Verifying Security Protocols in Tamarin

Ralf Sasse
Institute of Information Security
ETH Zurich

Tamarin Day 2, v.1

Jan 26, 2016

Outline

Outline

Motivation

Term Rewriting is

- a useful and flexible formalism in general.
  - ★ Programming languages
  - ★ Automated deduction
  - ★ Rewriting logic
- used for representing protocols formally in this course!

Signature

Definition (Signature)

An unsorted signature Σ is a set of function symbols, each having an arity $n \geq 0$. We call function symbols of arity 0 constants.

Signature

Definition (Signature)

An unsorted signature Σ is a set of function symbols, each having an arity $n \geq 0$. We call function symbols of arity 0 constants.

Example (Peano notation for natural numbers)

$\Sigma = \{0, s, +\}$, where 0 is a constant, s has arity 1 and represents the successor function, and $+$ has arity 2 and represents addition. Note that for binary operators we sometimes will use infix notation.

Term Algebra

Definition (Term Algebra)

Let Σ be a signature, $\mathcal{X}$ a set of variables, and $\Sigma \cap \mathcal{X} = \emptyset$. We call the set $\mathcal{T}_\Sigma(\mathcal{X})$ the term algebra over Σ. It is the least set such that:

- $\mathcal{X} \subseteq \mathcal{T}_\Sigma(\mathcal{X})$.
- If $t_1, \ldots, t_n \in \mathcal{T}_\Sigma(\mathcal{X})$ and $f \in \Sigma$ with arity n, then $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})$.

The set of ground terms $\mathcal{T}_\Sigma$ consists of terms built without variables, i.e., $\mathcal{T}_\Sigma := \mathcal{T}_\Sigma(\emptyset)$.

Term Algebra

Definition (Term Algebra)

Let Σ be a signature, $\mathcal{X}$ a set of variables, and $\Sigma \cap \mathcal{X} = \emptyset$. We call the set $\mathcal{T}_\Sigma(\mathcal{X})$ the term algebra over Σ. It is the least set such that:

- $\mathcal{X} \subseteq \mathcal{T}_\Sigma(\mathcal{X})$.
- If $t_1, \ldots, t_n \in \mathcal{T}_\Sigma(\mathcal{X})$ and $f \in \Sigma$ with arity n, then $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})$.

The set of ground terms $\mathcal{T}_\Sigma$ consists of terms built without variables, i.e., $\mathcal{T}_\Sigma := \mathcal{T}_\Sigma(\emptyset)$.

Exercise: constants are included in $\mathcal{T}_\Sigma$ and $\mathcal{T}_\Sigma(\mathcal{X})$.

Term Algebra

Definition (Term Algebra)

Let Σ be a signature, $\mathcal{X}$ a set of variables, and $\Sigma \cap \mathcal{X} = \emptyset$. We call the set $\mathcal{T}_\Sigma(\mathcal{X})$ the term algebra over Σ. It is the least set such that:

- $\mathcal{X} \subseteq \mathcal{T}_\Sigma(\mathcal{X})$.
- If $t_1, \ldots, t_n \in \mathcal{T}_\Sigma(\mathcal{X})$ and $f \in \Sigma$ with arity n, then $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})$.

The set of ground terms $\mathcal{T}_\Sigma$ consists of terms built without variables, i.e., $\mathcal{T}_\Sigma := \mathcal{T}_\Sigma(\emptyset)$.

Exercise: constants are included in $\mathcal{T}_\Sigma$ and $\mathcal{T}_\Sigma(\mathcal{X})$.

Example (Peano notation for natural numbers (ctd.))

$s(0) \in \mathcal{T}_\Sigma$
$s(s(0)) + s(X) \in \mathcal{T}_\Sigma(\mathcal{X})$
$+s(0)+ \notin \mathcal{T}_\Sigma(\mathcal{X})$

Equational Theory

Definition (Equation)

An equation is a pair of terms, written: $t = t'$, and a set of equations is called an equational theory (Σ, E). An equation can be oriented as $t \rightarrow t' \in \overrightarrow{E}$ or as $t \leftarrow t' \in \overleftarrow{E}$.

Equations are usually oriented from left to right for use in simplification.

Equational Theory

Definition (Equation)

An equation is a pair of terms, written: $t = t'$, and a set of equations is called an equational theory (Σ, E). An equation can be oriented as $t \rightarrow t' \in \overrightarrow{E}$ or as $t \leftarrow t' \in \overleftarrow{E}$.

Equations are usually oriented from left to right for use in simplification.

Example (Peano natural numbers (ctd.))

The equations E defining the Peano natural numbers are:
$X + 0 = X$
$X + s(Y) = s(X + Y)$
Using $\overrightarrow{E}$ on $s(s(0)) + s(0)$ yields the equational derivation:
$s(s(0)) + s(0) =$

Equational Theory

Definition (Equation)

An equation is a pair of terms, written: $t = t'$, and a set of equations is called an equational theory (Σ, E). An equation can be oriented as $t \rightarrow t' \in \vec{E}$ or as $t \leftarrow t' \in \overleftarrow{E}$.

Equations are usually oriented from left to right for use in simplification.

Example (Peano natural numbers (ctd.))

The equations E defining the Peano natural numbers are:
$X + 0 = X$
$X + s(Y) = s(X + Y)$
Using $\vec{E}$ on $s(s(0)) + s(0)$ yields the equational derivation:
$s(s(0)) + s(0) = s(s(s(0)) + 0) =$

Equational Theory

Definition (Equation)

An equation is a pair of terms, written: $t = t'$, and a set of equations is called an equational theory (Σ, E). An equation can be oriented as $t \rightarrow t' \in \overrightarrow{E}$ or as $t \leftarrow t' \in \overleftarrow{E}$.

Equations are usually oriented from left to right for use in simplification.

Example (Peano natural numbers (ctd.))

The equations E defining the Peano natural numbers are:
$X + 0 = X$
$X + s(Y) = s(X + Y)$
Using $\overrightarrow{E}$ on $s(s(0)) + s(0)$ yields the equational derivation:
$s(s(0)) + s(0) = s(s(s(0)) + 0) = s(s(s(0)))$.

Cryptographic Messages

We generally denote variables with upper case names $X, Y, \ldots$, and function symbols (including constants) with lower case names $a, b, \ldots$

Definition (Messages)

A message is a term in $\mathcal{T}_\Sigma(\mathcal{X})$, where
$\Sigma = \mathcal{A} \cup \mathcal{F} \cup \mathit{Func} \cup \{\mathit{pair}, \mathit{pk}, \mathit{aenc}, \mathit{senc}\}$. We call

$\mathcal{X}$	the set of variables A, B, X, Y, Z, ...,
$\mathcal{A}$	the set of agents a, b, c, ...,
$\mathcal{F}$	the set of fresh values na, nb, k (nonces, keys, ...),
Func	the set of user-defined functions (hash, exp, ...),
$\mathit{pair}(t_1, t_2)$	pairing, also denoted by $\langle t_1, t_2 \rangle$,
$\mathrm{pk}(t)$	public key,
$\mathit{aenc}(t_1, t_2)$	asymmetric encryption, also denoted by $\{t_1\}_{t_2}$,
$\mathit{senc}(t_1, t_2)$	symmetric encryption, also denoted by $\{\|t_1\|\}_{t_2}$.

Free Algebra

Definition (Free Algebra)

In the free algebra every term is interpreted by itself (syntactically).

Example (Equational theory for symmetric cryptography)

$\Sigma = \mathcal{A} \cup \mathcal{F} \cup \{senc, sdec\}$, with *senc* and *sdec* of arity 2.
(E: $sdec(senc(M, K), K) = M$)

- $t_1 =_{\text{free}} t_2$ iff $t_1 =_{\text{syntactic}} t_2$.
- $a \neq_{\text{free}} b$ for different constants a and b.
- For above example: $sdec(senc(X, Y), Y) \neq_{\text{free}} X$.

This is too coarse, as we obviously want to identify those two terms, which means we will need to reason modulo equations.

Algebraic Properties

Example (Equations E)

$$\begin{aligned} \{\{M\}_K\}_{(K)^{-1}} &= M & ((K)^{-1})^{-1} &= K \\ \{\!|\{\!|M|\!\}_K|\!\}_K &= M & \exp(\exp(B,X),Y) &= \exp(\exp(B,Y),X) \end{aligned}$$

Definition (Congruence, Equivalence, Quotient)

A set of equations E induces a congruence relation $=_E$ on terms and thus the equivalence class $[t]_E$ of a term modulo E. The quotient algebra $\mathcal{T}_\Sigma(\mathcal{X})/_{=_E}$ interprets each term by its equivalence class.

- Two terms are semantically equal iff that is a consequence of E.

Algebraic Properties

Example (Equations E)

$$\begin{aligned} \{\{M\}_K\}_{(K)^{-1}} &= M & ((K)^{-1})^{-1} &= K \\ \{\![\{\![M]\!\}_K]\!\}_K &= M & \exp(\exp(B,X),Y) &= \exp(\exp(B,Y),X) \end{aligned}$$

Definition (Congruence, Equivalence, Quotient)

A set of equations E induces a congruence relation $=_E$ on terms and thus the equivalence class $[t]_E$ of a term modulo E. The quotient algebra $\mathcal{T}_\Sigma(\mathcal{X})/_{=_E}$ interprets each term by its equivalence class.

- Two terms are semantically equal iff that is a consequence of E.
- For the above example equations:
  - ★ $a \neq_E b$ for any distinct constants a and b

Algebraic Properties

Example (Equations E)

$$
\begin{aligned}
\{\{M\}_K\}_{(K)^{-1}} &= M & ((K)^{-1})^{-1} &= K \\
\{\!|\{\!|M|\!\}_K|\!\}_K &= M & \exp(\exp(B,X),Y) &= \exp(\exp(B,Y),X)
\end{aligned}
$$

Definition (Congruence, Equivalence, Quotient)

A set of equations E induces a congruence relation $=_E$ on terms and thus the equivalence class $[t]_E$ of a term modulo E. The quotient algebra $\mathcal{T}_\Sigma(\mathcal{X})/_{=_E}$ interprets each term by its equivalence class.

- Two terms are semantically equal iff that is a consequence of E.
- For the above example equations:
  - $a \neq_E b$ for any distinct constants a and b
  - If $m_1 \neq_E m_2$ then also $h(m_1) \neq_E h(m_2)$

Algebraic Properties

Example (Equations E)

$$\{\{M\}_K\}_{(K)^{-1}} = M \qquad ((K)^{-1})^{-1} = K$$
$$\{\!|\{\!|M|\!\}_K|\!\}_K = M \qquad \exp(\exp(B,X),Y) = \exp(\exp(B,Y),X)$$

Definition (Congruence, Equivalence, Quotient)

A set of equations E induces a congruence relation $=_E$ on terms and thus the equivalence class $[t]_E$ of a term modulo E. The quotient algebra $\mathcal{T}_\Sigma(\mathcal{X})/_{=_E}$ interprets each term by its equivalence class.

- Two terms are semantically equal iff that is a consequence of E.
- For the above example equations:
  - ★ $a \neq_E b$ for any distinct constants a and b
  - ★ If $m_1 \neq_E m_2$ then also $h(m_1) \neq_E h(m_2)$
  - ★ $\{\{M\}_{(K)^{-1}}\}_K =_E M$

Algebraic Properties

Example (Equations E)

$$\begin{aligned} \{\{M\}_K\}_{(K)^{-1}} &= M & ((K)^{-1})^{-1} &= K \\ \{\!|\{\!|M|\!\}_K|\!\}_K &= M & \exp(\exp(B,X),Y) &= \exp(\exp(B,Y),X) \end{aligned}$$

Definition (Congruence, Equivalence, Quotient)

A set of equations E induces a congruence relation $=_E$ on terms and thus the equivalence class $[t]_E$ of a term modulo E. The quotient algebra $\mathcal{T}_\Sigma(\mathcal{X})/_{=_E}$ interprets each term by its equivalence class.

- Two terms are semantically equal iff that is a consequence of E.
- For the above example equations:
  - $a \neq_E b$ for any distinct constants a and b
  - If $m_1 \neq_E m_2$ then also $h(m_1) \neq_E h(m_2)$
  - $\{\{M\}_{(K)^{-1}}\}_K =_E M$
  - $\{\!|\{\!|M|\!\}_{\exp(\exp(g,Y),X)}|\!\}_{\exp(\exp(g,X),Y)} =_E M$

Substitution

Definition (Substitution)

A substitution σ is a function $\sigma : \mathcal{X} \to \mathcal{T}_\Sigma(\mathcal{X})$ where $\sigma(x) \neq x$ for finitely many $x \in \mathcal{X}$.
We write substitutions in postfix notation and homomorphically extend them to a mapping $\sigma : \mathcal{T}_\Sigma(\mathcal{X}) \to \mathcal{T}_\Sigma(\mathcal{X})$ on terms:

$$f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$$

Substitution

Definition (Substitution)

A substitution σ is a function $\sigma : \mathcal{X} \rightarrow \mathcal{T}_\Sigma(\mathcal{X})$ where $\sigma(x) \neq x$ for finitely many $x \in \mathcal{X}$.
We write substitutions in postfix notation and homomorphically extend them to a mapping $\sigma : \mathcal{T}_\Sigma(\mathcal{X}) \rightarrow \mathcal{T}_\Sigma(\mathcal{X})$ on terms:

$$f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$$

Example (Applying a substitution)

Given substitution $\sigma = \{X \mapsto senc(M, K)\}$ and the term $t = sdec(X, K)$ we can apply the substitution and get $t\sigma = sdec(senc(M, K), K)$.

Substitution (ctd.)

Definition (Substitution composition)

We denote with $\sigma\tau$ the composition of substitutions σ and τ, i.e., $\tau \circ \sigma$.

Example (Substitution composition)

For substitutions $\sigma = [x \mapsto f(y), y \mapsto z]$ and $\tau = [y \mapsto a, z \mapsto g(b)]$ we have $\sigma\tau = [x \mapsto f(a), y \mapsto g(b), z \mapsto g(b)]$.

Position

Definition (Position)

A position p is a sequence of positive integers. The subterm $t|_p$ of a term t at position p is obtained as follows.

- If $p = [\,]$ is the empty sequence, then $t|_p = t$.
- If $p = [i] \cdot p'$ for a positive integer i and a sequence p', and $t = f(t_1, \ldots, t_n)$ for $f \in \Sigma$ and $1 \leq i \leq n$ then $t|_p = t_i|_{p'}$, else $t|_p$ does not exist.

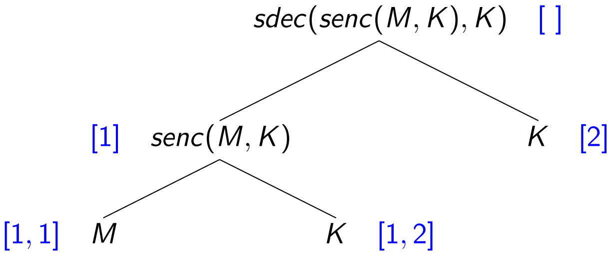
Example (Position in a term)

For the term $t = sdec(senc(M, K), K)$ we have five subterms:

$t|_{[\,]} = t$
$t|_{[1]} = senc(M, K)$
$t|_{[1,1]} = M$
$t|_{[1,2]} = K$
$t|_{[2]} = K$

Graphical representation of positions in a term

Tree of subterms of $sdec(senc(M, K))$ and their positions.



Matching and Application

Definition (Matching)

A term t matches another term l if there is a subterm of t, i.e., $t|_p$, such that there is a substitution σ so that $t|_p = l\sigma$. We call σ the matching substitution.

Matching and Application

Definition (Matching)

A term t matches another term l if there is a subterm of t, i.e., $t|_p$, such that there is a substitution σ so that $t|_p = l\sigma$. We call σ the matching substitution.

Definition (Application of a rule)

A rule (oriented equation) $l \to r$ is applicable on a term t, when t matches l.

The result of such a rule application is the term $t[r\sigma]_p$, where σ is the matching substitution.

Unification

Definition (Unification)

We say that $t \stackrel{?}{=} t'$ is unifiable in (Σ, E) for $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$, if there is a substitution σ such that $t\sigma =_E t'\sigma$ and we call σ a unifier.

For syntactic unification ($E = \emptyset$) there is a most general unifier for two unifiable terms, and it is decidable whether they are unifiable.

Unification modulo theories

- When considering other algebras, unifiability is in general undecidable, e.g., associativity and distributivity.
- Even when decidable, there is in general no unique most general unifier, e.g., $\{\exp(X, Y), \exp(X', c)\}$ ...
- Some unification problems are decidable but infinitary: in general, there is an infinite set of most general unifiers, e.g., associativity.

Equational Proofs

Definition (Equality Relation)

Given (Σ, E), an E-equality step for $u, v \in \mathcal{T}_\Sigma(\mathcal{X})$ is defined as $u \rightarrow_{(\vec{E} \cup \overleftarrow{E})} v$ and denoted as $u \leftrightarrow_E v$.
The transitive-reflexive closure of $\leftrightarrow_E$ is the E-equality relation $=_E$.

Definition (Equality Proof)

A sequence of steps $t_0 \leftrightarrow_E t_1 \leftrightarrow_E \ldots \leftrightarrow_E t_n$, witnessing n-step equality of $t_0 \leftrightarrow_E^+ t_n$ is an equality proof.

Equality for Peano natural numbers

Example (Equality reasoning for Peano natural numbers)

Consider how to prove $s(s(0)) + s(0) = s(0) + s(s(0))$:

$$\underline{s(s(0)) + s(0)} = s(\underline{s(s(0)) + 0}) = s(s(\underline{s(0)}))$$

$$= s(\underline{s(s(0) + 0)}) = \underline{s(s(0) + s(0))} = s(0) + s(s(0))$$

Complicated! Using termination and confluence, we could have instead computed the normal form of both sides, and simply compared them! (See next slides.)

See also: Assignment 2.2.

Termination of $\vec{E}$

Definition (Termination)

$(\Sigma, \vec{E})$ has infinite computations, if there is a function $a : \mathbb{N} \to \mathcal{T}_\Sigma(\mathcal{X})$ such that

$$a(0) \to_{\vec{E}} a(1) \to_{\vec{E}} a(2) \to_{\vec{E}} \ldots \to_{\vec{E}} a(n) \to_{\vec{E}} a(n+1) \ldots$$

We say it is terminating, when it does not have infinite computations.

Example (Termination)

For $E = \{a = b\}$, $\vec{E}$ is terminating.
For $E = \{a = b, b = a\}$, $\vec{E}$ is not terminating.

Confluence of $\vec{E}$

Definition (Confluence)

Confluence is the property that guarantees the order of applying equalities is immaterial, formally:

$\forall t, t_1, t_2 . t \rightarrow^* t_1 \wedge t \rightarrow^* t_2 \Rightarrow \exists s . t_1 \rightarrow^* s \wedge t_2 \rightarrow^* s$

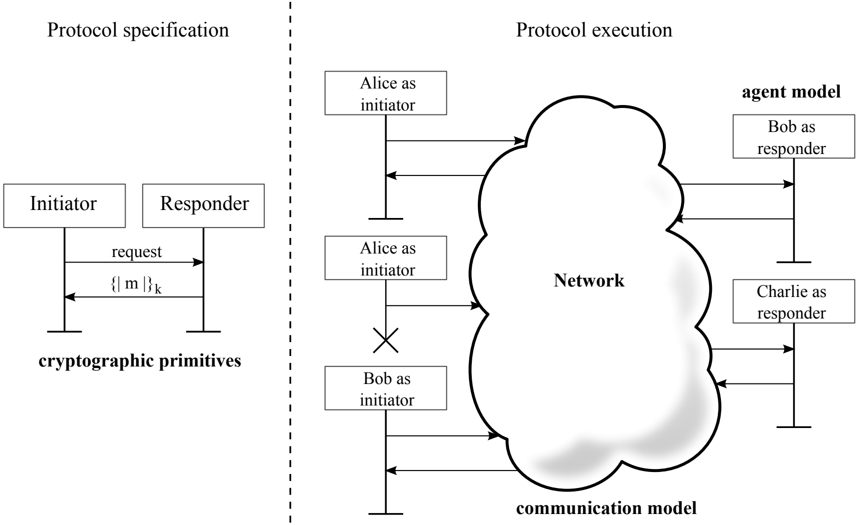
$$\begin{tikzcd}
& t \arrow[dl] \arrow[dr] & \\
t_1 \arrow[dr, dotted] & & t_2 \arrow[dl, dotted] \\
& s &
\end{tikzcd}$$

Example (Confluence)

For $E = \{a = b, a = c\}$, we have that $\vec{E}$ is not confluent, as b and c are reachable from a, but not joinable.
For $E = \{a = b, a = c, b = c\}$, then $\vec{E}$ is confluent.

Outline

Modeling the Adversary



Danny Dolev & Andrew C. Yao

On the Security of Public Key Protocols (IEEE Trans. Inf. Th., 1983)

- Consider a public key system in which for every user X
  - ★ there is a public encryption function E_X
    — every user can apply this function.
  - ★ and a private decryption function D_X
    — only X can apply this function.
  - ★ These functions have the property that $E_X D_X = D_X E_X = 1$.
- The **Dolev-Yao adversary**:
  - ★ Controls the network (read, intercept, send)
  - ★ Is also a user, called Z
  - ★ Can apply E_X for any X
  - ★ Can apply D_Z

Dolev-Yao Deduction

Definition (Adversary Knowledge)

We represent the adversary knowing a term t by a fact $\mathsf{K}(t)$. The set of the adversary's knowledge is $\mathcal{K}$ and contains facts of the form $\mathsf{K}(t)$, all of which are persistent.

Definition (Adversary Knowledge Derivation)

The adversary can use the following inference rules on the state:

$$\frac{\mathsf{Fr}(x)}{\mathsf{K}(x)} \qquad \frac{\mathsf{Out}(x)}{\mathsf{K}(x)} \qquad \frac{\mathsf{K}(x)}{\mathsf{In}(x)}$$

$$\frac{\mathsf{K}(t_1)\ldots\mathsf{K}(t_k)}{\mathsf{K}(f(t_1,\ldots,t_k))}\ \forall f \in \Sigma(\text{k-ary})$$

Note that terms are used modulo the equational theory. So, given $\mathsf{K}(< t_1, t_2 >)$ the operator *fst* can be applied, and the result is $\mathsf{K}(t_1)$.

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$
\dfrac{\overline{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \overline{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}}{\mathsf{K}(b, n)}\,E}{\mathsf{K}(snd(b, n))}}{\mathsf{K}(n)}\,E \qquad \overline{\mathsf{K}(x)}}{\mathsf{K}(prf(n, x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}
$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$\dfrac{\overline{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \overline{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}}{\mathsf{K}(b, n)}\,E}{\mathsf{K}(snd(b, n))}}{\mathsf{K}(n)}\,E \qquad \overline{\mathsf{K}(x)}}{\mathsf{K}(prf(n, x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$\dfrac{\dfrac{\dfrac{}{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{K}(\{\!|b,n|\!\}_k)} \quad \dfrac{}{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b,n|\!\}_k|\!\}_k)}}{\mathsf{K}(b,n)}\,E}{\mathsf{K}(snd(b,n))}}{\mathsf{K}(n)}\,E \qquad \dfrac{}{\mathsf{K}(x)}}{\mathsf{K}(prf(n,x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}}{}$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$
\dfrac{\dfrac{}{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \dfrac{}{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}}{\mathsf{K}(b, n)}\,E}{\mathsf{K}(snd(b, n))}}{\mathsf{K}(n)}\,E \qquad \dfrac{}{\mathsf{K}(x)}}{\mathsf{K}(prf(n, x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}
$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$
\dfrac{\overline{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\overline{\mathsf{K}(snd(b,n))}}{\mathsf{K}(n)}\,E \qquad \overline{\mathsf{K}(x)}}{\mathsf{K}(prf(n,x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}
$$

$$
\dfrac{\dfrac{\overline{\mathsf{K}(\{\!|b, n|\!\}_k)} \qquad \overline{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}}{\mathsf{K}(b, n)}\,E
$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$
\dfrac{
  \overline{\mathsf{K}(m)}
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{
            \dfrac{\overline{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \overline{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}
          }{\mathsf{K}(b, n)}\ E
        }{\mathsf{K}(snd(b, n))}
      }{\mathsf{K}(n)}\ E
    \qquad
    \overline{\mathsf{K}(x)}
  }{\mathsf{K}(prf(n, x))}
}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}
$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$
\dfrac{\dfrac{}{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \dfrac{}{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}}{\mathsf{K}(b, n)}\,E}{\mathsf{K}(snd(b, n))}}{\mathsf{K}(n)}\,E \qquad \dfrac{}{\color{red}{\mathsf{K}(x)}}}{\mathsf{K}(prf(n, x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}
$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$\dfrac{\dfrac{}{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \dfrac{}{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}}{\mathsf{K}(b, n)}\,E}{\mathsf{K}(snd(b, n))}}{\mathsf{K}(n)}\,E \qquad \dfrac{}{\mathsf{K}(x)}}{\mathsf{K}(prf(n, x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$
\dfrac{\overline{\mathsf{K}(m)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \overline{\mathsf{K}(k)}}{\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)}}{\mathsf{K}(b, n)}\,E}{\dfrac{\overline{\mathsf{K}(snd(b, n))}}{\mathsf{K}(n)}\,E} \qquad \overline{\mathsf{K}(x)}}{\mathsf{K}(prf(n, x))}}{\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})}
$$

Dolev-Yao Deduction

Example

Given $\mathsf{K}(x), \mathsf{K}(\{\!|b, n|\!\}_k), \mathsf{K}(k), \mathsf{K}(m) \in \mathcal{K}$. Use the equational theory E (containing decryption and pairing) to derive $\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})$

$$
\dfrac{
\dfrac{}{\mathsf{K}(m)}
\qquad
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\mathsf{K}(\{\!|b, n|\!\}_k)} \quad \dfrac{}{\mathsf{K}(k)}
}{
\mathsf{K}(\{\!|\{\!|b, n|\!\}_k|\!\}_k)
}
}{\mathsf{K}(b, n)}\; E
}{
\dfrac{\mathsf{K}(snd(b, n))}{\mathsf{K}(n)}\; E
}
\qquad
\dfrac{}{\mathsf{K}(x)}
}{\mathsf{K}(prf(n, x))}
}{
\mathsf{K}(\{\!|m|\!\}_{prf(n,x)})
}
$$

Dolev-Yao Deduction

Definition (Adversary Knowledge Derivation as rewrite rules)

$$[\mathsf{Fr}(x)] \rightarrow [\mathsf{K}(x)]$$

$$[\mathsf{Out}(x)] \rightarrow [\mathsf{K}(x)]$$

$$[\mathsf{K}(x)] \xrightarrow{\mathsf{K}(x)} [\mathsf{In}(x)]$$

$$[\mathsf{K}(t_1), \ldots, \mathsf{K}(t_k)] \rightarrow [\mathsf{K}(f(t_1, \ldots, t_k))] \quad \forall f \in \Sigma(\text{k-ary})$$

As you see, the adversary deriving a message and then sending it (via In) is annotated with the action fact K (identical to its state fact of the same name!), and we use this for our reasoning later.

Outline

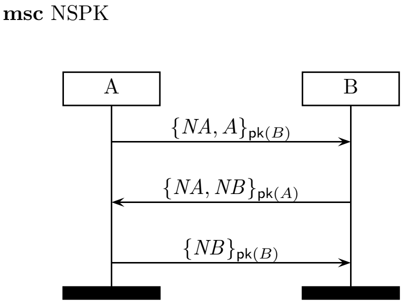
Outline

Basic ideas:

- We express the semantics of an AnB specification by a finite set P of role descriptions.
- Additionally, define an initial state $([], IK_0, th_0)$ with an infinite number of threads.
- Then the semantics of role-descriptions defines an infinite-state transition system.

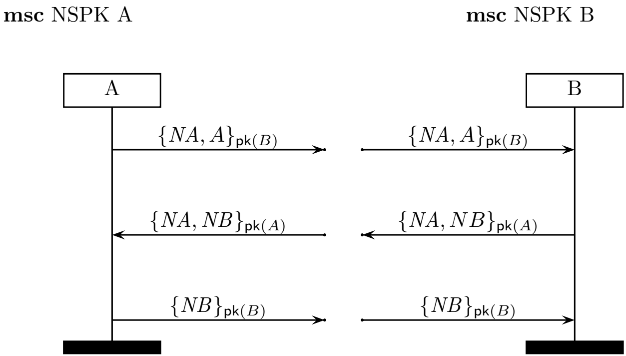
Recall initial idea

Split a message sequence chart into roles:



Recall initial idea

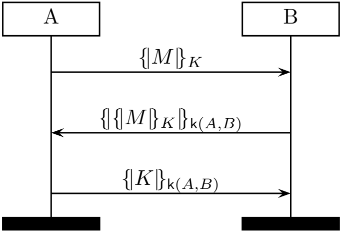
Split a message sequence chart into roles:



Recall initial idea

Not trivial for all protocols:

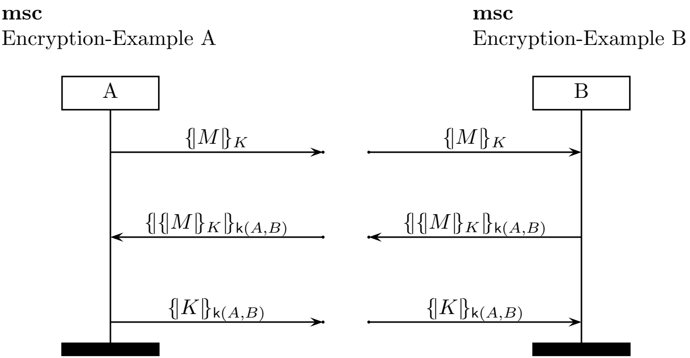
msc Encryption-Example



Here, $\mathsf{k}(A, B)$ is a shared key of A and B, K is fresh.

Recall initial idea

Not trivial for all protocols:



This is wrong: B cannot decrypt/check the format of the first message... before receiving the third!

Problems with the naive translation

- All protocols where agents cannot fully decrypt messages they receive: Kerberos, NSCK, many other shared-key examples.
- Diffie-Hellman.
- All these protocols would give unrealistic models.
- No executability check: can the agents generate all messages as they are supposed to?
- Construction of messages depends on agents' view of the messages and algebraic properties.

A running example for the semantics of AnB

$$
\begin{array}{l}
\texttt{Protocol}: \textit{Diffie-Hellman}\\
\texttt{Types}:\\
\quad \mathcal{A}\ A, B;\\
\quad \texttt{Number}\ g, X, Y, Msg;\\
\quad \texttt{Function}\ \mathsf{pk};\\
\texttt{Knowledge}:\\
\quad A:\ \ A, B, g, \mathsf{pk}, (\mathsf{pk}(A))^{-1};\\
\quad B:\ \ B, g, \mathsf{pk}, (\mathsf{pk}(B))^{-1};\\
\texttt{Actions}:\\
\qquad \begin{array}{lcl}
A \to B & : & \{\mathsf{exp}(g, X)\}_{(\mathsf{pk}(A))^{-1}}\\
B \to A & : & \{\mathsf{exp}(g, Y)\}_{(\mathsf{pk}(B))^{-1}}\\
A \to B & : & \{\!|A, Msg|\!\}_{\mathsf{exp}(\mathsf{exp}(g,X),Y)}
\end{array}\\
\texttt{Goals}:\\
\qquad A \bullet\!\!\rightarrow\!\!\bullet\ B\ :\ Msg
\end{array}
$$

Construction of Messages

Consider the set of messages M that an agent knows at a certain stage of the protocol execution:

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A, B, \mathrm{pk}, (\mathrm{pk}(A))^{-1}}_{\text{Initial Knowledge}}, \underbrace{X, Msg}_{\text{created}} \underbrace{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}$.

Crucial questions for defining the semantics:

- What can she check about M?
- Can she construct m from knowledge M? Executability.
- If she can construct m: how?

Construction of Messages

Consider the set of messages M that an agent knows at a certain stage of the protocol execution:

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathrm{pk}^{\mathcal{X}_2}, (\mathrm{pk}(A)^{\mathcal{X}_3})^{-1}}_{\text{Initial Knowledge}}, \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}} \underbrace{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}$.

Crucial questions for defining the semantics:

- What can she check about M?
- Can she construct m from knowledge M? Executability.
- If she can construct m: how?

To formally define this, we begin by labeling each element of M with a new variable $\mathcal{X}_i$.

Labeled Adversary Deduction

We define a variant $\mathcal{DY}_l$ of the Dolev-Yao closure for labeled terms:

Definition

$$\frac{}{m^l \in \mathcal{DY}_l(M)} \text{ Axiom } (m^l \in M) \qquad \frac{s^k \in \mathcal{DY}_l(M)}{t^l \in \mathcal{DY}_l(M)} \text{ Algebra } (s \approx t, l \approx k)$$

$$\frac{t_1^{l_1} \in \mathcal{DY}_l(M) \quad \dots \quad t_n^{l_n} \in \mathcal{DY}_l(M)}{f(t_1, \dots, t_n)^{f(l_1, \dots, l_n)} \in \mathcal{DY}_l(M)} \text{ Composition } (f \in \Sigma_p)$$

We push implicit decryption under the carpet here (a bit tricky)...

Construction of Messages

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathsf{pk}^{\mathcal{X}_2}, (\mathsf{pk}(A)^{\mathcal{X}_3})^{-1}}_{\text{Initial Knowledge}}, \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}} \underbrace{\{\exp(g,Y)\}_{(\mathsf{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{|A, Msg|\}_{\exp(\exp(g,X),Y)}$.

Alice can derive m:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\exp(g,Y)\}_{(\mathsf{pk}(B))^{-1}}{}^{\mathcal{X}_6}}}{\mathsf{open}(\{\exp(g,Y)\}_{(\mathsf{pk}(B))^{-1}})^{\mathsf{open}(\mathcal{X}_6)}}}{\exp(g,Y)^{\mathsf{open}(\mathcal{X}_6)}} \qquad \dfrac{}{X^{\mathcal{X}_4}}}{\exp(\exp(g,Y),X)^{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}}}{\exp(\exp(g,X),Y)^{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}} \qquad \ldots}{\{|A, Msg|\}_{\exp(\exp(g,X),Y)}^{\{|\mathcal{X}_0,\mathcal{X}_5|\}_{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}}}$$

... as $\{|\mathcal{X}_0, \mathcal{X}_5|\}_{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}$.

Construction of Messages

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathsf{pk}^{\mathcal{X}_2}, (\mathsf{pk}(A)^{\mathcal{X}_3})^{-1}}_{\text{Initial Knowledge}}, \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}} \underbrace{\{\exp(g,Y)\}_{(\mathsf{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}$.

Alice can derive m:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\{\exp(g,Y)\}_{(\mathsf{pk}(B))^{-1}}{}^{\mathcal{X}_6}}}{\mathsf{open}(\{\exp(g,Y)\}_{(\mathsf{pk}(B))^{-1}})^{\mathsf{open}(\mathcal{X}_6)}}}{\exp(g,Y)^{\mathsf{open}(\mathcal{X}_6)}} \quad \overline{X^{\mathcal{X}_4}}}{\exp(\exp(g,Y),X)^{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}}}{\exp(\exp(g,X),Y)^{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}} \quad \ldots}{\{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}^{\{\!|\mathcal{X}_0,\mathcal{X}_5|\!\}_{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}}}$$

... as $\{\!|\mathcal{X}_0, \mathcal{X}_5|\!\}_{\exp(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}$.

Construction of Messages

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathrm{pk}^{\mathcal{X}_2}, (\mathrm{pk}(A)^{\mathcal{X}_3})^{-1}}_{\text{Initial Knowledge}}, \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}} \underbrace{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{|A, Msg|\}_{\exp(\exp(g,X),Y)}$.

Alice can derive m:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}}{\mathrm{open}(\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}})^{\mathrm{open}(\mathcal{X}_6)}}}{\exp(g, Y)^{\mathrm{open}(\mathcal{X}_6)}} \quad \dfrac{}{X^{\mathcal{X}_4}}}{\exp(\exp(g, Y), X)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}{\exp(\exp(g, X), Y)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}} \quad \ldots}{\{|A, Msg|\}_{\exp(\exp(g,X),Y)}^{\{|\mathcal{X}_0,\mathcal{X}_5|\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}$$

... as $\{|\mathcal{X}_0, \mathcal{X}_5|\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}$.

Construction of Messages

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathsf{pk}^{\mathcal{X}_2}, (\mathsf{pk}(A)^{\mathcal{X}_3})^{-1}}_{\text{Initial Knowledge}}, \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}} \underbrace{\{\mathsf{exp}(g, Y)\}_{(\mathsf{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{\!|A, Msg|\!\}_{\mathsf{exp}(\mathsf{exp}(g,X),Y)}$.

Alice can derive m:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\mathsf{exp}(g, Y)\}_{(\mathsf{pk}(B))^{-1}}{}^{\mathcal{X}_6}}}{\mathsf{open}(\{\mathsf{exp}(g, Y)\}_{(\mathsf{pk}(B))^{-1}})^{\mathsf{open}(\mathcal{X}_6)}}}{\mathsf{exp}(g, Y)^{\mathsf{open}(\mathcal{X}_6)}} \qquad \dfrac{}{X^{\mathcal{X}_4}}}{\dfrac{\mathsf{exp}(\mathsf{exp}(g, Y), X)^{\mathsf{exp}(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}}{\mathsf{exp}(\mathsf{exp}(g, X), Y)^{\mathsf{exp}(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}}} \qquad \ldots}{\{\!|A, Msg|\!\}_{\mathsf{exp}(\mathsf{exp}(g,X),Y)}^{\{\!|\mathcal{X}_0,\mathcal{X}_5|\!\}_{\mathsf{exp}(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}}}$$

... as $\{\!|\mathcal{X}_0, \mathcal{X}_5|\!\}_{\mathsf{exp}(\mathsf{open}(\mathcal{X}_6),\mathcal{X}_4)}$.

Construction of Messages

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathrm{pk}^{\mathcal{X}_2}, (\mathrm{pk}(A)^{\mathcal{X}_3})^{-1},}_{\text{Initial Knowledge}} \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}} \underbrace{\{\exp(g,Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}$.

Alice can derive m:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\exp(g,Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}}{\mathrm{open}(\{\exp(g,Y)\}_{(\mathrm{pk}(B))^{-1}})^{\mathrm{open}(\mathcal{X}_6)}}}{\exp(g,Y)^{\mathrm{open}(\mathcal{X}_6)}} \qquad \dfrac{}{X^{\mathcal{X}_4}}}{\exp(\exp(g,Y),X)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}{\color{red}{\exp(\exp(g,X),Y)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}} \qquad \ldots}{\{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}^{\{\!|\mathcal{X}_0,\mathcal{X}_5|\!\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}$$

... as $\{\!|\mathcal{X}_0, \mathcal{X}_5|\!\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}$.

Construction of Messages

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathrm{pk}^{\mathcal{X}_2}, (\mathrm{pk}(A)^{\mathcal{X}_3})^{-1}}_{\text{Initial Knowledge}}, \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}}\, \underbrace{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}$.

Alice can derive m:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}}{\mathrm{open}(\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}})^{\mathrm{open}(\mathcal{X}_6)}}}{\exp(g, Y)^{\mathrm{open}(\mathcal{X}_6)}} \quad \dfrac{}{X^{\mathcal{X}_4}}}{\exp(\exp(g, Y), X)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}{\exp(\exp(g, X), Y)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}} \quad \color{red}{\dots}}{\color{red}{\{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}^{\{\!|\mathcal{X}_0,\mathcal{X}_5|\!\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}}$$

. . . as $\{\!|\mathcal{X}_0, \mathcal{X}_5|\!\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}$.

Construction of Messages

Example (Diffie-Hellman, Alice, receiving msg. 2)

$$M = \{\underbrace{A^{\mathcal{X}_0}, B^{\mathcal{X}_1}, \mathrm{pk}^{\mathcal{X}_2}, (\mathrm{pk}(A)^{\mathcal{X}_3})^{-1}}_{\text{Initial Knowledge}}, \underbrace{X^{\mathcal{X}_4}, Msg^{\mathcal{X}_5}}_{\text{created}} \underbrace{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}_{\text{received}}\}$$

The next outgoing message of Alice is $m = \{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}$.

Alice can derive m:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}}{}^{\mathcal{X}_6}}}{\mathrm{open}(\{\exp(g, Y)\}_{(\mathrm{pk}(B))^{-1}})^{\mathrm{open}(\mathcal{X}_6)}}}{\exp(g, Y)^{\mathrm{open}(\mathcal{X}_6)}} \quad \overline{X^{\mathcal{X}_4}}}{\exp(\exp(g, Y), X)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}{\exp(\exp(g, X), Y)^{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}} \quad \dots}{\{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}^{\{\!|\mathcal{X}_0,\mathcal{X}_5|\!\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}}}$$

. . . as $\{\!|\mathcal{X}_0, \mathcal{X}_5|\!\}_{\exp(\mathrm{open}(\mathcal{X}_6),\mathcal{X}_4)}$.

Checking Messages

Crucial questions for defining the semantics:

- What can she check about M?
- ✓ Can she construct m from knowledge M? Executability.
- ✓ If she can construct M: how?

Checking Messages

Checking is quite tricky, again:

- In general, all pairs (l_1, l_2) of distinct derivations the agent can do and that should give the same term t according to the protocol:

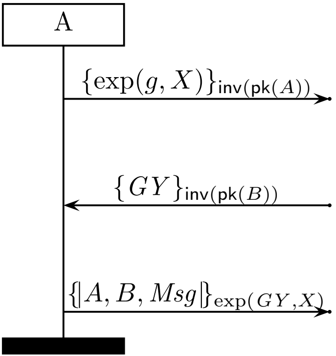
$$t^{l_1}, t^{l_2} \in \mathcal{DY}_I(M)$$

- In general, there are infinitely many checks.
- For many algebraic theories (e.g. exponentiation) we can reduce this to an equivalent finite set of checks.
- These checks and the explicit destructors can, for many examples, be translated into pattern matching, e.g.

$$\boxed{\begin{array}{c}\mathrm{rcv}(\mathcal{X}_6) \text{ where } \mathit{verify}(\mathrm{pk}(B), \mathcal{X}_6) \approx \mathrm{true} \\ \mathrm{snd}(\ldots, \mathrm{open}(\mathcal{X}_6), \ldots)\end{array}} \mapsto \boxed{\begin{array}{c}\mathrm{rcv}(\{\mathcal{X}'_6\}_{(\mathrm{pk}(B))^{-1}}) \\ \mathrm{snd}(\ldots, \mathcal{X}'_6, \ldots)\end{array}}$$

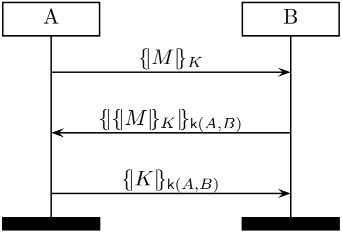
Result on Diffie-Hellman:

msc DH A

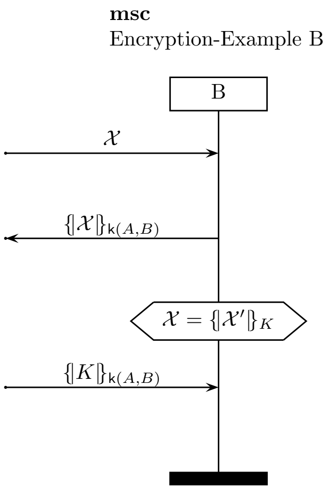


Our problem from before

msc Encryption-Example



Our problem from before



. . . requires some extension of role-descriptions!

Initial state

Definition

- Let $Agent \subset \Sigma_0$ be the set of all (constant) agent names, including the adversary i.
- Let V be the set of all variables in the initial knowledge of the roles (which are of type agent according to AnB syntax).
- Let Sub_V be the set of all substitutions σ with $dom(\sigma) = V$ and $ran(\sigma) \subset Agent$.
- $IK_0 = \bigcup_{\sigma \in Sub_V \wedge R\sigma = i} init(R)\sigma$ where $init(R)$ is the initial knowledge of role R in the AnB spec.

Example

Let $Agent = \{a, b, i\}$. For NSPK, we the set of roles $V = \{A, B\}$.
$Sub_V = \{\ [A \mapsto a, B \mapsto b], [A \mapsto b, B \mapsto a], [A \mapsto a, B \mapsto i], \ldots\}$.
$IK_0 = \{a, b, i, pk, (\mathrm{pk}(i))^{-1}\}$.

Initial state (cont.)

Definition

Consider a protocol P with roles $dom(P) = \{R_1, \ldots, R_k\}$ and let $Sub_V = \{\sigma_1, \sigma_2, \ldots\}$

- Let $TID = (\{1, \ldots, k\} \times \mathbb{N} \times \mathbb{N})$
- For each $(r, i, n) \in TID$, let $\sigma_{(r,i,n)}$ a substitution with domain $fv(R_r)$ where $v\sigma_{(r,i,n)} = v\sigma_i$ for all role names $v \in V \cap fv(R_r)$ and where the remaining free variables, i.e. $fv(R_r) \setminus V$, are mapped to fresh constants (disjoint over all $\sigma_{(r,i,n)}$).
- $role((r, i, n)) = R_r$ for all $(r, i, n) \in TID$
- $player((r, i, n)) = R_r\sigma_{(r,i,n)}$ for all $(r, i, n) \in TID$
- $th_0((r, i, n)) = P(R_r)\sigma_{(r,i,n)}$ for all $(r, i, n) \in TID$ where $player((r, i, n)) \neq i$.

Initial state (cont.)

For the NSPK attack, we need the following two threads, where $\sigma_1 = [A \mapsto a, B \mapsto b]$, $\sigma_3 = [A \mapsto a, B \mapsto i]$

Example

$$\begin{aligned} \sigma_{(1,3,0)} &= [A \mapsto a, B \mapsto i, NA \mapsto na_{(1,3,0)}] \\ \sigma_{(2,1,0)} &= [B \mapsto b, NB \mapsto nb_{(2,1,0)}] \end{aligned}$$

Actually, since $A \notin fvB$, also $\sigma_{2,3,0}$ would equally work for the attack.

Overview

- Introduction
- Two formal specification languages:

- Security Properties
- Landscape of Protocol Models: a quick tour.

Outline

Restricted Tamarin syntax with explicit send/receive

A protocol defines the behavior of a set of roles. Every role has a name R and consists of a set of rules, specifying the sending and receiving of messages, and the generation of fresh constants.

Restricted Tamarin syntax with explicit send/receive

A protocol defines the behavior of a set of roles. Every role has a name R and consists of a set of rules, specifying the sending and receiving of messages, and the generation of fresh constants. Such a rule is of the form

$$[\mathrm{St\_R\_s}(A, id, k_1, \ldots, k_n), \ldots] \xrightarrow{a} [\mathrm{St\_R\_s}'(A, id, k'_1, \ldots, k'_m), \ldots]$$

Restricted Tamarin syntax with explicit send/receive

A protocol defines the behavior of a set of roles. Every role has a name R and consists of a set of rules, specifying the sending and receiving of messages, and the generation of fresh constants. Such a rule is of the form

$$[\mathrm{St\_R\_s}(A, id, k_1, \ldots, k_n), \ldots] \xrightarrow{a} [\mathrm{St\_R\_s'}(A, id, k'_1, \ldots, k'_m), \ldots]$$

where R is the role name, $s \in \mathbb{N}$ the index for the present protocol step of the role, $s' = s + 1$ the index for the subsequent step. A is the agent name, id the thread identifier for this instantiation of role R, and the $k_i, k'_j \in \mathcal{T}_\Sigma(\mathcal{X})$ are terms in the agent's knowledge.
We call $\mathrm{St\_R\_s}(A, \ldots)$ an agent state fact for role R.

Nomenclature

Definition (Facts)

We call the top-level operators of the left- and right-hand sides of rules state facts, e.g., St_R_s(...), and we call the top-level operators in the rule label *a* the action facts. All arguments of facts are terms in $\mathcal{T}_{\Sigma}(\mathcal{X})$.

Nomenclature

Definition (Facts)

We call the top-level operators of the left- and right-hand sides of rules state facts, e.g., St_R_s(...), and we call the top-level operators in the rule label a the action facts. All arguments of facts are terms in $\mathcal{T}_\Sigma(\mathcal{X})$.

Definition (Events)

For a protocol rule $l \xrightarrow{a} r$ the actions a include all the information we will reason about. Thus, our traces of events will consist of sequences of such labels.

Communication

Messages are sent and received via In and Out facts, respectively, and any rule with such a fact also will have a matching Send and Recv action, respectively.

Example (Rule examples)

Receive rule example

$$[\mathsf{St\_I\_2}(A, 17, k), \mathsf{In}(m)] \xrightarrow{\mathsf{Recv}(A,m)} [\mathsf{St\_I\_3}(A, 17, k, m)]$$

Send rule example

$$[\mathsf{St\_I\_3}(A, 17, k, m)] \xrightarrow{\mathsf{Send}(A,\{m\}_k))} [\mathsf{St\_I\_4}(A, 17, k, m), \mathsf{Out}(\{m\}_k)]$$

Fresh and public Terms

Definition (Fresh terms)

Agents generate fresh terms using fresh facts, denoted by Fr. These fresh terms represent randomness being used, are assumed unguessable and unique, i.e., can represent nonces.

There is a countable supply of fresh terms, each as argument of a fresh fact, usable in rules.

Definition (Public terms)

We define public terms to be terms known to all participants of a protocol. These include all agent names and all constants.

Well-formedness

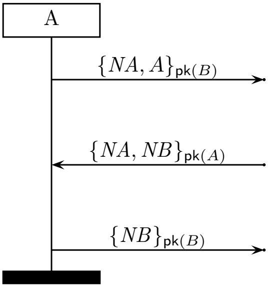
For a protocol rule $l \xrightarrow{a} r$ to be well-formed, the following conditions must be satisfied (except initialization rules):

1. Only In, Fr, and state facts occur in l.
2. Only Out and state facts occur in r.
3. Exactly one state fact occurs in each of l and r.
4. Either In or Out facts occur in the rule, never both.
5. If $\mathrm{St\_R\_s}(A, id, k_1, \ldots, k_n)$ occurs in l, then
   - **(i)** every In fact is of the form $\mathrm{In}(x)$, where $x \in \mathcal{T}_\Sigma(\mathcal{X})$,
   - **(ii)** every Out fact is of the form $\mathrm{Out}(x)$, where $x \in \mathcal{T}_\Sigma(\mathcal{X})$ and x is derivable from public terms, terms in Fr facts occurring in l and the terms $k_1, \ldots, k_n$.
   - **(iii)** the fact $\mathrm{St\_R\_s}'(A, id, k'_1, \ldots, k'_m)$ occurs in r, where $s' = s + 1$ and $k'_1, \ldots, k'_m$ are derivable from public terms, terms in Fr facts occurring in l, and the terms $k_1, \ldots, k_n$.
6. Every variable in r that is not public must occur in l.

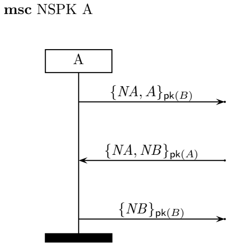
Role Syntax

Graphical:

msc NSPK A



Role specification rules



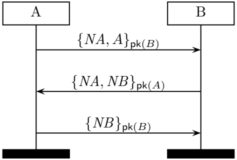
$$[\mathrm{St\_A\_1}(A, tid, skA, pk(skB)), \quad \mathrm{Fr}(NA)] \rightarrow$$
$$[\mathrm{St\_A\_2}(A, tid, skA, pk(skB), NA), \quad \mathrm{Out}(\{NA, A\}_{pk(skB)})]$$

$$[\mathrm{St\_A\_2}(A, tid, skA, pk(skB), NA), \quad \mathrm{In}(\{NA, NB\}_{pk(skA)})] \rightarrow$$
$$[\mathrm{St\_A\_3}(A, tid, skA, pk(skB), NA, NB)]$$

$$[\mathrm{St\_A\_3}(A, tid, skA, pk(skB), NA, NB)] \rightarrow$$
$$[\mathrm{St\_A\_4}(A, tid, skA, pk(skB), NA, NB), \quad \mathrm{Out}(\{NB\}_{pk(skB)})]$$

PKIs and longterm data

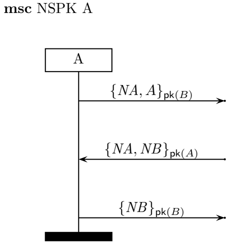
msc NSPK



Generate longterm keys and public keys.

$$[\mathsf{Fr}(skR)] \rightarrow [\mathsf{Ltk}(R, skR), \mathsf{Out}(pk(skR))]$$

Initialization of roles



For each role R there must be an initialization rule which is instantiated with a name A and a thread identifier id:

$$[\mathrm{Fr}(id), \mathrm{Ltk}(A, skA), \mathrm{Ltk}(B, skB)] \xrightarrow{\text{Create\_R(A,id)}} [\mathrm{St\_R\_1}(A, id, skA, pk(skB)), \mathrm{Ltk}(A, skA), \mathrm{Ltk}(B, skB)]$$

Role-based Protocol Property Specifications

Definition (Events for property specification)

$$\begin{aligned} \mathit{Event}(\mathit{Term}) \quad = \quad & \mathsf{Send}(\mathcal{R}, \mathit{Term}) \mid \mathsf{Recv}(\mathcal{R}, \mathit{Term}) \mid \\ & \mathsf{Claim\_}\mathtt{claimtype}(\mathcal{R}, \mathit{Term}^*) \mid \\ & \mathsf{Create\_R}(\mathcal{R}, \mathit{id}) \end{aligned}$$

We use Claim actions for property specification. Verification uses claims and messages.

Outline

Outlook

We will define a trace semantics for protocols in terms of labeled transition systems.

Labeled Multiset Rewriting

Definition (Multiset)

A multiset is a set of elements, each imbued with a multiplicity. Instead of stating an explicit multiplicity, we may also simply write elements multiple times.
We use $\setminus^\sharp$ for the multiset difference, and $\cup^\sharp$ for the union.

Definition (Labeled multiset rewriting)

A labeled multiset rewriting rule is a triple, l, a, r, each of which is a multisets of facts, and written as:

$$l \xrightarrow{a} r$$

State

Definition (State)

A state is a multiset of facts.

Example (State)

$$\mathrm{St\_R\_1}(A, id, k_1, k_2), \mathrm{Out}(k_1), \mathrm{Out}(k_2), \mathrm{Out}(k_2)$$

Ground substitution

Definition (Ground substitutition)

A substitution is called ground when each variable is mapped to a ground term.

Definition (Ground instances)

We call the ground instances of a term t all those terms $t\sigma$ that are ground for some (ground) substitution.

A fact F is ground if all its terms are ground. The multiset of all ground facts is $\mathcal{G}^{\sharp}$.

For a rule, its ground instances are those where all facts are ground, and we use

$$ginsts(R)$$

for the set of all ground instances of the set of rules R.

Fresh rule

Definition (Fresh rule)

We define a special rule for the creation of fresh facts. This is the only rule allowed to produce fresh facts and has no precondition:

$$[] \rightarrow [\mathrm{Fr}(N)]$$

Note that each created nonce N is fresh, and thus unique.

Labeled operational semantics - single step

Definition (Steps)

For a multiset rewrite system R we define the labeled transition relation step, *steps*$(R) \subseteq \mathcal{G}^\sharp \times ginsts(R) \times \mathcal{G}^\sharp$, as follows:

$$\frac{l \xrightarrow{a} r \in ginsts(R), \qquad l \subseteq^\sharp S, \qquad S' = (S \setminus^\sharp l) \cup^\sharp r}{(S, l \xrightarrow{a} r, S') \in steps(R)}$$

Executions

Definition (Execution)

An execution of R is an alternating sequence

$$S_0, (l_1 \xrightarrow{a_1} r_1), S_1, \ldots, S_{k-1}(l_k \xrightarrow{a_k} r_k), S_k$$

of states and multiset rewrite rule instances with

(1) $S_0 = \emptyset$

(2) $\forall i : S_{i-1}, (l_i \xrightarrow{a_i} r_i), S_i \in \mathit{steps}(R)$

(3) Fresh names are unique, i.e., for n fresh, and $(l_i \xrightarrow{a_i} r_i) = (l_j \xrightarrow{a_j} r_j) = ([] \rightarrow [\mathrm{Fr}(n)])$ it holds that $i = j$.

Trace

Definition (Trace)

The trace of an execution

$$S_0, (l_1 \xrightarrow{a_1} r_1), S_1, \ldots, S_{k-1}(l_k \xrightarrow{a_k} r_k), S_k$$

is defined by the sequence of the multisets of its action labels, i.e.:

$$a_1; a_2; \ldots; a_k$$

Semantics of a rule

Two parts:

- State transition
- Trace event

Semantics of a rule

Two parts:

- State transition
- Trace event

Example (Transition example)

$$[\mathrm{St\_I\_2}(A, 17, k), \mathrm{In}(m)] \xrightarrow{\mathrm{Recv}(A,m)} [\mathrm{St\_I\_3}(A, 17, k, m)]$$

Agent state changes, and In fact is consumed, while Recv action is added to trace.

References

- John Clark and Jeremy Jacob. *A survey of authentication protocol literature*, 1997. Available at http://www.cs.york.ac.uk/ jac/
- Gavin Lowe. *A hierarchy of authentication specifications*. In Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97), pages 31–43. IEEE CS Press, 1997.
- Benedikt Schmidt. *Formal analysis of key exchange protocols and physical protocols*. Ph.D. thesis, ETH Zürich, 2012. http://dx.doi.org/10.3929/ethz-a-009898924
- Peter Ryan, Steve Schneider, Michael Goldsmith, Gawin Lowe, and Bill Roscoe. *Modelling and Analysis of Security Protocols*, Addison-Wesley, 2000.

Explicit vs. Implicit Destructors

Implicit Destructor Rules (no destruction operation)

$$\frac{\langle m_1, m_2\rangle \in \mathcal{DY}(M)}{m_i \in \mathcal{DY}(M)}\ \text{Proj}_i \qquad \frac{\{\!|m|\!\}_k \in \mathcal{DY}(M) \quad k \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{DecSym}$$

$$\frac{\{m\}_k \in \mathcal{DY}(M) \quad (k)^{-1} \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{DecAsym} \qquad \frac{\{m\}_{(k)^{-1}} \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{OpenSig}$$

versus

Explicit Destructors with algebraic properties

$$\begin{array}{rcl@{\qquad}rcl} \pi_1(\langle m_1, m_2\rangle) & \approx & m_1 & \{\{m\}_k\}_{(k)^{-1}} & \approx & m \\ \pi_2(\langle m_1, m_2\rangle) & \approx & m_2 & \text{open}(\{m\}_{(k)^{-1}}) & \approx & m \\ \{\!|\{\!|m|\!\}_k|\!\}_k & \approx & m & & & \end{array}$$

- Implicit destructor rules are redundant with these properties
- Explicit has strictly more derivable messages
- Considerably more difficult to handle