# Seven Years of Verifying Security Protocols

Lawrence C Paulson

UNIVERSITY OF CAMBRIDGE

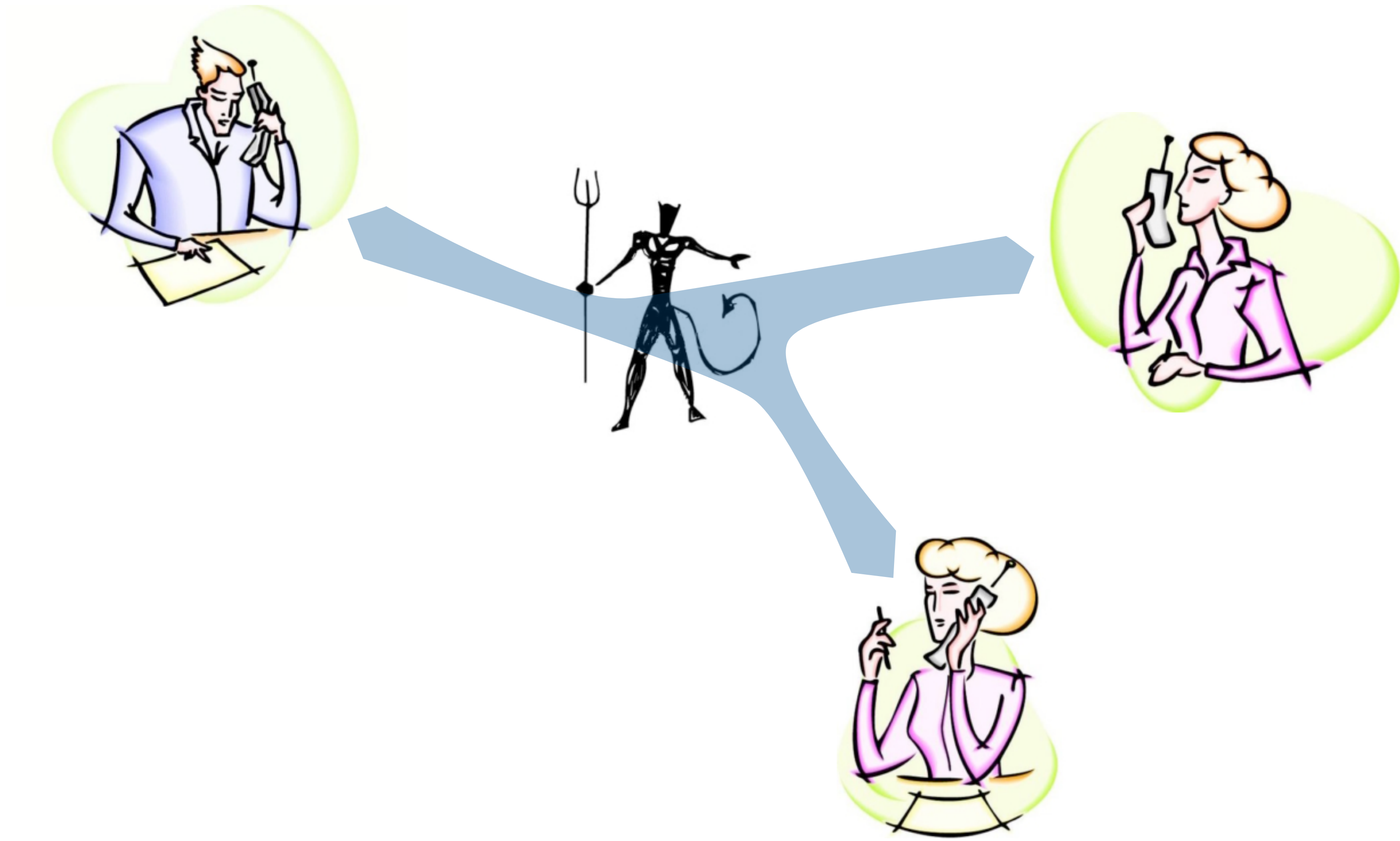
# Functions of Security Protocols

For secure communications on an open network in the presence of adversaries.

They ...

- authenticate the other party
- protect messages from tampering
- share sensitive information appropriately
- provide credentials that others can verify

# Is This Communication Secure?



# Operational Models of Systems

Used in model-checking and theorem-proving

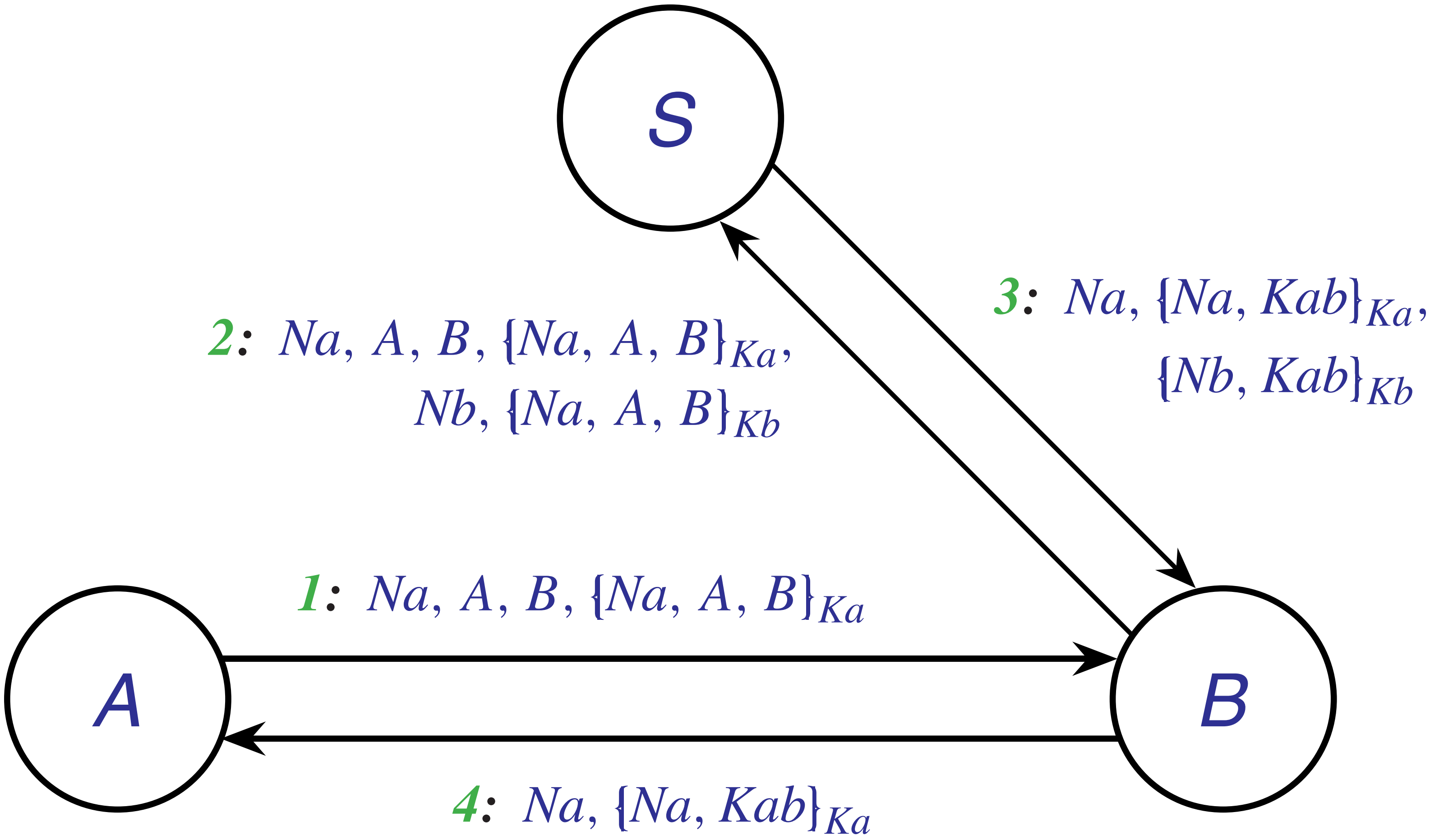
- Free algebra of message constructors: concatenation, encryption, etc.
- “Part-of” and similar relations on messages
- Perfect encryption and hashing
- Semantics based on traces of events

*Advantages*: Easy to formalize and to explain

# The Inductive Approach

- Each protocol specified by an inductive definition—a sort of logic program
- A common specification of the Dolev-Yao adversary: controls the network, etc.
- Security properties expressed in higher-order logic
- Theorems proved interactively by induction and simplification, using Isabelle

# A Variant Otway-Rees Protocol



# Formalization of Message 2

a fresh nonce

reference to the first message

```
OR2:  "[| evs2 ∈ otway;  Nonce NB ∉ used evs2;
       Gets B {|Nonce NA, Agent A, Agent B, X|} ∈ set evs2 |]
     ==> Says B Server
            {|Nonce NA, Agent A, Agent B, X, Nonce NB,
              Crypt (shrK B) {|Nonce NA, Agent A, Agent B|}|}
          # evs2 ∈ otway"
```

adding the next message to the trace

# A Secrecy Theorem

If KAB is a session key...

```
"[| evs ∈ otway;  KAB ∉ range shrK |] ==>
    (Key K ∈ analz (insert (Key KAB) (knows Spy evs))) =
    (K = KAB | Key K ∈ analz (knows Spy evs))"
```

...then a key can be broken with the help of *KAB* iff it *is KAB* or it can be broken anyway.

We can prove this theorem even though the protocol is flawed!

# Protocols Analysed Inductively

Classic authentication protocols:
*Otway-Rees, etc.*

Smartcard protocols

Multi-party protocols:
*recursive authentication, delegation, roving agents*

Non-repudiation protocols:
*Zhou-Gollmann, certified e-mail*

Industrial protocols:
*Kerberos, SSL, SET*

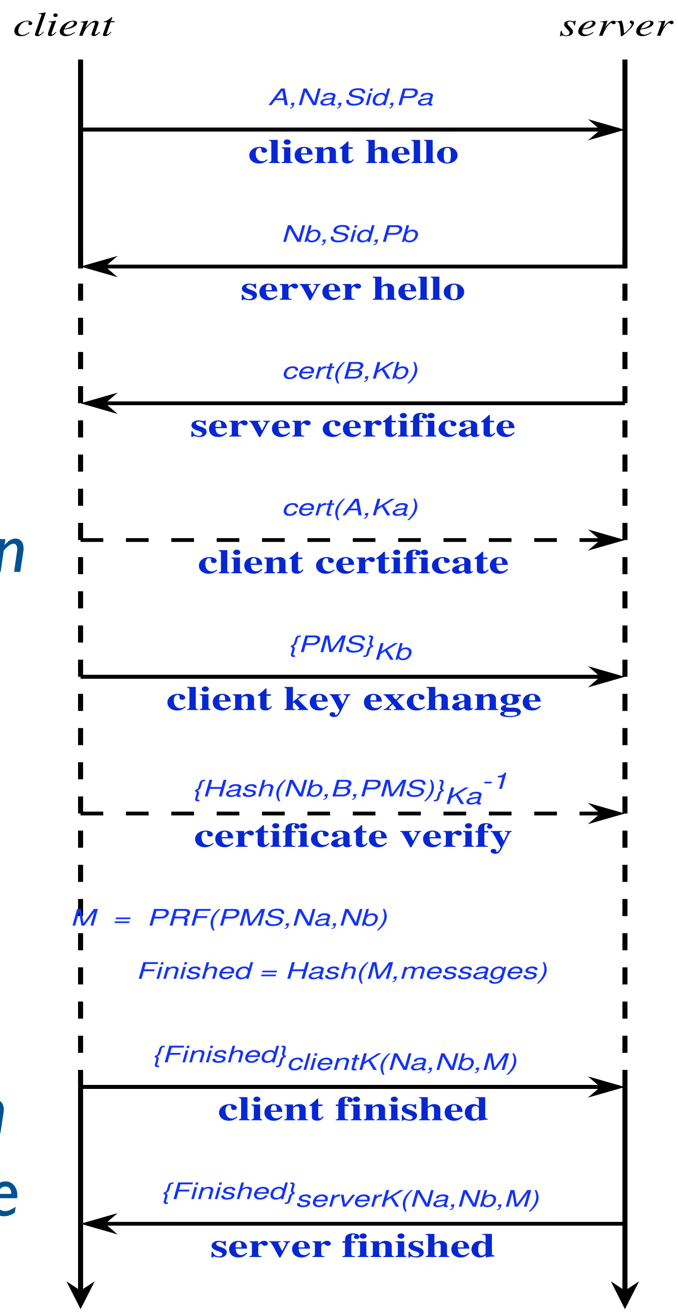
# Verifying TLS (or SSL 3.1)

- A detailed model including client authentication and session resumption.
- Eight messages; two optional paths; no limits on concurrent sessions.
- Elaborate system for creating session keys.
- From an 80 page official specification
- Proof done over six weeks in 1997

# The Message Flow of TLS

*Client Authentication (Optional)*

*A session resumption jumps straight to here*



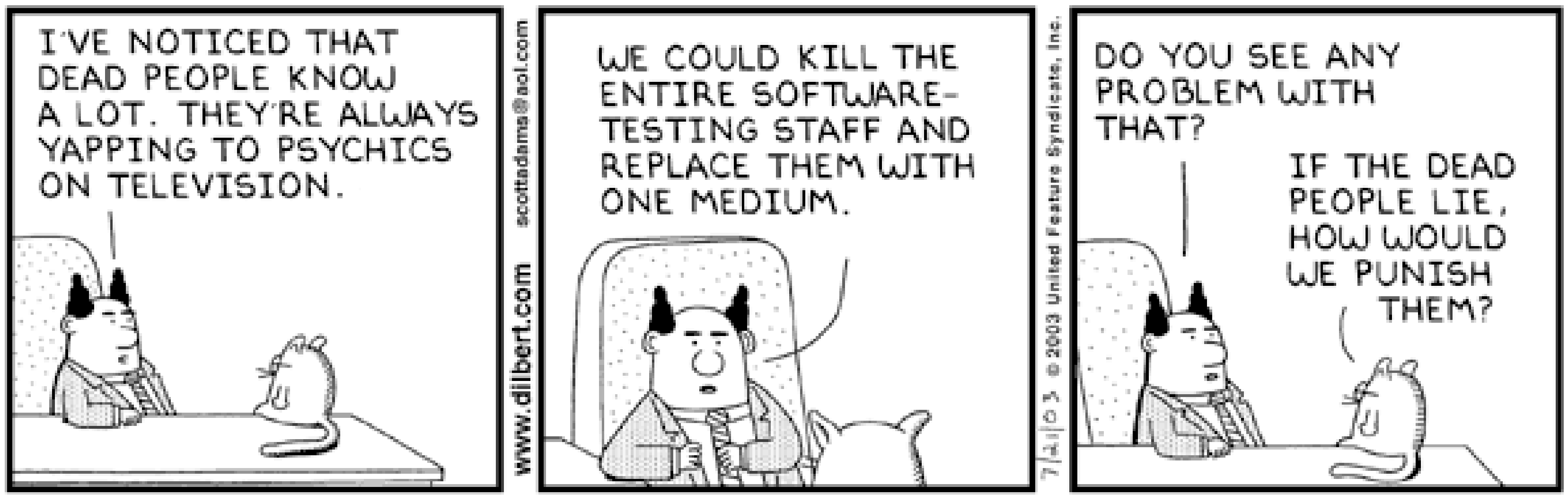
# Verifying the SET Protocols

- Several sub-protocols
- Complex cryptographic primitives
- Many types of principal: *Cardholders, Merchants, Payment Gateways, CAs*
- *Dual signatures*: partial sharing of secrets
- 1000 pages of specification and description
- The upper limit of realistic verification

# A Signed SET Purchase

```
[|evsPReqS ∈ set_pur;
  C = Cardholder k;
  CardSecret k ≠ 0;  Key KC2 ∉ used evsPReqS;  KC2 ∈ symKeys;
  Transaction = {|Agent M, Agent C, Number OrderDesc, Number PurchAmt|};
  HOD = Hash{|Number OrderDesc, Number PurchAmt|};
  OIData = {|Number LID_M, Number XID, Nonce Chall_C, HOD, Nonce Chall_M|};
  PIHead = {|Number LID_M, Number XID, HOD, Number PurchAmt, Agent M,
             Hash{|Number XID, Nonce (CardSecret k)|}|};
  PANData = {|Pan (pan C), Nonce (PANSecret k)|};
  PIData = {|PIHead, PANData|};
  PIDualSigned = {|sign (priSK C) {|Hash PIData, Hash OIData|},
                   EXcrypt KC2 EKj {|PIHead, Hash OIData|} PANData|};
  OIDualSigned = {|OIData, Hash PIData|};
  Gets C (sign (priSK M)
             {|Number LID_M, Number XID,
               Nonce Chall_C, Nonce Chall_M,
               cert P EKj onlyEnc (priSK RCA)|})
    ∈ set evsPReqS;
  Says C M {|Number LID_M, Nonce Chall_C|} ∈ set evsPReqS;
  Notes C {|Number LID_M, Transaction|} ∈ set evsPReqS |]
==> Says C M {|PIDualSigned, OIDualSigned|}
      # Notes C {|Key KC2, Agent M|}
      # evsPReqS ∈ set_pur"
```

# A Different Verification Method



# Benefits of Theorem Proving

*Yes, proofs are a lot of work, but they give ...*

- Flexibility:
  - specifying new types of system
  - choice in what to prove
- Expressiveness: no need to "program" the protocol and its desired guarantees
- Proof runs offer justification and insight

# Open Problems

- Formalization of large documents, identifying protocol goals and assumptions
  - two weeks for TLS; unending for SET
  - no technical solutions
- Relaxing the need for perfect encryption
- Understanding composition of primitives

# Protocol Implicit Assumptions

- The basis of many doubtful attacks
  - Needham-Schroeder: correct in its threat model
  - Viewing mobile phone protocols as network protocols (many false attacks against TMN)
  - Assuming distinct items to have the same length
  - Deliberately omitting required checks
  - Deliberately discarding essential records
- Modelling requires fair, informed judgement

# Beyond Perfect Encryption?

- Separation of concerns: protocol flaws versus crypto flaws
- Provable security: a more detailed model based on problem reduction
- *Abstract Cryptographic Library* (Backes et al.): a provably secure black-box abstraction
- Similar work by Abadi and Rogaway

# Composition of Primitives

- For protocols that assume secure channels established by another protocol
- For protocols that use digital envelopes and similar constructions
- Much work in progress, e.g. Datta et al.

# Conclusions

- Many substantial protocols can be analysed.
- Automatic tools make this almost easy.
- Theorem proving remains useful for modelling novel systems.
- Open questions are being pursued.