

Engineering Secure Software Systems

Winter 2020, Weeks \approx 5 – 7: Automatic Analysis: Theory

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Part I: Crypto Protocols

Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms



Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms



Goal: Automatic Analysis

protocols formalizes

- protocol
- messages
- protocol execution
- successful attack

formal problem

Problem: **INSECURE**

Input: protocol P , initial adversary knowledge I

Question: is there a successful attack on P ?

algorithm for INSECURE?

is the problem decidable?



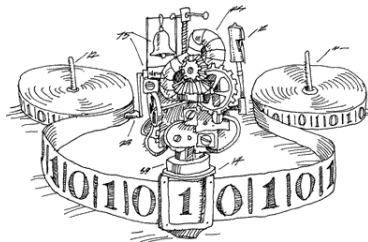
Obstacles to Decidability

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 “language”
- 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.

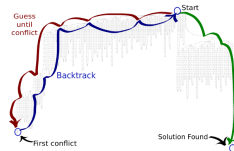
Michaël Rusinowitch and Mathieu Turuani. “Protocol insecurity with a finite number of sessions, composed keys is NP-complete”. In: *Theoretical Computer Science* 1-3.299 (2003), pp. 451–475

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 $\Pi: \mathcal{V}(\varphi) \rightarrow \{0, 1\}$

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



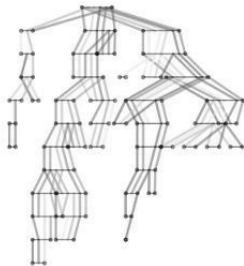
Proof of Rusinowitch-Turuani Theorem

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” attack
2. attack can be verified efficiently



Rusinowitch-Turuani Algorithm

NP-algorithm for INSECURE

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

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

(nondeterministic) polynomial time?

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



Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms



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 $[x_1, \dots, x_i] \rightarrow \text{enc}_k^s([t_1, t_2])$, with
$$t_1 = [x_1, [x_2, [x_3, [x_4, [\dots, [x_{i-1}, [x_i, \mathbf{0}]] \dots]]]]$$
$$t_2 = [[[[[\dots [[\mathbf{0}, x_i], x_{i-1}], \dots], x_4], x_3], x_2], x_1].$$
 2. \mathcal{I}_2 has a single receive/send action $\text{enc}_k^s(y, y) \rightarrow \text{FAIL}$.
- The initial adversary knowledge is the set $\{\mathbf{0}, \mathbf{1}\}$.

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?



Short Representation of σ

fact

“exponentially long attacker terms” needed for some protocols

↪ 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

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

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



Term Compression: Example

example term

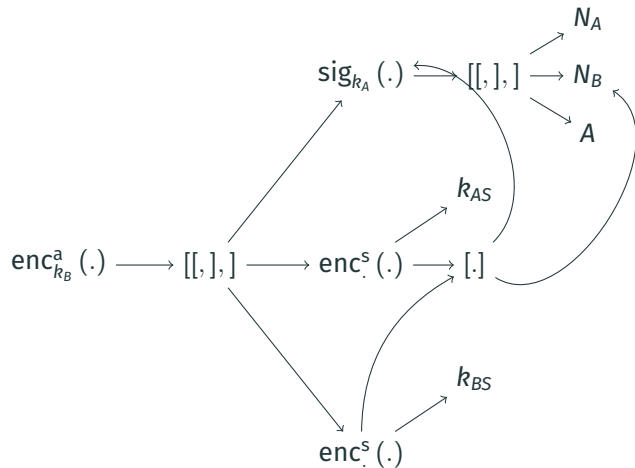
$$\text{enc}_{k_B}^a \left(\text{sig}_{k_A} (N_A, N_B, A), \text{enc}_{k_{AS}}^s (N_B, \text{sig}_{k_A} (N_A, N_B, A)), \text{enc}_{k_{BS}}^s (N_B, \text{sig}_{k_A} (N_A, N_B, A)) \right)$$

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 an attack on $P = \{\mathcal{I}_0, \dots, \mathcal{I}_{n-1}\}$, then there is an attack (\mathbf{o}, σ) such that

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

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)





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 $\sigma: \sigma(x) = t, \sigma(y) = y$ for $y \neq x$

lemma

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

corollary

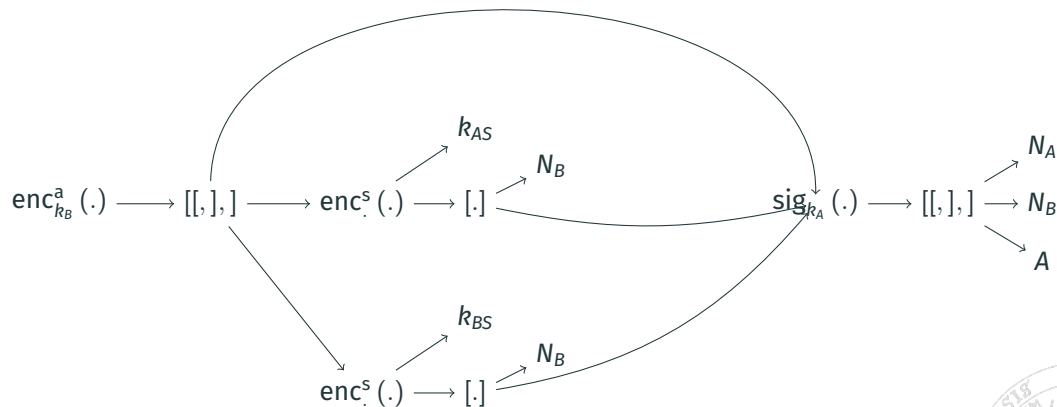
$S \subseteq \mathcal{T}$, σ ground substitution on variables x_1, \dots, x_k , then $|S\sigma|_{\text{DAG}} \leq |S \cup \{\sigma(x_1), \dots, \sigma(x_k)\}|_{\text{DAG}}$.

relevance

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



Replacing All Variable Occurrences With Same Term



note something odd? identical nodes would be compressed.



Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms



Size of an Attack

want to show

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

definition

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

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

definition

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

remark

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

Exercise

Task (no unique minimal attack)

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

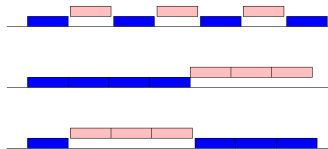




simplification

attack consists of

- substitution σ
- execution order \mathbf{o}



rule application

- $r_{\#o(o)}^{o(o)} \rightarrow s_{\#o(o)}^{o(o)}$
- $r_{\#o(1)}^{o(1)} \rightarrow s_{\#o(1)}^{o(1)}$
- \vdots
- $r_{\#o(n)}^{o(n)} \rightarrow s_{\#o(n)}^{o(n)}$

simplification

write $r_i \rightarrow s_i$ instead of $r_{\#o(i)}^{o(i)} \rightarrow s_{\#o(i)}^{o(i)}$.



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

question

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



Example

protocol fragment (cp. Woo Lam Protocol)

$$\begin{aligned} A &\rightarrow B && \text{enc}_{k_B}^a \left(A, \text{enc}_{k_{AS}}^a (A, \text{MAC}_{k_{AS}} (N_A, A, S)) \right) \\ B &\rightarrow S && \text{enc}_{k_S}^a \left(B, \text{verify}, \text{enc}_{k_{AS}}^a (A, \text{MAC}_{k_{AS}} (N_A, A, S)) \right) \\ S &\rightarrow A && \text{enc}_{k_B}^a (\text{MAC}_{k_{BS}} (N_A, A)) \end{aligned}$$

attack

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

...

represent as receive/send actions

$$\begin{aligned} A &\epsilon && \rightarrow \text{enc}_{k_B}^a \left(A, \text{enc}_{k_{AS}}^a (A, \text{MAC}_{k_{AS}} (N_A, A, S)) \right) \\ B &\text{enc}_{k_B}^a (A, x) && \rightarrow \text{enc}_{k_S}^a (B, \text{verify}, x) \\ S &\text{enc}_{k_S}^a (B, \text{verify}, \text{enc}_{k_{AS}}^a (A, \text{MAC}_{k_{AS}} (y, A, S))) && \rightarrow \text{enc}_{k_B}^a (\text{MAC}_{k_{BS}} (y, a)) \end{aligned}$$

term complexity

- Bob cannot build $\text{enc}_{k_{AS}}^s (\dots)$, needs variable
- $\sigma(x)$ is a “complex” term
- can be made arbitrarily complex \rightarrow 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

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

statement

terms in variables represent steps in the protocol

informal justification

assume this is not true for x with $|\sigma(x)| > 1$.

- for all $t \in \text{Sub}(P)$ 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 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

- $t \in \text{Sub}(r_0, \dots, r_n, s_0, \dots, s_n)$,
- 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 appearance 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

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

Errata for Video “Parsing Lemma Proof”

corrections

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

additions

- third handwritten page:
 - $T_o = \{\sigma s_o, \dots, \sigma s_{j-1}\}, T'_o = \{\sigma s'_o, \dots, \sigma s'_{j-1}\}$
 - $\sigma(r_j) \in T_n, \sigma(r'_j) \in T'_n$

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 :-)



theorem

(σ, \mathbf{o}) minimal attack on P , then $|\sigma(\mathbf{x})|_{\text{DAG}} \leq |\{r_0, \dots, r_n, s_0, \dots, s_n\}|_{\text{DAG}}$.

completes NP membership proof

- each \mathbf{x} : representation of $\sigma(\mathbf{x})$ bound by protocol length
- number of variables is polynomial

therefore: polynomial
representation of attack

proof (sketch)

- for every \mathbf{x} with $\sigma(\mathbf{x}) > 1$ ex. subterm \mathbf{t} of P with $\sigma(\mathbf{t}) = \sigma(\mathbf{x})$, \mathbf{t} no variable
- every “long” $\sigma(\mathbf{x})$ is obtained by applying terms from the protocol
- replace long terms with references to protocol
 \rightsquigarrow small DAG size

Proof of Rusinowitch Turuani Theorem

theorem

(σ, \mathbf{o}) minimal attack on P , then $|\sigma(\mathbf{x})|_{\text{DAG}} \leq |\{r_0, \dots, r_n, s_0, \dots, s_n\}|_{\text{DAG}}$.

proof

V set of variables, then $\bar{V} = \{\sigma(\mathbf{x}) \mid \mathbf{x} \in V\}$.

- construct $S_p \subseteq \text{Sub}(\{r_0, \dots, s_n\})$, $V_p \subseteq \mathcal{V}$:

$$|\sigma(\mathbf{x})|_{\text{DAG}} \leq |S_p \cup \bar{V}_p|_{\text{DAG}}$$

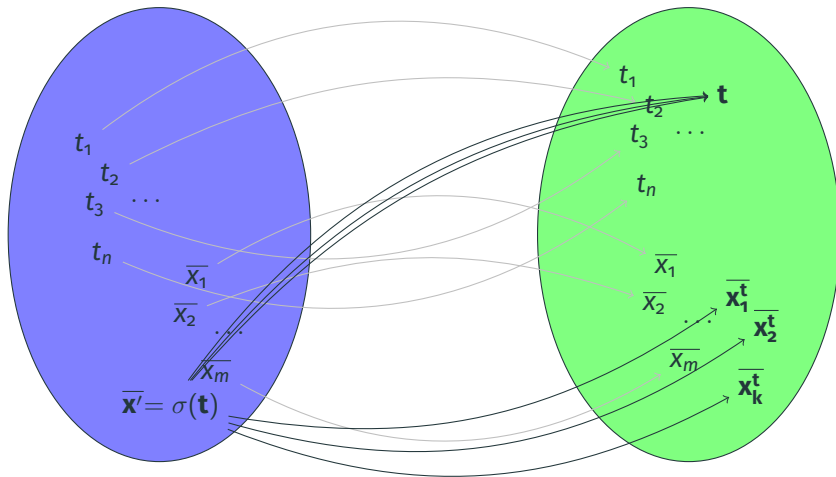
- 1. $p = 0$: choose $S_0 = \emptyset$, $V_0 = \{\mathbf{x}\}$.
- 2. $p \rightarrow p + 1$
 - choose $\mathbf{x}' \in V_p$, \mathbf{t} from protocol with $\sigma(\mathbf{x}') = \sigma(\mathbf{t})$
 - $S_{p+1} = S_p \cup \{\mathbf{t}\}$
 - $V_{p+1} = V_p \setminus \{\mathbf{x}'\} \cup \mathcal{V}(\mathbf{t})$
 - ind: $|\sigma(\mathbf{x})|_{\text{DAG}} \leq |S_p \cup \bar{V}_p|_{\text{DAG}}$
 - now: $|\sigma(\mathbf{x})|_{\text{DAG}} \leq |S_p \cup \bar{V}_p|_{\text{DAG}} \leq |S_{p+1} \cup \bar{V}_{p+1}|_{\text{DAG}}$

start/end, termination

- $p = 0$: $|\sigma(\mathbf{x})|_{\text{DAG}} \leq |\overline{\{\mathbf{x}\}}|_{\text{DAG}}$
- $V_p = \emptyset$: $|\mathbf{x}|_{\text{DAG}} \leq |S_p|_{\text{DAG}} \leq |r_0, \dots, s_n|_{\text{DAG}}$
- why do we reach $V_p = \emptyset$?
- $\sum_{z \in V_p} |\sigma(\mathbf{z})|$ decreases

Proof of Rusinowitch-Turuani Theorem

$|S_p \cup \overline{V_p}|_{\text{DAG}} \leq |S_{p+1} \cup \overline{V_{p+1}}|_{\text{DAG}}$ via injection $f: \text{Sub}(S_p \cup \overline{V_p}) \rightarrow \text{Sub}(S_{p+1} \cup \overline{V_{p+1}})$



Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms

Proof of Rusinowitch Turuani Theorem

theorem

INSECURE is NP-complete.

two parts

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

second part: INSECURE is NP-hard

recall TGI: reduce from NP-complete problem

NP hardness proof

TGI refresher

L_1, L_2 languages, L_1 NP-hard and $L_1 \leq_m^P L_2$, then L_2 NP-hard.

reduction

$L_1 \leq_m^P L_2$ means:

L_1 -question can be “efficiently translated” into L_2 -questions

formally

there is a total, P-computable function $f: \Sigma^ \rightarrow \Sigma^*$ with*

$$x \in L_1 \text{ iff } f(x) \in L_2 \text{ for all } x.$$

NP-complete problem: 3SAT

SAT for formulas $\varphi = \bigwedge_{i=1}^n (l_1^i \vee l_2^i \vee l_3^i)$, where l_j^i literals over $\{x_1, \dots, x_m\}$

NP hardness reduction

3SAT

- **nondeterministic step:** guess assignment I for variables of formula φ
- **deterministic step:** check that assignment satisfies φ

INSECURE

- **nondeterministic step:** guess *one* adversary message (encoding I)
- **deterministic step:** let honest participants check that assignment satisfies φ

note

φ can be hard-coded into **INSECURE** instance

issues

- adversary can interfere with communication between honest principals
- use cryptography to ensure secure communication

NP hardness reduction

input formula

$$\varphi = \bigwedge_{i=1}^n (l_1^i \vee l_2^i \vee l_3^i)$$

honest principals

A expects assignment from adversary and distributes it

B_j^i for each clause in formula, participants verifying that clause is satisfied, giving “certificate”

F collects certificates and releases **FAIL**-constant if successful

intended protocol run

- adversary: send *single* propositional assignment I to participant A
- A / B_j^i / F : coordinate response to adversary



input formula

- $\varphi = \bigwedge_{i=1}^n (l_1^i \vee l_2^i \vee l_3^i)$
- $r_{i,j}$: index of x that l_j^i is based on, i.e., $l_j^i \in \{x_{r_{i,j}}, \overline{x_{r_{i,j}}}\}$

honest participants

A job: receive assignment I from adversary, distribute among B_j^i

B_j^i job: checks j -th satisfying assignment for clause $l_1^i \vee l_2^i \vee l_3^i$ (7 total), awards k_i if successful

F collects k_1, \dots, k_n from adversary, awards **FAIL** if successful

Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms

Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms

Rusinowitch-Turuani Theorem [RTo3]

INSECURE is NP-complete

model

INSECURE: instances given

- theorem only covers fixed number of instances
- instance $\hat{=}$ protocol session

reality

unbounded number of sessions

- many users for single server
- different (or same) users at different servers

number of concurrent TLS sessions?

Exercise

Task (applying the Rusinowitch Turuani Theorem)

In the lecture, we discussed how to model the Needham-Schroeder protocol formally as an input to **INSECURE** such that the attack can be detected. Can you come up with a general mechanism translating a natural representation of a protocol (for example, as the list of “intended instances” for a single session) into an instance that can be used as input for **INSECURE**? If not, why not?

Towards Automatic Analysis

seen in lecture

- formalization of NS protocol must contain sessions to find attack
 - sender instance of $A \rightarrow C$
 - receiver instance of $A \rightarrow B$
- unsatisfying: this “tells the algorithm where to look”

possible way out: over-approximate

- observation: more instances only make the situation worse (more insecure)
- therefore: let algorithm analyze the following:
 - sender instance of $A \rightarrow B, A \rightarrow C, B \rightarrow C$
 - receiver instance of $A \rightarrow B, A \rightarrow C, B \rightarrow C$
- issues?

“parallel” attacks

Rusinowitch-Turuani analysis

- instances fixed
- hence, protocol sessions fixed

problem

there are issues in protocols that need an “arbitrary” number of sessions

reference

Jonathan K. Millen. “A Necessarily Parallel Attack”. In: *In Workshop on Formal Methods and Security Protocols*. 1999



protocol

1 $A \rightarrow B$ A
2 $B \rightarrow A$ $[N_1, N_2]$
3 $A \rightarrow B$ $\text{enc}_{k_B}^a([N_1, \underbrace{N_2}_{=:x}, \underbrace{FAIL}_{=:y}])$
4 $B \rightarrow A$ $[N_1, x, \text{enc}_{k_B}^a([x, y, N_1])]$

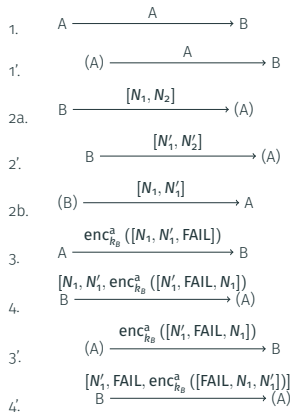
more precisely

step 3:

- B verifies N_1
- B does **not** verify correctness of N_2
- matches N_2 with variable x , **FAIL** with variable y



attack



security

- there is an attack
- attack requires 2 responder instances
- fact: protocol is secure if there is only one instance

consequence

- analysing a single instance is not enough
- generalization: arbitrarily many instances
- analysis of unbounded number of instances required
- not covered by Rusinowitch Turuani

Exercise

Task (The FFGG protocol: too complicated?)

Can you come up with a simpler protocol that is secure when only one session is running, but becomes insecure if the adversary can start as many instances as she wishes? Is there an “advantage” of the ffgg protocol (as an example illustrating the need for the analysis of parallel sessions) over your example?

Unbounded Version of INSECURE

required

analysis of extension of **INSECURE** to (arbitrarily many) parallel sessions

formalization

- input to algorithm may not contain explicit sessions anymore
- alternative: “template” for instances
 - instance $\mathcal{I}_{A \rightarrow B}$ may be started arbitrarily often
 - “between **A** and **B**”
 - “between **A** and **C**”
 - ...

issue

FAIL-rule may only be contained in “relevant” instance

Unbounded Version of INSECURE

approach

- specify instances, initial attacker knowledge as usual
- mark one instance as **goal** (usually contains FAIL constant)

definition

protocol P_{unb} **based** on P , if P_{unb} obtained from P by

- replicating instances (with fresh variables)
- changing identities in non-goal instances

issues

- changing identities must “respect” knowledge of keys
- straight-forward for asymmetric keys, more technical for symmetric keys
- see discussion in exercise class

this lecture

- no formal definition
- follow these ideas in practical security specifications
- case-study later: modeling of Needham-Schroeder in ProVerif

Exercise

Task (unbounded instances formalization)

Specify the Needham-Schroeder protocol as an instance of the decision problem

UNBOUNDED-INSECURE, and show that it is insecure in this formalization. Discuss the differences between expressing the protocol using this formalism compared to the earlier formalization using the decision problem **INSECURE**.

Undecidability

Theorem

the following problem is undecidable:

Problem: **UNBOUNDED-INSECURE**

Input: protocol $P = (\{\mathcal{I}_0, \dots, \mathcal{I}_{n-1}\}, I)$

Question: is there an insecure protocol P_{unb} based on P ?

?

missing something? we didn't even define **UNBOUNDED-INSECURE**!

note

result true for very simple modeling of **UNBOUNDED-INSECURE**

Undecidability Result

formalization for undecidability

“simplest” formalization of unbounded sessions: result covers more expressive models as well

“minimal requirements”

- protocol consists of instances $\{\mathcal{I}_0, \dots, \mathcal{I}_{n-1}\}$, each instance has a single receive/send rule
- adversary may activate each instance as often as she wishes
- there is only a single symmetric key K (no identities)

undecidability proof

works for this model

Undecidability Proof

TGI refresher

L_1, L_2 languages, L_1 undecidable and $L_1 \leq L_2$, then L_2 undecidable.

reduction

$L_1 \leq L_2$ means:

L_1 -questions can be translated into L_2 -questions.

formally:

there is a total, computable function $f: \Sigma^ \rightarrow \Sigma^*$ such that for all x :*

$$x \in L_1 \text{ iff } f(x) \in L_2.$$

Post's Correspondence Problem

seen in TGI

halting problem, Rice's theorem

drawback

talk about encodings of Turing machines

classical problem

Problem: PCP (Post's correspondence problem)

Input: $(x_1, y_1), \dots, (x_n, y_n)$ with $x_i, y_i \in \{0, 1\}^*$

Question: Is there a sequence i_1, \dots, i_ℓ with $x_{i_1}x_{i_2} \dots x_{i_\ell} = y_{i_1}y_{i_2} \dots y_{i_\ell}$?

theorem

PCP is undecidable.

Emil L. Post. "A variant of a recursively unsolvable problem". In: *Bull. Amer. Math. Soc.* 52.4 (Apr. 1946), pp. 264–268. URL: <https://projecteuclid.org:443/euclid.bams/1183507843>

Undecidability Proof

want to show

UNBOUNDED-INSECURE is undecidable

proof (sketch)

- show $\text{PCP} \leq \text{UNBOUNDED-INSECURE}$
- describe **computable** function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ such that
 $x \in \text{PCP} \text{ iff } f(x) \in \text{UNBOUNDED-INSECURE}$



PCP

infinite search space: find $i_1 \dots i_\ell$

with $x_{i_1} \dots x_{i_\ell} = y_{i_1} \dots y_{i_\ell}$

UNBOUNDED-INSECURE

infinite search space: choice of instances in

- protocol P_{unb} based on P
- execution order of attack

let instances perform “concatenation” of PCP strings

note

$x_1, \dots, x_n, y_1, \dots, y_n$ can be hard-coded into UNBOUNDED-INSECURE instance

issues

- adversary can use “fake PCP substrings”
- use cryptography to authenticate substrings and concatenation from PCP instance

The Edge of Decidability

lecture results

- Rusinowitch Turuani: **bounded** sessions
decidable
- PCP reduction: **unbounded** sessions
undecidable

middle ground?

- “restricted” unbounded sessions?
- simple loops in protocol?
- data structure processing?
- more complex protocol goals?

results

there is a lot!

The Edge of Decidability: References I

- Ralf Küsters and Tomasz Truderung. “On the Automatic Analysis of Recursive Security Protocols with XOR”. In: [STACS](#). Ed. by Wolfgang Thomas and Pascal Weil. Vol. 4393. Lecture Notes in Computer Science. Springer, 2007, pp. 646–657. ISBN: 978-3-540-70917-6
- Detlef Kähler, Ralf Küsters, and Tomasz Truderung. “Infinite State AMC-Model Checking for Cryptographic Protocols”. In: [LICS](#). IEEE Computer Society, 2007, pp. 181–192
- Henning Schnoor. “Deciding Epistemic and Strategic Properties of Cryptographic Protocols”. In: [ESORICS](#). Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Vol. 7459. Lecture Notes in Computer Science. Springer, 2012, pp. 91–108. ISBN: 978-3-642-33166-4

- Steve Kremer and Robert Künnemann. “Automated analysis of security protocols with global state”. In: *Journal of Computer Security* 24.5 (2016), pp. 583–616. DOI: 10.3233/JCS-160556. URL: <https://doi.org/10.3233/JCS-160556>
- Jannik Dreier, Charles Duménil, Steve Kremer, and Ralf Sasse. “Beyond Subterm-Convergent Equational Theories in Automated Verification of Stateful Protocols”. In: *Principles of Security and Trust - 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*. Ed. by Matteo Maffei and Mark Ryan. Vol. 10204. Lecture Notes in Computer Science. Springer, 2017, pp. 117–140. ISBN: 978-3-662-54454-9. DOI: 10.1007/978-3-662-54455-6

The Edge of Decidability: References III

- Jannik Dreier, Lucca Hirschi, Sasa Radomirovic, and Ralf Sasse. “Automated Unbounded Verification of Stateful Cryptographic Protocols with Exclusive OR”. In: *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 359–373. ISBN: 978-1-5386-6680-7. URL: <https://ieeexplore.ieee.org/xpl/conhome/8428826/proceeding>
- Robert Künnemann, Ilkan Esiyok, and Michael Backes. “Automated Verification of Accountability in Security Protocols”. In: *CoRR* abs/1805.10891 (2018). arXiv: 1805.10891. URL: <http://arxiv.org/abs/1805.10891>
- ...

Undecidability: Consequences

result

(in)security with arbitrary many sessions is undecidable

consequences

- no complete “push-button” analysis of security
 - hardly unexpected
- justification for “user-unfriendly” input for Rusinowitch Turuani algorithm
 - some automatic “preprocessing” possible, but does not solve problem

analysis still required

what are options for practice?

Rusinowitch Turuani Analysis

approach

- fixed choice of instances
 - fixes identities, roles (e.g., “Alice as initiator in session with Bob”)
 - fixes number of sessions
 - fixes max. number of messages
- attack found: protocol insecure
- no attack found: secure **in this scenario**

usual security approach

- worst-case assumptions
- “unusual” attacks are exactly what we do automatic analysis for
- situation not satisfying

justification

- most attacks found by checking small systems
- unusual for an attack to require “many” sessions

manual approach

- proof using protocol structure
- for every message: *if* accepted, *then* earlier ...
- then “protocol run as intended”

expensive and error-prone

automatic analysis

- problem is undecidable
- cannot have both
 - **soundness** result “protocol secure” is correct
 - **completeness** if protocol secure, this is recognized
- need to look at “incomplete” algorithms

Part I: Crypto Protocols

Foundations

Cryptography

An Example and an Attack

More Examples

Formal Protocol Model

Automatic Analysis: Theoretical Foundations

Decidability: The Rusinowitch-Turuani Theorem

DAGs

Short Attacks

NP hardness

Automatic Analysis: Undecidability

Arbitrarily Many Sessions

Incomplete Algorithms



construct security proof

- algorithm searches for security proof
- on failure: abort or endless loop
- algorithm is correct (sound)

consequence

secure protocols are recursively enumerable
(semi-decidable)

construct attack

- algorithm searches for attack
- on failure: abort or endless loop
- algorithm is correct (sound)

consequence

insecure protocols are recursively enumerable
(semi-decidable)

what's wrong?

something does not add up! (*aka don't cite this slide!*)

Incomplete Algorithms

seen

- searching for security proofs and attacks can never cover everything
- way out: **heuristics** (cp. NP-complete problems)

heuristics

- there is always a price!
- what do we give up?

over-approximate attacker

- simplified attacker model
- gives “too much power” to attacker
- constructs “over-approximated” attack
- user must check attack
- algorithm sound, not complete (for security)

abstractions

- over-approximation of attacker
- leads to finite model
- apply model checking

lecture: skipped due to time constraints

logic-based modeling

- models protocol properties in (FO Horn) logic
- leads to Horn theory
- apply satisfiability testing

lecture: cover this in practice (ProVerif), brief look at theory

Unbounded Analysis

approaches

- **model checking** “abstraction approach” (Lowe, [Low99])
- **logic modeling** “Horn approach” (Blanchet, [Bla11])

model checking

- state-based system representation
- for infinite systems
 - infinite-state model checking
 - bisimulation arguments
- analysis by state space exploration
 - needs “shortcut arguments”

logic modeling

- logic-based representation
- facts and clauses
- model protocol steps as implications
- analysis by logical deduction rules
 - needs “computationally nice” logic

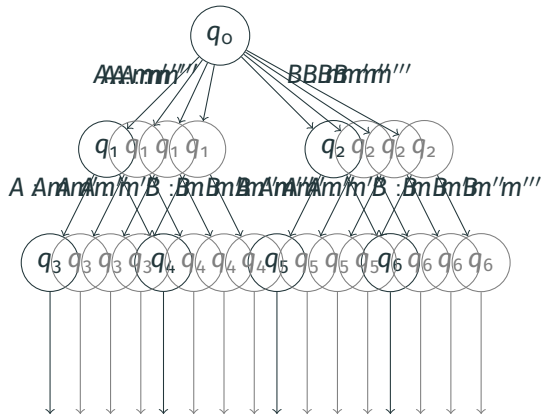
Model Checking Outline

state space

- steps performed by protocol
- messages sent so far $\hat{=}$ substitution
- transitions: message delivery

properties

- trace properties
 - $G \neg \text{FAIL}$
 - $G(\text{acc} \rightarrow \text{Psend})$
- epistemic properties
 - $\mathcal{K}_{A,B}(v_{B,1}) \wedge \neg \mathcal{K}_C(v_{B,1})$
- ...



issue

- state space explosion
- reduction technique needed (e.g., RT)

Computationally Nice Logics

propositional logic

- $\varphi = \exists x_1 \forall y_1 \exists x_2 \dots \forall y_n$
 $(x_1 \vee \overline{x_9} \vee y_4) \wedge \dots \wedge (y_6 \vee \overline{x_3} \vee \overline{y_{44}})$
- relevant algorithmic problems:
decidable, NP-complete

first-order logic

- $\varphi = \exists x_1 \forall y_1 \exists x_2 \dots \forall y_n$
 $R_1(x_1, x_9, y_4) \vee (R_2(x_3, y_1, x_{13}) \wedge \dots)$
- relevant algorithmic problems: undecidable

complexity reduction

syntactically defined sub-logic with “nicer” complexity? Horn clauses

- allows unit resolution
- “largest” sublogic for which propositional satisfiability is PTIME-solvable [Sch78]
- still undecidable first-order theory, but “better behaved”

Logic Modeling Outline

facts

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

DY deductions

- $d(\text{enc}_{k_C}^a(x)) \wedge d(\hat{k}_C) \rightarrow d(x)$
- $d(x) \wedge d(y) \rightarrow d([x, y])$
- $d(x) \rightarrow d(\text{hash}(x))$
- ...

protocol deductions

- $d(\text{enc}_{k_B}^a([A, x])) \rightarrow d(\text{enc}_{k_A}^a([B, x]))$
- $d(\text{enc}_{k_A}^a([B, x, y])) \rightarrow (\text{enc}_{k_B}^a(y))$
- ...

Horn clauses

$$(x_1 \wedge x_2 \wedge \cdots \wedge x_n \rightarrow y) \leftrightarrow (\overline{x_1} \vee \overline{x_2} \vee \cdots \vee \overline{x_n} \vee y)$$

target clause

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

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

 Jannik Dreier, Charles Duménil, Steve Kremer, and Ralf Sasse. “Beyond Subterm-Convergent Equational Theories in Automated Verification of Stateful Protocols”. In: *Principles of Security and Trust - 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*. Ed. by Matteo Maffei and Mark Ryan. Vol. 10204. Lecture Notes in Computer Science. Springer, 2017, pp. 117–140. ISBN: 978-3-662-54454-9. DOI: 10.1007/978-3-662-54455-6.



Jannik Dreier, Lucca Hirschi, Sasa Radomirovic, and Ralf Sasse. “Automated Unbounded Verification of Stateful Cryptographic Protocols with Exclusive OR”. In: **31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018**. IEEE Computer Society, 2018, pp. 359–373. ISBN: 978-1-5386-6680-7. URL:





<https://ieeexplore.ieee.org/xpl/conhome/8428826/proceeding>.



Robert Künnemann, Ilkan Esiyok, and Michael Backes. “Automated Verification of Accountability in Security Protocols”. In: **CoRR** abs/1805.10891 (2018). arXiv: 1805.10891. URL: <http://arxiv.org/abs/1805.10891>.



Steve Kremer and Robert Künnemann. “Automated analysis of security protocols with global state”. In: **Journal of Computer Security** 24.5 (2016), pp. 583–616. DOI: 10.3233/JCS-160556. URL: <https://doi.org/10.3233/JCS-160556>.

-  Detlef Kähler, Ralf Küsters, and Tomasz Truderung. “Infinite State AMC-Model Checking for Cryptographic Protocols”. In: **LICS**. IEEE Computer Society, 2007, pp. 181–192.
-  Ralf Küsters and Tomasz Truderung. “On the Automatic Analysis of Recursive Security Protocols with XOR”. In: **STACS**. Ed. by Wolfgang Thomas and Pascal Weil. Vol. 4393. Lecture Notes in Computer Science. Springer, 2007, pp. 646–657. ISBN: 978-3-540-70917-6.
-  Gavin Lowe. “Towards a Completeness Result for Model Checking of Security Protocols”. In: **Journal of Computer Security** 7.1 (1999), pp. 89–146. URL: <http://content.iospress.com/articles/journal-of-computer-security/jcs118>.
-  Jonathan K. Millen. “A Necessarily Parallel Attack”. In: **In Workshop on Formal Methods and Security Protocols**. 1999.



Emil L. Post. “A variant of a recursively unsolvable problem”. In: **Bull. Amer. Math. Soc.** 52.4 (Apr. 1946), pp. 264–268. URL:

<https://projecteuclid.org:443/euclid.bams/1183507843>.



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.



Henning Schnoor. “Deciding Epistemic and Strategic Properties of Cryptographic Protocols”. In: **ESORICS**. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Vol. 7459. Lecture Notes in Computer Science. Springer, 2012, pp. 91–108. ISBN: 978-3-642-33166-4.



T. J. Schaefer. “The complexity of satisfiability problems”. In: **Proceedings 10th Symposium on Theory of Computing**. ACM Press, 1978, pp. 216–226.