Using Rely/Guarantee to Pinpoint Assumptions underlying Security Protocols

Nisansala Yatapanage ANU, Australia
Cliff Jones Newcastle University, UK

Overture Workshop
Milano
2024-09-10

Plan

- challenges of formally describing security protocols and their assumptions
- quick reminder(?) of rely/guarantee idea
  - rely: as assumptions on environment
  - fault-tolerance = layered assumptions?
- our (incomplete) journey
- conclusions

Warning: more questions than answers!
about how to model?

Challenge of security protocols

- e.g. Needham/Schroeder (N-S) [NS78]

$(a1) A: enc([A, NA], pkeym(B))$
$(b1) B: enc([NA, NB], pkeym(A))$
$(a2) A: enc([NB], pkeym(B))$

    - flawed!
    - clear reasoning is non-trivial because of [Low95]
- challenge = proper specification!
    - including assumptions (about attackers, etc.)
    - contrast with listing the intended steps
      ... and looking for counter examples
    - it is clear that reasoning is non-trivial because N-S was around 18 years before Lowe's attack found
- assumptions
    - there are assumptions under which N-S is correct!
    - what are the assumptions for Lowe's "correction"?
    - can assumptions be used to identify run-time checks?
- most appropriate mental/metal tools for this study?

Our mental tools

vs. metal tools (JJH)

- abstraction
- abstraction
- $\Sigma / pre / post$
- (data) abstraction/reification
- Rely-Guarantee conditions
  - assumption/commitment disctinction
  - nested for fault tolerance
- abstraction, abstraction, abstraction, . . . , abstraction

Data: abstraction, reification + data type invariants (DTIs)

- data abstraction/reification in development methods
  more important than operation decomposition?
- most specifications use same collection of base types
- predicate restriction = DTI
  - useful (especially for future proofing)
  - DTIs as “meta pre/post conditions”
- R/G can became long (difficult to understand)
  - DTI as meta rely/guarantee conditions
  - reduces length/complexity of R/G conditions

Some mental tools to respond to the challenges

- one (common) idea is to abstract from encryption
- π-calculus, applied-π, spi-calculus, . . . ?
  - I have used π-calculus (e.g. Mondex paper with KGP)
  - but, I feel PAs wrong-level of abstraction
- special “belief” logics
  - ??
  - we try to avoid “belief/thinks” terminology
    the protocols are, after all, just code
- so, we’re trying to use:
  - $\Sigma/pre/rely/guar/post$

Rely/Guarantee “thinking”

- “top down” design/record from abstract specification
- basic idea (specs as relations):

$$\underbrace{\overbrace{\sigma_0}^{pre} \quad \cdots \quad \overbrace{\sigma_i\ \sigma_{i+1}}^{rely} \quad \cdots \quad \underbrace{\sigma_j\ \sigma_{j+1}}_{guar} \quad \cdots \quad \sigma_f}_{post}$$

- (skip proof rules here, just matching R/G)
- in a sense, just “think about assumptions”
- restricted expressiveness — has proved useful

“Relying on” the environment

- R/G originated as a (top-down) decomposition rule
- since applied to rely on non-developed components
  - physical components
  - can even “derive the spec of control system” [BHJ20]
- of course, don’t just “rely on”
  customer/deployer has to agree the assumptions

Furthermore:

- layered R/G for fault-tolerance
  - optimistic rely + ideal behaviour
  - weaker rely + less desirable guarantee

Needham/Schroeder (N-S)

$(a1) A: enc([A, NA], pkeym(B))$
$(b1) B: enc([NA, NB], pkeym(A))$
$(a2) A: enc([NB], pkeym(B))$

N-S is a testbed, not our final goal

$$NS(from, to) = sender(to) \parallel receiver()$$

would be easy, but we are interested in:

$$NS(from, to) = sender(to) \parallel receiver() \parallel other$$

this is where R/G come in?

Some modelling decisions

- Σ has complete *history* of all *Action*s (*Invent*/*Msg*)
  - *history* can only extend
  - $Invent :: Uid\ Nonce$
  - nonces are unique: *unique-nonces* is a fudge (probabilistic)
  - $Msg :: rec\colon Uid\ sender\colon Uid\ content\colon Item^*$
  - *sender* is a ghost variable (not knowable) except . . .
  - $Item = Uid \mid Nonce$
- Σ also has (for *post-NS*): $users\colon Uid \xrightarrow{m} User$
- *User* has $intPartner\colon Uid$ and $knows\colon Nonce$-**set**

an (optimistic) specification

- *post-NS* says *intPartner*s tie up;
  from/*to* have same *knows*?
  no other user has those *Nonce*s
- strong assumptions that would make N-S work:
  no-leaks $\triangleq$ can only send invented or received
  no-forge $\triangleq$ sign honestly

Lowe's attack

$(a1) A\colon enc([A, NA], pkeym(I))$
$(d1) I\colon enc([A, NA], pkeym(B))$
$(b1) B\colon enc([NA, NB], pkeym(A))$
$(d2) I\colon$ **skip**
$(a2) A\colon enc([NB], pkeym(I))$
$(d3) I\colon enc([NB], pkeym(B))$

- oddities:
  - A sends to (miscreant) I
  - only message $a1$ is signed (properly)
  - message $d1$ has a forged signature (important for attack)
  - message $a2$ actually gives NB to I!

Lowe's "correction"

$(a1) A\colon enc([A, NA], pkeym(I))$
$(d1) I\colon enc([A, NA], pkeym(B))$
$(b1) B\colon enc([B, NA, NB], pkeym(A))$
A **aborts** because $B \neq I$

- but this is a (post facto) test case
  telling, but not a spec
- what **is** the spec?
  authentication vs. key establishment [BMS19]

Layered R/G

- question each assumption:
  can it be checked at run time?
  if not, consequences and alternative assumptions
  e.g. $\mathit{no\text{-}leaks}$, can't check, so introduce $\mathit{conforms}$ (not honest)
- weaker assumptions
  - extra check
  - **abort** if intrusion detected
  - implementation has to satisfy both (all) layers of spec
    Lowe's correction still satisfies optimistic spec
- closing in on assumptions: $\mathit{conforms} \Rightarrow \cdots$

. . . onwards

- getting to encryption
  - certainly not unique to abstract away [SB10]
  - postponement also delays $dec(enc(\cdots))$
  - introduce $Skey$ in $User$ and $Pkey$ per Uid in Σ
  - new assumptions about visibility, uniqueness, . . .
- proof issues
  - $\nexists u \in \cdots\cdots\cdots$ prompts *reductio*
  - tempting, but . . .
- the "current version" of the paper (not as accepted!)
  - widen view of system to look at "context"
  - looks at $conforms(sender) + \neg\,\exists u \in Uid \cdot \cdots$
  - also $conforms(sender) \vee conforms(receiver)$
  - introduces $sess$ions, . . .

Back to metal tools

- Overture tool extensions?
- mechanisations of R/G
    - Diego [MD17]
    - Ian [HMWC19]
    - vs. POG for, say, Isabelle
- come and join us in the search?

Conclusions

- there's work to do!
- choice of best mental tools is not decided?
- tool support will matter (cf. CryptoVerif)

References

Alan Burns, Ian J. Hayes, and Cliff B. Jones.
Deriving specifications of control programs for cyber physical systems.
The Computer Journal, 63(5):774–790, 2020.

Colin Boyd, Anish Mathuria, and Douglas Stebila.
Protocols for authentication and key establishment.
Springer, 2nd edition, 2019.

Ian J. Hayes, Larissa A. Meinicke, Kirsten Winter, and Robert J. Colvin.
A synchronous program algebra: a basis for reasoning about shared-memory and event-based concurrency.
Formal Aspects of Computing, 31(2):133–163, 2019.
Online 6 August 2018.

Gavin Lowe.
An attack on the Needham-Schroeder public-key authentication protocol.
Information processing letters, 56(3), 1995.

Diego Machado Dias.
Mechanising an algebraic rely-guarantee refinement calculus.
PhD thesis, Newcastle University, 2017.

Roger M Needham and Michael D Schroeder.
Using encryption for authentication in large networks of computers.
Communications of the ACM, 21(12):993–999, 1978.

Christoph Sprenger and David Basin.
Developing security protocols by refinement.
In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 361–374, 2010.