

Models and analysis of security protocols

1st Semester 2007-2008

Active Intruder

Pascal Lafourcade

Université Joseph Fourier, Verimag

Master: November 12th 2007

Thanks to Steve Kremer

Last Time (I)

Lecture

- Man in the Middle Needham-Schroeder
- Dolev Yao Model
- Notion of Locality
- Undecidability Result
- Diffie-Hellman

Last Time (I)

Exercise

- Properties of Syntactic Subterms
- Locality Result
- Security against Passive Intruder
- Logical Passive Attack on Shamir 3-Pass Protocol

Outline of Today

- 1 Active Intruder: Security Problem
- 2 Unification Notions
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions
- 4 NP-Hardness for Bounded Number of Sessions
- 5 Tools
- 6 Complexity
- 7 Conclusion

Outline

- 1 Active Intruder: Security Problem
- 2 Unification Notions
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions
- 4 NP-Hardness for Bounded Number of Sessions
- 5 Tools
- 6 Complexity
- 7 Conclusion

The Intruder is the Network (Worst Case)



Listen

Passive: Intruder deduction problem

The Intruder is the Network (Worst Case)

Listen



Passive: Intruder deduction problem

Active Intruder Security problem

- intercept messages (add messages to his knowledge)
- Play messages from his knowledge
- Start new sessions

Execution tree has:

- infinite branching (size of messages is not bounded)
- infinite depth (number of sessions is not bounded)

Active Intruder with bounded number of sessions

- Theoretically: **decidable**
- Interesting **practically**:
 - **Find flaws**
 - Usually attacks use **few sessions** !

Dolev-Yao Deduction System

Deduction System : $T_0 \vdash^? s$

$$(A) \quad \frac{u \in T_0}{T_0 \vdash u}$$

$$(UL) \quad \frac{T_0 \vdash \langle u, v \rangle}{T_0 \vdash u}$$

$$(P) \quad \frac{T_0 \vdash u \quad T_0 \vdash v}{T_0 \vdash \langle u, v \rangle}$$

$$(UR) \quad \frac{T_0 \vdash \langle u, v \rangle}{T_0 \vdash v}$$

$$(C) \quad \frac{T_0 \vdash u \quad T_0 \vdash v}{T_0 \vdash \{u\}_v}$$

$$(D) \quad \frac{T_0 \vdash \{u\}_v \quad T_0 \vdash v}{T_0 \vdash u}$$

Model: actions, roles and protocol

Definition (Action)

An **action** is a couple $(recv(u), send(v))$ such that $u \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{init\}$, $v \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{stop\}$. Denoted $(u \rightarrow v)$.

Model: actions, roles and protocol

Definition (Action)

An **action** is a couple $(recv(u), send(v))$ such that $u \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{init\}$, $v \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{stop\}$. Denoted $(u \rightarrow v)$.

Definition (Role)

A **role** is a finite sequence of actions:

$$(u_1 \rightarrow v_1), \dots, (u_n \rightarrow v_n)$$

such that $vars(v_i) \subseteq \bigcup_{1 \leq j \leq i} vars(u_j)$.

Model: actions, roles and protocol

Definition (Action)

An **action** is a couple $(recv(u), send(v))$ such that $u \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{init\}$, $v \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{stop\}$. Denoted $(u \rightarrow v)$.

Definition (Role)

A **role** is a finite sequence of actions:

$$(u_1 \rightarrow v_1), \dots, (u_n \rightarrow v_n)$$

such that $vars(v_i) \subseteq \bigcup_{1 \leq j \leq i} vars(u_j)$.

Definition (Protocol)

A **protocol** P is a finite set of roles: $P = \{R_1, \dots, R_k\}$

1st Example: second version more general

Example (Needham-schroeder)

1. $A \rightarrow B : \{N_a, A\}_{pk(B)}$
2. $B \rightarrow A : \{N_a, N_b\}_{pk(A)}$
3. $A \rightarrow B : \{N_b\}_{pk(B)}$

Write down each agent's role description, this A talks with anybody.

$$R_A = (init, X_b \rightarrow \{N_a, A\}_{pk(X_b)}, \\ (\{N_a, X_{N_b}\}_{pk(A)} \rightarrow \{X_{N_b}\}_{pk(X_b)}),$$

$$R_B = (\{X_{N_a}, X_A\}_{pk(B)} \rightarrow \{X_{N_a}, N_b\}_{pk(X_A)}) \\ (\{N_b\}_{pk(B)} \rightarrow stop)$$

Scyther Notation

```
A:  const Na: Nonce;
     var Nb: Nonce;

     send(A,B, {Na,A}pk(B));
     recv(B,A, {Na,Nb}pk(A));
     send(A,B, {Nb}pk(B));

B:  const Nb: Nonce;
     var Na: Nonce;

     recv(A,B, {Na,A}pk(B));
     send(B,A, {Na,Nb}pk(A));
     recv(A,B, {Nb}pk(B));
```

Exercise

Denning-Sacco Protocol

1. $A \rightarrow S : \langle A, B \rangle$
2. $S \rightarrow A : \{ \langle \langle B, N_{AB} \rangle, \langle N_S, \{ \langle N_{AB}, \langle A, N_S \rangle \} \rangle_{K_{BS}} \rangle \}_{K_{AS}}$
3. $A \rightarrow B : \{ \langle N_{AB}, \langle A, N_S \rangle \} \}_{K_{BS}}$
4. $B \rightarrow A : \{ S_{AB} \}_{N_{AB}}$

$P_{DS} = \{R_A, R_B, R_S\}$ models one session of A, B and S .

Exercise

Denning-Sacco Protocol

1. $A \rightarrow S : \langle A, B \rangle$
2. $S \rightarrow A : \{ \{ \langle B, N_{AB} \rangle, \langle N_S, \{ \langle N_{AB}, \langle A, N_S \rangle \} \}_{K_{BS}} \} \}_{K_{AS}}$
3. $A \rightarrow B : \{ \langle N_{AB}, \langle A, N_S \rangle \} \}_{K_{BS}}$
4. $B \rightarrow A : \{ S_{AB} \}_{N_{AB}}$

$P_{DS} = \{R_A, R_B, R_S\}$ models one session of A , B and S .

$$R_A = (init, X_B \rightarrow \langle A, X_B \rangle), \\ (\{ \{ \langle X_B, X_{N_{AB}} \rangle, \langle X_{N_S}, Z_A \rangle \} \}_{K_{AS}} \rightarrow Z_A), \\ (\{ W_A \}_{X_{N_{AB}}} \rightarrow stop)$$

$$R_B = (\{ Y_{N_{AB}}, \langle X_A, Y_{N_S} \rangle \}_{K_{BS}} \rightarrow \{ S_{AB} \}_{Y_{N_{AB}}})$$

$$R_S = (\langle X_A, X_B \rangle \rightarrow \{ \langle X_B, N_{AB}, \langle N_S, \{ \langle N_{AB}, \langle X_A, N_S \rangle \} \}_{K_{BS}} \} \}_{K_{AS}})$$

Semantic

Definition (States and Transitions)

A **state** is a couple (T, P) where T is a set of ground terms (intruder knowledge) and P a protocol.

We define a **transition relation** between states $(T, P) \rightarrow (T', P')$ by:

- $R_i \in P, R_i = (u \rightarrow v)$
- $T \vdash u\sigma \quad (dom(\sigma) = vars(u))$
- $T' = T \cup \{v\sigma\}$
- $R'_i \in P', R'_i = (P \setminus \{R_i\}) \cup R_i\sigma$

Example

Example

Let $T = \{a, b, k_l\}$ and $P = \{R\}$ where
 $R = (\langle x, y \rangle \rightarrow \langle \{y\}_k, x \rangle), (z \rightarrow \langle x, \langle y, z \rangle \rangle).$

- $(T, P) \rightarrow (T \cup \{\langle \{b\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle b, z \rangle \rangle)\})$
- $(T, P) \rightarrow (T \cup \{\langle \{\{a\}_{k_l}\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle \{a\}_{k_l}, z \rangle \rangle)\})$
- $(T, P) \not\rightarrow (T \cup \{\langle \{\{a\}_k\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle \{a\}_k, z \rangle \rangle)\})$

Example

Example

Let $T = \{a, b, k_l\}$ and $P = \{R\}$ where

$R = (\langle x, y \rangle \rightarrow \langle \{y\}_k, x \rangle), (z \rightarrow \langle x, \langle y, z \rangle \rangle)$.

- $(T, P) \rightarrow (T \cup \{\langle \{b\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle b, z \rangle \rangle)\})$
- $(T, P) \rightarrow (T \cup \{\langle \{\{a\}_{k_l}\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle \{a\}_{k_l}, z \rangle \rangle)\})$
- $(T, P) \not\rightarrow (T \cup \{\langle \{\{a\}_k\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle \{a\}_k, z \rangle \rangle)\})$

Each branch has a **finite depth**, but **possibly a infinite branching**.

Preservation of the secrecy

Definition (Secrecy)

Let T_1 be a ground set of terms (Initial knowledge of the intruder). A protocol P **does not preserve the secrecy** of a ground term s for T_1 if there does not exist a state (T', P') , such that

- $T' \vdash s$
- $(T_1, P) \rightarrow^* (T', P')$

where \rightarrow^* is the reflexive and transitive closure of \rightarrow .

If there does not exist a such state (T', P') we say that P **preserves the secrecy** of s for the initial intruder knowledge T_1 .

Interleaving

Definition (Partial Order $<_P$)

A protocol P define a **partial order** $<_P$ on actions of P , s.t

$$(u_i \rightarrow v_i) <_P (u_j \rightarrow v_j)$$

if $R \in P$, $R = (u_1 \rightarrow v_1) \dots (u_i \rightarrow v_i) \dots (u_j \rightarrow v_j) \dots (u_n \rightarrow v_n)$ ($1 \leq i \leq j \leq n$).

Interleaving

Definition (Partial Order $<_P$)

A protocol P define a **partial order** $<_P$ on actions of P , s.t

$$(u_i \rightarrow v_i) <_P (u_j \rightarrow v_j)$$

if $R \in P$, $R = (u_1 \rightarrow v_1) \dots (u_i \rightarrow v_i) \dots (u_j \rightarrow v_j) \dots (u_n \rightarrow v_n)$ ($1 \leq i \leq j \leq n$).

Definition (Execution Order $<_E$)

An execution order $<_E$ of P is a total order on the subset A of actions of P , compatible with $<_P$ and stable by predecessor, i.e.

$$\text{if } b \in A \text{ et } a <_P b \text{ then } a \in A \text{ and } a <_E b$$

It corresponds to an interleaving of roles.

Secrecy

Definition (Secrecy over \langle_E)

Let an execution order \langle_E of P . We assume that

$$(u_1 \rightarrow v_1) \langle_E \dots \langle_E (u_n \rightarrow v_n)$$

\langle_E does not preserve the secrecy of s , given T_1 if there exists $\sigma_1, \dots, \sigma_n$ such that

$$(P, T_1) \rightarrow (P_1, T_1 \cup \{v_1\sigma_1\}) \rightarrow \dots \rightarrow (P_n, T_1 \cup \{v_1\sigma_1, \dots, v_n\sigma_n\})$$

and $T_1 \cup \{v_1\sigma_1, \dots, v_n\sigma_n\} \vdash s$.

Outline

- 1 Active Intruder: Security Problem
- 2 Unification Notions**
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions
- 4 NP-Hardness for Bounded Number of Sessions
- 5 Tools
- 6 Complexity
- 7 Conclusion

Arity

Definition

- \mathcal{F} is a finite set
 - *Arity* is a mapping from \mathcal{F} into \mathbb{N}
 - $(\mathcal{F}, \textit{Arity})$ is a **ranked alphabet** or **signature** denoted Σ
-
- The **arity** of a symbol $f \in \mathcal{F}$ is $\textit{Arity}(f)$
 - The set of symbols of arity p is denoted by \mathcal{F}_p .
 - Elements of arity 0, 1, \dots p are respectively called constants, unary, \dots p -ary symbols.

Example

Example

Let $\mathcal{F} = \{\mathbf{enc}, \mathbf{pair}, \mathbf{k}_1, \mathbf{k}_2, \mathbf{0}, \mathbf{1}\}$

$Ariety(\mathbf{enc}) = Ariety(\mathbf{pair}) = 2$

$Ariety(\mathbf{k}_1) = Ariety(\mathbf{k}_2) = Ariety(\mathbf{0}) = Ariety(\mathbf{1}) = 0$

We also denote $\mathcal{F} = \{\mathbf{enc}/2, \mathbf{pair}/2, \mathbf{k}_1/0, \mathbf{k}_2/0, \mathbf{0}/0, \mathbf{1}/0\}$

Terms

Let \mathcal{X} be a set of symbols called **variables**.

Definition

The set $\mathcal{T}(\mathcal{F}, \mathcal{X})$ of **terms** over the ranked alphabet \mathcal{F} and the set of variables \mathcal{X} is the smallest set defined by:

- $\mathcal{F}_0 \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
- $\mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
- if $p \geq 1$, $f \in \mathcal{F}_p$ and $t_1, \dots, t_p \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, then $f(t_1, \dots, t_p) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

- If $\mathcal{X} = \emptyset$ then $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is also written $\mathcal{T}(\mathcal{F})$. Terms in $\mathcal{T}(\mathcal{F})$ are called **ground terms**.
- A term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is **linear** if each variable occurs at most once in t .

Example

Example

Let $\mathcal{F} = \{\mathbf{enc}/2, \mathbf{pair}/2, \mathbf{k}_1/0, \mathbf{k}_2/0, \mathbf{0}/0, \mathbf{1}/0\}$ and $\mathcal{X} = \{x, y, z\}$
 $\mathbf{pair}(x, \mathbf{1})$, $\mathbf{enc}(\mathbf{pair}(y, z), \mathbf{k}_1)$ and $\mathbf{enc}(\mathbf{0}, \mathbf{k}_1)$ are terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$
 $\mathbf{pair}(\mathbf{0}, \mathbf{1})$, $\mathbf{enc}(\mathbf{0}, \mathbf{k}_1)$ are terms in $\mathcal{T}(\mathcal{F})$, i.e., ground terms

We also denote $\{-\}_-$ for $\mathbf{enc}(-, -)$ and $\langle -, - \rangle$ for $\mathbf{pair}(-, -)$.

Substitution

Definition

- A **substitution** (respectively a **ground substitution**) σ is a mapping from \mathcal{X} into $\mathcal{T}(\mathcal{F}, \mathcal{X})$ (respectively into $\mathcal{T}(\mathcal{F})$) where there are only finitely many variables not mapped to themselves.
- Substitutions can be extended to $\mathcal{T}(\mathcal{F}, \mathcal{X})$ in such a way that $\forall f \in \mathcal{F}_n, \forall t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$:

$$\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n)).$$

The **domain** of a substitution σ is the subset of variables $x \in \mathcal{X}$ such that $\sigma(x) \neq x$.

Example:

Let $\sigma = \{x \leftarrow N_A, y \leftarrow \{\langle N_A, N_B \rangle\}_{k_B}\}$ and $t = \langle x, \langle y, \langle x, x \rangle \rangle \rangle$.

Then,

$$\sigma(t) = \langle N_A, \{\langle N_A, N_B \rangle\}_{k_B}, \langle N_A, N_A \rangle \rangle$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

$$s = a \quad t = p(X)$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

$$s = a \quad t = p(X) \quad \text{No unifier}$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

$$s = a \quad t = p(X) \quad \text{No unifier}$$

$$s = p(a, X) \quad t = p(Y, b)$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

$$s = a \quad t = p(X) \quad \text{No unifier}$$

$$s = p(a, X) \quad t = p(Y, b) \quad \sigma = \{X \leftarrow b; Y \leftarrow a\}$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

$$s = a \quad t = p(X) \quad \text{No unifier}$$

$$s = p(a, X) \quad t = p(Y, b) \quad \sigma = \{X \leftarrow b; Y \leftarrow a\}$$

$$s = p(f(X), g(Z)) \quad t = p(f(a), Y)$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

$$s = a \quad t = p(X) \quad \text{No unifier}$$

$$s = p(a, X) \quad t = p(Y, b) \quad \sigma = \{X \leftarrow b; Y \leftarrow a\}$$

$$s = p(f(X), g(Z)) \quad t = p(f(a), Y)$$

$$\sigma = \{X \leftarrow a; Y \leftarrow g(Z)\}$$

Unification

Definition

Two t and u are unifiable if there exists a substitution σ such that $\sigma s = \sigma t$

Examples:

$$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$$

$$s = a \quad t = p(X) \quad \text{No unifier}$$

$$s = p(a, X) \quad t = p(Y, b) \quad \sigma = \{X \leftarrow b; Y \leftarrow a\}$$

$$s = p(f(X), g(Z)) \quad t = p(f(a), Y)$$

$$\sigma = \{X \leftarrow a; Y \leftarrow g(Z)\} \text{ or } \sigma = \{X \leftarrow a; Y \leftarrow g(b); Z \leftarrow b\}$$

Most General Unifier

Definition

The most general unification between two terms s and t , denoted by $mgu(s, t)$ if: $\forall \sigma$ such that $s\sigma = t\sigma, \exists \theta$ such that $\sigma = mgu(s, t)\theta$

$$s = p(f(X), g(Z)) \quad t = p(f(a), Y)$$

$$\sigma_1 = \{X \leftarrow a; Y \leftarrow g(Z)\} \quad \sigma_2 = \{X \leftarrow a; Y \leftarrow g(b); Z \leftarrow b\}$$

Goal

Design an algorithm that for a given unification problem $s =? t$

- returns an mgu of s and t if they are unifiable.
- reports failure otherwise.

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

- 1 Move the markers simultaneously, one symbol at a time, until both move off the end of the term (success), or until they point to two different symbols;
- 2 If the two symbols are both non-variables, then fail; otherwise, one is a variable (call it x) and the other one is the first symbol of a subterm (call it t):
 - If x occurs in t , then fail;
 - Otherwise, replace x everywhere by t (including in the solution), write down " $x \leftarrow t$ " as a part of the solution, and return to 1.

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$f(x, g(a), g(z))$

$f(g(y), g(y), g(g(x)))$

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$f(x, g(a), g(z))$

$f(g(y), g(y), g(g(x)))$

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$f(x, g(a), g(z))$

$f(g(y), g(y), g(g(x)))$

$\sigma = \{x \leftarrow g(y)\}$

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$f(g(y), g(a), g(z))$

$f(g(y), g(y), g(g(g(y))))$

$\sigma = \{x \leftarrow g(y)\}$

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$f(g(y), g(a), g(z))$

$f(g(y), g(y), g(g(g(y))))$

$\sigma = \{x \leftarrow g(y)\}$

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$$f(g(a), g(a), g(z))$$

$$f(g(a), g(a), g(g(g(a))))$$

$$\sigma = \{x \leftarrow g(a), y \leftarrow a\}$$

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$f(g(a), g(a), g(z))$

$f(g(a), g(a), g(g(g(a))))$

$\sigma = \{x \leftarrow g(a), y \leftarrow a\}$

Example: $f(x, g(a), g(z)) \stackrel{?}{=} f(g(y), g(y), g(g(x)))$

$f(g(a), g(a), g(g(g(a))))$

$f(g(a), g(a), g(g(g(a))))$

$\sigma = \{x \leftarrow g(a), y \leftarrow a, z \leftarrow g(g(a))\}$

Questions

- 1 Correctness:
 - Does the algorithm always terminate?
 - Does it always produce an mgu for two unifiable terms, and fail for non-unifiable terms?
 - Do these answers depend on the order of operations?
- 2 Complexity:
 - How much space does this take, and how much time?
- 3 Extension with equational theory, e.g., $ab = ba$.

Syntactic Unification is Unitary

Theorem (Robinson)

Without equational theory there exists a unique mgu for syntactic unification (modulo renaming). Unification is called unitary.

Herbrand, Martelli, Montanari, Plotkin, Robinson, Huet, Knuth, Bendix, Siekman, Baader.

Outline

- 1 Active Intruder: Security Problem
- 2 Unification Notions
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions**
- 4 NP-Hardness for Bounded Number of Sessions
- 5 Tools
- 6 Complexity
- 7 Conclusion

Constraints System

Symbolic representation of execution tree by constraints system.

Definition (Constraints System)

A **constraint** is an expression $T \Vdash u$ where T is a set of terms and u a term.

A **constraints system** C is a finite set of constraints $\cup_{1 \leq i \leq n} T_i \Vdash u_i$ such that

- $T_i \subseteq T_{i+1}$ ($1 \leq i \leq n$)
- if $T_i \Vdash u_i \in C$ and $x \in \text{vars}(T_i)$ then $T_j = \min\{T' \mid T' \Vdash v \in C, x \in \text{vars}(v)\}$ exists and $j < i$

A substitution σ is a **solution** of C if $T\sigma \vdash u\sigma$ for all $T \Vdash u \in C$.

We denote by \perp a constraints system unsatisfiable.

From Protocols to Constraints system

Let P a protocol, $<_E$ an execution order of P and s a secret term.

$$(u_1 \rightarrow v_1) <_E (u_2 \rightarrow v_2) <_E \dots <_E (u_n \rightarrow v_n)$$

We associate C :

$$\begin{array}{rcl} T_1 & \Vdash & u_1 \\ T_2 = T_1 \cup \{v_1\} & \Vdash & u_2 \\ & \vdots & \\ T_n = T_{n-1} \cup \{v_{n-1}\} & \Vdash & u_n \\ T_{n+1} = T_n \cup \{v_n\} & \Vdash & s \end{array}$$

We show that C has a solution iff $<_E$ does not preserve the secret of the term s .

Exercises

Exercise 1

$A \rightarrow B : \langle A, N_A \rangle$

$B \rightarrow A : \{\langle N_A, N_B \rangle\}_{K_{ab}}$

$A \rightarrow B : N_B$

$B \rightarrow A : \{\langle K, N_B \rangle\}_{K_{ab}}$

$A \rightarrow B : \{s\}_K$

Intruder knows only identities of A and B .

- Give role specification of this protocol of an instance of execution between A and B .
- Give a constraint system associated to this protocol between A and B .

Solution

$$\begin{aligned}
 A \rightarrow B &: \langle A, N_A \rangle \\
 B \rightarrow A &: \{\langle N_A, N_B \rangle\}_{K_{ab}} \\
 A \rightarrow B &: N_B \\
 B \rightarrow A &: \{\langle K, N_B \rangle\}_{K_{ab}} \\
 A \rightarrow B &: \{s\}_K
 \end{aligned}$$

$T_1 =$

$\{A, B, \langle A, N_A \rangle, \{\langle N_A, N_B \rangle\}_{K_{ab}}, N_B, \{\langle K, N_B \rangle\}_{K_{ab}}, \{s\}_K, \text{init}, \text{stop}\}$

Roles

$$\begin{aligned}
 R_A = & (\text{init} \rightarrow \langle A, N_A \rangle), \\
 & (\{\langle N_A, X_{N_B} \rangle\}_{K_{(A, X_B)}} \rightarrow X_{N_B}), \\
 & (\{\langle X_K, X_{N_B} \rangle\}_{K_{(A, X_B)}} \rightarrow \{s\}_{X_K})
 \end{aligned}$$

$$\begin{aligned}
 R_B = & (\langle X_A, X_{N_A} \rangle \rightarrow \{\langle X_{N_A}, N_B \rangle\}_{K_{(X_A, B)}}) \\
 & (N_B \rightarrow \{\langle K, N_B \rangle\}_{K_{(X_A, B)}}), \\
 & (\{X_s\}_K \rightarrow \text{stop})
 \end{aligned}$$

Solution

$$\begin{aligned}
 A \rightarrow B &: \langle A, N_A \rangle \\
 B \rightarrow A &: \{\langle N_A, N_B \rangle\}_{K_{ab}} \\
 A \rightarrow B &: N_B \\
 B \rightarrow A &: \{\langle K, N_B \rangle\}_{K_{ab}} \\
 A \rightarrow B &: \{s\}_K
 \end{aligned}$$

$T_1 =$

$\{A, B, \langle A, N_A \rangle, \{\langle N_A, N_B \rangle\}_{K_{ab}}, N_B, \{\langle K, N_B \rangle\}_{K_{ab}}, \{s\}_K, \text{init}, \text{stop}\}$

Constraint System

T_1	\Vdash	init
$T_2 = T_1 \cup \{\langle A, N_A \rangle\}$	\Vdash	$\langle X_A, X_{N_A} \rangle$
$T_3 = T_2 \cup \{\{\langle X_{N_A}, N_B \rangle\}_{K_{(X_A, B)}}\}$	\Vdash	$\{\langle N_A, X_{N_B} \rangle\}_{K_{(A, X_B)}}$
$T_4 = T_3 \cup \{X_{N_B}\}$	\Vdash	N_B
$T_5 = T_4 \cup \{\{\langle K, N_B \rangle\}_{K_{(X_A, B)}}\}$	\Vdash	$\{\langle X_K, X_{N_B} \rangle\}_{K_{(A, X_B)}}$
$T_6 = T_5 \cup \{\{s\}_{X_K}\}$	\Vdash	$\{X_s\}_K$
$T_7 = T_6 \cup \{\text{stop}\}$	\Vdash	s

Resolution of Constraints systems

Definition (Rules of simplification: $C \rightsquigarrow_{\sigma} C'$)

R_1	$C \cup \{T \Vdash u\}$	\rightsquigarrow	C	if $T \cup \{x \mid T' \Vdash x \in C, T' \subset T\} \vdash u$
R_2	$C \cup \{T \Vdash u\}$	$\rightsquigarrow_{\sigma}$	$C\sigma \cup \{T\sigma \Vdash u\sigma\}$	$\sigma = mgu(t, u), t \in st(T),$ t, u no variables
R_3	$C \cup \{T \Vdash u\}$	$\rightsquigarrow_{\sigma}$	$C\sigma \cup \{T\sigma \Vdash u\sigma\}$	$\sigma = mgu(t_1, t_2), t_1, t_2 \in st(T),$ t_1, t_2 no variables
R_4	$C \cup \{T \Vdash \{u\}_v\}$	\rightsquigarrow	$C \cup \{T \Vdash u, T \Vdash v\}$	
R_5	$C \cup \{T \Vdash \langle u, v \rangle\}$	\rightsquigarrow	$C \cup \{T \Vdash u, T \Vdash v\}$	
R_6	$C \cup \{T \Vdash u\}$	\rightsquigarrow	\perp	if $T = \emptyset$ or $var(T) = var(u) = \emptyset$ and $T \not\vdash u$

Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Lemma (Correctness)

If $C \rightsquigarrow_{\sigma} C'$ then if θ is a solution of C' , $\sigma\theta$ is also a solution of C .

Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Lemma (Correctness)

If $C \rightsquigarrow_{\sigma} C'$ then if θ is a solution of C' , $\sigma\theta$ is also a solution of C .

Lemma (Termination)

Simplification rules always terminate: There does not exist infinite chain $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} C_2 \rightsquigarrow_{\sigma_3} \dots$

Properties

Definition (Solved Form)

A constraints system C is in **solved form** if $C = \perp$ or if each constraint is of the following form $T \Vdash x$ where x is a variable $T \neq \emptyset$.

Lemma

All constraints systems in solved form different of \perp has at least one solution.

Properties

Definition (Solved Form)

A constraints system C is in **solved form** if $C = \perp$ or if each constraint is of the following form $T \Vdash x$ where x is a variable $T \neq \emptyset$.

Lemma

All constraints systems in solved form different of \perp has at least one solution.

Lemma (Completeness)

If C is a constraint system not in solved form and if σ is a solution of C then there exists θ, τ such that $C \rightsquigarrow_{\theta} C'$, $\sigma = \theta\tau$ and τ is a solution of C' .

Decidability

Theorem

Preservation of the secrecy for protocol with bounded number of sessions is decidable.

- Guess an interleaving and build constraints system associated.
- Using previous lemma C has a solution iff there exists C' in solved form such that $C' \neq \perp$ and $C \rightsquigarrow_{\tau} C'$
- Using termination lemma to conclude.

We also can show that the problem is in co-NP.

Exercises

Exercise 1

$$\begin{aligned}A &\rightarrow B : \langle A, N_A \rangle \\B &\rightarrow A : \{\langle N_A, N_B \rangle\}_{K_{ab}} \\A &\rightarrow B : N_B \\B &\rightarrow A : \{\langle K, N_B \rangle\}_{K_{ab}} \\A &\rightarrow B : \{s\}_K\end{aligned}$$

Intruder knows only identities of A and B .

- Use simplification rules to transform the system in solved form.
- There exists an easy attack, can you find it ?

Solution

$$T_1 = \{A, B, \langle A, N_A \rangle, \{\langle N_A, N_B \rangle\}_{K_{ab}}, N_B, \{\langle K, N_B \rangle\}_{K_{ab}}, \{s\}_K, \text{init}, \text{stop}\}$$

C_1	T_1	\Vdash	init
C_2	$T_2 = T_1 \cup \{\langle A, N_A \rangle\}$	\Vdash	$\langle X_A, X_{N_A} \rangle$
C_3	$T_3 = T_2 \cup \{\{\langle X_{N_A}, N_B \rangle\}_{K_{(X_A, B)}}\}$	\Vdash	$\{\langle N_A, X_{N_B} \rangle\}_{K_{(A, X_B)}}$
C_4	$T_4 = T_3 \cup \{X_{N_B}\}$	\Vdash	N_B
C_5	$T_5 = T_4 \cup \{\{\langle K, N_B \rangle\}_{K_{(X_A, B)}}\}$	\Vdash	$\{\langle X_K, X_{N_B} \rangle\}_{K_{(A, X_B)}}$
C_6	$T_6 = T_5 \cup \{\{s\}_{X_K}\}$	\Vdash	$\{X_s\}_K$
C_7	$T_7 = T_6 \cup \{\text{stop}\}$	\Vdash	s

Road book

Interleaving: $(u_1^A, v_1^A)(u_1^B, v_1^B)(u_2^A, v_2^A)(u_2^B, v_2^B)(u_3^A, v_3^A)(u_3^B, v_3^B)$

$$R_2 \quad C \cup \{T \Vdash u\} \rightsquigarrow_{\sigma} C\sigma \cup \{T\sigma \Vdash u\sigma\} \quad \sigma = \text{mgu}(t, u), t \in \text{st}(T), \\ t, u \text{ no variables}$$

- Apply nothing on C_1 , already in resolved form.
- Apply R_2 on C_2 give $\sigma_1 = \{X_{N_A} \leftarrow N_A, X_A \leftarrow A\}$ and R_1

Solution

$$T_1 = \{A, B, \langle A, N_A \rangle, \{\langle N_A, N_B \rangle\}_{K_{ab}}, N_B, \{\langle K, N_B \rangle\}_{K_{ab}}, \{s\}_K, \text{init}, \text{stop}\}$$

$$\begin{array}{ll}
 C_3\sigma_1 & T_3 = T_2 \cup \{\{\langle N_A, N_B \rangle\}_{K_{(A,B)}}\} \quad \Vdash \{\langle N_A, X_{N_B} \rangle\}_{K_{(A,X_B)}} \\
 C_4\sigma_1 & T_4 = T_3 \cup \{X_{N_B}\} \quad \Vdash N_B \\
 C_5\sigma_1 & T_5 = T_4 \cup \{\{\langle K, N_B \rangle\}_{K_{(A,B)}}\} \quad \Vdash \{\langle X_K, X_{N_B} \rangle\}_{K_{(A,X_B)}} \\
 C_6\sigma_1 & T_6 = T_5 \cup \{\{s\}_{X_K}\} \quad \Vdash \{X_s\}_K \\
 C_7\sigma_1 & T_7 = T_6 \cup \{\text{stop}\} \quad \Vdash s
 \end{array}$$

Road book $\sigma_1 = \{X_{N_A} \leftarrow N_A, X_A \leftarrow A\}$

- Apply R_2 on C_3 gives $\sigma_2 = \{X_{N_B} \leftarrow N_B, X_B \leftarrow B\}$ (or N_A) and R_1

Solution

$$T_1 = \{A, B, \langle A, N_A \rangle, \{\langle N_A, N_B \rangle\}_{K_{ab}}, N_B, \{\langle K, N_B \rangle\}_{K_{ab}}, \{s\}_K, \text{init}, \text{stop}\}$$

$$\begin{array}{lll} C_5\sigma_1\sigma_2 & T_5 = T_4 \cup \{\{\langle K, N_B \rangle\}_{K_{(A,B)}}\} & \Vdash \{\langle X_K, N_B \rangle\}_{K_{(A,B)}} \\ C_6\sigma_1\sigma_2 & T_6 = T_5 \cup \{\{s\}_{X_K}\} & \Vdash \{X_S\}_K \\ C_7\sigma_1\sigma_2 & T_7 = T_6 \cup \{\text{stop}\} & \Vdash s \end{array}$$

Road book $\sigma_1 = \{X_{N_A} \leftarrow N_A, X_A \leftarrow A\}$ $\sigma_2 = \{X_{N_B} \leftarrow N_B, X_B \leftarrow B\}$

- Apply R_2 on $C_5\sigma_1\sigma_2$ give $\sigma_3 = \{X_K \leftarrow N_A\}$
- Apply R_2 , on $\sigma_1\sigma_2\sigma_3C_6$ give $\sigma_4 = \{X_S \leftarrow s\}$

Solution

- 1 $A \rightarrow B : \langle A, N_A \rangle$
- 2 $B \rightarrow A : \{\langle N_A, N_B \rangle\}_{K_{ab}}$
- 3 $A \rightarrow B : N_B$
- 4 $B \rightarrow A : \{\langle K, N_B \rangle\}_{K_{ab}}$
- 5 $A \rightarrow B : \{s\}_K$

The resolution of constraint system gives the following attack:
Send 2nd message $\{\langle N_A, N_B \rangle\}_{K_{ab}}$ instead of the 4th message
 $\{\langle K, N_B \rangle\}_{K_{ab}}$. Hence A will replay $\{s\}_{N_A}$ because intruder knows
 N_A

Exercises

Exercise 2

$$A \rightarrow B : \{\langle A, K \rangle\}_{K_{ab}}$$

$$B \rightarrow A : \{s\}_{K_{ab}}$$

Intruder knows only identities of A and B . Show that the secret data s is preserved by one single session between A and B .

Solution

$$\begin{aligned} A \rightarrow B &: \{\langle A, K \rangle\}_{K_{ab}} \\ B \rightarrow A &: \{s\}_{K_{ab}} \end{aligned}$$

$$T_1 = \{A, B, \{\langle A, K \rangle\}_{K_{ab}}, \{s\}_{K_{ab}}\}$$

Constraint System

$$\begin{array}{lll} C_1 & T_1 & \Vdash \{\langle A, X_K \rangle\}_{K_{ab}} \\ C_2 & T_2 = T_1 \cup \{\langle A, X_K \rangle\}_{K_{ab}} & \Vdash \{s\}_{X_{K_{ab}}} \\ C_3 & T_3 = T_2 \cup \{s\}_{X_{K_{ab}}} & \Vdash s \end{array}$$

Solution

$$\begin{array}{ll}
 C_1 & T_1 \qquad \qquad \qquad \Vdash \{ \langle A, X_K \rangle \}_{X_{K_{ab}}} \\
 C_2 & T_2 = T_1 \cup \{ \langle A, X_K \rangle \}_{X_{K_{ab}}} \quad \Vdash \{ s \}_{X_{K_{ab}}} \\
 C_3 & T_3 = T_2 \cup \{ s \}_{X_{K_{ab}}} \quad \Vdash s
 \end{array}$$

$$T_1 = \{ A, B, \{ \langle A, K \rangle \}_{K_{ab}}, \{ s \}_{K_{ab}} \}$$

Road book

- Apply nothing or R_4 or R_5 and R_2 on C_1 give $\sigma_0 = \{ X_K \leftarrow K, X_{K_{ab}} \leftarrow K_{ab} \}$
- Apply R_5 or nothing and R_2 , on $\sigma_0 C_2$ give $\sigma_1 = \{ X_{N_B} \leftarrow N_B \}$ (or N_A)

Each time you meet a solved form of the form \perp with R_6 .

Outline

- 1 Active Intruder: Security Problem
- 2 Unification Notions
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions
- 4 NP-Hardness for Bounded Number of Sessions**
- 5 Tools
- 6 Complexity
- 7 Conclusion

NP-hardness

Theorem

Decide if a protocol P does not preserve the secrecy of a ground term s from an initial knowledge T_1 is NP-difficult.

Recall 3-SAT Problem

Definition

Input: set of propositional variables $\{x_1, \dots, x_n\}$ and a conjunction of clauses with 3 literals.

$$f(\vec{x}) = \bigwedge_{1 \leq i \leq l} (x_{i,1}^{\epsilon_{i,1}} \vee x_{i,2}^{\epsilon_{i,2}} \vee x_{i,3}^{\epsilon_{i,3}})$$

where $\epsilon_{i,j} \in \{+, -\}$ and $x^+ = x, x^- = \neg x$.

Question : Does exist a valuation V of $\{x_1, \dots, x_n\}$, such that $V(f(\vec{x})) = \top$.

Theorem

3-SAT problem is NP-complete.

NP-difficulty

We build a protocol such that an intruder can deduce s iff $f(\vec{x})$ is satisfaisable.

$$g(x_{i,j}^{\epsilon_{i,j}}) = \begin{cases} x_{i,j} & \text{if } \epsilon_{i,j} = + \\ \{x_{i,j}\}_K & \text{if } \epsilon_{i,j} = - \end{cases}$$

$$\forall 1 \leq i \leq l : f_i(\vec{x}) = \langle g(x_{i,1}^{\epsilon_{i,1}}), g(x_{i,2}^{\epsilon_{i,2}}), g(x_{i,3}^{\epsilon_{i,3}}) \rangle$$

We suppose Initial intruder knowledge is $\{\perp, \top\}$.

$$A : \langle x_1, \langle \dots, x_n \rangle \rangle \rightarrow \{ \langle f_1(\vec{x}), \langle f_2(\vec{x}), \langle \dots, \langle f_n(\vec{x}), end \rangle \dots \rangle \rangle \}_p$$

$$\forall 1 \leq i \leq l :$$

$$B_i : \{ \langle \langle \top, \langle x, y \rangle \rangle, z \rangle \}_p \rightarrow \{z\}_p$$

$$\overline{B}_i : \{ \langle \langle \{ \perp \}_K, \langle x, y \rangle \rangle, z \rangle \}_p \rightarrow \{z\}_p$$

$$C_i : \{ \langle \langle x, \langle \top, y \rangle \rangle, z \rangle \}_p \rightarrow \{z\}_p$$

$$\overline{C}_i : \{ \langle \langle x, \langle \{ \perp \}_K, y \rangle \rangle, z \rangle \}_p \rightarrow \{z\}_p$$

$$D_i : \{ \langle \langle x, \langle y, \top \rangle \rangle, z \rangle \}_p \rightarrow \{z\}_p$$

$$\overline{D}_i : \{ \langle \langle x, \langle y, \{ \perp \}_K \rangle \rangle, z \rangle \}_p \rightarrow \{z\}_p$$

$$E : \{ end \}_p \rightarrow s$$

Outline

- 1 Active Intruder: Security Problem
- 2 Unification Notions
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions
- 4 NP-Hardness for Bounded Number of Sessions
- 5 Tools**
- 6 Complexity
- 7 Conclusion

- **Avispa** :

OFMC: On-the-fly Model-Checker employs several symbolic techniques to explore the state space in a demand-driven way.

CL-AtSe: Constraint-Logic-based Attack Searcher applies constraint solving with simplification heuristics and redundancy elimination techniques.

SATMC: SAT-based Model-Checker builds a propositional formula encoding all the possible traces (of bounded length) on the protocol and uses a SAT solver.

TA4SP: Tree Automata based on Automatic Approximations for the Analysis of Security Protocols approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations.

- **Proverif**: Analyses unbounded number of session using over-approximation with Horn Clauses.

- **Scyther**: Verifies bounded and unbounded number of session with backwards search based on partially ordered patterns.

AVISPA

- Common language for specifying protocols and security properties.
- Supports symmetric and asymmetric keys, cryptographic hash functions, key-tables, user-definable algebraic functions, etc.

Input	Output (<1 second)
<pre> PROTOCOL Needham-Schroeder; Identifiers A, B: user; Na, Nb: nonce; Ka, Kb: public_key; Messages 1. A -> B: {A,Na}Kb 2. B -> A: {Na,Nb}Ka 3. A -> B: {Nb}Kb Intruder_knowledge Spy, b, ka, kb, kspy; Goal correspondence_between A B; </pre>	<pre> A -> Spy: {A,Na}Kspy Spy -> B: {A,Na}Kb B -> A: {Na,Nb}Ka A -> Spy: {Nb}Kspy Spy -> B {Nb}Kb </pre>

Scyther

- Alternative: backwards search based on patterns
 - Security properties represented by claim events in the protocol.
 - Supports symmetric and asymmetric keys, cryptographic hash functions, key-tables, multiple protocols in parallel, composed keys, etc (but no user-definable algebraic functions)
 - Can perform unbounded verification of protocols
 - Provides *complete characterization* of protocol roles:
Answer to: “after execution of a protocol role, what events must also have occurred?”
- Also state-of-art. Freely available for download for Windows, Linux and Mac OS X.
- Will be used in the exercise sessions.

Input

```

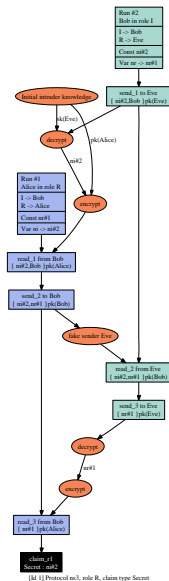
protocol ns3(I,R) {
  role I {
    const ni: Nonce;
    var nr: Nonce;
    send_1(I,R, {ni,I}pk(R) );
    read_2(R,I, {ni,nr}pk(I) );
    send_3(I,R, {nr}pk(R) );

    claim_i1(I,Secret,ni);
    claim_i2(I,Nisynch);
  }
  role R {
    var ni: Nonce;
    const nr: Nonce;
    read_1(I,R, {ni,I}pk(R) );
    send_2(R,I, {ni,nr}pk(I) );
    read_3(I,R, {nr}pk(R) );

    claim_r1(R,Secret,ni);
    claim_r2(R,Nisynch);
  }
}

```

Output (<0.02 seconds)

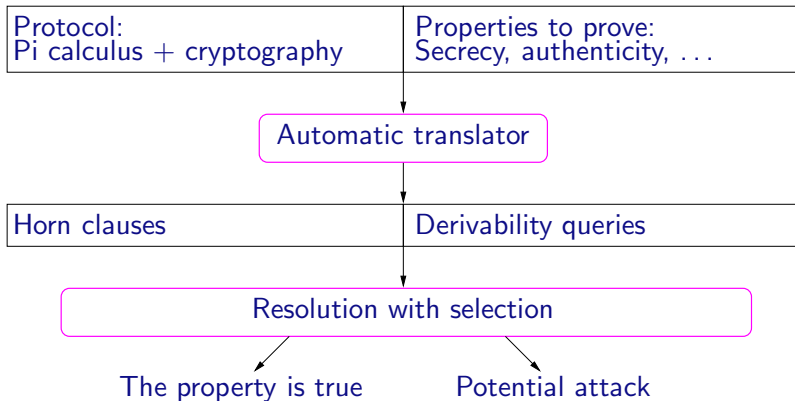


Our goal

Goal: a verifier for cryptographic protocols

- Fully automatic
- For an unbounded number of sessions and an unbounded message size
- Handles many cryptographic primitives
- Proves various properties: secrecy, correspondences, equivalences
- Efficient

Overview of the protocol verifier ProVerif



Proverif uses Horn Clauses

A Horn clause is a logical formula of the form

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

Formalism simple and homogeneous for

- modeling intruder capabilities
- modeling protocol rules
- checking an unbounded number of sessions

This formalism is used like intermediary representation (translation from high level language “Pi-calculus like”) in the Tool ProVerif [Blanchet2001]

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

Our solution

Two ideas (extending [Weidenbach, CADE'99]):

- a **simple abstract representation** of these protocols, by a set of Horn clauses;
- an **efficient solving algorithm to find which facts can be derived from these clauses.**

Using this, we can prove **secrecy** properties of protocols, or exhibit attacks showing why a message is not secret.

We handle in particular shared- and public-key cryptography, hash functions, Diffie-Hellman key agreements.

Protocol representation

- Messages \rightsquigarrow terms

$$M ::= x \mid f(M_1, \dots, M_n) \mid k[M_1, \dots, M_n]$$

$\text{pencrypt}(c_0, \text{pk}(sk_A))$.

- Properties \rightsquigarrow facts

$$F ::= \text{attacker}(M).$$

- Protocol, attacker \rightsquigarrow Horn clauses

$$F_1 \wedge \dots \wedge F_n \rightarrow F$$

$\text{attacker}(m) \wedge \text{attacker}(pk) \rightarrow \text{attacker}(\text{pencrypt}(m, pk)).$

Example - Cryptographic primitives

Public-key encryption:

- Encryption $\text{pencrypt}(m, pk)$.
 $\text{attacker}(m) \wedge \text{attacker}(pk) \rightarrow \text{attacker}(\text{pencrypt}(m, pk))$
- Public key generation $\text{pk}(sk)$.
(builds a public key from a secret key)
 $\text{attacker}(sk) \rightarrow \text{attacker}(\text{pk}(sk))$
- Decryption $\text{pdecrypt}(\text{pencrypt}(m, \text{pk}(sk)), sk) \rightarrow m$.
 $\text{attacker}(\text{pencrypt}(m, \text{pk}(sk))) \wedge \text{attacker}(sk) \rightarrow \text{attacker}(m)$

General treatment of primitives

- **Constructors** $f(M_1, \dots, M_n)$
 $\text{attacker}(x_1) \wedge \dots \wedge \text{attacker}(x_n) \rightarrow \text{attacker}(f(x_1, \dots, x_n))$
- **Destructors** $g(M_1, \dots, M_n) \rightarrow M$
 $\text{attacker}(M_1) \wedge \dots \wedge \text{attacker}(M_n) \rightarrow \text{attacker}(M)$

(There may be several reductions defining a function.)

Example (Exercise)

Give clauses for shared-key encryption and signatures

Names

Normally, fresh names are created each time the protocol is run. Here, we only distinguish two names when they are created after receiving different messages.

Each name k becomes a **function** of the messages previously received:

$$k[M_1, \dots, M_n].$$

(Skolemisation)

These functions can only be applied by the principal that creates the name, not by the attacker.

Approximations

- **Nonces** are modeled by functions of previous received messages
If intruder send same messages then same nonces will be used.
- One **step of the protocol can be executed several times** if previous steps are executed at least once.

Example :

1. Intruder sends to A the message M_1
2. A answers by M_2
3. Intruder sends to A the message M_3
4. A answers by M_4
5. Intruder sends to A the message M'_3 (**without executing the 2 first steps**)
6. A replies with M'_4

Proverif: Horn Clauses

```
(* Needham Shroeder Lowe *)
pred c/1 elimVar,decompData.
nounif c:x.
fun pk/1.
fun encrypt/2.

query c:secret [].
reduc

(* Initialization *)
c:c[];
c:pk(sA[]);
c:pk(sB[]);

c:x & c:encrypt(m,pk(x)) -> c:m;
c:x -> c:pk(x);
c:x & c:y -> c:encrypt(x,y);
```

Proverif: Horn Clauses

```
(* The protocol *)
```

```
(* A *)
```

```
c:pk(x) -> c:encrypt((Na[pk(x)], pk(sA[])), pk(x));
```

```
c:pk(x) & c:encrypt((Na[pk(x)], y), pk(sA[]))
-> c:encrypt((y,k[pk(x)]), pk(x));
```

```
(* B *)
```

```
c:encrypt((x,y), pk(sB[]))
```

```
-> c:encrypt((x, Nb[x,y], pk(sB[])), y);
```

```
c:encrypt((x, pk(sA[])))
```

```
& c:encrypt((Nb[x, pk(sA[])], z), pk(sB[]))
```

```
-> c:encrypt(secret[], pk(z)).
```

Tools

```

goal reachable: c:secret[]

rule 7 c:secret[]
  any c:x_182
  rule 1 c:encrypt(secret [],pk(x_182))
    rule 5 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(sB[]))
      2-tuple c:(Na[pk(x_168)],pk(sA[]))
      0-th c:Na[pk(x_168)]
      rule 7 c:(Na[pk(x_168)],pk(sA[]))
        any c:x_168
        rule 4 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(x_168))
          rule 6 c:pk(x_168)
            any c:x_168
            rule 9 c:pk(sA[])
            rule 8 c:pk(sB[])
            rule 5 c:encrypt((Nb[Na[pk(x_168)],pk(sA[])],x_182),pk(sB[]))
              2-tuple c:(Nb[Na[pk(x_168)],pk(sA[])],x_182)
              0-th c:Nb[Na[pk(x_168)],pk(sA[])]
              rule 7 c:(Nb[Na[pk(x_168)],pk(sA[])],k[pk(x_168)])
                any c:x_168
                rule 3 c:encrypt((Nb[Na[pk(x_168)],pk(sA[])],k[pk(x_168)]),pk(x_168))
                  rule 6 c:pk(x_168)
                    any c:x_168
                    rule 2 c:encrypt((Na[pk(x_168)],Nb[Na[pk(x_168)],pk(sA[])]),pk(sA[]))
                      rule 5 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(sB[]))
                        2-tuple c:(Na[pk(x_168)],pk(sA[]))
                        0-th c:Na[pk(x_168)]
                        rule 7 c:(Na[pk(x_168)],pk(sA[]))
                          any c:x_168
                          rule 4 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(x_168))
                            rule 6 c:pk(x_168)
                              any c:x_168
                              rule 9 c:pk(sA[])
                              rule 8 c:pk(sB[])
                            any c:x_182
                            rule 8 c:pk(sB[])
                          any c:x_182
                          rule 8 c:pk(sB[])
                    any c:x_182
                    rule 8 c:pk(sB[])
              any c:x_182
              rule 8 c:pk(sB[])
            any c:x_182
            rule 8 c:pk(sB[])
          any c:x_182
          rule 8 c:pk(sB[])
        any c:x_182
        rule 8 c:pk(sB[])
      any c:x_182
      rule 8 c:pk(sB[])
    any c:x_182
    rule 8 c:pk(sB[])
  any c:x_182
  rule 8 c:pk(sB[])

```

RESULT goal reachable: c:secret[]

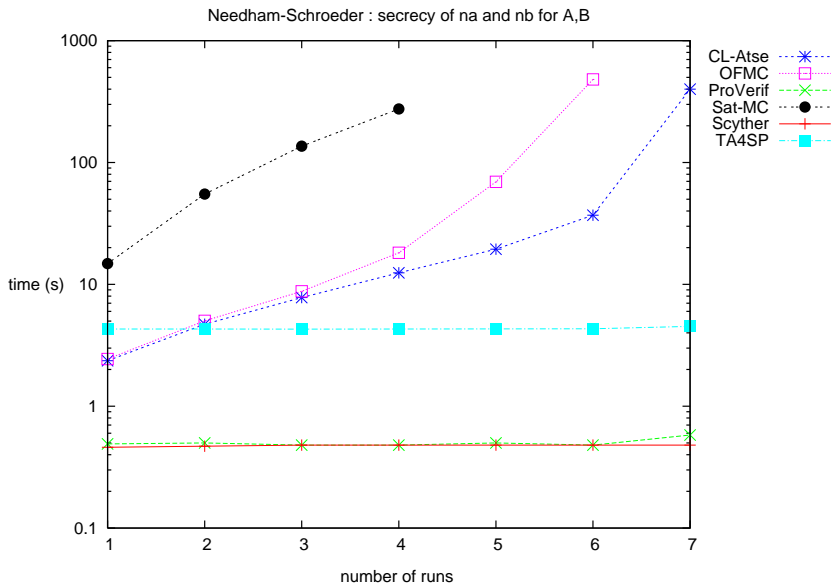
Outline

- 1 Active Intruder: Security Problem
- 2 Unification Notions
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions
- 4 NP-Hardness for Bounded Number of Sessions
- 5 Tools
- 6 Complexity**
- 7 Conclusion

Complexity

Complexity depends of intruder capabilities. In classical Dolev-Yao intruder model we (pair + encryption) we have the following results:

- **Passive Intruder**
Problem is **polynomial**
- **Bounded Number of sessions**
Problem is **NP-complete**
Tools can verify 3-4 sessions: useful to **finds flaws** ! OFMC, CI-Atse, SATMC, FDR, Athena...
- **Unbounded Number of sessions**
Problem is in general **undecidable**
Tools are **corrects, but uncomplete** (can find false attacks, can not terminate) Proverif, TA4SP, Scyther.



Bibliography

- **Time performance comparison of AVISPA Tools**
L. Vigano “Automated Security Protocol Analysis With the AVISPA Tool” ENTCS 2006.
- **Usability comparison between AVISPA and HERMES**
M. Hussain and D. Seret “A Comparative study of Security Protocols Validation Tools: HERMES vs. AVISPA”.In the 8th International Conference Advanced Communication Technology, ICACT’06.
- Cas Cremers and Pascal Lafourcade. **Comparing State Spaces in Automatic Security Protocol Verification**. In Michael Goldsmith and Bill Roscoe (eds.), Proceedings of the 7th International Workshop on Automated Verification of Critical Systems (AVoCS’07), Oxford, UK, September 2007, Electronic Notes in Theoretical Computer Science, pages 49-63. Elsevier Science Publishers.

Outline

- 1 Active Intruder: Security Problem
- 2 Unification Notions
 - Terms and Messages
 - Unification
- 3 Bounded Number of Sessions
- 4 NP-Hardness for Bounded Number of Sessions
- 5 Tools
- 6 Complexity
- 7 Conclusion**

Summary

Today

- Active Intruder
- Bounded Number of Sessions
- NP-Hardness
- Tools

Next Time

- Playing with Tools:
 - Scyther
 - Avispa: OFMC, CI-Atse, SATMC, TA4SP
 - Proverif

Thank you for your attention



Questions ?