

Formal Verification of Cryptographic Protocols

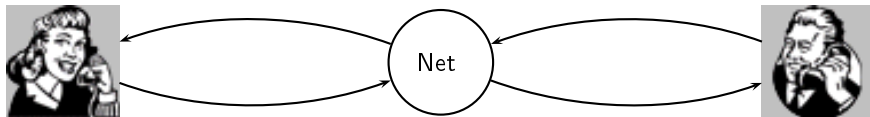
Tutorial

Steve Kremer

Laboratoire Spécification et Vérification
ENS Cachan



Cryptographic Protocols



Protocol

↔ rules describing message exchanges

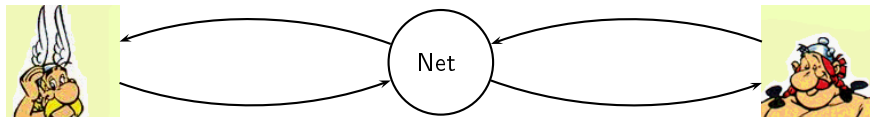
Goal

↔ secure communications: *secret*, *authentication*, *anonymity* ...

Applications

↔ mobile phones, electronic voting, homebanking, electronic commerce, ...

Cryptographic Protocols



Protocol

↔ rules describing message exchanges

Goal

↔ secure communications: *secret*, *authentication*, *anonymity* ...

Applications

↔ mobile phones, electronic voting, homebanking, electronic commerce, ...

Cryptographic Protocols



Protocol

↔ rules describing message exchanges

Goal

↔ secure communications: *secret*, *authentication*, *anonymity* ...

Applications

↔ mobile phones, electronic voting, homebanking, electronic commerce, ...

Symmetric key and public key encryption

- Symmetric key encryption



Symmetric key and public key encryption

- Symetric key encryption



- Public key encryption



The Needham-Schroeder protocol (1978)



- $$\begin{array}{lll} A & \rightarrow & B : \{A, N_a\}_{\text{pub}(B)} \\ B & \rightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \rightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$



The Needham-Schroeder protocol (1978)



- $$\begin{array}{lll} A & \rightarrow & B : \{A, N_a\}_{\text{pub}(B)} \\ B & \rightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \rightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$



The Needham-Schroeder protocol (1978)



$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$
• $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



The Needham-Schroeder protocol (1978)


$$\begin{array}{lll} A & \rightarrow & B : \quad \{A, N_a\}_{\text{pub}(B)} \\ B & \rightarrow & A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\ A & \rightarrow & B : \quad \{N_b\}_{\text{pub}(B)} \end{array}$$


Questions

- Is N_b a shared secret between A et B ?
- When B receives $\{N_b\}_{\text{pub}(B)}$, does this message really come from A ?

The Needham-Schroeder protocol (1978)


$$\begin{array}{lll} A & \rightarrow & B : \quad \{A, N_a\}_{\text{pub}(B)} \\ B & \rightarrow & A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\ A & \rightarrow & B : \quad \{N_b\}_{\text{pub}(B)} \end{array}$$


Questions

- Is N_b a shared secret between A et B ?
- When B receives $\{N_b\}_{\text{pub}(B)}$, does this message really come from A ?

An **attack** has been found on this protocol **17 years** after its publication !

Attack on the Needham-Schroeder protocol



Agent *A*



Intruder *I*



Agent *B*

$$\begin{array}{lll} A & \longrightarrow & B : \{N_a, A\}_{\text{pub}(B)} \\ B & \longrightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \longrightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$

Attack on the Needham-Schroeder protocol



Agent A

$\xrightarrow{\{N_a, A\}_{\text{pub}(I)}}$



Intruder I



Agent B

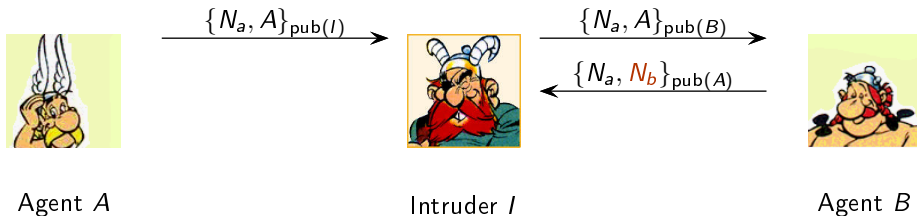
- $$\begin{array}{lll} A & \longrightarrow & B : \{N_a, A\}_{\text{pub}(B)} \\ B & \longrightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \longrightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$

Attack on the Needham-Schroeder protocol

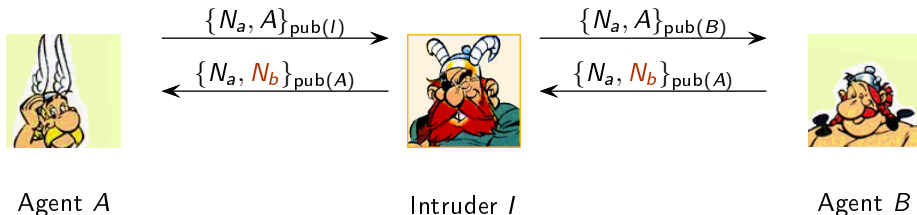


- - $A \longrightarrow B : \{N_a, A\}_{\text{pub}(B)}$
 - $B \longrightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$
 - $A \longrightarrow B : \{N_b\}_{\text{pub}(B)}$

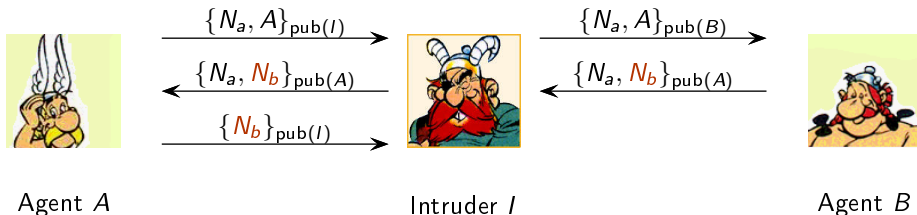
Attack on the Needham-Schroeder protocol



Attack on the Needham-Schroeder protocol

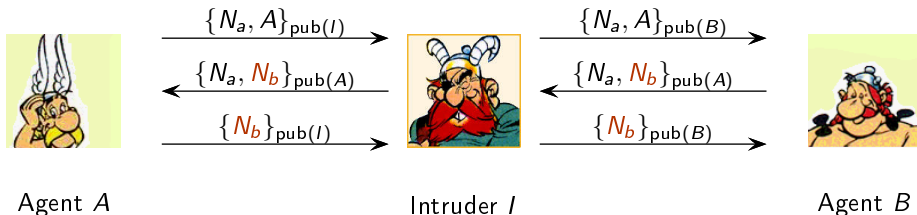


Attack on the Needham-Schroeder protocol



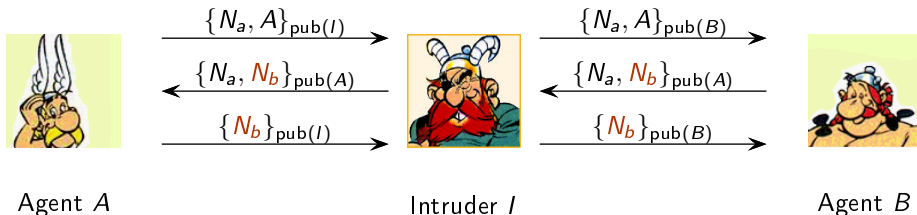
- $A \longrightarrow B : \{N_a, A\}_{\text{pub}(B)}$
- $B \longrightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$
- $A \longrightarrow B : \{N_b\}_{\text{pub}(B)}$

Attack on the Needham-Schroeder protocol



$A \longrightarrow B : \{N_a, A\}_{\text{pub}(B)}$
 $B \longrightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$
• $A \longrightarrow B : \{N_b\}_{\text{pub}(B)}$

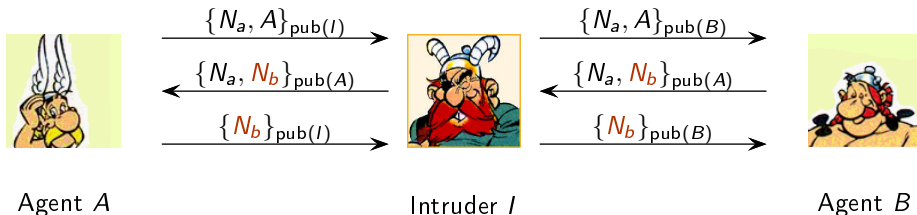
Attack on the Needham-Schroeder protocol



Answers

- Is N_b a shared secret between A et B ?
 \hookrightarrow No

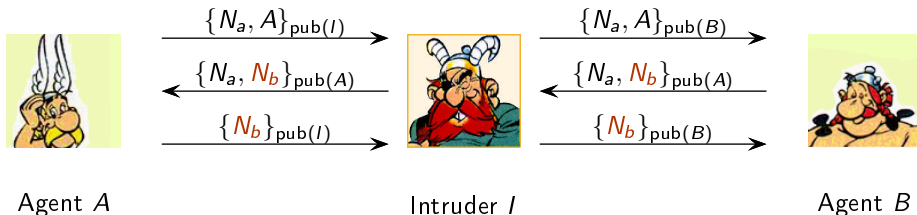
Attack on the Needham-Schroeder protocol



Answers

- Is N_b a shared secret between A et B ?
 \hookrightarrow No
- When B receives $\{N_b\}_{\text{pub}(B)}$, does this message really come from A ?
 \hookrightarrow No

Attack on the Needham-Schroeder protocol



Answers

- Is N_b a shared secret between A et B ?
 \hookrightarrow No
- When B receives $\{N_b\}_{\text{pub}(B)}$, does this message really come from A ?
 \hookrightarrow No

Remark: The encryption algorithms have not been broken
 \hookrightarrow the flaw is in the **logic** of the protocol

Security protocols are 3 line programs that people still manage to get wrong

Roger Needham

For more security protocols, attacks and info consult

SPORE (Security Protocols Open Repository)

<http://www.lsv.ens-cachan.fr/spore/index.html>

Need for rigorous methods to **analyze** and **proof** protocols **correct**, preferably **automated**

Seminal paper by Dolev and Yao in 1981 defines an abstract model for reasoning about security protocols [DolevYao81]

- Protocol messages are modelled using **abstract term algebras**
 - ↪ **perfect cryptography** assumption: *“A message can only be decrypted if the right decryption key is known”*
- the adversary has complete **control of the network**
 - ↪ all messages are sent to the intruder; hence the intruder can
 - **remove** message
 - **alter** messages
 - **insert** new messages which he can construct
- the intruder can initiate **new sessions**
 - ↪ the intruder decides who executes a protocol with whom: **any number of interleaved sessions**

Modelling messages using abstract term algebras

Term algebra

- a **signature** Σ with a finite set of **function symbols** $\{f_1, \dots, f_m\}$, each given with its arity $ar(f_i)$; **constants** are functions with arity 0
- an infinite set of **names**: \mathcal{N}
- an infinite set of **variables**: \mathcal{X}
- **terms** are generated by the following grammar

T	$::=$	term
	x	variable $x \in \mathcal{X}$
	a	name $a \in \mathcal{N}$
	$f(T_1, \dots, T_k)$	application of function symbol $f \in \Sigma$ ($ar(f) = k$)

$names(T)$ and $vars(T)$ denote the set of names and variables of term T

A term T is said to be **ground** when $vars(T) = \emptyset$.

Examples of terms

Let $\Sigma = \{enc, dec, pair, fst, snd\}$ such that

- $ar(enc) = ar(dec) = ar(pair) = 2$
- $ar(fst) = ar(snd) = 1$

Examples

- $t_1 = enc(m, k)$ is a **ground** term ($names(t_1) = \{m, k\}$ and $vars(t) = \emptyset$)
- $t_2 = dec(snd(pair(x, enc(m, k))), k)$ where $names(t_2) = \{m, k\}$ and $vars(t_2) = x$

We would like to give a semantics to terms such that $dec(snd(pair(x, enc(m, k))), k)$ and m are equivalent

Semantics of terms via equational theories

Let E be an **equational theory** over the symbols in Σ .

Example: Consider the following equational theory E

$$\begin{aligned} \text{dec}(\text{enc}(x, y), y) &= x \\ \text{enc}(\text{dec}(x, y), y) &= x \\ \text{fst}(\text{pair}(x, y)) &= x \\ \text{snd}(\text{pair}(x, y)) &= y \end{aligned}$$

E partitions the set of terms generated by Σ into (an infinite number of) **equivalence classes**. When two terms M and N are in the same equivalence class, this is noted $M =_E N$.

Example:

We have that $\text{dec}(\text{snd}(\text{pair}(x, \text{enc}(m, k))), k) =_E \text{dec}(\text{enc}(m, k), k) =_E m$
(while $\text{dec}(\text{snd}(\text{pair}(x, \text{enc}(m, k))), k) \neq m$)

Free term algebras vs explicit destructors

Many works consider a **free term algebra**, rather than equational theories

- each term has a **unique representation**
- **no** explicit destructors

Example

Semantics are given by intruder deduction rules

$$\frac{T \vdash \text{enc}(a, k) \quad T \vdash k}{T \vdash a}$$

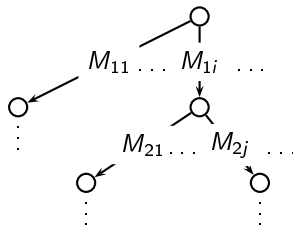
The destructor symbol **dec** does not exist explicitly and hence a term **dec**(*t*, *k*) does not exist either

Here we consider **explicit destructors**

- some **attacks** are not discovered without explicit destructors [Millen03], [MeadowsLynch04]
- **protocol specification** is easier and more natural

Difficulties in the automated verification

- equational theories: even $=_E$ may be undecidable
- the intruder can construct an infinite number of messages: infinite branching
- the intruder can initiate an unbounded number of sessions: infinite depth



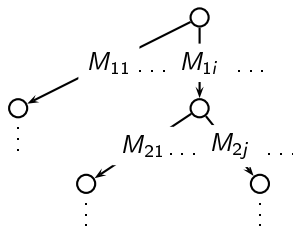
Deciding secrecy of a term is **undecidable** in general!

Possible approaches

- loose **termination** and/or **completeness**
- take **restrictions**: particular equational theories, bounded number of sessions, passive adversaries

Difficulties in the automated verification

- **equational theories**: even $=_E$ may be undecidable
- the intruder can construct an infinite number of messages: infinite branching
- the intruder can initiate an unbounded number of sessions: infinite depth



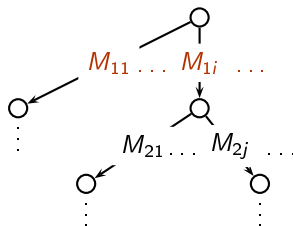
Deciding secrecy of a term is **undecidable** in general!

Possible approaches

- loose **termination** and/or **completeness**
- take **restrictions**: particular equational theories, bounded number of sessions, passive adversaries

Difficulties in the automated verification

- **equational theories**: even $=_E$ may be undecidable
- the intruder can construct an **infinite number of messages**: infinite branching
- the intruder can initiate an unbounded number of sessions: infinite depth



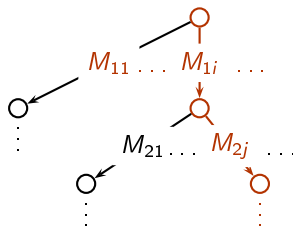
Deciding secrecy of a term is **undecidable** in general!

Possible approaches

- loose **termination** and/or **completeness**
- take **restrictions**: particular equational theories, bounded number of sessions, passive adversaries

Difficulties in the automated verification

- **equational theories**: even $=_E$ may be undecidable
- the intruder can construct an **infinite number of messages**: infinite branching
- the intruder can initiate an **unbounded number of sessions**: infinite depth



Deciding secrecy of a term is **undecidable** in general!

Possible approaches

- loose **termination** and/or **completeness**
- take **restrictions**: particular equational theories, bounded number of sessions, passive adversaries

Outline

- 1 Introduction
- 2 Passive adversary
- 3 Active adversary with finite number of sessions
- 4 Active adversary with unbounded number of sessions
- 5 Conclusion and perspectives

Outline

- 1 Introduction
- 2 **Passive adversary**
- 3 Active adversary with finite number of sessions
- 4 Active adversary with unbounded number of sessions
- 5 Conclusion and perspectives

Deducibility: is a given term secret?

Let $\varphi = \{x_1 = T_1, \dots, x_n = T_n\}$ be a substitution where T_i are ground terms **observed by the intruder** and $\text{dom}(\varphi) = \{x_1, \dots, x_n\}$.

The variables x_1, \dots, x_n are **handles** by the means of which the intruder accesses the corresponding terms.

Definition : deducibility

A ground term T is deducible under an equational theory E from $\varphi = \{x_1 = T_1, \dots, x_n = T_n\}$, denoted $\varphi \vdash_E T$, iff there exists a term M such that $\text{vars}(M) \subseteq \text{dom}(\varphi)$ and $\text{names}(M) \cap \text{names}(\varphi) = \emptyset$ and $M\varphi =_E T$.

Example:

$$\varphi = \{x_1 = \text{enc}(m, k), x_2 = k\}$$

We have that $\varphi \vdash_E m$ as $\text{dec}(x_1, x_2)\varphi = \text{dec}(\text{enc}(m, k), k) =_E m$.

Notations for manipulating terms

The set of **positions** $Pos(t)$ of a term t is inductively defined as follows

$$\begin{aligned} Pos(x) = Pos(n) = Pos(c) &= \{\epsilon\} \quad (x \in \mathcal{X}, n \in \mathcal{N}, c/0 \in \Sigma) \\ Pos(f(t_1, \dots, t_n)) &= \{\epsilon\} \cup_{1 \leq i \leq n} i \cdot Pos(t_i) \end{aligned}$$

Example

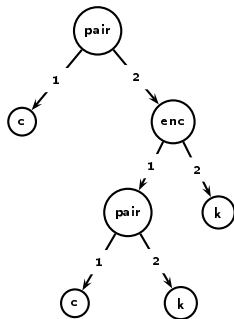
Let $M = pair(c, enc(pair(c, k), k))$

$Pos(t) = \{\epsilon, 1, 2, 21, 22, 211, 212\}$

$t|_p$ denotes the subterm of t rooted at position p , e.g. $M|_{21} = pair(c, k)$

$t[s]_p$ denotes the term in which $t|_p$ has been replaced by s , e.g. $M[c]_{21} = pair(c, enc(c, k))$

$st(t) = \{t|_p \mid p \in Pos(t)\}$ are the **subterms** of t (also extended to sets of terms)



Compact representations of sets of terms

Sets of terms can be represented in a compact way as **DAGs with maximal sharing**

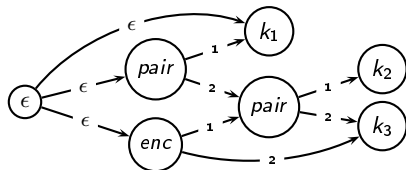
The DAG $(\mathcal{V}, \mathcal{E})$ representing the set of terms T is defined as

- $\mathcal{V} = st(T) \cup \{\epsilon\}$
- $\mathcal{E} = \{v_s \xrightarrow{i} v_e \mid v_s, v_e \in \mathcal{V}, v_s = f(t_1, \dots, t_n), v_e = t_i\} \cup \{\epsilon \xrightarrow{\epsilon} v \mid v \in T\}$

The DAG representing T graph has $n + 1$ vertexes and at most $((n - 1) * m) + n$ edges where $n = |st(T)|$ and m is the maximal arity of function symbols

Example

$$T = \{ \text{pair}(k_1, \text{pair}(k_2, k_3)), \\ \text{enc}(\text{pair}(k_2, k_3), k_1) \}$$



$\|T\|_d$ denotes the DAG-size of the set of terms T

Term rewriting systems

Given an equational theory $E = \{l_i = r_i\}$, we associate to E the TRS $\mathcal{R}_E = \{l_i \rightarrow r_i\}$

Given a TRS \mathcal{R}

- $t \rightarrow_{\mathcal{R}} s$ if $l \rightarrow r \in \mathcal{R}$ and there exists a position p of T and a substitution σ , such that $t|_p = l\sigma$ and $s = t[r\sigma]_p$
- $\rightarrow_{\mathcal{R}}^*$ is the reflexive, transitive closure of $\rightarrow_{\mathcal{R}}$
- \mathcal{R} is **terminating** if there exists no infinite chain $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \dots$
- \mathcal{R} is **confluent** if for all t_1, t_2, t_3 , such that $t_1 \rightarrow_{\mathcal{R}}^* t_2$, $t_1 \rightarrow_{\mathcal{R}}^* t_3$, there exists t_4 , such that $t_2 \rightarrow_{\mathcal{R}}^* t_4$ and $t_3 \rightarrow_{\mathcal{R}}^* t_4$.
- \mathcal{R} is **convergent** if both confluent and terminating
- A term t is in **\mathcal{R} -normal form** if there is no term s such $t \rightarrow_{\mathcal{R}} s$
- $t = s \downarrow_{\mathcal{R}}$ is the normal form of s if $s \rightarrow_{\mathcal{R}}^* t$ and t is in \mathcal{R} -normal form

Convergent public collapsing equational theories

Definition [Convergent public key collapsing]

[DelauneJacquemard04]

An equational theory E is **convergent public key collapsing** if \mathcal{R}_E is convergent and for any rule $\ell \rightarrow r \in \mathcal{R}_E$

- $r \in \text{vars}(\ell)$ or r is ground and \mathcal{R}_E -normal
- if $\ell = f(\ell_1, \dots, \ell_n)$ then for any position p in ℓ_i ($1 \leq i \leq n$), such that $\ell_i|_p = g(t_1, \dots, t_m)$ either $g(t_1, \dots, t_m)$ is ground and \mathcal{R}_E -normal or there exists j ($1 \leq j \leq m$), such that $t_j = r$

Examples:

- ✓ Pairing: $\text{fst}(\text{pair}(x, y)) = x$, $\text{snd}(\text{pair}(x, y)) = y$
- ✓ Symmetric encryption: $\text{dec}(\text{enc}(x, y), y) = x$
- ✓ Probabilistic symmetric encryption: $\text{dec}(\text{enc}(x, y, r), y) = x$
- ✗ XOR: $x \oplus x = 0$, $x \oplus 0 = x$, $x \oplus y = y \oplus x$, $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
- ✗ Blind signatures: $\text{unblind}(\text{sig}(\text{blind}(x, r), y), r) = \text{sig}(x, y)$

Locality lemma

following [McAllester93](#)

Let E be a convergent public collapsing theory, $\varphi = \{x_1 = M_1, \dots, x_n = M_n\}$ and $\varphi \vdash_E T$. Every minimal size proof M is such that for any $t \in st(M\varphi)$, we have that $t \downarrow_{\mathcal{R}_E} \in st(\{M_1, \dots, M_n\} \cup \{T\} \cup \{c \mid c \in \Sigma, ar(c) = 0\})$.

Proof: The proof is done by induction on M . The difficult case is when a destructor is applied: done by a case analysis on the conditions on the TRS.

Example

Let $\varphi = \{x_1 = enc(a, k_1), x_2 = pair(k_1, k_2)\}$. We have that $\varphi \vdash_E a$, because of the (minimal size) proof $M = dec(x_1, fst(x_2))$. The proof indeed only includes subterms of φ and T . We have that $M\varphi = dec(enc(a, k_1), fst(pair(k_1, k_2)))$ and

$$st(M\varphi) = \{ \quad dec(enc(a, k_1), fst(pair(k_1, k_2))) \downarrow_{\mathcal{R}_E} = a, enc(a, k_1), a, k_1, \\ fst(pair(k_1, k_2)) \downarrow_{\mathcal{R}_E} = k_1, pair(k_1, k_2), k_2 \}$$

Theorem

[DelauneJacquemard04]

Let $\varphi = \{x_1 = M_1, \dots, x_n = M_n\}$. Let t be a ground term. When E is convergent public collapsing $\varphi \vdash_E t$ can be decided in polynomial time in $||\{M_1, \dots, M_n, t\}||_d$.

Proof: Due to the locality Lemma the proof contains only subterms of φ and t or constants. Define the set of Horn clauses

$$S = \left\{ \begin{array}{ll} \Rightarrow p(s) & \begin{array}{l} | s \in \{M_1 \downarrow_{\mathcal{R}_E}, \dots, M_n \downarrow_{\mathcal{R}_E}\} \\ \text{or } s/0 \in \Sigma \end{array} \\ p(s_1), \dots, p(s_n) \Rightarrow p(f(s_1, \dots, s_n) \downarrow_{\mathcal{R}_E}) & \begin{array}{l} | s_1, \dots, s_n, f(s_1, \dots, s_n) \downarrow_{\mathcal{R}_E} \\ \in st(\{M_1, \dots, M_n, y\}), f \in \Sigma \end{array} \\ p(t \downarrow_{\mathcal{R}_E}) \Rightarrow & \end{array} \right\}$$

We have that $\varphi \vdash_E t$ iff S is not satisfiable. HORN-SAT can be decided in linear time in $|S|$ and $|S|$ is polynomial in $||\{M_1, \dots, M_n, t\}||_d$ (the degree is the maximum arity of Σ).

Many other results for different families of equational theories:

[Comon-LundhTreinen03], [AbadiCortier04], [LafourcadeLugiezTreinen04], ...

Outline

- 1 Introduction
- 2 Passive adversary
- 3 Active adversary with finite number of sessions
- 4 Active adversary with unbounded number of sessions
- 5 Conclusion and perspectives

Representing protocols as roles

Represent the **local view** of the protocol

Role A

$$\begin{array}{l} \text{protocol step 1} \\ \vdots \\ \text{protocol step } n \end{array} \quad \left\{ \begin{array}{l} \text{receive}(x_1) \\ \{M_1^1 = N_1^1, \dots, M_{\ell_1}^1 = N_{\ell_1}^1\} \\ \text{send}(t_1) \\ \\ \text{receive}(x_n) \\ \{M_1^n = N_1^n, \dots, M_{\ell_n}^n = N_{\ell_n}^n\} \\ \text{send}(t_n) \end{array} \right.$$

We suppose that $\text{vars}(M_j^i, N_j^i, t_i) \subseteq \{x_k \mid k \leq i\}$.

A **scenario** is a finite set of roles

Example: the Needham Schroeder protocol

$$\begin{aligned} A \rightarrow B & : \text{enc}(\text{pair}(N_A, \text{pk}(A)), \text{pk}(B)) \\ B \rightarrow A & : \text{enc}(\text{pair}(N_A, N_B), \text{pk}(A)) \\ A \rightarrow B & : \text{enc}(N_B, \text{pk}(B)) \end{aligned}$$

Role1($N_A, A, \text{pk}B$)

Role2(N_B, B)

$\text{send}(\text{enc}(\text{pair}(N_A, \text{pk}(A)), \text{pk}(B)))$

$\text{recv}(x_1^2)$
 $\text{send}(\text{enc}(\text{pair}(\text{fst}(\text{dec}(x_1^2, \text{sk}(B))), N_B),$
 $\text{snd}(\text{dec}(x_1^2, \text{sk}(B)))))$

$\text{recv}(x_1^1)$
 $\{\text{fst}(\text{dec}(x_1^1, \text{sk}(A))) = N_A\}$
 $\text{send}(\text{enc}(\text{snd}(\text{dec}(x_1^1, \text{sk}(A)))))$

$\text{recv}(x_2^2)$
 $\{\text{fst}(\text{dec}(x_2^2, \text{sk}(B))) = N_B\}$
 $\text{send}(\text{enc}(\text{snd}(\text{dec}(x_2^2, \text{sk}(A)))))$

Scenario: $\{\text{Role1}(na1, a, \text{pk}(b)), \text{Role1}(na2, a, \text{pk}(i)), \text{Role2}(nb, b)\}$

A **configuration** is of the form r_1, \dots, r_n, ϕ

Concrete **execution step**

$$r_1, \dots, r_i, \dots, r_n, \phi \rightarrow r_1, \dots, r'_i, \dots, r_n, \phi'$$

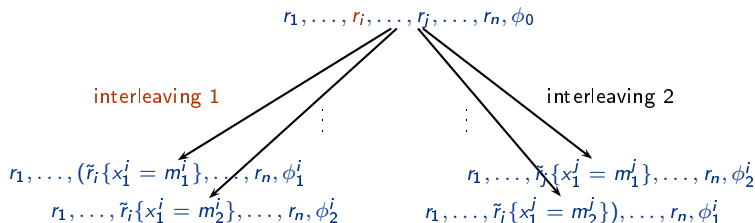
- $r_i = \text{recv}(x), \{M_j = N_j\}, \text{send}(t). \tilde{r}_i$
- $\phi \vdash_E m$, such that $M_j\{x = m\} =_E N_j\{x = m\}$
- $\phi' = \phi \circ \{y = t\{x = m\}\}$
- $r'_i = \tilde{r}_i\{x = m\}$

Given a scenario $\{r_1, \dots, r_n\}$ the **initial state** corresponds to r_1, \dots, r_n, ϕ_0 , where ϕ_0 represents the initial knowledge of the intruder, e.g. public keys, identities, ...

Insecurity of the protocol w.r.t. a secret s : a configuration r_1, \dots, r_n, ϕ is reachable, such that $\phi \vdash_E s$

Infinite state system

The concrete semantics yields an **infinite state system**



The resulting transition system

- **infinite branching**: the intruder may deduce an infinite number of terms
- exponential number of **interleavings**
- each path is **finite** (fixed number of roles, each finite)

Symbolic semantics and lazy intruders

Main idea: keep **variables** and **constraints** instead of ground messages

An intruder constraint is either $\phi \vdash x$ or an equation $M = N$

A symbolic configuration: $r_1, \dots, r_n, \mathcal{C}, \phi$

Symbolic execution step: $r_1, \dots, r_i, \dots, r_n, \mathcal{C}, \phi \rightarrow r_1, \dots, r'_i, \dots, r_n, \mathcal{C}', \phi'$

- $r_i = \text{recv}(x), \{M_j = N_j\}, \text{send}(t).r'_i$
- $\mathcal{C}' = \mathcal{C} \cup \{\phi \vdash x\} \cup \{M_j = N_j\}$
- $\phi' = \phi \circ \{y = t\}$

The symbolic transition system has only **one branch** per interleaving

The intruder instantiates variables **lazily**

Intruder constraint systems

Any **interleaving** of roles leads to an **intruder constraint system** is of the form

$$\begin{array}{rcl} \phi_0 & \Vdash & x_1 \\ \phi_1 = \phi_0 \circ \{y_1 = t_1\} & \Vdash & x_2 \\ \phi_2 = \phi_1 \circ \{y_2 = t_2\} & \Vdash & x_3 \\ & \vdots & \\ \phi_n = \phi_{n-1} \circ \{y_n = t_n\} & \Vdash & x_n \\ & \{M = N\} & \end{array}$$

A **solution** of \mathcal{C} is a grounding substitution σ , such that $\phi_j \vdash_E x_j$ ($1 \leq j \leq n$) and $M\sigma =_E N\sigma$.

σ is a solution iff it yields a valid concrete execution

To decide the secrecy of a term s , add an additional constraint

$$\phi_{n+1} = \phi_n \circ \{y_{n+1} = t_{n+1}\} \Vdash s$$

Deciding insecurity

Theorem

Deciding insecurity with respect to a secret s in presence of a convergent public collapsing theory is NP-complete.

NP-easy

- guess an interleaving of the roles
- construct the corresponding constraint system
- solving the constraint system is in NP [DelauneJacquemard04]

NP-hard

Let X_1, \dots, X_n be propositional variables and consider the following instance of 3-SAT: $\bigwedge_{1 \leq i \leq m} (X_{\alpha_{i,1}}^{\epsilon_{i,1}} \vee X_{\alpha_{i,2}}^{\epsilon_{i,2}} \vee X_{\alpha_{i,3}}^{\epsilon_{i,3}})$ where $\epsilon_{i,j} \in \{0, 1\}$ and $X^1 = X$ and $X^0 = \neg X$.

Consider $\Sigma = \{\top/0, \perp/0, \text{tup}/n, \pi_1/1, \dots, \pi_n/1, \neg/1, \wedge/2, \vee/2\}$ and the expected convergent public collapsing equational theory

$$\text{recv}(x), \{\bigwedge_{1 \leq i \leq m} (\pi_{\alpha_{i,1}}(x)^{\epsilon_{i,1}} \vee \pi_{\alpha_{i,2}}(x)^{\epsilon_{i,2}} \vee \pi_{\alpha_{i,3}}(x)^{\epsilon_{i,3}}) = \top\}, \text{send}(\text{secret})$$

Constraint solving procedure


Initially: $\mathcal{D} = \emptyset$ and $\mathcal{S} = \emptyset$

$(\mathcal{C}; \mathcal{D}; \mathcal{S})$

Constraint solving procedure

Initially: $\mathcal{D} = \emptyset$ and $\mathcal{S} = \emptyset$

Narrowing
Syntactic Unification



$(\mathcal{C}; \mathcal{D}; \mathcal{S})$

$$\frac{\mathcal{C} \cup \{e[u]\}; \mathcal{D}; \mathcal{S}}{\mathcal{C} \cup \{e[r]\}; \mathcal{D}\eta; \mathcal{S}\eta \cup \eta}$$

Narrowing

e is an equation or an intruder constraint,
 $\ell \rightarrow r$ is a fresh variant of a rule of \mathcal{R}_E ,
 $\eta = mgu(\ell\mathcal{S}, u\mathcal{S})$,
 $root(\ell) = root(u)$.

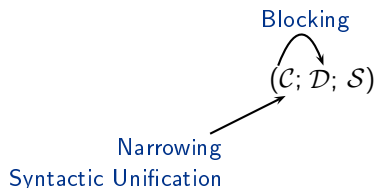
$$\frac{\mathcal{C} \cup \{t_1 = t_2\}; \mathcal{D}; \mathcal{S}}{\mathcal{C}; \mathcal{D}\eta; \mathcal{S}\eta \cup \eta}$$

Syntactic Unification

$\eta = mgu(t_1\mathcal{S}, t_2\mathcal{S})$.

Constraint solving procedure

Initially: $\mathcal{D} = \emptyset$ and $\mathcal{S} = \emptyset$

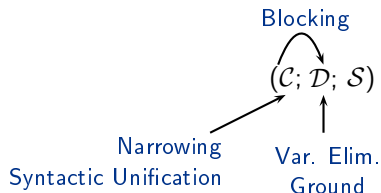


$$\frac{\mathcal{C} \cup \{c\}; \mathcal{D}; \mathcal{S}}{\mathcal{C}; \mathcal{D} \cup \{c\mathcal{S}\}; \mathcal{S}}$$

Blocking
 c is an intruder constraint.

Constraint solving procedure

Initially: $\mathcal{D} = \emptyset$ and $\mathcal{S} = \emptyset$



$$\frac{\mathcal{C}; \mathcal{D}; \mathcal{S}}{\mathcal{C}; \mathcal{D}[x = t]; \mathcal{S}[x = t] \cup [x = t]}$$

$$\frac{\mathcal{C}; \mathcal{D} \cup \{T \Vdash u\}; \mathcal{S}}{\mathcal{C}; \mathcal{D}; \mathcal{S}}$$

Variable Elimination

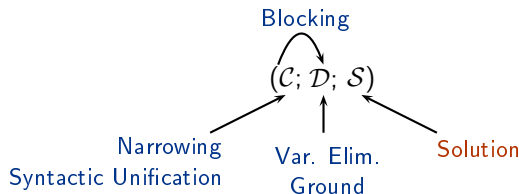
$x \in \text{vars}(\mathcal{D}), t \in \text{st}(\mathcal{D}) \setminus \text{vars}(\mathcal{D})$,
there is no occurrence of x in t .

Ground

if $T \vdash_E u$.

Constraint solving procedure

Initially: $\mathcal{D} = \emptyset$ and $\mathcal{S} = \emptyset$



$$\frac{\mathcal{C}; \mathcal{D}; \mathcal{S}}{\mathcal{C}; \mathcal{D}[x = t]; \mathcal{S}[x = t] \cup [x = t]}$$

Variable Elimination

$x \in \text{vars}(\mathcal{D}), t \in \text{st}(\mathcal{D}) \setminus \text{vars}(\mathcal{D})$,
there is no occurrence of x in t .

$$\frac{\mathcal{C}; \mathcal{D} \cup \{T \Vdash u\}; \mathcal{S}}{\mathcal{C}; \mathcal{D}; \mathcal{S}}$$

Ground

if $T \vdash_E u$.

Finally: $\mathcal{C} = \emptyset$ and $\mathcal{D} = \emptyset$ if a solution exists

Tools based on constraint solving

Several tools exploit similar ideas based on **constraint solving**

- Constraint solver by Millen and Shmatikov

<http://www.csl.sri.com/users/millen/capsl/constraints.html>

- **CoProVe** - Constraint-based Security Protocol Verifier: optimised and extended version by Corin and Etalle

<http://wwwes.cs.utwente.nl/coprove/>

- **AVISPA tool suite** - Automated Validation of Internet Security Protocols and Applications: several tools are based on constraint solving

<http://www.avispa-project.org/>

Outline

- 1 Introduction
- 2 Passive adversary
- 3 Active adversary with finite number of sessions
- 4 Active adversary with unbounded number of sessions
- 5 Conclusion and perspectives

Horn clauses

A **Horn clause** is a logical formula of the form

$$\frac{L_1, \dots, L_n}{L} \quad (\equiv \neg L_1 \vee \dots \vee \neg L_n \vee L)$$

First order Horn clauses provide a simple and uniform formalism to

- model **abilities of the attacker**
- model the **rules of the protocol**
- verify an **unbounded** number of sessions: the intruder can create new sessions

Horn clauses are for instance used as a low level representation (translation from a high-level language to Horn clauses) in the **ProVerif** tool [Blanchet2001]

<http://www.di.ens.fr/~blanchet/crypto.html>

T	$::=$	term
	x	variable x
	$a[T_1, \dots, T_n]$	name a
	$f(T_1, \dots, T_k)$	application of $f \in \Sigma$ ($ar(f) = k$)
F	$::=$	facts
	$p(M_1, \dots, M_n)$	application of predicate p
R	$::=$	rule
	$F_1 \wedge \dots \wedge F_n \rightarrow F$	implication

Intruder capacities as Horn clauses

To model the capacities of the intruder

- introduction of a special predicate $I(m)$ to model intruder knowledge
- $I(m)$ is true iff the intruder knows message m

Let $f \in \Sigma$ be a function symbol with $ar(f) = n$. f is modeled by the rule

$$\frac{I(x_1), \dots, I(x_n)}{I(f(x_1, \dots, x_n))}$$

Suppose that we are given the equational theory E and the associated rewriting system \mathcal{R}_E . E is modeled by the rules

$$\frac{I(\ell)}{I(r)}$$

where $\ell \rightarrow r \in \mathcal{R}_E$

Initial knowledge: if a ground term t is initially known we add the rule $I(t)$

Example: Intruder capacities

Consider the **signature**

$$\Sigma = \{enc/2, dec/2, pair/2, fst/1, snd/1\}$$

and the (convergent) **equational theory**

$$E = \{dec(enc(x, y), y) = x, fst(pair(x, y)) = x, snd(pair(x, y)) = y\}$$

We obtain the following **rules**

$$\frac{I(x) \quad I(y)}{I(enc(x, y))} \quad \frac{I(x) \quad I(y)}{I(dec(x, y))} \quad \frac{I(x) \quad I(y)}{I(pair(x, y))} \quad \frac{I(x)}{I(fst(x))} \quad \frac{I(x)}{I(snd(x))}$$
$$\frac{I(dec(enc(x, y), y))}{I(x)} \quad \frac{I(fst(pair(x, y)))}{I(x)} \quad \frac{I(snd(pair(x, y)))}{I(y)}$$

Protocol rules as Horn clauses

Example: the Needham Schroeder protocol modeled in terms of Horn clauses

$$\begin{array}{lll} A & \longrightarrow & B : \{N_a, A\}_{\text{pub}(B)} \\ B & \longrightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \longrightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$

$$\begin{array}{c} I(pk(x)) \\ \hline I(enc((Na[pk(x)], pk(sA[])), pk(x))) \\ \\ \frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))} \\ \\ \frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))} \end{array}$$

Initiating a new session

The intruder chooses with whom A starts the protocol

Protocol rules as Horn clauses

Example: the Needham Schroeder protocol modeled in terms of Horn clauses

$$\begin{array}{lll} A & \longrightarrow & B : \{N_a, A\}_{\text{pub}(B)} \\ B & \longrightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \longrightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$

$$\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))}$$

$$\frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}$$

$$\frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))}$$

Modelling fresh values

Fresh values are functions of the “parameters” of the protocol

Protocol rules as Horn clauses

Example: the Needham Schroeder protocol modeled in terms of Horn clauses

$$\begin{array}{lll} A & \longrightarrow & B : \{N_a, A\}_{\text{pub}(B)} \\ B & \longrightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \longrightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$

$$\frac{\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))} \quad \frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}}{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))} \\ I(encrypt(y, pk(x)))$$

Protocol rules as Horn clauses

Example: the Needham Schroeder protocol modeled in terms of Horn clauses

$$\begin{array}{lll} A & \longrightarrow & B : \{N_a, A\}_{\text{pub}(B)} \\ B & \longrightarrow & A : \{N_a, N_b\}_{\text{pub}(A)} \\ A & \longrightarrow & B : \{N_b\}_{\text{pub}(B)} \end{array}$$

$$\frac{\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))} \quad \frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}}{\frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))}}$$

Approximations

Modeling security protocols by Horn clauses introduces **approximations**

- **fresh values** are modelled as **names**, which are functions of previously received values

if the intruder sends the same previous value, the same “fresh” name will be used

- there is **no order** on the rules

for instance a protocol step can be executed **several times**

These approximations can lead to **false attacks**

In practice, false attacks are very seldom

The approximations are **correct**

If a correction proof is given in the model of Horn clauses, the protocol is **also correct** in a **precise model**

Definition [Implication between rules]

$(H_1 \rightarrow C_1) \Rightarrow (H_2 \rightarrow C_2)$ iff there exists a substitution σ such that $C_1\sigma = C_2$ and $H_1\sigma \subseteq H_2$ (H_1 and H_2 are sets of hypotheses)

Definition [Derivability]

Let F be a ground fact and B a set of rules. F is derivable from B iff there exists a finite tree such that

1. all nodes (except the root) are labelled by a rule $R \in B$
2. edges are labelled by facts
3. if a tree contains a node labelled by a rule R with an incoming edge, labelled F_0 and n outgoing edges labelled F_1, \dots, F_n then $R \Rightarrow \{F_1, \dots, F_n\} \rightarrow F_0$
4. The root has an outgoing edge labelled by F

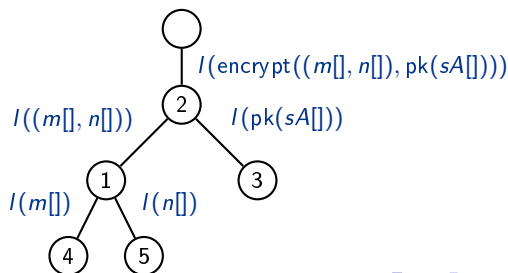
A ground term S is **secret** if it is not possible to **derive** the fact $I(S)$ from the rules describing the intruder capacities and the protocol rules

Example

Suppose that we are given the following rules

$$\begin{array}{ll} I(x) \wedge I(y) \rightarrow I((x, y)) & (1) \\ I(x) \wedge I(y) \rightarrow I(\text{encrypt}(x, y)) & (2) \\ I(\text{pk}(sA[])) & (3) \\ I(m[]) & (4) \\ I(n[]) & (5) \end{array}$$

We can derive $I(\text{encrypt}((m[], n[]), \text{pk}(sA[])))$ as follows:



Automatisation ?

Given a set of rules (Horn clauses), can a fact F be derived from these rules

This problem corresponds to the problem solved by **Prolog**

But: the classic Prolog resolution algorithm does **not terminate** given classical rules used in cryptographic protocols

In [Blanchet2001], Blanchet presents a novel resolution algorithm which “guides” the resolution and is well suited for cryptographic protocols

Preliminary definitions...

Definition [rule combination]

Let $R = H \rightarrow C$ and $R' = H' \rightarrow C'$ be two rules. We suppose that there exists a fact $F_0 \in H'$, such that F_0 and C can be unified, and σ is the most general unifier for C and F_0 . Then

$$R \circ_{F_0} R' = (H \cup (H' \setminus F_0))\sigma \rightarrow C'\sigma$$

Example:

$$R = I(\text{pk}(x)) \rightarrow I(\text{encrypt}(\text{sign}(\text{msg}[], \text{skA}[]), \text{pk}(x)))$$

$$R' = I(\text{encrypt}(m, \text{pk}(\text{sk}))) \wedge I(\text{sk} \rightarrow I(m))$$

Let $F_0 = I(\text{encrypt}(m, \text{pk}(\text{sk})))$. Then

$$R \circ_{F_0} R' = I(\text{pk}(x)) \wedge I(x) \rightarrow I(\text{sign}(\text{msg}[], \text{skA}[]))$$

where $\sigma = \{sk = x, m = \text{sign}(\text{msg}[], \text{skA}[])\}$

Guiding the algorithm

Let S be a finite set of facts. We say that $F \in_r S$ iff there exists a substitution σ of variables by other variables such that $F\sigma \in S$.

In the algorithm S will **guide** the choices for combining rules: one **does not** combine R and R' by $R \circ_{F_0} R'$ if $F_0 \in_r S$

Example: By default $S = \{I(x)\}$ to avoid the following situation. Suppose $I(x) \notin_r S$ and consider the rules

$$I(x) \rightarrow I(pk(x)) \quad (1)$$

$$I(pk(x)) \wedge I(y) \rightarrow \text{encrypt}(y, pk(x)) \quad (2)$$

If we apply the combination $(2) \circ_{I(x)} (1)$ we obtain

$$I(pk(x)) \wedge I(y) \rightarrow pk(\text{encrypt}(y, pk(x))) \quad (3)$$

We can then apply the combination $(3) \circ_{I(x)} (1)$ and obtain

$$I(pk(x)) \wedge I(y) \rightarrow pk(pk(\text{encrypt}(y, pk(x)))) \quad (4)$$

The successive combinations **do not terminate**. Similar problem if $F_0 = I(y)$.

Resolution algorithm : phase 1

$$\text{Let } add(R, B) = \begin{cases} B & \text{if } \exists R' \in B, R' \Rightarrow R \\ \{R\} \cup \{R' \in B \mid R \not\Rightarrow R'\} & \text{else} \end{cases}$$

Let B_0 be the set of rules describing the protocol and the intruder

1. For all $R \in B_0$, $B = add(R, B)$
2. Let $R \in B$, $R = H \rightarrow C$ and $R' \in B$, $R' = H' \rightarrow C'$. Suppose there exists $F_0 \in H'$ such that
 - (a) $R \circ_{F_0} R'$ is defined
 - (b) $\forall F \in H, F \in_r S$ # by default $S = \{I(x)\}$
 - (c) $F_0 \notin_r S$

Then $B = add(R \circ_{F_0} R', B)$

Execute step 2. until reaching a fixed point

3. $B' = \{(H \rightarrow C) \in B \mid \forall F \in H, F \in_r S\}$

After executing phase 1, we have that a ground fact F can be derived from B' iff F can be derived from B_0

Resolution algorithm : phase 2

$\text{derivablerec}(R, B'')$

1. $\text{derivablerec}(R, B'') = \emptyset$ if $\exists R' \in B''. R' \Rightarrow R$ # loop: backtrack
2. else, $\text{derivablerec}(\emptyset \rightarrow C, B'') = \{C\}$ # proof of C
3. else, $\text{derivablerec}(R, B'') = \cup \{ \text{derivablerec}(R' \circ_{F_0} R), \{R\} \cup B'' \mid R' \in B', F_0 \text{ is such that } R' \circ_{F_0} R \text{ is defined} \}$

$\text{derivable}(F) = \text{derivablerec}(\{F\} \rightarrow F, \emptyset)$

Intuitively,

- the hypotheses of R contain the facts that we are currently trying to prove
- the conclusion of R is an instance of F that we initially wanted to prove
- the set B'' contains the rule already encountered during the search

F is derivable from B_0 iff $F \in \text{derivable}(F)$

Remarks on the algorithm

The fixed point in phase 1 may **not terminate**

In practice, this phase terminates on nearly all examples!

The ProVerif tool implements this algorithm with numerous extensions and optimisations...

Conclusions

Verification of cryptographic protocols has **direct application to concrete problems**

Many interesting **theoretical questions**: complexity, algorithms, ...

Beyond deducibility:

- **stronger notions** of secrecy in terms of undistinguishability
- **other properties**: authentication, anonymity, ...

Dolev-Yao like models

- Is the perfect cryptography assumption **sound**?
- Link with more **detailed** models, modeling adversaries as PPT Turing Machines

Indistinguishability properties

Protocol executions P_1 and P_2 are **observationally equivalent**, i.e. cannot be distinguished by any adversary

$$P_1 \approx P_2$$

Strong secrecy:

$$P\{s \leftarrow t_1\} \approx P\{s \leftarrow t_2\}$$

for any terms t_1, t_2 . No partial information is leaked

Anonymity: For instance in a voting system

$$V_1\{v \leftarrow 0\} \mid V_2\{v \leftarrow 1\} \approx V_1\{v \leftarrow 1\} \mid V_2\{v \leftarrow 0\}$$

Few results on automation for the verification of \approx

Computational soundness of formal methods

The formal, symbolic approach (this talk):

- data are represented as **terms**
- **idealized** adversary and cryptography represented as deduction rules or equational theories,
- **automated tools** for analyzing large, complex protocols with multiple or unbounded number of sessions

The computational, cryptographic approach:

- data are represented as **bitstrings**
- adversaries are **PPT Turing Machines**
- cryptographic primitives are **PT algorithms**; the adversary has **negligible probability to break the security**
- proofs are **tedious, by hand and error-prone**

Link between the two approaches ?

Goal: combine the advantages of both approaches, i.e. automatic proofs of complex protocols with strong guarantees in a cryptographic model