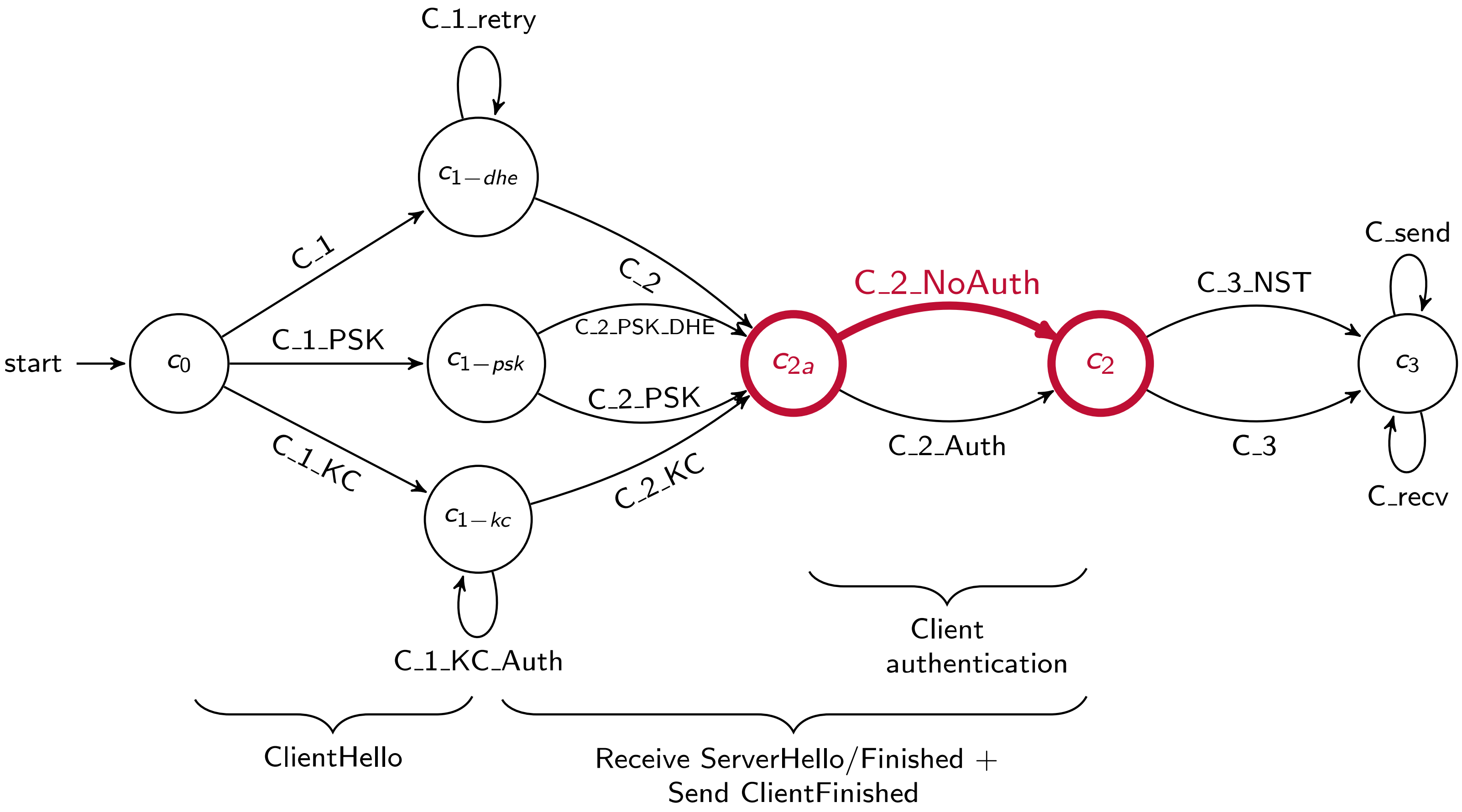
Step 1: Building a model



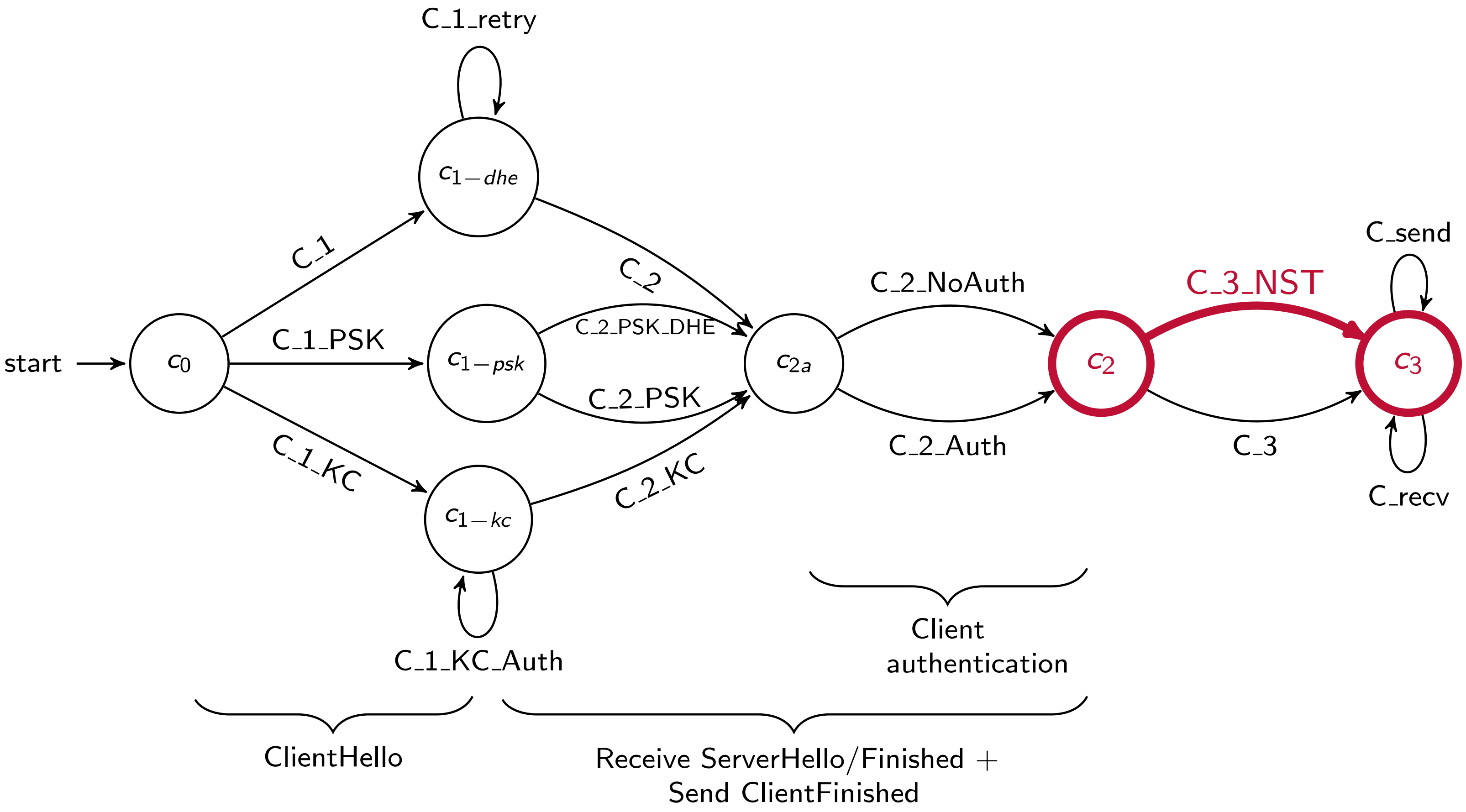
Step 1: Building a model



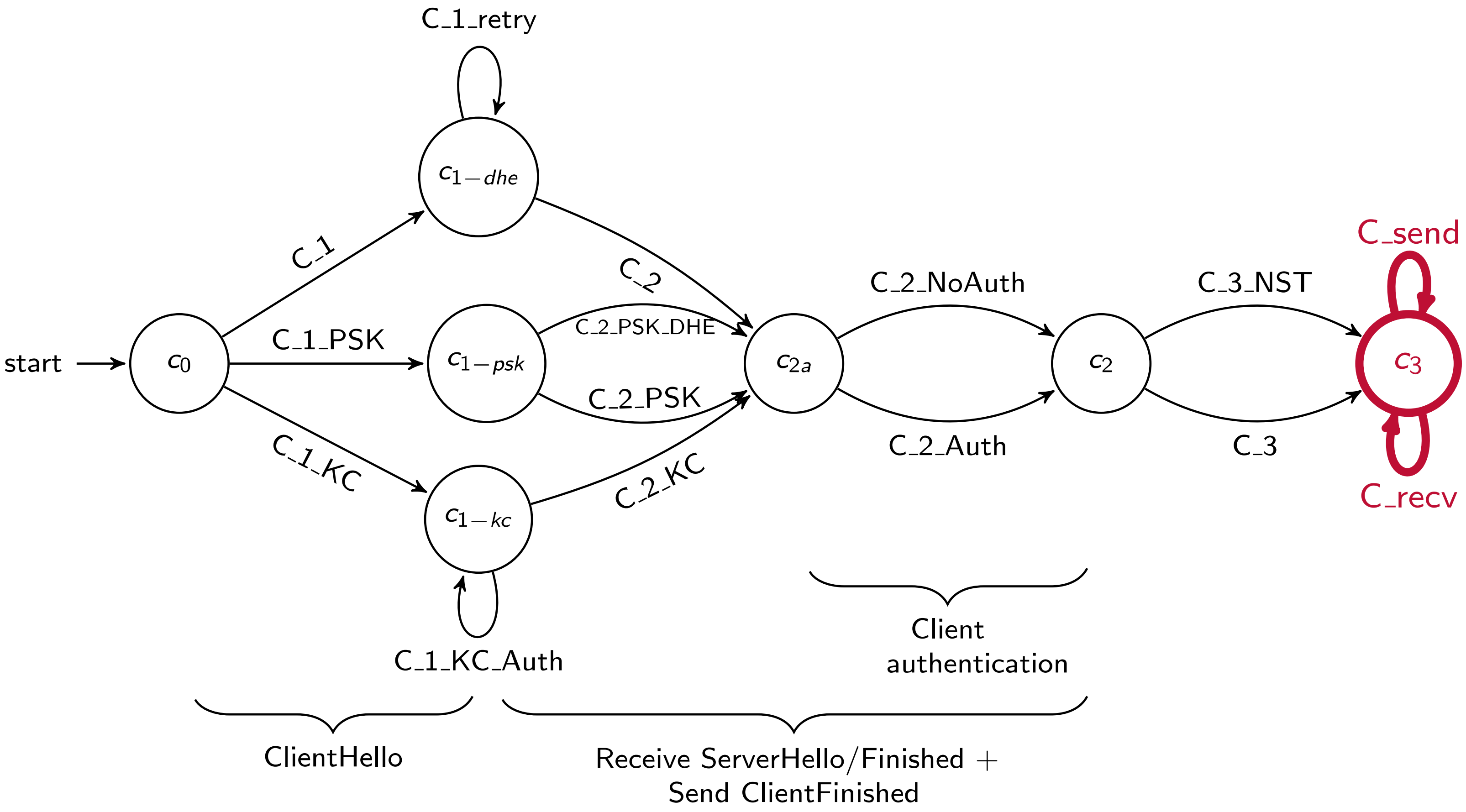
Step 1: Building a model



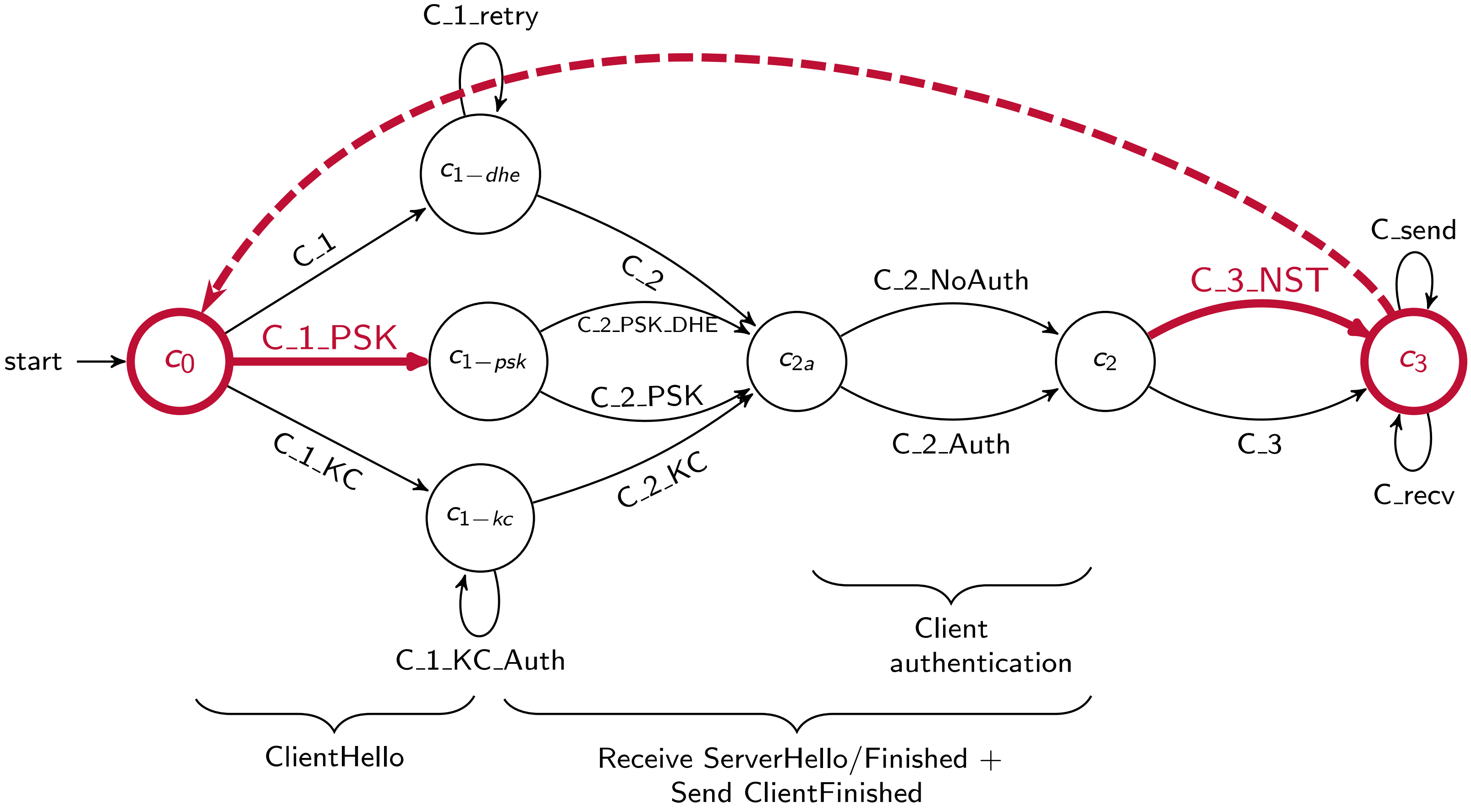
Step 1: Building a model



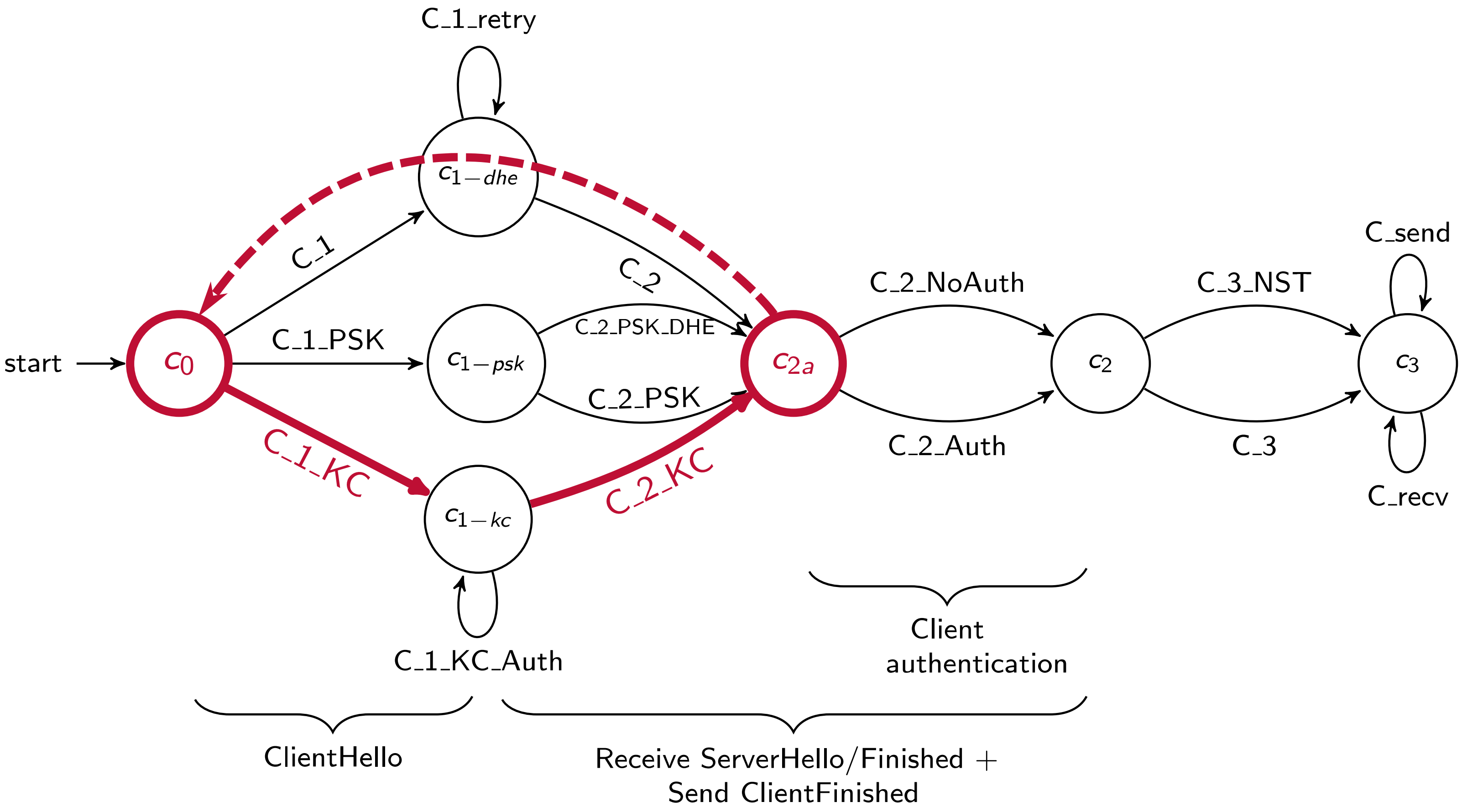
Step 1: Building a model



Step 1: Building a model



Step 1: Building a model



Step 1: Building a model

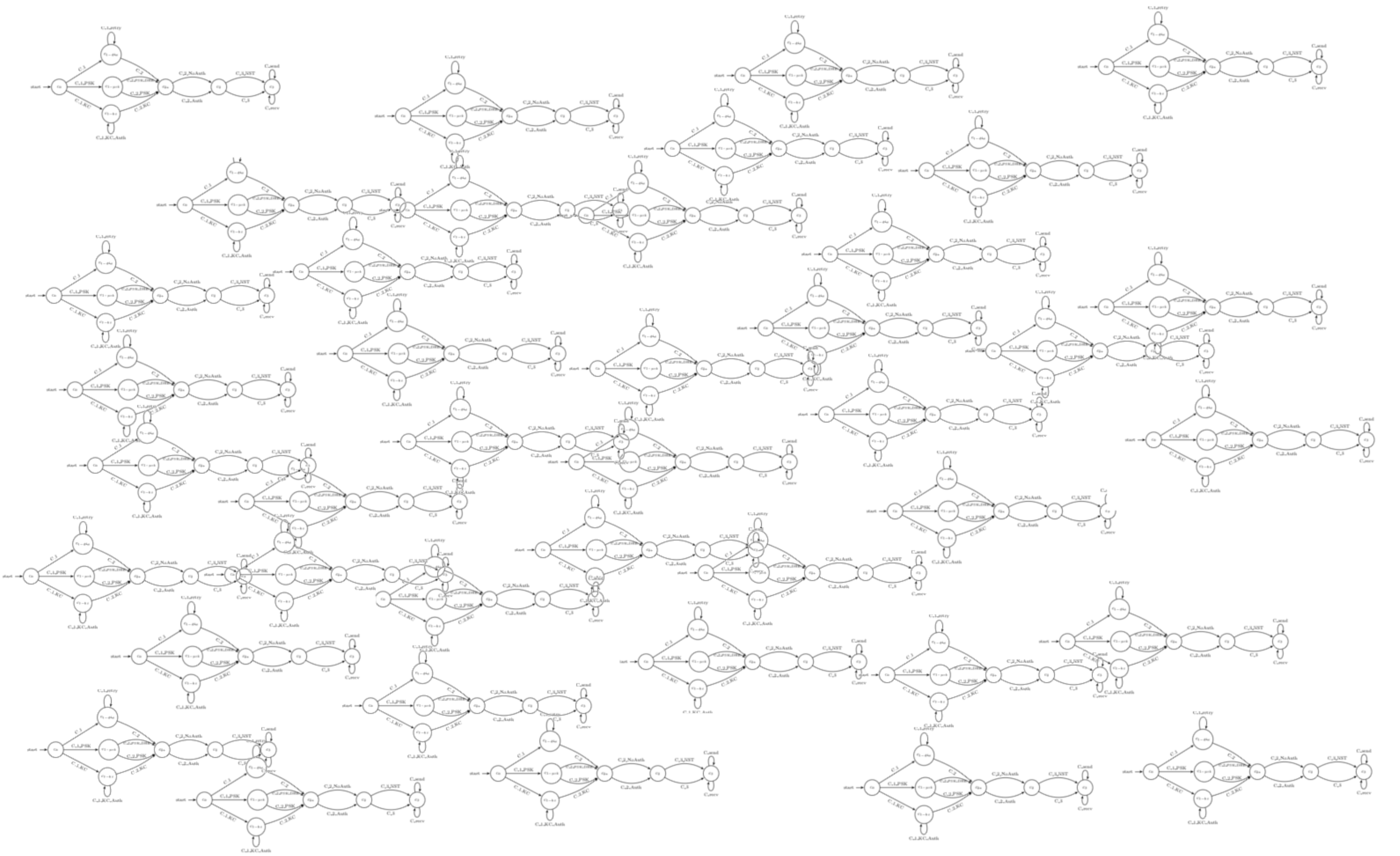
```
rule C_1:
let
    // Default C1 values
    tid = ~nc

    // 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
  - **integrity** of handshake messages

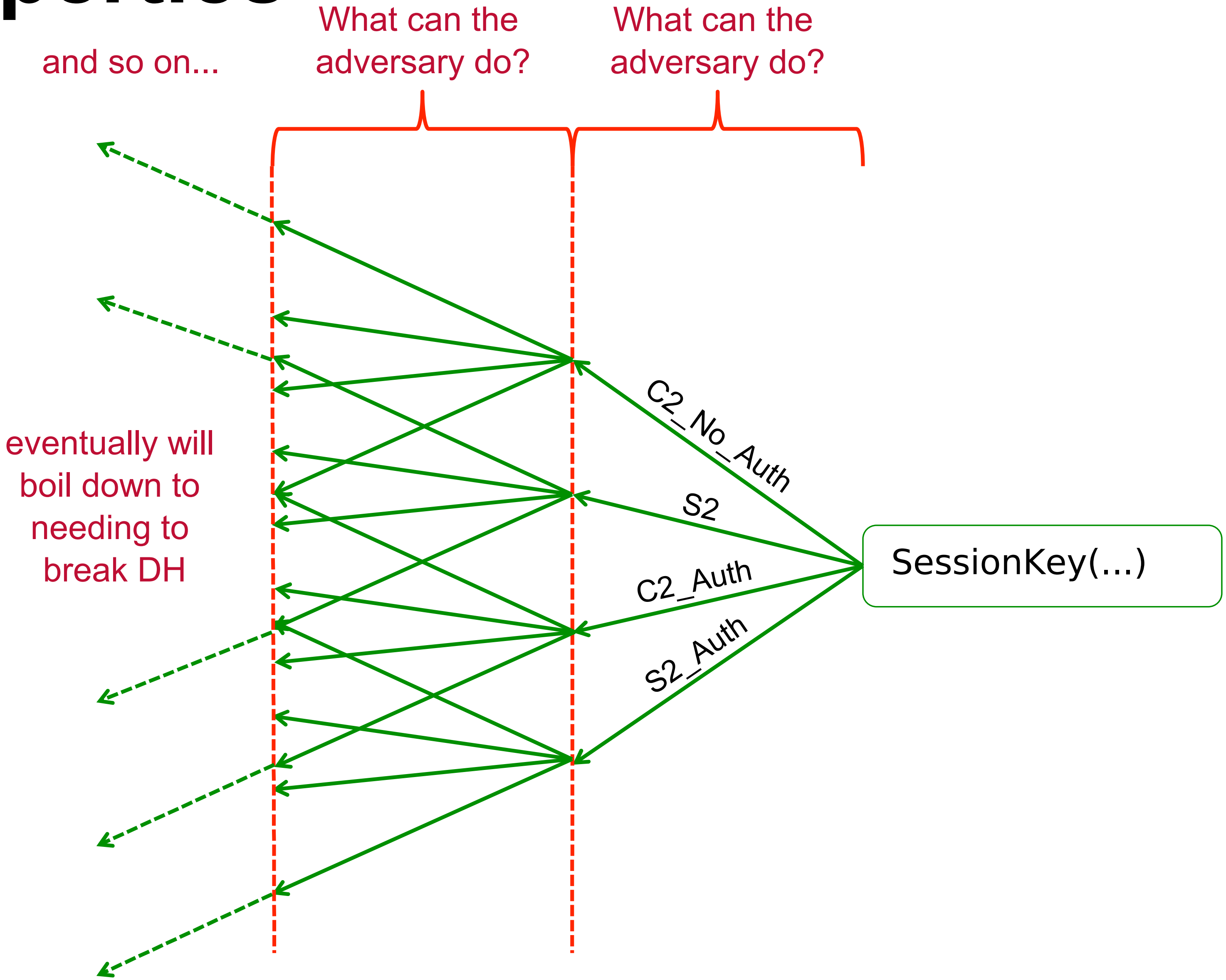
Step 2: Encoding security properties

```
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



and so on...

What can the adversary do?

What can the adversary do?

eventually will boil down to needing to break DH

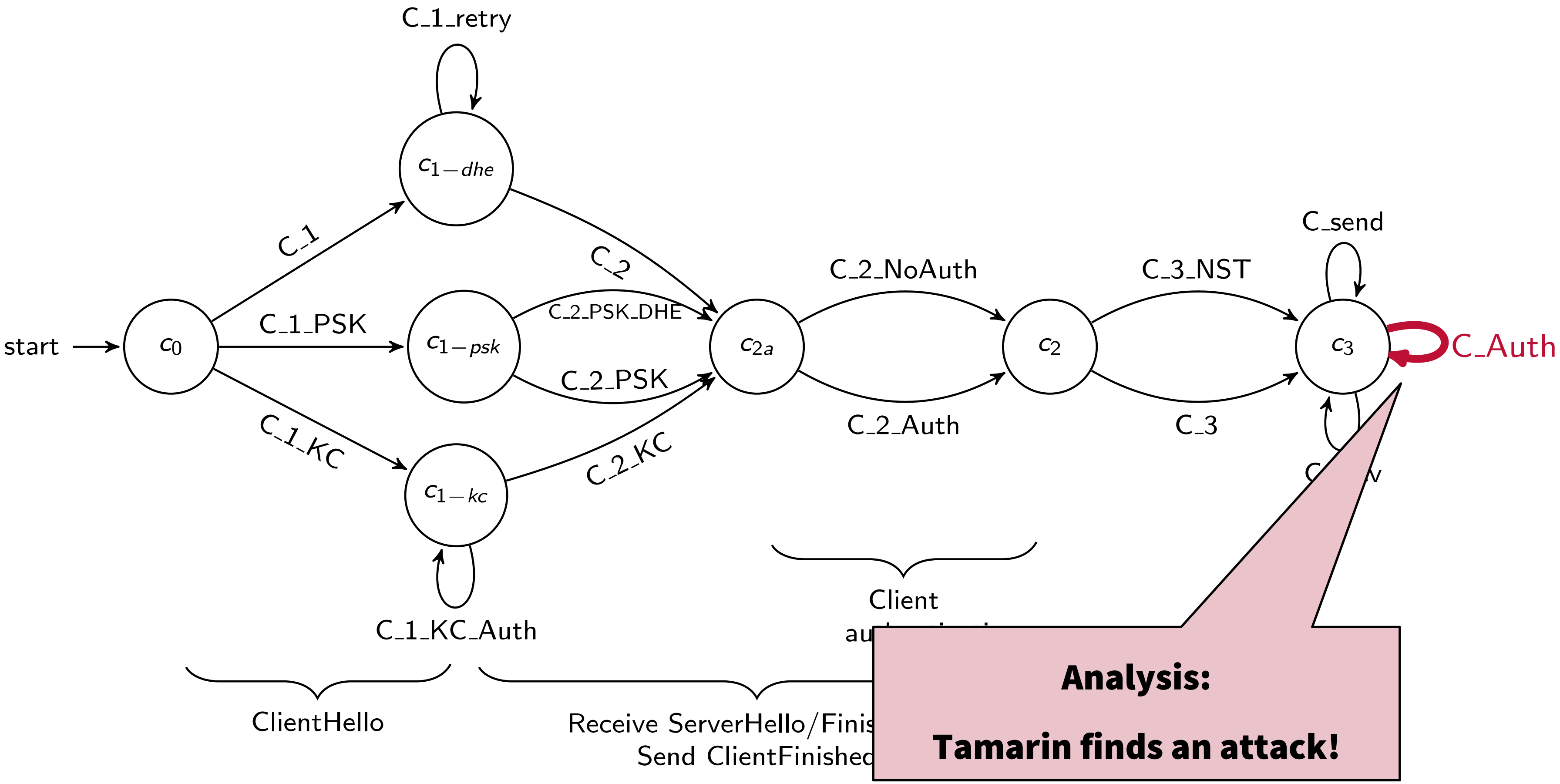
Step 3: Proving security properties

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

Step 3: Proving security properties

- We verified the core properties of TLS 1.3 draft 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+)



Analysis:

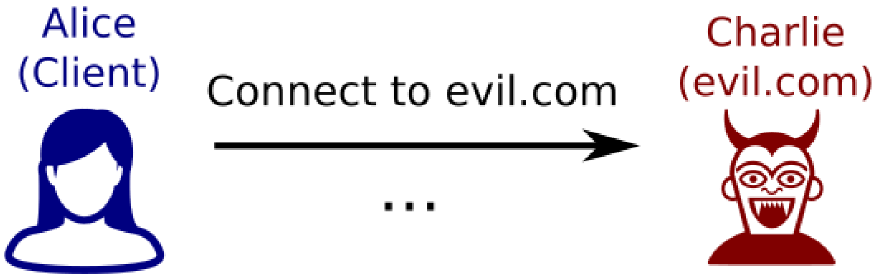
Tamarin finds an attack!

Attacking client authentication

Alice
(Client)

Charlie
(evil.com)

Attacking client authentication



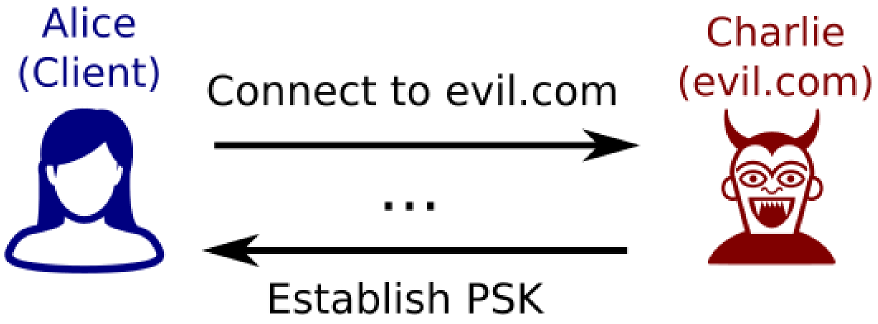
Alice
(Client)

Connect to evil.com

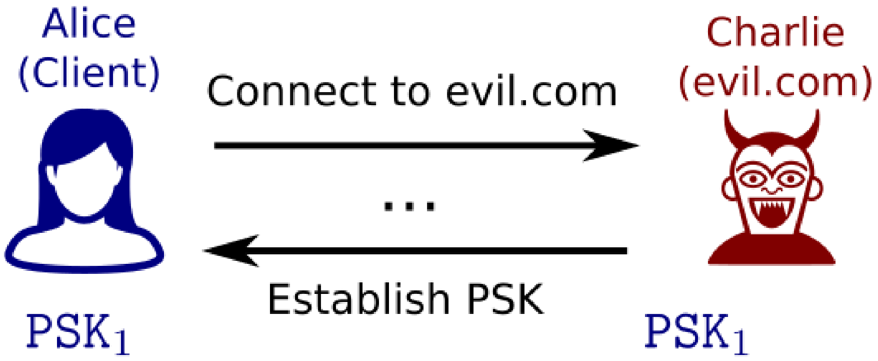
...

Charlie
(evil.com)

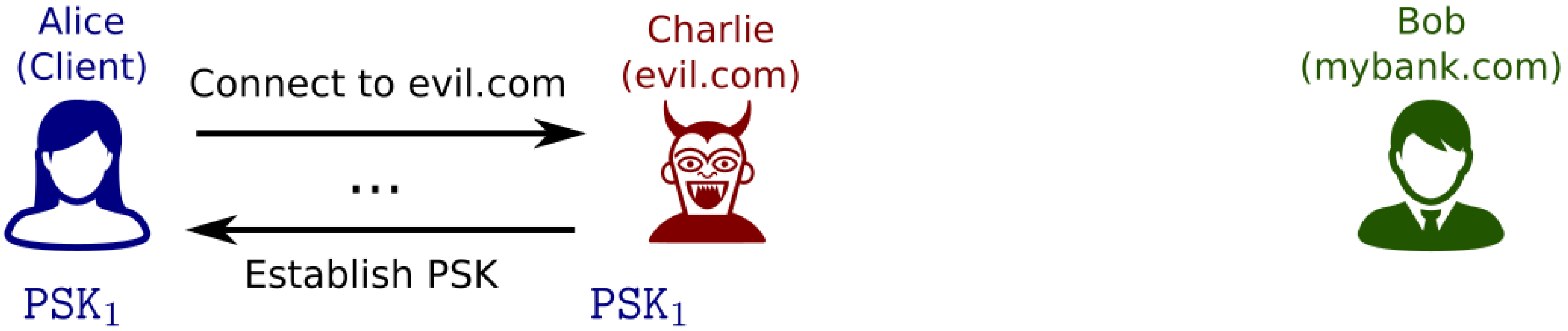
Attacking client authentication



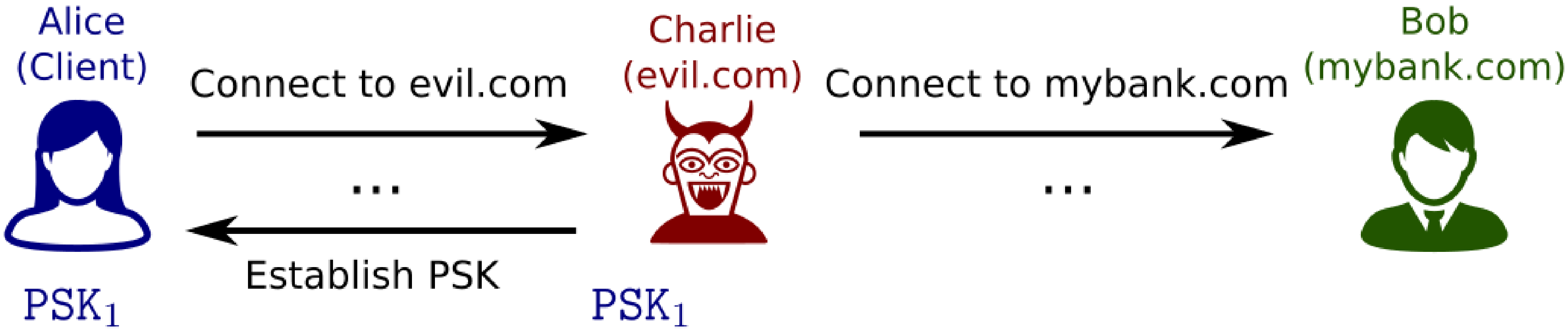
Attacking client authentication



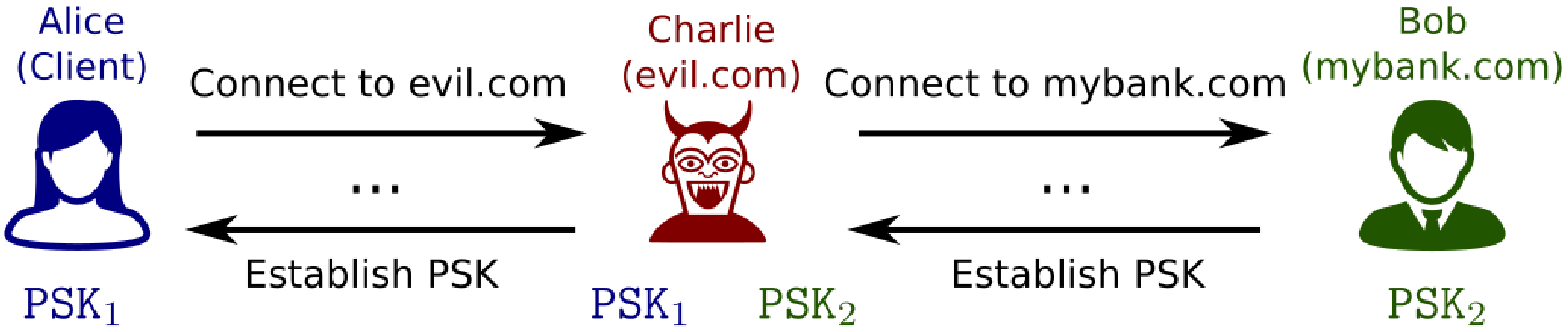
Attacking client authentication



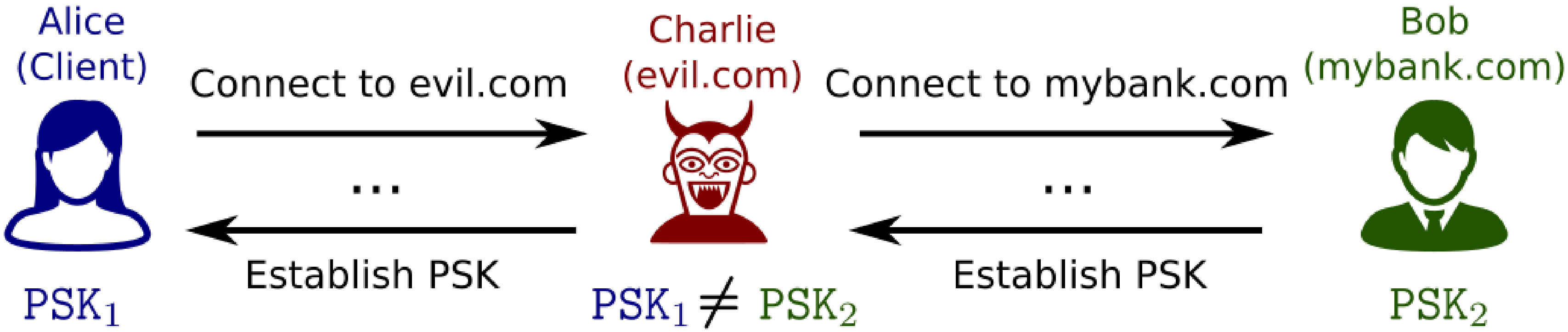
Attacking client authentication



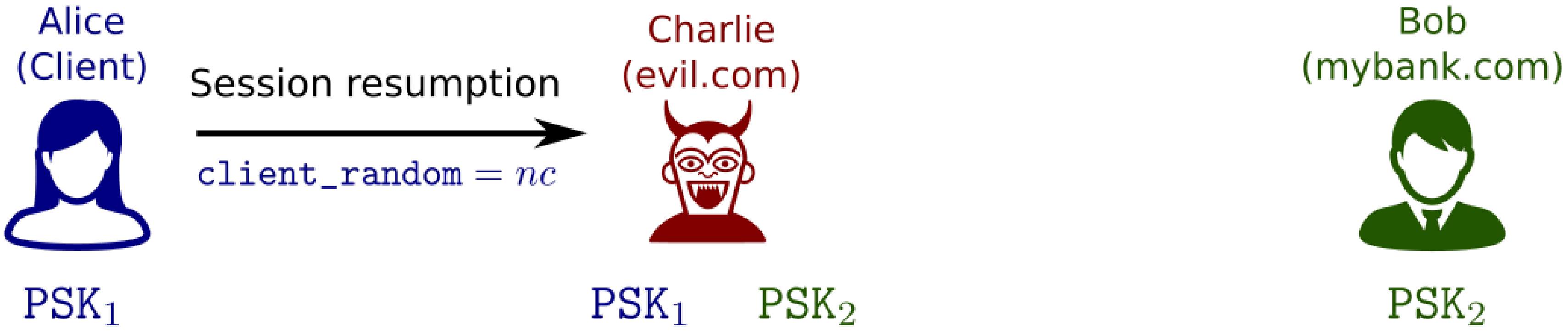
Attacking client authentication



Attacking client authentication



Attacking client authentication



Alice
(Client)

Session resumption

$\texttt{client\_random} = nc$

Charlie
(evil.com)

Bob
(mybank.com)

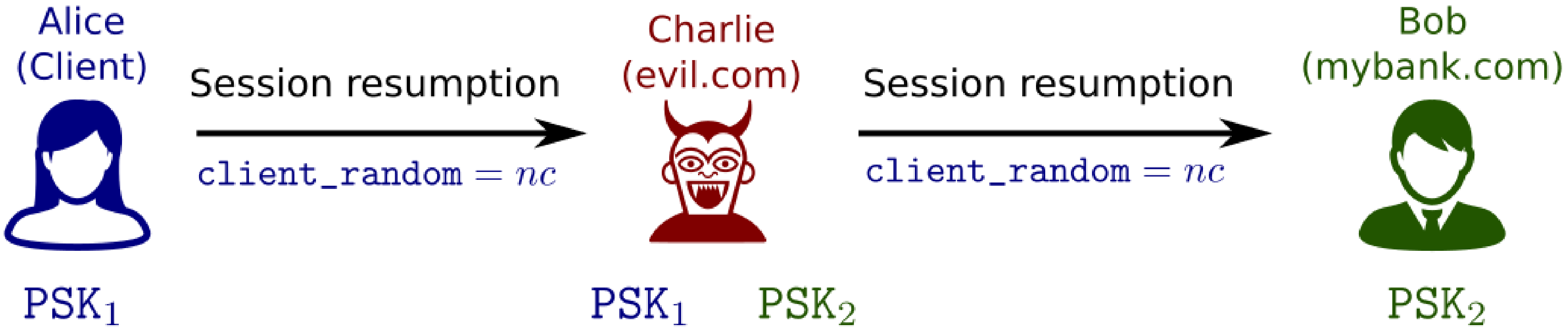
PSK_1

PSK_1 PSK_2

PSK_2

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

Attacking client authentication



Alice (Client)

Session resumption

$\texttt{client\_random} = nc$

Charlie (evil.com)

Session resumption

$\texttt{client\_random} = nc$

Bob (mybank.com)

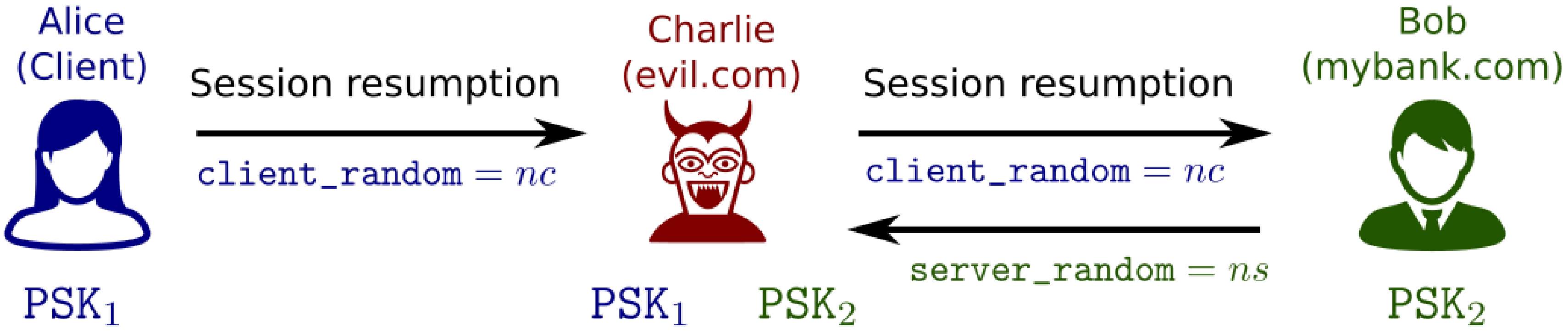
PSK_1

PSK_1 PSK_2

PSK_2

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

Attacking client authentication



Alice (Client)

Session resumption

$\texttt{client\_random} = nc$

Charlie (evil.com)

Session resumption

$\texttt{client\_random} = nc$

$\texttt{server\_random} = ns$

Bob (mybank.com)

PSK_1

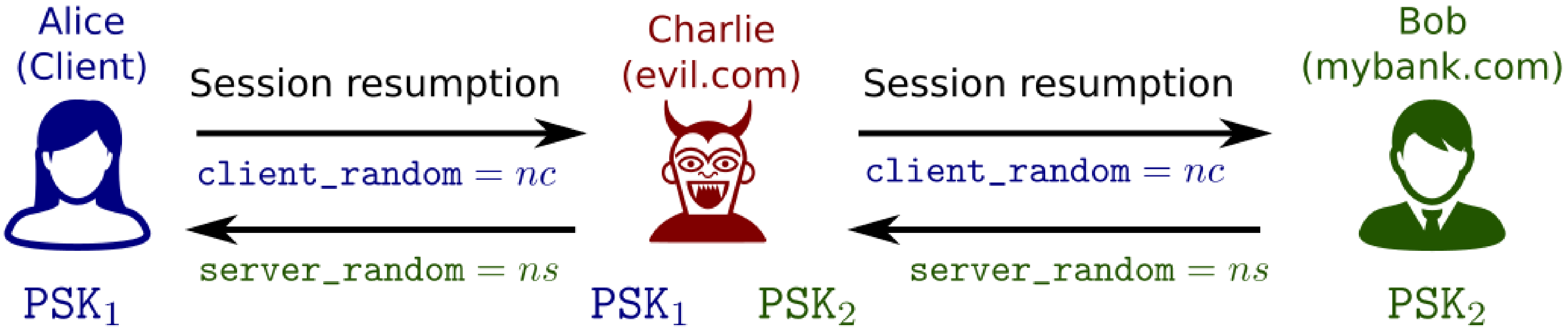
PSK_1 PSK_2

PSK_2

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

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

Attacking client authentication



Alice (Client)

Session resumption

$\texttt{client\_random} = nc$

$\texttt{server\_random} = ns$

PSK_1

Charlie (evil.com)

PSK_1 PSK_2

Session resumption

$\texttt{client\_random} = nc$

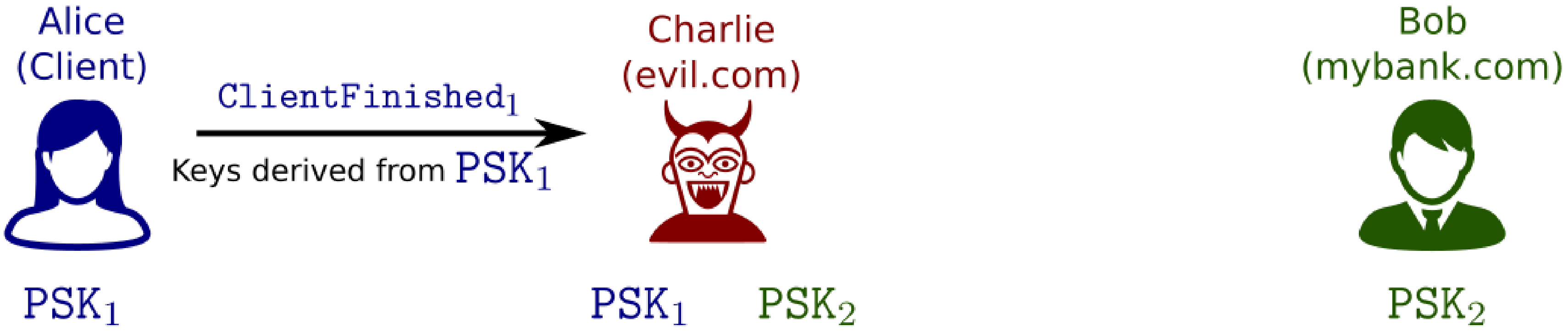
$\texttt{server\_random} = ns$

Bob (mybank.com)

PSK_2

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

Attacking client authentication



Alice
(Client)

$\texttt{ClientFinished}_1$

Keys derived from PSK_1

Charlie
(evil.com)

Bob
(mybank.com)

PSK_1

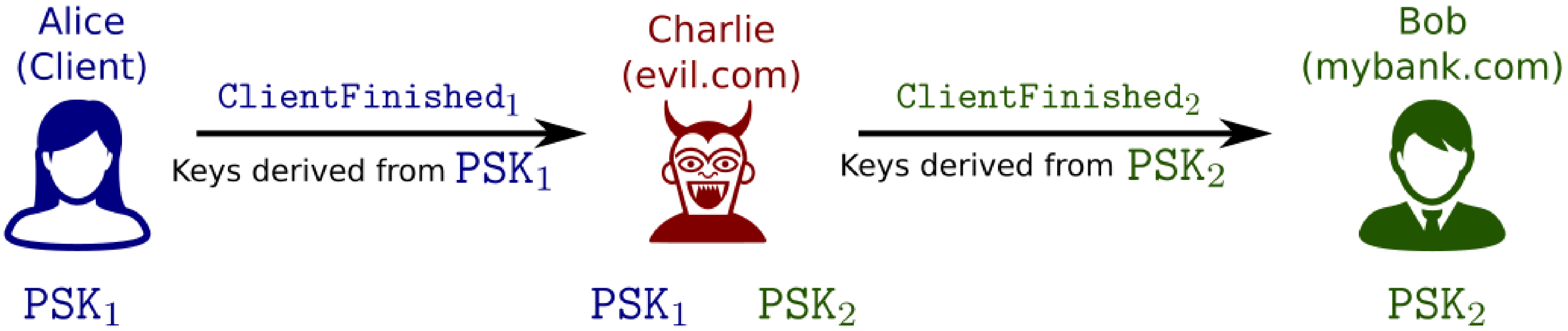
PSK_1 PSK_2

PSK_2

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

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

Attacking client authentication



Alice (Client)

$\texttt{ClientFinished}_1$

Keys derived from PSK_1

Charlie (evil.com)

$\texttt{ClientFinished}_2$

Keys derived from PSK_2

Bob (mybank.com)

PSK_1

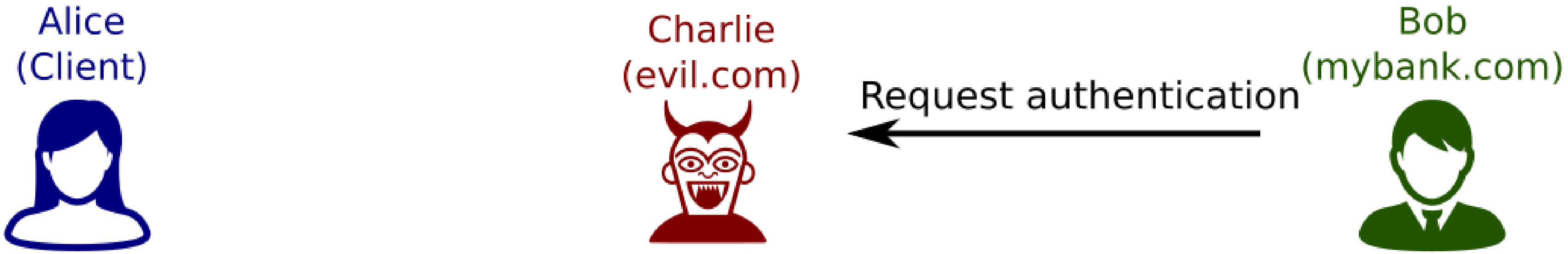
PSK_1 PSK_2

PSK_2

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

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

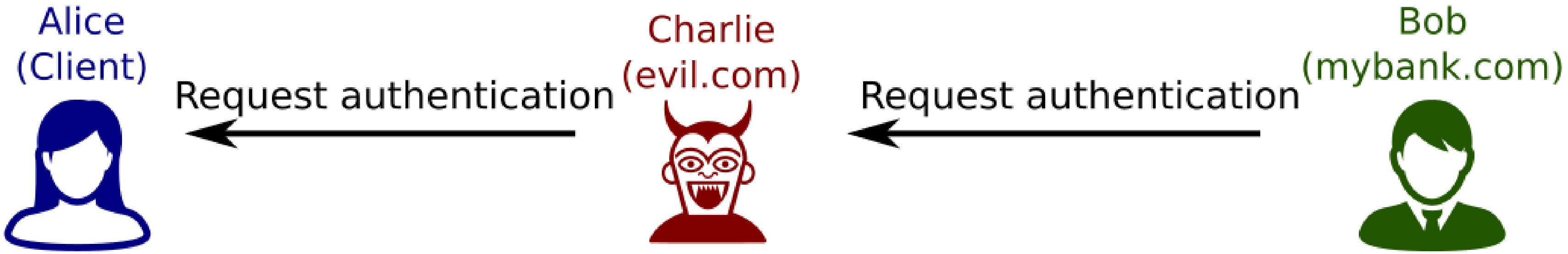
Attacking client authentication



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

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

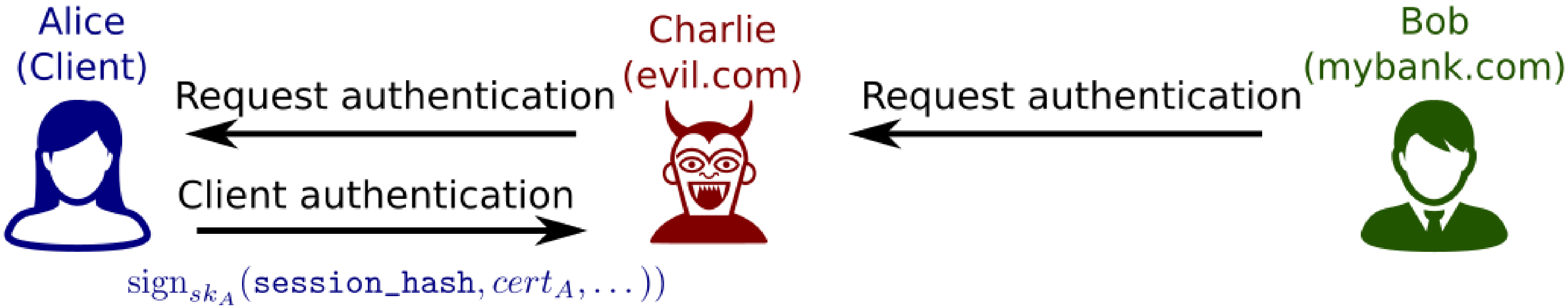
Attacking client authentication



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

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

Attacking client authentication

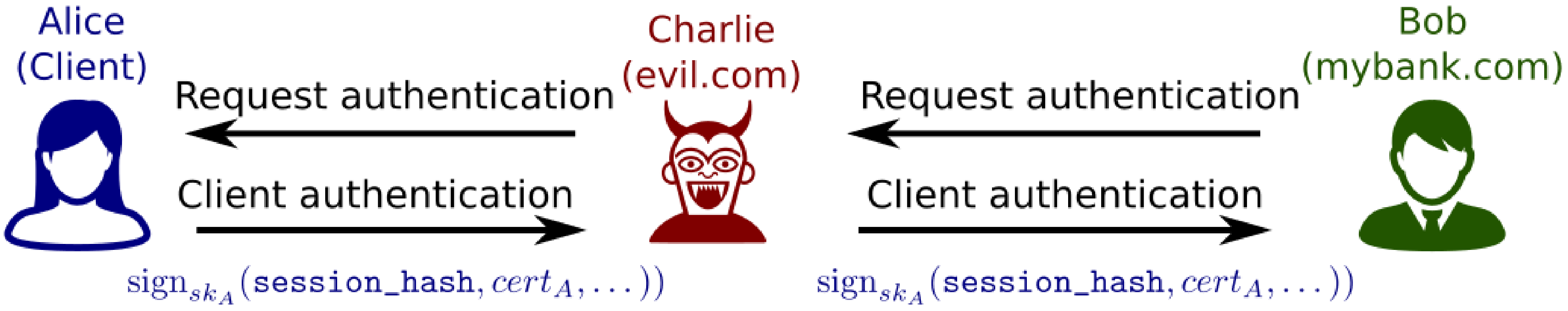


$\mathrm{sign}_{sk_A}(\texttt{session\_hash}, cert_A, \ldots))$

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

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

Attacking client authentication



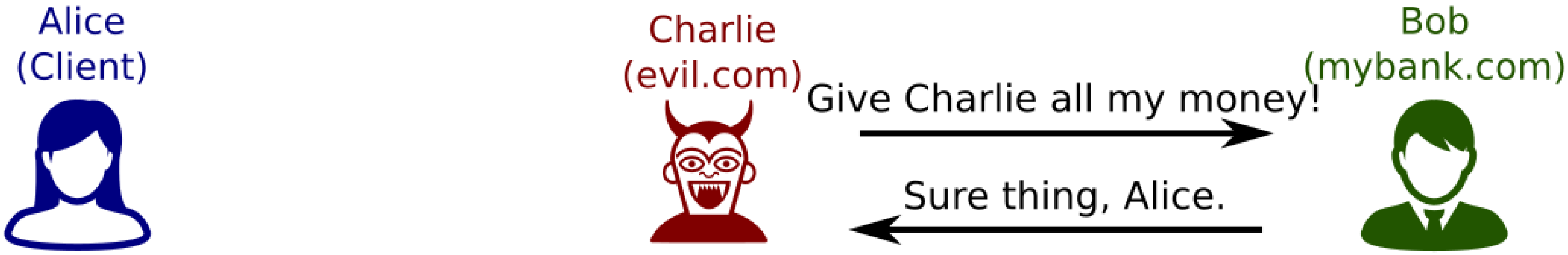
$\mathrm{sign}_{sk_A}(\texttt{session\_hash}, cert_A, \ldots))$

$\mathrm{sign}_{sk_A}(\texttt{session\_hash}, cert_A, \ldots))$

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

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

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
  - requires 18 messages to set up
  - involves 2 handshakes, 2 resumptions, 1 client auth...
- 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.

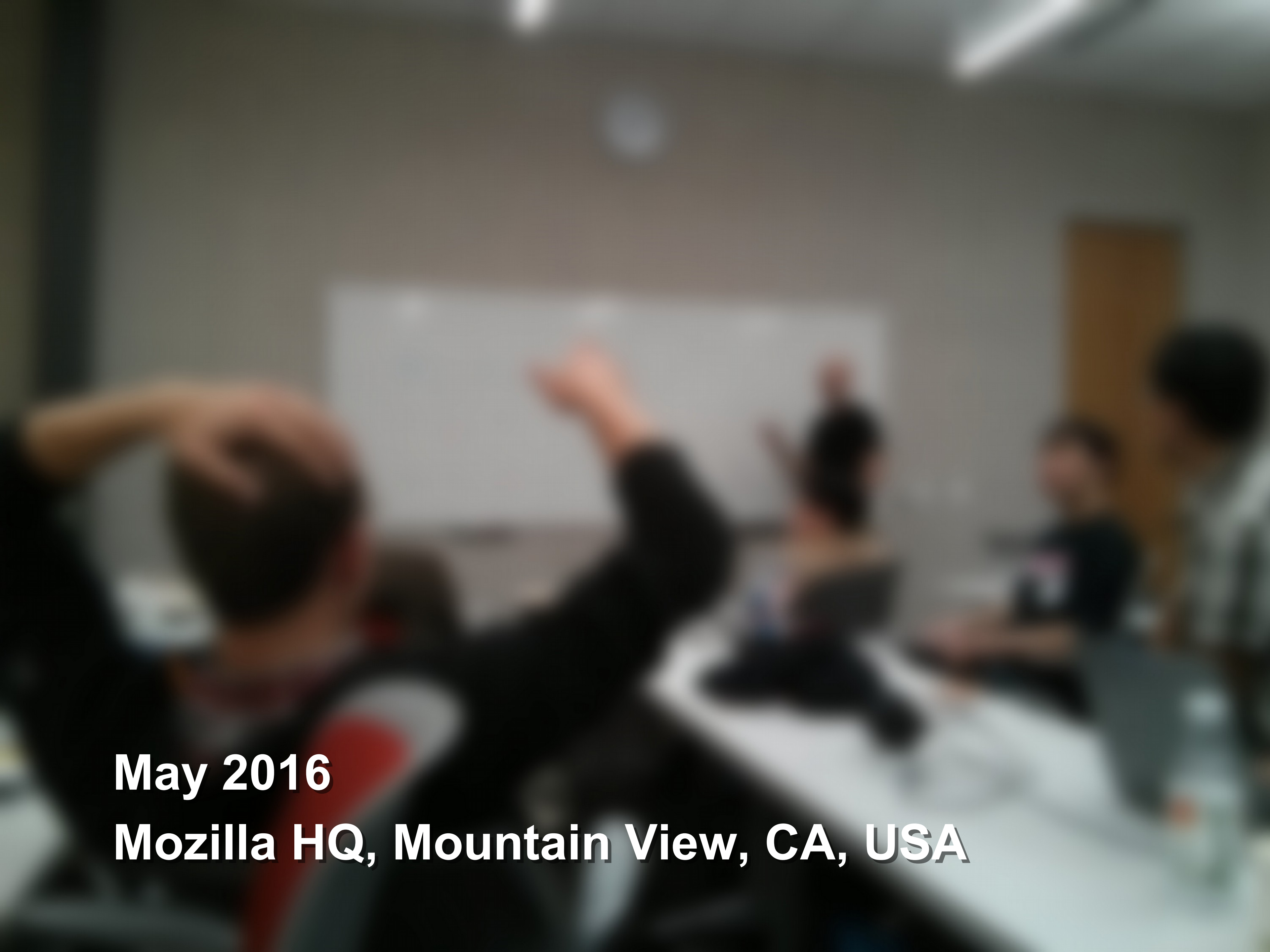
While revision 10 does not yet appear to permit certificate-based client authentication in PSK (and in particular resumption using PSK), we modelled what we believe is the intended functionality. By enabling client authentication either in the initial handshake, or with a post- handshake signature over 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:

IETF WG mailing list reactions

“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.”

“The 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, which of course we'll be discussing in Yokohama.”



May 2016

Mozilla HQ, Mountain View, CA, USA

The Future?

What I didn’t talk about...

- In parallel, we work on **computational (cryptographic) models and proofs**
- More fine-grained guarantees…

  … in the property and models
- **BUT:** Manual (pen and paper) proofs are often surprisingly coarse
  - many side cases not considered well
  - ongoing work on automation, but often partial or hard to scale
- Ongoing: first cryptographic proof of the core of the Signal Protocol
  - As used by TextSecure, Facebook, WhatsApp, …
  - Claims “future secrecy”… (See also our CSF 2016 paper on Post-Compromise Security)

Take away

- People design complex systems; hard to be confident
- Formal methods tools one way of increasing confidence in solutions
  - Now at a level where we impact real-world standards
  - Careful: One methodology not enough to provide high assurance; too error-prone
- Our tools all open source (github)
  - see my webpage etc. or drop me a mail (cas.cremers@cs.ox.ac.uk)