Engineering Secure Software Systems

December 1, 2020: Automatic Analysis of Crypto Protocols—The Rusinowitch-Turuani Theorem

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Part I: Crypto Protocols

Overview

Overview

(IN)SECURE Examples and Security Proofs

seen

definition of security for protocols

examples

- Needham-Schroeder protocol is insecure
- Needham-Schroeder-Lowe protocol is secure

really? missing?

- NSL formalization with BREAK/FAIL (easy)
- **security proof**

security proofs in the lecture

- past: proof of example protocol as discussion on blackboard
- in class: no complete security proof (time/benefit tradeoff)
- example: see solution of exercise 2.2, contains “handwaving”

key point in lecture

- manual analysis: expensive and error-prone
- alternative: automatic analysis

consequence

focus on automatic analysis

Exercise

Task (Formal Protocol Model: Features and Omissions)

There are a couple of usually assumed properties of cryptographic systems that are not explicitly expressed in our protocol model. Which of the following properties are implicitly expressed in our model, and which are not? Are any of the “omissions” problematic?

- Nonces are indeed used only once, and are freshly generated for each session.
- Private keys are never sent over the network.
- There is a complete public-key infrastructure PKI available.
- Nonces are long enough so that the adversary cannot guess them correctly.
- The adversary knows the involved algorithms, including the protocol (Kerckhoffs’s principle).
- In the absence of an adversary, the network simply forwards the protocol participant’s messages as intended.

Overview

Overview

Goal: Automatic Analysis

security definition

A protocol P is secure if there is no successful attack on P.
(Recall complex definition of successful attack)

computational decision problem

Problem:	INSECURE
Input:	protocol P (including initial adversary knowledge I)
Question:	is there a successful attack on P?

algorithm for INSECURE?

is the problem decidable?

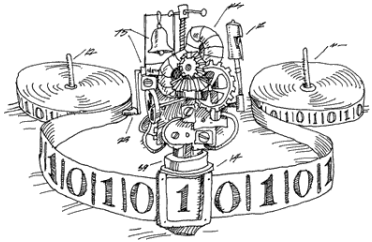
Obstacles to Decidability

related problems

- undecidable problems for term rewriting systems
- term unifiers can be exponentially large (exercise)
- can express **if**/**then** — encode halting problem?

simpler setting

- simple scenario: fixed number of sessions
- fixed number of r/s actions
- simple term replacement rules



The Rusinowitch-Turuani Theorem

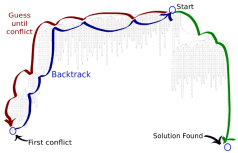
theorem

INSECURE is NP-complete.

Michaël Rusinowitch and Mathieu Turuani. “Protocol insecurity with a finite number of sessions, composed keys is NP-complete”. In: Theoretical Computer Science 1-3.299 (2003), pp. 451–475

consequences

- problem is decidable
- (probably) no efficient algorithm, but
  - possible for small instances (“three line programs”)
  - approach with SAT-solver, constraint solver, …
- many extensions proved since then, see later



proof (today & next week)

- presentation based on original paper
- consider only encryption and nonces

P and NP: A Brief Reminder

machine model

Turing Machine (TM): abstraction of (e.g.) Java programs

P: determinism
- problems solvable in polynomial time on deterministic machines
- “normal” efficient algorithms

NP: nondeterminism
- problems solvable in polynomial time on non-deterministic machines
- “magic guess” algorithms

examples for NP problems
- satisfiability (NP-c)
- clique (NP-c)
- graph isomorphism
- subgraph isomorphism (NP-c)
- integer factorization
- bin-packing (NP-c)
- scheduling problems (NP-c)
- subset-sum (NP-c)
- traveling salesman (NP-c)
- …

Example NP algorithms

Satisfiability for Propositional Logic

Input: propositional formula φ
  guess assignment $\Pi\colon \mathcal{V}(\varphi) \to \{0,1\}$
  verify that $\Pi \models \varphi$

Clique Search in Graphs

Input: graph $G = (V, E)$, number k
  guess $C \subseteq V$ with $|C| = k$
  verify that $C \times C \subseteq E$

Generic NP problem

Input: instance I
  guess poly-length "witness" for I
  verify that witness is correct in polynomial time

generic witness? path through nondeterministic configurations
theorem NP is class of "efficiently verifiable" problems

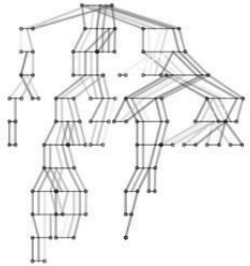
Proof of Rusinowitch-Turuani Theorem

theorem

INSECURE is NP-complete.

two parts

1. **INSECURE** $\in$ NP
2. **INSECURE** is NP-hard



more interesting part: INSECURE $\in$ NP

approach: guess & verify

Input: protocol P, initial knowledge I

guess attack

verify that attack is successful

need to show

1. if P is insecure, there is a “short successful attack”
2. attack can be verified efficiently

Rusinowitch-Turuani Algorithm

NP-algorithm for INSECURE

input: protocol $P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$, with initial knowledge I

1. guess execution order o of P
2. guess short representation of substitution σ for variables in P
3. verify $\sigma(r^{o(k)}_{\#o(k)}) \in \mathsf{DY}\left(I \cup \left\{\sigma(s^{o(\ell)}_{\#o(\ell)}) \mid 0 \leq \ell < k\right\}\right)$ for all k
4. verify $\mathsf{FAIL} \in \mathsf{DY}\left(I \cup \left\{\sigma(s^{o(\ell)}_{\#o(\ell)}) \mid 0 \leq \ell < |o|\right\}\right)$
5. accept if all checks successful

(nondeterministic) polynomial time?

- length of o and representation of σ polynomial in input length?
- verification possible in (deterministic) polynomial time?

Overview

Short Representation of σ

fact

"exponentially long attacker terms" needed for some protocols $\rightsquigarrow$ exercise

needed: short representation of terms

- required terms "long," but structurally simple
- "many copies" of identical subterms

idea: "compress" by avoiding repetitions

definition (DAG representation)

DAG representation of $S \subseteq \mathcal{T}$: edge-labeled graph $G = (V, E)$ with

- $V = \mathsf{Sub}(S)$,
- $E = \left\{ v_s \xrightarrow{\text{left}} v_e \mid \exists b, v_s = [v_e, b] \text{ or } v_s = \mathsf{enc}^{\mathsf{s}}_{v_e}(b) \text{ or } v_s = \mathsf{enc}^{\mathsf{a}}_{v_e}(b) \right\}$
  $\cup \left\{ v_s \xrightarrow{\text{right}} v_e \mid \exists b, v_s = [b, v_e] \text{ or } v_s = \mathsf{enc}^{\mathsf{s}}_{b}(v_e) \text{ or } v_s = \mathsf{enc}^{\mathsf{a}}_{b}(v_e) \right\}$

$|S|_{\mathsf{DAG}}$: number of nodes in DAG representation of S

Term Compression: Example

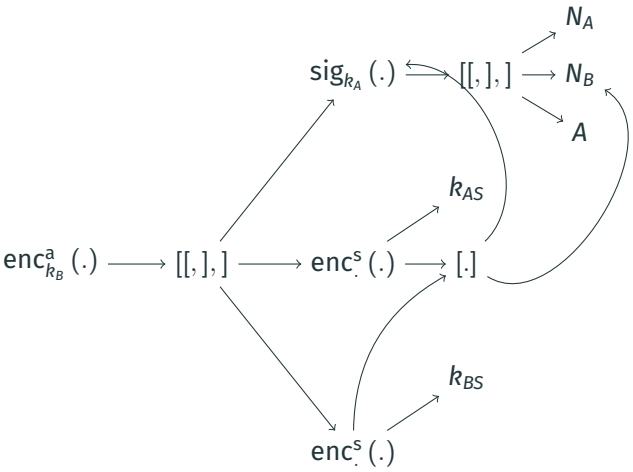
example term

$$\mathrm{enc}^{\mathrm{a}}_{k_B}\left(\mathrm{sig}_{k_A}\left(N_A, N_B, A\right), \mathrm{enc}^{\mathrm{s}}_{k_{AS}}\left(N_B, \mathrm{sig}_{k_A}\left(N_A, N_B, A\right)\right), \mathrm{enc}^{\mathrm{s}}_{k_{BS}}\left(N_B, \mathrm{sig}_{k_A}\left(N_A, N_B, A\right)\right)\right)$$

term representation as graph

- following slide: complete term / graph representation, compression of repeated elements
- (sets of) terms: DAGS with fan-in 1 (i.e., forests)
- note modeling of symmetric / asymmetric encryption, sequences

Term Compression: Trees and DAGs



compression

- tree: 29
- DAG: 14

(3-tuple counts as 2 nodes)

Goal

want to show

if there is a successful attack on P, then there is a successful attack (o, σ) such that

$$|\{\sigma(x) \mid x \text{ variable in } P\}|_{\mathsf{DAG}} \leq p(|P|),$$

for a fixed polynomial p.

crucial role in NP membership proof

shows: if there is an attack, then there is one with a “short” representation

forgot something?

also need: our algorithms work with DAG representation

- (easy to see, standard techniques)

Exercise

Task (exponential attack size)

For $i \in \mathbb{N}$, the protocol P_i is defined as follows:

- There are two instances:
  1. $\mathcal{I}_1$ has a single receive/send action $[x_1, \ldots, x_i] \to \mathsf{enc}^{\mathsf{s}}_k([t_1, t_2])$, with
     $t_1 = [x_1, [x_2, [x_3, [x_4, [\ldots, [x_{i-1}, [x_i, \mathsf{o}]] \ldots]]]]]$
     $t_2 = [[[[[\ldots [[\mathsf{o}, x_i], x_{i-1}], \ldots], x_4], x_3], x_2], x_1]$.
  2. $\mathcal{I}_2$ has a single receive/send action $\mathsf{enc}^{\mathsf{s}}_k(y, y) \to \mathsf{FAIL}$.
- The initial adversary knowledge is the set $\{\mathsf{o}, \mathsf{1}\}$.

Show that each protocol P_i is insecure, but a successful attack requires terms of exponential length. How can you use DAGs to obtain a shorter representation of the involved terms?

“Compression” Properties of DAG representation

notation

- t term, S set of terms, σ substitution, then $S\sigma = \{\sigma(t) \mid t \in S\}$
- t term, x variable, then $[x \leftarrow t]$ substitution σ: $\sigma(x) = t$, $\sigma(y) = y$ for $y \neq x$

lemma

$S \subseteq \mathcal{T}$, x variable, t message, then $|S[x \leftarrow t]|_{\mathsf{DAG}} \leq |S \cup \{t\}|_{\mathsf{DAG}}$.

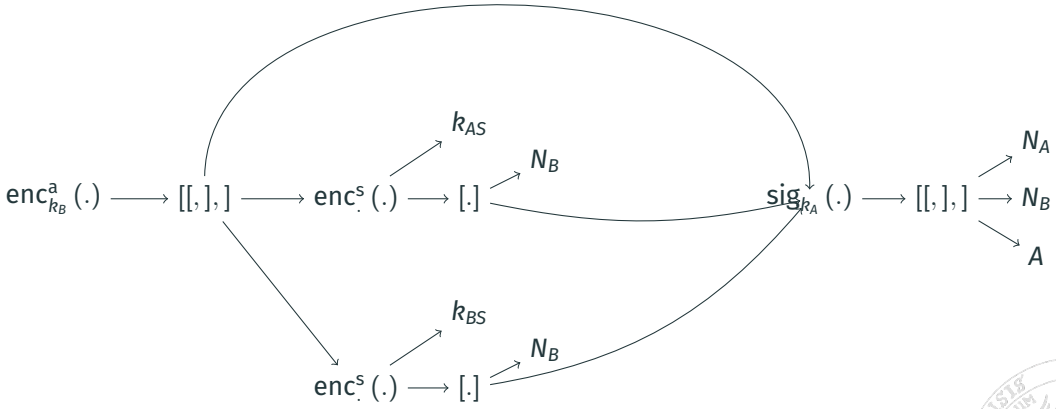
corollary

$S \subseteq \mathcal{T}$, σ ground substitution on variables $x_1, \ldots, x_k$, then $|S\sigma|_{\mathsf{DAG}} \leq |S \cup \{\sigma(x_1), \ldots, \sigma(x_k)\}|_{\mathsf{DAG}}$.

relevance

assigning t to many occurrences is not more expensive than adding t once

Replacing All Variable Occurrences With Same Term



note something odd? equivalent nodes would be identified.

Overview

Size of an Attack

want to show

if P insecure, then there is successful attack with “short” representation (choose the “shortest”)

definition

(σ, o) attack on protocol P. Then the size of σ (denoted with $|\sigma|$) is

$$\sum_{x \text{ variable in } P} |\sigma(x)|_{\mathsf{DAG}} .$$

definition

a successful attack (σ, o) on P is minimal, if $|\sigma| \leq |\sigma'|$ for every successful attack (σ', o') on P.

remark

- every insecure protocol has a minimal successful attack
- protocols can have several minimal successful attacks (see exercise)

Exercise

Task (no unique successful minimal attack)

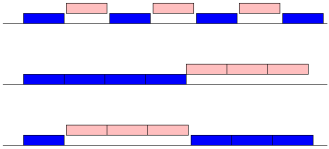
Show that in general, there is no unique minimal successful attack on a protocol. That is, construct a protocol and two different successful attacks on it that both have minimal size.

Notation for Proof

simplification

attack consists of

- substitution σ
- execution order o



rule application

- $r^{o(0)}_{\#o(0)} \to s^{o(0)}_{\#o(0)}$
- $r^{o(1)}_{\#o(1)} \to s^{o(1)}_{\#o(1)}$

$\vdots$

- $r^{o(n)}_{\#o(n)} \to s^{o(n)}_{\#o(n)}$

simplification

- write $r_i \to s_i$ instead of $r^{o(i)}_{\#o(i)} \to s^{o(i)}_{\#o(i)}$.
- note: this fixes execution order from attack
- used in proof of parsing Lemma, Rusinowitch-Turuani Theorem

Attacks Against Protocols: Simple Terms in Substitutions

seen up to now

- substitutions: reference nonces, identities, keys, …from protocols
- in particular: $\sigma(x)$ **atomic** in most examples
- atomic case: short attacks given ($|\sigma(x)|_{\mathsf{DAG}} = 1$)

question

when do more complex terms show up for $\sigma(x)$? how complex does it get?

Example

protocol fragment (cp. Woo Lam Protocol)

$$
\begin{array}{lcll}
A & \to & B & \mathsf{enc}^{\mathsf{a}}_{k_B}\left(A, \mathsf{enc}^{\mathsf{a}}_{k_{AS}}\left(A, \mathsf{MAC}_{\mathsf{k}_{\mathsf{AS}}}\left(N_A, A, S\right)\right)\right) \\
B & \to & S & \mathsf{enc}^{\mathsf{a}}_{k_S}\left(B, \mathsf{verify}, \mathsf{enc}^{\mathsf{a}}_{k_{AS}}\left(A, \mathsf{MAC}_{\mathsf{k}_{\mathsf{AS}}}\left(N_A, A, S\right)\right)\right) \\
S & \to & A & \mathsf{enc}^{\mathsf{a}}_{k_B}\left(\mathsf{MAC}_{\mathsf{k}_{\mathsf{BS}}}\left(N_A, A\right)\right)
\end{array}
$$

attack

$\sigma(x) = \mathsf{enc}^{\mathsf{a}}_{k_{AS}}\left(A, \mathsf{MAC}_{\mathsf{k}_{\mathsf{AS}}}\left(N_A, A, S\right)\right)$

$\ldots$

represent as receive/send actions

$$
\begin{array}{lll}
A & \epsilon & \to \quad \mathsf{enc}^{\mathsf{a}}_{k_B}\left(A, \mathsf{enc}^{\mathsf{a}}_{k_{AS}}\left(A, \mathsf{MAC}_{\mathsf{k}_{\mathsf{AS}}}\left(N_A, A, S\right)\right)\right) \\
B & \mathsf{enc}^{\mathsf{a}}_{k_B}\left(A, x\right) & \to \quad \mathsf{enc}^{\mathsf{a}}_{k_S}\left(B, \mathsf{verify}, x\right) \\
S & \mathsf{enc}^{\mathsf{a}}_{k_S}\left(B, \mathsf{verify}, \mathsf{enc}^{\mathsf{a}}_{k_{AS}}\left(A, \mathsf{MAC}_{\mathsf{k}_{\mathsf{AS}}}\left(y, A, S\right)\right)\right) & \to \quad \mathsf{enc}^{\mathsf{a}}_{k_B}\left(\mathsf{MAC}_{\mathsf{k}_{\mathsf{BS}}}\left(y, a\right)\right)
\end{array}
$$

term complexity

- Bob cannot build $\mathsf{enc}^{\mathsf{s}}_{k_{AS}}(\ldots)$, needs variable
- $\sigma(x)$ is a "complex" term
- can be made arbitrarily complex $\to$ cannot prove "$|\sigma(x)| < c$" for any constant c"
- **reason** why term must be "complex?" structure of $\sigma(x)$ appears in r/s rules
- $\sigma(x)$ will be **parsed** by matching with a protocol rule

question

other reasons why $\sigma(y)$ must be "complex?"

Parsing Lemma: Terms Always Match with Protocol

parsing lemma

P protocol, (o, σ) minimal successful attack on P, x variable in P with $|\sigma(x)| > 1$. Then there is a term t such that

- $t \in \mathsf{Sub}(r_0, \ldots, r_n, s_0, \ldots, s_n)$,
- t is not a variable,
- $\sigma(t) = \sigma(x)$.

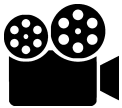
statement

terms in variables represent steps in the protocol

informal justification

assume x with $|\sigma(x))| > 1$ counter-example

- for all $t \in \mathsf{Sub}(P)$ with $\sigma(t) = \sigma(x)$: t is variable.
- outmost operator in $\sigma(x)$ does not match with protocol.
  - then $\sigma(x)$ computed by $\mathcal{A}$
  - then $\sigma(x)$ does not get "parsed"
  - $\sigma(x)$ more complex than needed
  - contradiction, since (σ, o) minimal attack



Parsing Lemma Proof

https://cloud.rz.uni-kiel.de/index.php/s/TW4sLpCNbSwe7QB

video content

- Proof of Parsing Lemma by explicit construction of a “smaller” attack
- note errata slide!

study

- watch video—feedback welcome!
- video slides contained in slide set (gray background), additional material in lecture notes
- next week: discussion of content (in small groups), bring questions!

Parsing Lemma Proof Overview

parsing lemma statement

P protocol, (o, σ) minimal successful attack on P, x variable in P with $|\sigma(x)| > 1$. Then there is a term t such that

- $t \in \mathsf{Sub}(r_0, \ldots, r_n, s_0, \ldots, s_n)$,
- t is not a variable,
- $\sigma(t) = \sigma(x)$.

proof structure

1. assume (o, σ) counter-example: minimal attack, $\sigma(x)$ matches no term in protocol, $|\sigma(x)| > 1$
2. collect facts about appearence of $\sigma(x)$ in protocol run
3. show that $\sigma(x)$ can be derived by adversary
4. replace $\sigma(x)$ with ϵ in attack, this is a smaller successful attack on P

reference

Michaël Rusinowitch and Mathieu Turuani. “Protocol insecurity with a finite number of sessions, composed keys is NP-complete”. In: Theoretical Computer Science 1-3.299 (2003), pp. 451–475

Exercise

Task (parsing lemma proof)

In the proof of the Parsing Lemma, we showed that in that particular setting, the term $\sigma(x)$ is constructed by the adversary. Is this generally true? More precisely: Is there a protocol P with initial knowledge I and a successful minimal attack (o, σ) such that there is a variable x with $\sigma(x) \neq x$ and $\sigma(x) \notin \mathsf{DY}(S)$, where S is the set of terms available to the adversary at the step where the first term containing $\sigma(x)$ is sent?

Errata for Video “Parsing Lemma Proof”

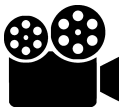
corrections

- second handwritten page:
  - induction step $i \rightsquigarrow i - 1$: should be $\sigma(\mathbf{x}) \in \mathsf{Sub}(\mathsf{S}_i)$
- third handwritten page:
  - last line should be “So, (o, σ') is successful attack”

additions

- third handwritten page:
  - $T_0 = \{\sigma(s_0), \ldots, \sigma(s_{j-1})\}$, $T'_0 = \{\sigma'(s_0), \ldots, \sigma'(s_{j-1})\}$
  - $\sigma(r_j) \in T_n$, $\sigma'(r_j) \in T'_n$

Video Lecture: Feedback wanted



questions

- audio/video quality?
- proof presentation as screenshots, or “live writing?”
- better as video or “live Zoom session?”
- any suggestions?

feedback crucial

- your perspective very different from mine!
- constructive criticism always welcome
- review after week 6!

remember

- we’re all still learning this
- new tools, concepts
- big playground :-)