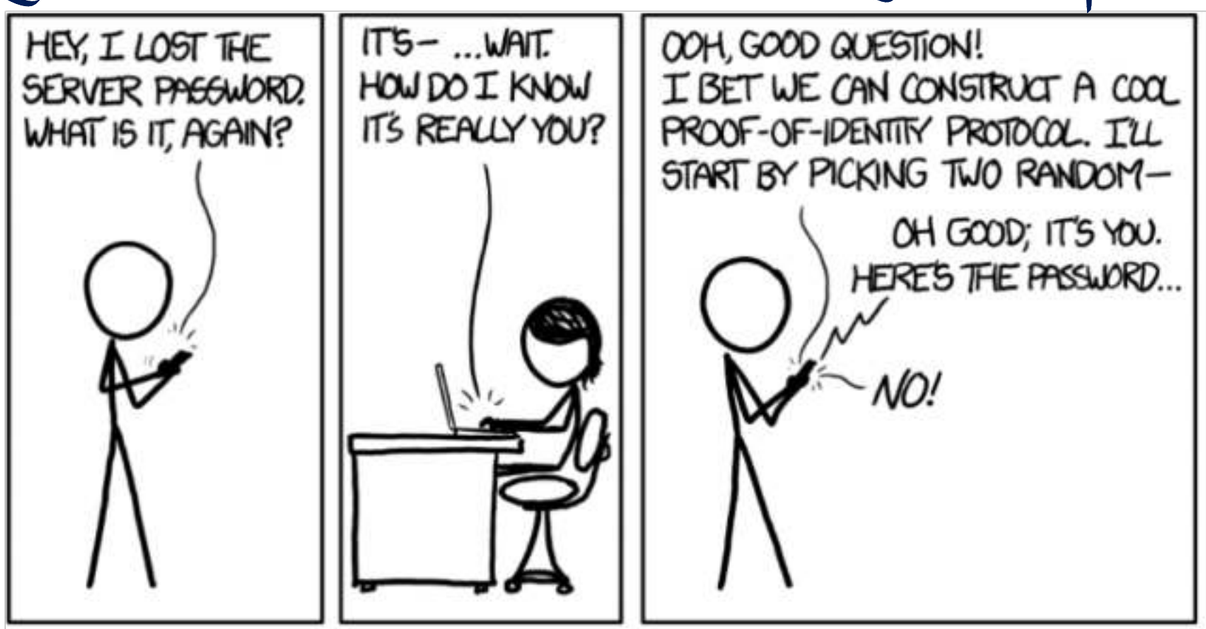
Proving Security Protocols Correct

– Vaishnavi Sundararajan

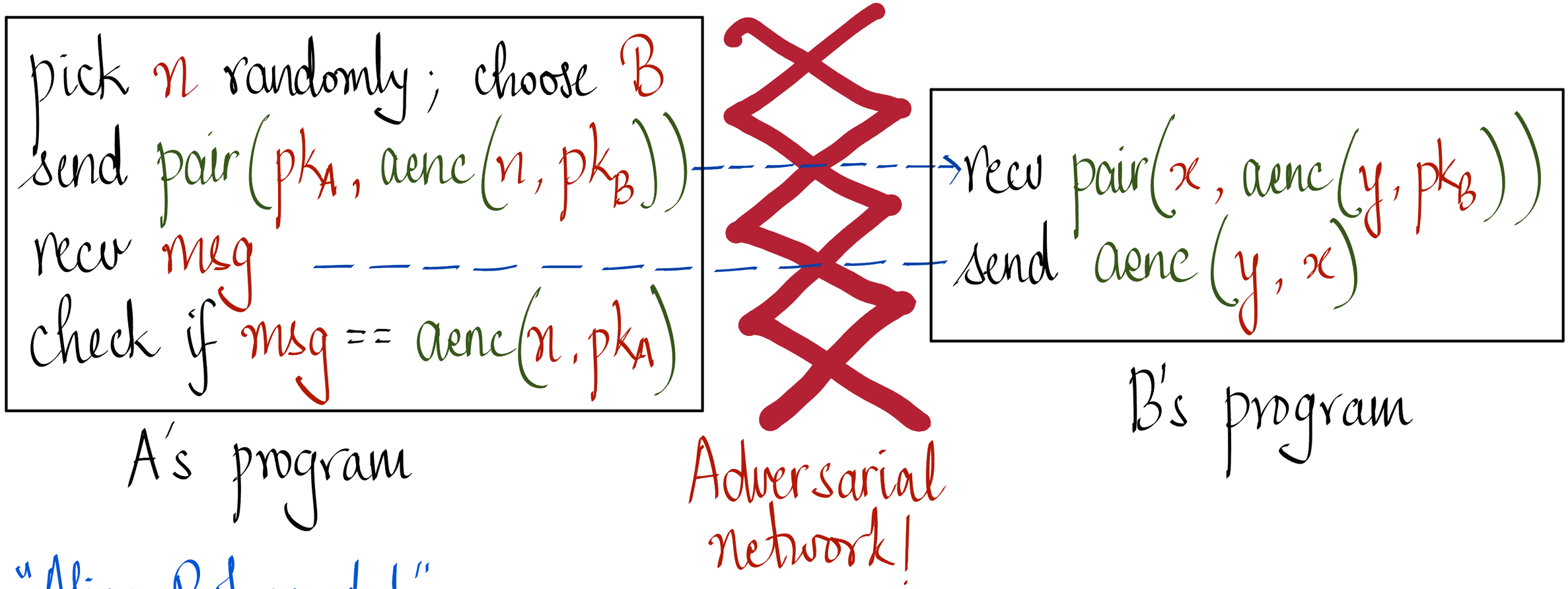
> "Security protocols are three-line programs that people still manage to get wrong"
>
> – Roger Needham

- What exactly is a security protocol?
- Any agreed-upon template for how to communicate securely
- Security protocols have existed forever
- Ciphers, passphrases, code books, Enigma....
- .... all the way into the digital age today!

- How exactly do we design a security protocol?
- Lots of considerations: what is the application?
  what crypto do we have access to?
  what machines will this run on?
  what threat model do we assume?
- Mostly, a lot of trial and error!
- If lucky, many stakeholders start using the protocol.



- Suppose many people do end up using it
- Then we need the protocol to guarantee secure communication
- Long history of proving cryptographic schemes secure
- But is secure crypto enough to guarantee secure communication?
- Often not! Hackers can exploit logical flaws
- Most recently, hacks on cryptocurrency like ETH (DAO, Ronin etc.)
- Safety-critical systems cannot be guaranteed correct by testing alone
- Needs rigorous guarantees; enter formal verification
- Example...

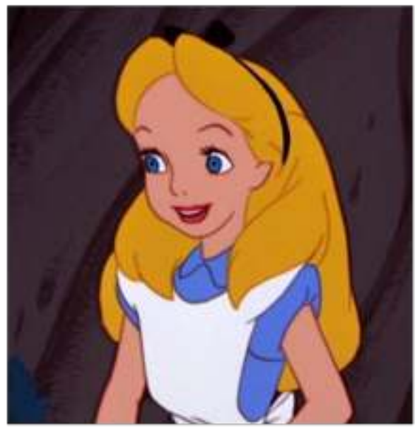


"Alice-Bob model"

A, B placeholder names, initialized at runtime

- What does it mean for this communication to be secure?
- Property of interest:
  After a valid execution of this protocol between A and B, nobody but A and B knows n.

Assume encryption to be perfect. Does this property hold?



pair(Alice, aenc(5, pk_{Bob}))

pair(Evee, aenc(5, pk_{Bob}))

aenc(5, pk_{Evee})

aenc(5, pk_{Alice})

What can the adversarial network do?

Flagship work from 1983:
(Dolev-Yao model)


On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

The "standard" adversary can:

see any message,
block any message,
redirect any message,
masquerade as anyone,
initiate new communication,
mix-and-match messages,
generate new messages,

every send goes to the adversary
} network itself assumed to be adversarial
every recv comes from the adversary
according to the protocol
} according to some rules

but cannot break the underlying "perfect" cryptography!

- How do we formally prove/disprove properties in this model?
- Collect information, perform inference

- But there is a lot of information! What are we interested in?
- Only the relevant parts of the communicated messages
- Actual messages might have metadata, headers etc.
- Formal model abstracts away unnecessary details

- Every communicated message modelled as a term in an algebra

$$t_1, t_2 := m \mid k \mid pk(k) \mid pair(t_1, t_2) \mid aenc(t_1, pk(k))$$

- Can generate new messages according to some rules of inference
- Suppose the adversary has some set X of terms already

$$\frac{}{X \vdash t}\ Ax\ [t \in X]$$

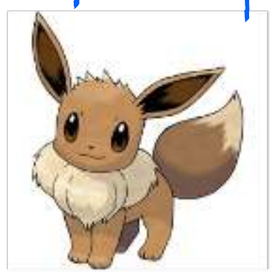
$$\frac{X \vdash t_0 \quad X \vdash t_1}{X \vdash pair(t_0, t_1)}\ pair \qquad \frac{X \vdash pair(t_0, t_1)}{X \vdash t_i}\ split_i$$

$$\frac{X \vdash t \quad X \vdash pk(k)}{X \vdash aenc(t, pk(k))}\ aenc \qquad \frac{X \vdash aenc(t, pk(k)) \quad X \vdash k}{X \vdash t}\ adec$$

$X \vdash t$: There is a proof of t from X using some sequence of these rules

- Can we formally show that n is not secret in our example?
- Need a notion of how the adversary's knowledge grows.
- Each send is captured by the adversary
- Valid execution: each receive needs to be derivable by the adversary
- Property of interest:
  After a valid execution of this protocol between A and B, nobody but A and B knows n.
- Equivalent formulation:
  Is there a valid execution of this protocol between A and B where a Dolev-Yao adversary can get to know n?

$$X_0 = \{ \text{Alice}, \text{Bob}, \text{Evee}, pk_{\text{Alice}}, pk_{\text{Bob}}, pk_{\text{Evee}}, sk_{\text{Evee}} \}$$



$= pk(sk_{\text{Evee}})$

$pair(\text{Alice}, aenc(5, pk_{\text{Bob}}))$ $\quad X_1 = X_0 \cup \{pair(\text{Alice}, aenc(5, pk_{\text{Bob}}))\}$

Does $X_1 \vdash^{?} pair(\text{Evee}, aenc(5, pk_{\text{Bob}}))$

$X_2 = X_1 \cup \{aenc(5, pk(sk_{\text{Evee}}))\}$ $\quad aenc(5, pk(sk_{\text{Evee}}))$

$$\dfrac{\dfrac{}{X_1 \vdash \text{Evee}}\,Ax \qquad \dfrac{\dfrac{}{X_1 \vdash pair(\text{Alice}, aenc(5, pk_{\text{Bob}}))}\,Ax}{X_1 \vdash aenc(5, pk_{\text{Bob}})}\,split_1}{X_1 \vdash pair(\text{Evee}, aenc(5, pk_{\text{Bob}}))}\,pair$$

Claim: $X_2 \vdash 5$

- Given X and t, how easy/difficult is it to check if $X \vdash t$?
- For some notion of a "well-behaved" proof of $X \vdash t$, not that difficult
- Polynomial in the input* → no "unnecessary detours"
- This is the Passive Intruder Problem
  - If the adversary does nothing other than observe network traffic, what can they still get to know?
- Most protocols are secure against passive adversaries
- More interesting problem: Active Intruder Problem
  - Is there any valid execution where a Dolev-Yao adversary can violate the property required of the protocol?
- How difficult is it to solve this problem?

- **Active Intruder Problem**: Is there any valid execution where a Dolev-Yao adversary can violate the property required of the protocol?
- What can a valid execution involve? Two sources of infinity!
  1. Unboundedly many copies of each role ("sessions")
  2. Unboundedly large terms injected by the adversary in a recv
- Get around (1) by only considering a bounded number of sessions
- Given boundedly many sessions, (2) solved by Rusinowitch & Turuani:
  - The adversary will not inject a larger term if a small one will provide the same "effect" [RT03]
- Active intruder problem for boundedly many sessions is NP-complete
  Guess an interleaving of roles and small terms injected by adversary
  Polynomially many derivability checks, each done in PTIME.

- Lots of tools built on the Dolev-Yao model to automate verification
  - ProVerif, Tamarin, DeepSec, DY*....
- Allow you to add new functions (XOR, homomorphic encryption, signatures..)
- Good for properties like secrecy
  - Safety property (trace-based) "Is there any run where...."
  - Only need to examine one execution at a time
- Not so good for properties like privacy

  "Nobody should be able to link my actions to my name"
  - Equivalence property (multiple runs are indistinguishable to adversary)
  - Need to examine multiple runs simultaneously
  - Often results in non-termination!

A Formal Analysis of 5G Authentication

David Basin
Department of Computer Science
ETH Zurich
Switzerland
basin@inf.ethz.ch

Jannik Dreier
Universite de Lorraine
CNRS, Inria, LORIA
Nancy, France
jannik.dreier@loria.fr

Lucca Hirschi
Department of Computer Science
ETH Zurich
Switzerland
lucca.hirschi@inf.ethz.ch

Saša Radomirović
School of Science and Engineering
University of Dundee
UK
s.radomirovic@dundee.ac.uk

Ralf Sasse
Department of Computer Science
ETH Zurich
Switzerland
ralf.sasse@inf.ethz.ch

Vincent Stettler
Department of Computer Science
ETH Zurich
Switzerland
svincent@student.ethz.ch

A Formal Security Analysis of the Signal Messaging Protocol

Extended Version, July 2019†

Katriel Cohn-Gordon*, Cas Cremers†, Benjamin Dowling‡, Luke Garratt§, Douglas Stebila¶
*Independent Scholar me@katriel.co.uk
†CISPA Helmholtz Center for Information Security, Germany cremers@cispa.saarland
‡ETH Zürich, Switzerland benjamin.dowling@inf.ethz.ch
§Cisco Systems, USA lgarratt@cisco.com
¶University of Waterloo, Canada dstebila@uwaterloo.ca

Formal Analysis of EDHOC Key Establishment for Constrained IoT Devices

Karl Norrman[1,2], Vaishnavi Sundararajan[3] and Alessandro Bruni[4]
[1]KTH Royal Institute of Technology, Stockholm, Sweden
[2]Ericsson Research, Security, Stockholm, Sweden
[3]University of California Santa Cruz, USA
[4]IT University of Copenhagen, Copenhagen, Denmark
karl.norrman@ericsson.com, vasundar@ucsc.edu, brun@itu.dk

A Symbolic Analysis of Privacy for TLS 1.3 with Encrypted Client Hello

Karthikeyan Bhargavan
Inria Paris
Paris, France
karthikeyan.bhargavan@inria.fr

Vincent Cheval
Inria Paris
Paris, France
vincent.cheval@inria.fr

Christopher Wood
Cloudflare
San Francisco, United States
chriswood@cloudflare.com

Formal Verification of Smart Contracts

Short Paper

Karthikeyan Bhargavan[1], Antoine Delignat-Lavaud[2], Cédric Fournet[2], Anitha Gollamudi[3], Georges Gonthier[2], Nadim Kobeissi[1], Natalia Kulatova[1], Aseem Rastogi[2], Thomas Sibut-Pinote[1], Nikhil Swamy[2], and Santiago Zanella-Béguelin[2]
[1]Inria
{karthikeyan.bhargavan,nadim.kobeissi,natalia.kulatova,thomas.sibut-pinote}@inria.fr
[2]Microsoft Research
{antdl,fournet,gonthier,aseemr,nswamy,santiago}@microsoft.com
[3]Harvard University
agollamudi@g.harvard.edu

- What about complicated, real-world authorization/delegation protocols?
- If I allow you to edit my Google Doc, I transfer an access credential
- Different in spirit from storing an encrypted file on the cloud
- But both would be modelled as terms in the same algebra!

- Some of my recent work involves adding a new algebra for certificates
- fragment of first-order logic: AND, EXISTS, list membership "assertions"
- Makes protocol specifications easier to read [RSS17]
- More expressive, but the secrecy problem remains NP-complete [RSS24]

- Consider a voting protocol
- Voter sends their identity credentials to an authority
- The authority authorizes the voter to vote
- The voter then anonymously sends their vote to the collector
- How is the collector to know that this came from an authorized voter?

- Authority often "modifies" an encrypted version of the vote
- Why encrypted? Because anonymity
  "Nobody should be able to link a voter to their vote"
- But encryption needs to be stripped off while sending to the collector

Foo E-voting protocol:

$V \to A: sign(enc(v, k_A), V)$

$A \to V: bsign(enc(v, k_A), A)$

$\to C: enc(bsign(v, A), k_C), k_C$

$bsign(enc(t,k), X) = enc(bsign(t, X), k)$

With assertions, this becomes

$V \to A: enc(v, k_A), V \text{ says } \exists xy.[enc(v,k_A) = enc(x,y) \wedge cand(x)]$

$A: assert\ elg(v)$

$A \to V: A \text{ says } [V \text{ says } \exists xy.[enc(v,k_A) = enc(x,y) \wedge cand(x)] \wedge elg(v)]$

$V \dashrightarrow C: enc(v, k_c), k_c,$

$\exists X w z.[[A \text{ says } [X \text{ says } \exists xy.[enc(w,z) = enc(x,y) \wedge cand(x)] \wedge elg(x)] \wedge (w = v)]$

Current and future work:

- Assertions implicitly link multiple terms together
- Appears to help simplify properties like privacy, anonymity etc.
  - ★ Can cast these as safety properties; only examine one run at a time
- Have example-specific casts for such properties
- Want a general, protocol-agnostic formulation
- How far can we extend the language but still stay NP-complete?
- Extending some tools with assertions for automation

Want to know more? Get in touch!

Vaishnavi@cse.iitd.ac.in
https://vaishs.github.io