Engineering Secure Software Systems

January 5, 2021: ProVerif: Background and First Examples (Lockdown & Power Outage Edition)

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Part I: Crypto Protocols

Overview

Overview

Logic Modeling Outline

facts

- $d(\hat{k}_C)$
- $d(\{0,1\})$
- $\ldots$

DY deductions

- $d(\mathrm{enc}^{\mathrm{a}}_{k_C}(x)) \land d(\hat{k}_C) \to d(x)$
- $d(x) \land d(y) \to d([x,y])$
- $d(x) \to d(\mathrm{hash}(x))$
- $\ldots$

protocol deductions

- $d(\mathrm{enc}^{\mathrm{a}}_{k_B}([A,x])) \to d(\mathrm{enc}^{\mathrm{a}}_{k_A}([B,x]))$
- $d(\mathrm{enc}^{\mathrm{a}}_{k_A}([B,x,y])) \to (\mathrm{enc}^{\mathrm{a}}_{k_B}(y))$
- $\ldots$

Horn clauses

$$(x_1 \land x_2 \land \cdots \land x_n \to y) \quad \leftrightarrow \quad (\overline{x_1} \lor \overline{x_2} \lor \cdots \lor \overline{x_n} \lor y)$$

target clause

$\neg d(\mathrm{FAIL})$

Exercise

Task (Needham-Schroeder as Horn clauses)

Model the Needham-Schroeder protocol as Horn clauses and use this formalism to show that the protocol is insecure. To do this, first list the facts, Dolev-Yao deductions, protocol deductions and the target clause. Then, use logical inference to show that the protocol is in fact insecure. Do you see any limits or imprecisions in this approach?

Overview

Applying the Theory: Analysis in Practice

so far in lecture

- formal model
- formal security properties
- decidability and complexity results
- Horn modeling
- ~~abstraction~~

now: application of the theory

different analysis tools

- **ProVerif (Bruno Blanchet, Vincent Cheval)**
- CryptoVerif (Bruno Blanchet, David Cadé)
- FDR (Formal Systems Europe)

ProVerif in Lecture I

outline

- ProVerif introduction (syntax, semantics, examples)
- generalized cryptographic primitives in ProVerif
- extended security properties in ProVerif

no detailed theory (literature references)

key references

paper Bruno Blanchet. “Using Horn Clauses for Analyzing Security Protocols”. In: Cryptology and Information Security Series 5 (2011), pp. 86–111

update Vincent Cheval, Véronique Cortier, and Mathieu Turuani. “A Little More Conversation, a Little Less Action, a Lot More Satisfaction: Global States in ProVerif”. In: 31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018. IEEE Computer Society, 2018, pp. 344–358. ISBN: 978-1-5386-6680-7. DOI: 10.1109/CSF.2018.00032. URL: https://doi.org/10.1109/CSF.2018.00032

tool http://prosecco.gforge.inria.fr/personal/bblanche/proverif/

ProVerif in Lecture II

learning goals

practice	analyzing protocols with ProVerif
security properties	study of complex properties (without detailed formal model)
conceptual	undecidable problems in real life
	consequences for tool application

not a learning goal

complete introduction to ProVerif (see reading exercise on Needham-Schroeder modeling)

ProVerif Applications

non-trivial examples

- certified email protocol, including ssh layer, JFK protocol (candidate for IKE replacement), secure filesystem Plutus, web service verification
- e-voting protocols, authenticated routing, zero knowledge protocols
- TLS (F# implementation for .NET), 5G EAP-TLS
- Telegram, Bitcoin Smart Contracts, Healthcare protocols
- ...(see Google Scholar, ProVerif since 2020)

original reference

Bruno Blanchet. “Using Horn Clauses for Analyzing Security Protocols”. In: Cryptology and Information Security Series 5 (2011), pp. 86–111

ProVerif

features

- analysis of protocols in symbolic model
- can handle an unbounded number of protocol sessions (necessarily incomplete)
- user-provided specification of cryptographic primitives and security properties

example security properties

- secrecy
- strong secrecy
- authentication
- correspondence
- observational equivalence

incompleteness consequences

- “don’t know”
- non-termination

Overview

ProVerif „Hello, World“

`hello.pv` (from PV doc)

```
free c:channel.

free Plain:bitstring [private].
free RSA:bitstring [private].

query attacker(RSA).
query attacker(Plain).

process
  out(c,RSA);
  0
```

elements

free	free algebraic variables
channel	communication channel
bitstring	data type
private	$\mathcal{A}$ cannot (directly) access
query	security property
out	send on channel

execution

- send “private” value RSA on public channel
- query: secrecy for
  - RSA
  - Plain

ProVerif usage

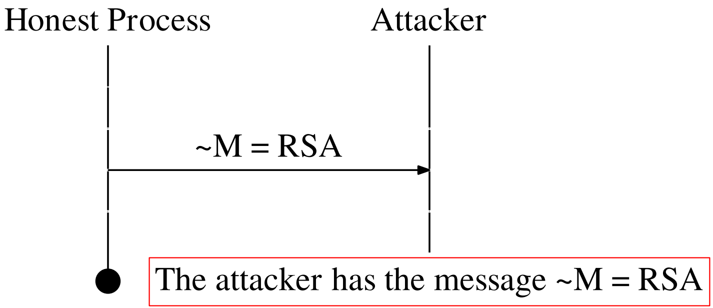
command line tool

```
$ proverif 2021_01_05_lecture_08/01_hello.pv
$ proverif -graph targetDir 2021_01_05_lecture_08/01_hello.pv
$ proverif -html targetDir 2021_01_05_lecture_08/01_hello.pv
```

live demo

ProVerif Analysis Result: Hello, World

A trace has been found.



example: handshake (from ProVerif tutorial)

$$
\begin{array}{ll}
A \to B & k_A \\
B \to A & \mathrm{enc}^{\mathrm{a}}_{k_A}\left(\mathrm{sig}_{k_B}([k_B, k])\right) \\
A \to B & \mathrm{enc}^{\mathrm{s}}_{k}(\mathrm{FAIL})
\end{array}
$$

properties?

- intended security properties?
- protocol secure?

attack

$$
\begin{array}{ll}
\mathcal{A} \to B & k_{\mathcal{A}} \\
B \to \mathcal{A} & \mathrm{enc}^{\mathrm{a}}_{k_{\mathcal{A}}}\left(\mathrm{sig}_{k_B}([k_B, k])\right) \\
A \to B & k_A \\
\mathcal{A} \to A & \mathrm{enc}^{\mathrm{a}}_{k_A}\left(\mathrm{sig}_{k_B}([k_B, k])\right) \\
A \to B & \mathrm{enc}^{\mathrm{s}}_{k}(\mathrm{FAIL})
\end{array}
$$

fix

add receiver to message

steps

1. specify cryptographic primitives
2. specify communication infrastructure
3. specify adversary goal
4. specify Alice & Bob
5. specify “main process”

lecture so far

behavior of crypto primitives

- asymmetric encryption
- symmetric encryption
- signatures
- hash functions

ProVerif: flexible modeling of primitives

technique: equational theories, more general than DY-like specification ısee example scripts

Overview

definition

- equation: pair (l, r) of terms, also written $l = r$
  - left: “complex” term
  - right: “simple” term
- equational theory: set E of equations

caveat: choose term signature matching to E (implicit)

rewrite relation

- $t_1 \twoheadrightarrow_E t_2$: t_2 obtained from t_1 by applying rule from E
- $\twoheadrightarrow_E^*$: closure of $\twoheadrightarrow_E$ under transitivity, reflexivity, application of function symbols
- $\equiv_E$: closure of $\twoheadrightarrow_E^*$ under symmetry and transitivity

Equational Theories: Examples

primitives

- asym. encryption $\text{dec}^{\text{a}}_{\hat{k}_A}\left(\text{enc}^{\text{a}}_{k_a}(x)\right) = x$
- sym. encryption $\text{dec}^{\text{s}}_{k}\left(\text{enc}^{\text{s}}_{k}(x)\right) = x$
- signature
    - $\text{check}(k_A, \text{sig}_{k_A}(x)) = \text{ok}$
    - $\text{extr} - \text{key}(\text{sig}_{k_A}(x)) = k_A$
    - $\text{extr} - \text{msg}(\text{sig}_{k_A}(x)) = x$
- hash function
- bit commitment $\text{open}\left(\text{bc}\left(k, b\right), k\right) = b$
- trapdoor commitments
    - $\text{open}\left(\text{tdc}\left(m, r, t_d\right), r\right) = m$
    - $\text{tdc}\left(m_2, f(m_1, r, t_d, m_2), t_d\right) = \text{tdc}\left(m_1, r, t_d\right)$

“Simple” Equational Theories

definition

$\twoheadrightarrow_E$ is

- confluent, if for all t, t_1, t_2 with $t \twoheadrightarrow_E^* t_1$ and $t \twoheadrightarrow_E^* t_2$ there is some t' with $t_1 \twoheadrightarrow_E^* t'$ and $t_2 \twoheadrightarrow_E^* t'$.
- terminating, if there is no infinite sequence $t_1, t_2, \ldots$ with $t_i \neq t_{i+1}$ and $t_i \twoheadrightarrow_E t_{i+1}$ for all i.
- E is convergent, if $\twoheadrightarrow_E$ is confluent and terminating
- E is convergent subterm theory, if
  - E is convergent, and
  - for all $(l, r) \in E$: r is subterm of l or constant

definition

A term t is in E-normal-form if $t = t'$ for all $t \twoheadrightarrow_E t'$.

lemma & definition

If E is convergent, then for every term t, there is a unique term $[t]$ with

- $[t]$ is in E-normal-form,
- $t \equiv_E [t]$.

lemma

$t \equiv_E t'$ iff $[t] = [t']$.

computation of normal form

How do we compute $[t]$ from t? for a convergent theory?

Equational Theories: Examples Revisited

primitives

- asym. encryption $\mathrm{dec}^{a}_{\hat{k}_A}\left(\mathrm{enc}^{a}_{k_a}(x)\right) = x$
- sym. encryption $\mathrm{dec}^{s}_{k}\left(\mathrm{enc}^{s}_{k}(x)\right) = x$
- signature
  - $\mathrm{check}(k_A, \mathrm{sig}_{k_A}(x)) = \mathrm{ok}$
  - $\mathrm{extr} - \mathrm{key}(\mathrm{sig}_{k_A}(x)) = k_A$
  - $\mathrm{extr} - \mathrm{msg}(\mathrm{sig}_{k_A}(x)) = x$
- hash function
- bit commitment $\mathrm{open}\left(\mathrm{bc}\left(k, b\right), k\right) = b$
- trapdoor commitments
  - $\mathrm{open}\left(\mathrm{tdc}\left(m, r, t_d\right), r\right) = m$
  - $\mathrm{tdc}\left(m_2, f(m_1, r, t_d, m_2), t_d\right) = \mathrm{tdc}\left(m_1, r, t_d\right)$

discussion

- complexity?
- observations?

remember

modeling primitives at this level of abstraction loses details

algorithms

algorithms discussed so far do not cover all of these

Exercise

Task (Missing Proof)

Prove the following lemma that was stated in the lecture without proof:

If E is a convergent equational theory, then:

1. For every term t, there is a unique term $[t]$ with
   - $[t]$ is in E-normal-form,
   - $t \equiv_E [t]$.
2. For terms t and t', we have that $t \equiv_E t'$ if and only if $[t] = [t']$.

Exercise

Task (“Badly-Behaved” Equational Theories)

Define equational theories for which the resulting rewrite relation $\twoheadrightarrow_E$ is not a convergent subterm theory, i.e., one that is not confluent, not terminating, or not a subterm theory.

Theorem

For convergent subterm theories, the following problems are polynomial-time decidable:

- given E, t, t', does $t \twoheadrightarrow_E t'$ hold?
- given E, t, t', is $t \equiv_E t'$?

Theorem

Also computable in polynomial time: given E, t, compute $[t]$ (DAG representation)

reference

Martın Abadi and Véronique Cortier. “Deciding knowledge in security protocols under equational theories”. In: Theoretical Computer Science 367.1-2 (2006), pp. 2–32

Needham Schroeder

$$
\begin{array}{llcl}
\text{Alice} & \epsilon & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_A) \\
 & \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, y) & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}(y) \\
 & & & \\
\text{Bob} & \mathrm{enc}^{\mathrm{a}}_{k_B}(A, x) & \to & \mathrm{enc}^{\mathrm{a}}_{k_A}(x, N_B)
\end{array}
$$

equational theory

$$\mathrm{dec}^{\mathrm{a}}_{\hat{k}_A}\left(\mathrm{enc}^{\mathrm{a}}_{k_a}(x)\right) = x$$

“simplification”

- new notation for protocols?
- advantage?

additional equations for pairing

- $\mathrm{split1}(\mathrm{pair}(x, y)) = x$
- $\mathrm{split2}(\mathrm{pair}(x, y)) = y$

Alice

$$
\begin{array}{lcl}
\epsilon & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_A) \\
x & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}\left(\mathrm{split2}\left(\mathrm{dec}^{\mathrm{a}}_{\hat{k}_A}(x)\right)\right)
\end{array}
$$

Bob

$$y \to \mathrm{enc}^{\mathrm{a}}_{k_A}\left(\mathrm{pair}\left(\mathrm{split2}\left(\mathrm{dec}^{\mathrm{a}}_{\hat{k}_B}(y)\right), N_B\right)\right)$$

Handshake-Protocol in ProVerif: Primitives

symmetric encryption

type key

fun senc(bitstring, key): bitstring

reduc forall m:bitstring, k:key; sdec(senc(m,k),k) = m

asymmetric encryption

type skey

type pkey

fun pk(skey): pkey

fun aenc(bitstring, pkey): bitstring

reduc forall m:bitstring, sk:skey; adec(aenc(m,pk(sk)),sk) = m

signatures

type sskey

type pskey

fun spk(sskey): spkey

fun sign(bitstring, sskey): bitstring

reduc forall m:bitstring, ssk:sskey; getmess(sign(m,ssk)) = m

reduc forall m:bitstring, ssk:sskey; checksign(sign(m,ssk),spk(ssk)) = m

Handshake-Protocol in ProVerif: Communication and Adversary Goal

channel, values, attacker goal

```
free c:channel
free FAIL:bitstring [private]
query attacker (FAIL)
```

models

- c is public channel
- FAIL: bitstring, private (not in initial $\mathcal{A}$ knowledge)
- security property: $\mathcal{A}$ cannot derive FAIL

protocol

$$\begin{array}{ll} A \rightarrow B & k_A \\ B \rightarrow A & \mathrm{enc}^{a}_{k_A}\left(\mathrm{sig}_{k_B}\left([k_B, k]\right)\right) \\ A \rightarrow B & \mathrm{enc}^{s}_{k}\left(\mathrm{FAIL}\right) \end{array}$$

client (Alice)

```
let clientA(pkA:pkey,skA:skey,pkB:spkey)=
    out (c,pkA)
    in (c,x:bitstring)
    let y=adec(x,skA) in
        let (=pkB,k:key)=checksign(y,pkB) in
            out (c,senc(FAIL,k))
```

features

- keys as arguments
- FAIL global value
- decryptions explicit (pattern matching in formal model)

protocol

$A \rightarrow B \quad k_A$

$B \rightarrow A \quad \mathrm{enc}^{a}_{k_A}\left(\mathrm{sig}_{k_B}\left([k_B, k]\right)\right)$

$A \rightarrow B \quad \mathrm{enc}^{s}_{k}\left(\mathrm{FAIL}\right)$

server (Bob)

```
let serverB(pkB:spkey,skB:sskey)=
    in (c,pkX:pkey)
    new k:key
    out (c,aenc(sign((pkB,k),skB),pkX))
    in c,x:bitstring
    let z=sdec(x,k) in 0
```

features

- type checking on message receive
- last line: “no match” if decryption fails

Handshake-Protocol in ProVerif: Main Process

previous processes

- clientA(pkA:pkey,skA:skey,pkB:spkey)
- serverB(pkB:spkey,skB:sskey)

main process

```
new skA:skey
new skB:sskey
let pkA=pk(skA) in out(c,pkA)
let pkB=spk(skB) in out(c,pkB)
( (!clientA(pkA,skA,pkB) | (!serverB(pkB,skB))) )
```

features

- key generation, sending of pkA, pkB on public channel
- call of Alice and Bob with matching parameters
- !: "replication operator"

ProVerif Script: Handshake Protocol

command line tool

```
$ proverif 2021_01_05_lecture_08/02_handshake.pv
$ proverif -graph targetDir 2021_01_05_lecture_08/02_handshake.pv
$ proverif -html targetDir 2021_01_05_lecture_08/02_handshake.pv
```

live demo

ProVerif Analysis Result: Handshake