Ecole JEUNES CHERCHEURS
en PROGRAMMATION

Juin 2007

Vérification des Protocoles Cryptographiques et de leurs Implémentations

Cédric Fournet
Microsoft Research, Cambridge & MSR—INRIA, Orsay
http://research.microsoft.com/~fournet

avec Martin Abadi, Pedro Adao, Karthik Bhargavan, Bruno Blanchet, Ricardo Corin, Andrew D. Gordon, Tamara Rezk, Stephen Tse

Ecole JEUNES CHERCHEURS
en PROGRAMMATION

Juin 2007

Vérification des Protocoles Cryptographiques et de leurs Implémentations

1. Modélisation des protocoles en pi calcul
2. Comment vérifier l'usage des protocoles? application aux services Web (outils, demo)
3. Comment vérifier leurs implémentations? (demo)
4. Cryptographie formelle/concrete

Some pi calculus

Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, parts I and II. *Information and Computation*, vol. 100, 1992.

Robin Milner. Communicating and Mobile Systems: The Pi-Calculus. Cambridge University Press, 1999.

Davide Sangiorgi and David Walker. The Pi-Calculus: a Theory of Mobile Processes Cambridge University Press, 2001.

Pi calculus: syntax

$u, v, w ::=$	**Terms**
$a, b, c, \ldots$	channel names
$x, y, z, \ldots$	channel variables
$P, Q, R ::=$	**Processes**
$\mathbf{0}$	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu a.P$	name restriction ("new", binding a)
$u(x).P$	message input (binding x)
$\overline{u}\langle v\rangle.P$	message output

Names can be dynamically created and communicated

Pi calculus: examples

$P = \overline{a}\langle b \rangle$

$Q = a(x).\overline{c}\langle x \rangle$ $\quad P \mid Q \rightarrow \overline{c}\langle b \rangle$

by communication on channel a

Pi calculus: semantics

$$\text{Comm}\quad \overline{a}\langle v\rangle.P \mid a(x).Q \to P \mid Q\{^{v}/_{x}\}$$

$$\text{Ctx}\ \frac{P \to P'}{E[P] \to E[P']} \qquad \text{Struct}\ \frac{P \equiv Q \quad Q \to Q' \quad Q' \equiv P'}{P \to P'}$$

We rely on structural equivalence
to rearrange processes before reduction:

New-$\mathbf{0}$	$\nu a.\mathbf{0}$	$\equiv$	$\mathbf{0}$	
New-C	$\nu a.\nu b.P$	$\equiv$	$\nu b.\nu a.P$	
New-Par	$P \mid \nu a.Q$	$\equiv$	$\nu a.(P \mid Q)$	when $a \notin fn(P)$
Par-$\mathbf{0}$	P	$\equiv$	$P \mid \mathbf{0}$	
Par-A	$P \mid (Q \mid R)$	$\equiv$	$(P \mid Q) \mid R$	
Par-C	$P \mid Q$	$\equiv$	$Q \mid P$	
Repl	$!P$	$\equiv$	$P \mid !P$	

$$\text{Ctx}\ \frac{P \equiv P'}{E[P] \equiv E[P']}$$

A protocol for sending *n* names

We code n-ary communication on channel a.

$$\mathit{Send}\ v_1 \ldots v_n = \nu c.\overline{a}\langle c\rangle.\overline{c}\langle v_1\rangle.\ldots.\overline{c}\langle v_n\rangle$$
$$\mathit{Recv}\ x_1 \ldots x_n = a(c).c(x).c(x_1).\ldots.c(x_n)$$

$$\mathit{Send}\ v_1 v_2 \mid \mathit{Recv}\ x_1 x_2.T \longrightarrow\longrightarrow\longrightarrow T\{{}^{v_1}/_{x_1}, {}^{v_2}/_{x_2}\}$$

We use a private channel c for each run of the protocol.

$$\mathit{Send}\ u_1 u_2 \mid \mathit{Send}\ v_1 v_2 \mid \mathit{Recv}\ x_1 x_2.T$$
$$\longrightarrow\longrightarrow\longrightarrow \mathit{Send}\ u_1 u_2 \mid T\{{}^{v_1}/_{x_1}, {}^{v_2}/_{x_2}\}$$
$$\text{or}\quad \longrightarrow\longrightarrow\longrightarrow \mathit{Send}\ v_1 v_2 \mid T\{{}^{u_1}/_{x_1}, {}^{u_2}/_{x_2}\}$$

The pi calculus (review)

- The pi calculus is a core language of concurrent processes
- Mobile names naturally represent dynamic capabilities, such as communication capabilities.
- Mobility for fresh names is very expressive.
  - Dynamic configuration
  - Local encodings
- There is a nice theory of observational equivalences in the pi calculus
  - When are two processes equivalent in all contexts?
  - When is an encoding correct?
  - How to prove such equivalences?
  - Can we use labelled transitions instead of reductions?

The applied pi calculus

M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, Jan. 1999.

M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, 2001.

Security in the pi calculus ?

- Domain: security protocols,
  with interactions between cryptographic computations, controlled usage of secrets, and communications.

- Process calculi are useful for such protocols, e.g.,
  - pi calculus, to reason on high-level security properties.
  - spi calculus [Abadi&Gordon], to tackle some cryptography.

- Still, there is a gap between typical security specifications (e.g. RFCs) and what can be represented in those calculi.

An applied pi calculus

Can we get a robust & uniform extension
of the pi calculus, and still use our favourite tools?

- Parameterise the pi calculus with computations on values.
- Keep communications and scopes!
- Uniformly develop equivalences and proof techniques.
  - Contexts representing active attackers
  - Observational equivalences
  - Automated verifiers (Bruno Blanchet's ProVerif)

Syntax for processes

$P, Q, R ::=$		**Processes**
	$\mathbf{0}$	null process
	$P \mid Q$	parallel composition
	$!P$	replication
	$\nu n.P$	name restriction ("new")
	$\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q$	conditional
	$u(x).P$	message input
	$\overline{u}\langle N \rangle.P$	message output

Processes are those of the plain pi calculus.

Communicated values are terms, rather than names.

The calculus is parameterized by an equational theory for terms.

Syntax for terms

$M, N ::=$	**Terms**
$a, b, c, \ldots, k, \ldots, m, n, \ldots, s$	name
x, y, z	variable
$f(M_1, \ldots, M_l)$	function application

We assume given:

- a signature: a set of function symbols with an arity;
- a sort system;
- an equational theory:
  - an equivalence relation ($=$) on terms;
  - closed by substitutions of terms for variables;
  - closed by one-to-one substitutions on names.

We distinguish three similar notions: constants, names, variables.

Example: pairs

- A constructor function "cons", written (M,N)
- Two selector functions, written fst(M) and snd(M)
- The equations

$$\begin{aligned} \mathsf{fst}((x,y)) &= x \\ \mathsf{snd}((x,y)) &= y \end{aligned}$$

+ all equations obtained by reflexivity, symmetry, transitivity, and substitutions.

We can now directly communicate pairs of terms:

$$\begin{aligned} A &= \overline{a}\langle (M_1, M_2) \rangle \\ B &= a(x).T\{^{\mathsf{fst}(x)}/_{x_1}, ^{\mathsf{snd}(x)}/_{x_2}\} \end{aligned}$$

Similarly, we can model tuples, arrays, lists, …

Shared-key cryptography

- To model shared-key cryptography,
  we can use two binary functions related with:

$$\mathsf{dec}(\mathsf{enc}(x, y), y) \quad = \quad x$$

- We can use restricted names as keys (or not)
- This is much as the spi calculus.

For each variant of the spi calculus, one can select an equational theory that yields an applied pi calculus with the same reductions.

Operational semantics

We use a standard chemical-style semantics:

- reduction step ($\rightarrow$) contains the rules

Comm $\quad \overline{a}\langle M\rangle.P \mid a(x).Q \;\rightarrow\; P \mid Q[^{M}/_{x}]$

Then $\quad \mathit{if}\ M = M\ \mathit{then}\ P\ \mathit{else}\ Q \;\rightarrow\; P$

Else $\quad \mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q \;\rightarrow\; Q$

when $M = N$ not in the theory and $fv(M, N) = \emptyset$

closed by structural equivalence
& application of evaluation contexts.

- structural equivalence ($\equiv$) is defined as usual, and also closed by equality on terms.

Token-based authentication

$$A = \overline{a}\langle(M, s)\rangle$$
$$B = a(x).\mathit{if}\ \mathsf{snd}(x) = s\ \mathit{then}\ \overline{b}\langle\mathsf{fst}(x)\rangle$$

- The name s in the pair acts as a capability for the forwarding.
- Expected behaviour:

$$\nu s.\,(A \mid B) \quad \rightarrow\rightarrow \quad \overline{b}\langle M\rangle$$

using the equations
$$\mathsf{fst}((x, y)) = x$$
$$\mathsf{snd}((x, y)) = y$$

Token-based authentication ?

$$
\begin{aligned}
A &= \overline{a}\langle(M, s)\rangle \\
B &= a(x).\mathit{if}\ \mathsf{snd}(x) = s\ \mathit{then}\ \overline{b}\langle\mathsf{fst}(x)\rangle \\
I &= a(x).\overline{a}\langle(N, \mathsf{snd}(x))\rangle
\end{aligned}
$$

- The name s in the pair acts as a capability for the forwarding.
- Expected behaviour:

$$\nu s.\,(A \mid B) \quad \rightarrow\rightarrow \quad \overline{b}\langle M\rangle$$

- The token is not protected; we can represent an (obvious) interception attack as the context I:

$$I \mid \nu s.\,(A \mid B) \quad \rightarrow\rightarrow\rightarrow \quad \overline{b}\langle N\rangle$$

Cryptographic hash

- A one-way, collision-free hash function is modelled as a constructor "h" with no equation.
- Example: message authentication code (MAC)

$$A = \overline{a}\langle(M, \boxed{\mathsf{h}(s, M)})\rangle$$
$$B = a(x).\mathit{if}\ \mathsf{h}(s, \mathsf{fst}(x)) = \mathsf{snd}(x)\ \mathit{then}\ \overline{b}\langle\mathsf{fst}(x)\rangle$$

$$\nu s.(A \mid B) \rightarrow\rightarrow \overline{b}\langle M\rangle$$

  - A sends a hash code that depends on the secret. (The secret is not communicated.)
  - B checks the authenticity of the received message by recomputing its hash code.
  - Attackers cannot produce another valid hash code.

Scope restriction for terms

- In the plain pi calculus,
  - new restricted names can be created ("fresh values");
  - scope restrictions nicely disappear when those names are passed to the environment ("scope extrusion").

$$\nu s.\left(\overline{a}\langle s\rangle \mid B\right) \xrightarrow{\overline{a}(s)} B$$

Scope restriction for terms

- In the plain pi calculus,

$$\nu s.\left(\overline{a}\langle s\rangle \mid B\right) \xrightarrow{\overline{a}(s)} B$$

- With terms instead of names,
  scope restriction gets more interesting:
  - How to represent the result of sending an opaque term?

$$\nu s.\left(\overline{a}\langle \mathsf{h}(s, M)\rangle \mid B\right) \xrightarrow{\overline{a}\,?} \begin{array}{l} B\ ? \\ \nu s.B\ ?? \end{array}$$

  - The environment can accumulate partial knowledge on restricted names, and use it later.
  - The problem already occurs in the spi calculus, when sending messages encrypted with a restricted key. [Abadi Gordon, Boreale deNicola Pugliese]

Scope restriction for terms

- In the plain pi calculus,

$$\nu s.\left(\overline{a}\langle s\rangle \mid B\right) \xrightarrow{\overline{a}(s)} B$$

- With terms instead of names,
  scope restriction gets more interesting:
  - How to represent the result of sending an opaque term?

$$\nu s.\left(\overline{a}\langle \mathsf{h}(s, M)\rangle \mid B\right) \xrightarrow{\overline{a}(x)} \nu s.\left(\{^{\mathsf{h}(s,M)}/_{x}\} \mid B\right)$$

  - We extend processes with active substitutions that keep track of the values passed to the environment.

Substitutions as processes

$A, B, C ::=$	**Extended processes**
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	**active substitution**

- Active substitutions map distinct variables to terms
  - They may appear under restrictions (not under guards)
  - They operate on the environment.
  - They represent terms passed to the environment "by reference", much as a floating $\mathbf{let}\ x = M \quad \mathbf{in} \ldots$

(There are well-formed conditions for extended processes.)

Operational semantics

- Structural equivalence $\equiv$ is extended with rules for active substitutions (reduction is defined as before).

Subst	$\{^M/_x\} \mid A$	$\equiv$	$\{^M/_x\} \mid A[^M/_x]$	
Alias	$\nu x.\{^M/_x\}$	$\equiv$	$\mathbf{0}$	
Rewrite	$\{^M/_x\}$	$\equiv$	$\{^N/_x\}$	when $M = N$

Par-$\mathbf{0}$	A	$\equiv$	$A \mid \mathbf{0}$
Par-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$
Par-C	$A \mid B$	$\equiv$	$B \mid A$
Repl	$!P$	$\equiv$	$P \mid !P$
New-$\mathbf{0}$	$\nu n.\mathbf{0}$	$\equiv$	$\mathbf{0}$
New-C	$\nu u.\nu v.A$	$\equiv$	$\nu v.\nu u.A$
New-Par	$A \mid \nu u.B$	$\equiv$	$\nu u.(A \mid B)$ when $u \notin fv(A) \cup fn(A)$

Substitutions as processes (2)

- Every closed extended process can be put in a normal form that separates its static and dynamic parts

$$A \equiv \nu\widetilde{n}.(\{\widetilde{M}/\widetilde{x}\} \mid P) \quad \text{where} \quad \begin{array}{l} fv(P) = \emptyset \\ fv(\widetilde{M}) = \emptyset \\ \{\widetilde{n}\} \subseteq fn(\widetilde{M}) \end{array}$$

  - The static part operates only on the environment
  - The dynamic part P is an ordinary process that describes communications
  - These two parts can share some restricted names

(However, "flattening" processes is not necessarily a good idea.)

Cryptographic hash, again

- Using active substitutions, we can represent a process that has MACed several messages using the secret s:

$$\nu s. \left(\begin{array}{l} \left\{ {}^{(M,\mathsf{h}(s,M))}/_{x} \right\} \mid \left\{ {}^{(N,\mathsf{h}(s,N))}/_{y} \right\} \mid \ldots \mid \\ a(x).\mathit{if}\ \mathsf{h}(s,\mathsf{fst}(x)) = \mathsf{snd}(x)\ \mathit{then}\ \overline{b}\langle \mathsf{fst}(x) \rangle \end{array} \right)$$

- What an attacker can effectively do with x and y depends on the equational theory being considered.

More encryption primitives

- To model shared-key cryptography,
  we used two binary functions related with:

$$\mathsf{dec}(\mathsf{enc}(x, y), y) \quad = \quad x$$

- There are many variants of encryption primitives,
  with diverse properties
  - Symmetric or not?
  - Detection of decryption errors?
  - Which-key concealing?

- We can select equations accordingly

Asymmetric encryption

- To model public-key cryptography,
  we generate public- and private-keys from a seed:

$$\mathsf{dec}(\mathsf{enc}(x, \mathsf{pk}(y)), \mathsf{sk}(y)) \quad = \quad x$$

- Using active substitutions, we can write a process that exports the public key and keeps the private key secret:

$$\nu s.\left(\{^{\mathsf{pk}(s)}/_{pk}\} \mid a(x).\overline{b}\langle \mathsf{dec}(x, \mathsf{sk}(s))\rangle\right)$$

- We can add "troublesome" equations for security protocols, for instance reflecting a typical weakness of RSA encryption:

$$\mathsf{dec}(\mathsf{enc}(x, y), z) = \mathsf{enc}(\mathsf{dec}(x, z), y)$$

Non-deterministic encryption

- To model probabilistic cryptography,
  we may add a third argument to the encryption function:

$$\mathsf{dec}(\mathsf{enc}(x, \mathsf{pk}(y), z), \mathsf{sk}(y)) \quad = \quad x$$

- With this variant, consider the protocol:

$$\begin{aligned} & \left(\nu m.\overline{b}\langle \mathsf{enc}(M, pk, m)\rangle\right) \\ \mid \quad & \left(\nu n.\overline{c}\langle \mathsf{enc}(N, pk, n)\rangle\right) \\ \mid \quad & \nu s.\left(\{^{\mathsf{pk}(s)}/_{pk}\} \mid R\right) \end{aligned}$$

Without access to the decryption key, an attacker cannot detect whether the underlying plaintexts are identical

Observational Equivalence

How to compare applied pi processes?

Contexts and Barbs

- Evaluation contexts are environments for running processes

$E[\_] ::=$	**Evaluation contexts**
$[\_]$	placeholder
$A \mid E[\_]$	parallel composition
$\nu n.E[\_]$	name restriction
$\nu x.E[\_]$	variable restriction

  They may contain processes, active substitutions…
  We will use them to represent classes of attackers

- Our basic observation predicate, or barb, tests whether the process A can send a message on the free channel a.

$$A \Downarrow_a \quad \text{when} \quad A \equiv E[\overline{a}\langle M \rangle]$$
$$\text{and} \quad E[\_] \text{ does not bind } a$$

Observational equivalence

- Observational equivalence ($\approx$) is the largest symmetric relation between closed extended processes defining the same variables such that A $\approx$ B implies:
  1. if $A \Downarrow_a$, then $B \Downarrow_a$
  2. if $A \rightarrow^* A'$ then $B \rightarrow^* B'$ and $A' \approx B'$
  3. for all evaluation contexts $E[\_]$, we have $E[A] \approx E[B]$

- Examples (in plain pi calculus)

$$\overline{a_1}\langle\rangle \not\approx \overline{a_2}\langle\rangle \qquad \nu a.\overline{a}\langle M\rangle \approx \mathbf{0}$$

$$\overline{b}\langle a_1\rangle \not\approx \overline{b}\langle a_2\rangle \qquad \nu a.(\overline{a}\langle\rangle \mid a(x).P) \approx \nu a.P$$

- How to prove observational equivalence?

Secrecy by equivalence

- With symmetric encryption,
  consider the simplistic protocol

$$A \stackrel{\text{def}}{=} \nu s.\overline{b}\langle \mathsf{enc}(M, s)\rangle$$

The attacker observes a fresh, opaque message, apparently unrelated to the term M

$$A \approx \nu n.\overline{b}\langle n\rangle$$

This second process is simpler & more abstract

Secrecy by equivalence (2)

- With asymmetric encryption, this doesn't work!

$$A \stackrel{\text{def}}{=} \nu s.\{^{\mathsf{pk}(s)}/_{pk}\} \mid \overline{b}\langle \mathsf{enc}(M, pk)\rangle$$

$$A \not\approx \nu s.\{^{\mathsf{pk}(s)}/_{pk}\} \mid \nu n.\overline{b}\langle n\rangle$$

$$I_M \stackrel{\text{def}}{=} b(x).\mathit{if}\ x = \mathsf{enc}(M, pk)\ \mathit{then}\ \overline{a}\langle\rangle$$

The attacker can guess the term M, then verify it

If M is a "weak secret", such as a password, then this inequation reflects a dictionary attack

Secrecy by equivalence (3)

- With non-deterministic encryption,
  we do have strong secrecy properties, e.g.

$$A \stackrel{\text{def}}{=} \nu s.\{^{\mathsf{pk}(s)}/_{pk}\} \mid \nu m.\{^{\mathsf{enc}(M,pk,m)}/_{x}\}$$

$$A \approx \nu n.\{^{n}/_{pk}\} \mid \nu m.\{^{m}/_{x}\}$$

The attacker observes two unrelated fresh values

The attacker learns nothing on M ,
and cannot detect even that x is an encryption

Equivalence for frames ?

- Frames are extended processes that only consist of active substitutions and restrictions.
  What is observational equivalence for frames?

- Consider two functions f and g, no equations, and frames:

$$
\begin{array}{lcll}
\varphi_0 & = & \nu k\, \nu s. & \{{}^{s}/_{x}\} \mid \{{}^{\mathrm{f}(k)}/_{y}\} \\
\varphi_1 & = & \nu k. & \{{}^{\mathrm{g}(k)}/_{x}\} \mid \{{}^{\mathrm{f}(k)}/_{y}\} \\
\varphi_2 & = & \nu k. & \{{}^{k}/_{x}\} \mid \{{}^{\mathrm{f}(k)}/_{y}\}
\end{array}
$$

φ_0 and φ_1 have the same observable behaviour: they provide two fresh, apparently independent values

φ_2 is visibly different: we have $y = \mathrm{f}(x)$ with φ_2 only

Static equivalence (definition)

- We write $(M = N)\varphi$ when the terms M and N are equal in the theory after alpha-conversion and substitution.

$$(M = N)\sigma \text{ when } \exists \tilde{n}, \sigma. \begin{cases} \varphi \equiv \nu \tilde{n}.\sigma \\ M\sigma = N\sigma \\ \{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset \end{cases}$$

- Two frames are statically equivalent when they agree on all term comparisons:

$$\varphi \approx_s \psi \text{ when } \forall M, N.(M = N)\varphi \text{ iff } (M = N)\psi$$

Two extended processes are statically equivalent when their frames are equivalent.

Static equivalence (properties)

- Static equivalence is closed by $\equiv$, $\rightarrow$, E[_].

- For extended processes,
  observational equivalence is finer than static equivalence.

- For frames,
  static equivalence and observational equivalence coincide.

  Hence, we can uniformly lift equational properties
  from (restricted) terms to (extended) processes.

We use special evaluation contexts instead of frame comparisons:

$\big((\mathit{if}\ \ M = N\ \ \mathit{then}\ \overline{a}\langle s\rangle) \mid A\big) \Downarrow a$ iff $(M = N)\ \varphi(A)$

Labelled semantics

- Can we characterize observational semantics using labelled transitions?
  - A good technical test for the calculus
  - Standard, effective proof techniques
    - No quantification over all contexts.
    - Proofs "up to active substitutions"

- We have two such labelled semantics that refine static equivalence.

- Theorem: for any equational theory, the labelled and observational semantics coincide.

  However, the generalization of the pi calculus LTS with scope extrusion (exporting terms instead of names) yields a labelled semantics that "sees through" all term constructors and discriminates too much.

A labelled semantics

In addition to $\rightarrow$ and $\equiv$, we use the rules

In $$a(x).P \xrightarrow{a(M)} P\{^M/_x\}$$

Out-Atom $$\overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} P$$

Open-Atom $$\frac{A \xrightarrow{\overline{a}\langle u\rangle} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\overline{a}\langle u\rangle} A'}$$

Scope $$\frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

Par $$\frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

Struct $$\frac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

Example transitions

- Labelled transitions systematically pass values by aliasing them to fresh variables
- The environment can use these values indirectly, by forming terms that contain these variables

$$\nu k.\ \overline{a}\langle \mathsf{enc}(M,k)\rangle.\overline{a}\langle k\rangle.a(z).\mathit{if}\ \ z = M\ \ \mathit{then}\ \overline{c}\langle\, \mathsf{oops}\,!\,\rangle$$

$$\xrightarrow{\nu x.\overline{a}\langle x\rangle}\quad \nu k.\Big(\{^{\mathsf{enc}(M,k)}/_{x}\} \mid \overline{a}\langle k\rangle.a(z).\mathit{if}\ \ z = M\ \ \mathit{then}\ \overline{c}\langle\, \mathsf{oops}\,!\,\rangle\Big)$$

$$\xrightarrow{\nu y.\overline{a}\langle y\rangle}\quad \nu k.\Big(\{^{\mathsf{enc}(M,k)}/_{x}\} \mid \{^{k}/_{y}\} \mid a(z).\mathit{if}\ \ z = M\ \ \mathit{then}\ \overline{c}\langle\, \mathsf{oops}\,!\,\rangle\Big)$$

$$\xrightarrow{a(\mathsf{dec}(x,y))}\quad \nu k.\Big(\{^{\mathsf{enc}(M,k)}/_{x}\} \mid \{^{k}/_{y}\} \mid \mathit{if}\ \ \mathsf{dec}(x,y) = M\ \ \mathit{then}\ \overline{c}\langle\, \mathsf{oops}\,!\,\rangle\Big)$$

$$\rightarrow\quad \nu k.\Big(\{^{\mathsf{enc}(M,k)}/_{x}\} \mid \{^{k}/_{y}\}\Big) \mid \overline{c}\langle\, \mathsf{oops}\,!\,\rangle$$

Labelled bisimilarity

- Labelled bisimilarity ($\approx_l$) is defined almost as usual: the largest symmetric relation such that $A \approx_l B$ implies
  1. $A \approx_s B$
  2. if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \approx_l B'$ for some B';
  3. if $A \xrightarrow{\alpha} A'$ and α has free variables in dom(A), and α has no bound names that are free in B, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A' \approx_l B'$ for some B'.

- Labelled bisimilarity is observational equivalence: $\approx_l = \approx$
- Labelled bisimilarity has nice technical properties (e.g. proofs up to frame simplification).

Symbolic bisimulations

- Labelled bisimulations make proofs easier by dealing abstractly with message outputs (active substitutions)

- Message inputs may also be troublesome:
  - The environment can supply arbitrary terms (infinite-branching transition system)
  - There is an infinite number of names
  - There is no bound on the nesting of functions in terms

  In contrast, many different terms are uniformly handled by security protocols (few tests)

- Symbolic transitions (and symbolic bisimulations) use abstract "environment" variables for inputs [Huimin & Hennessy; Boreale]

Symbolic bisimulations (example)

$$P \stackrel{\text{def}}{=} \mathit{if}\ x = \mathsf{mac}(k, M)\ \mathit{then}\ Q$$

$$a(x).P \xrightarrow{a(\underline{x})} P$$

$$\rightarrow \underline{x = \mathsf{mac}(k, M)} \mid Q$$

- Symbolic transitions (and symbolic bisimulations) use abstract "environment" variables for inputs
- Symbolic reductions introduce constraints on those variables.
  - Equality between open terms
  - Occur-checks on output variables (no causality loop)
- Constraints must be solvable to obtain concrete reductions.

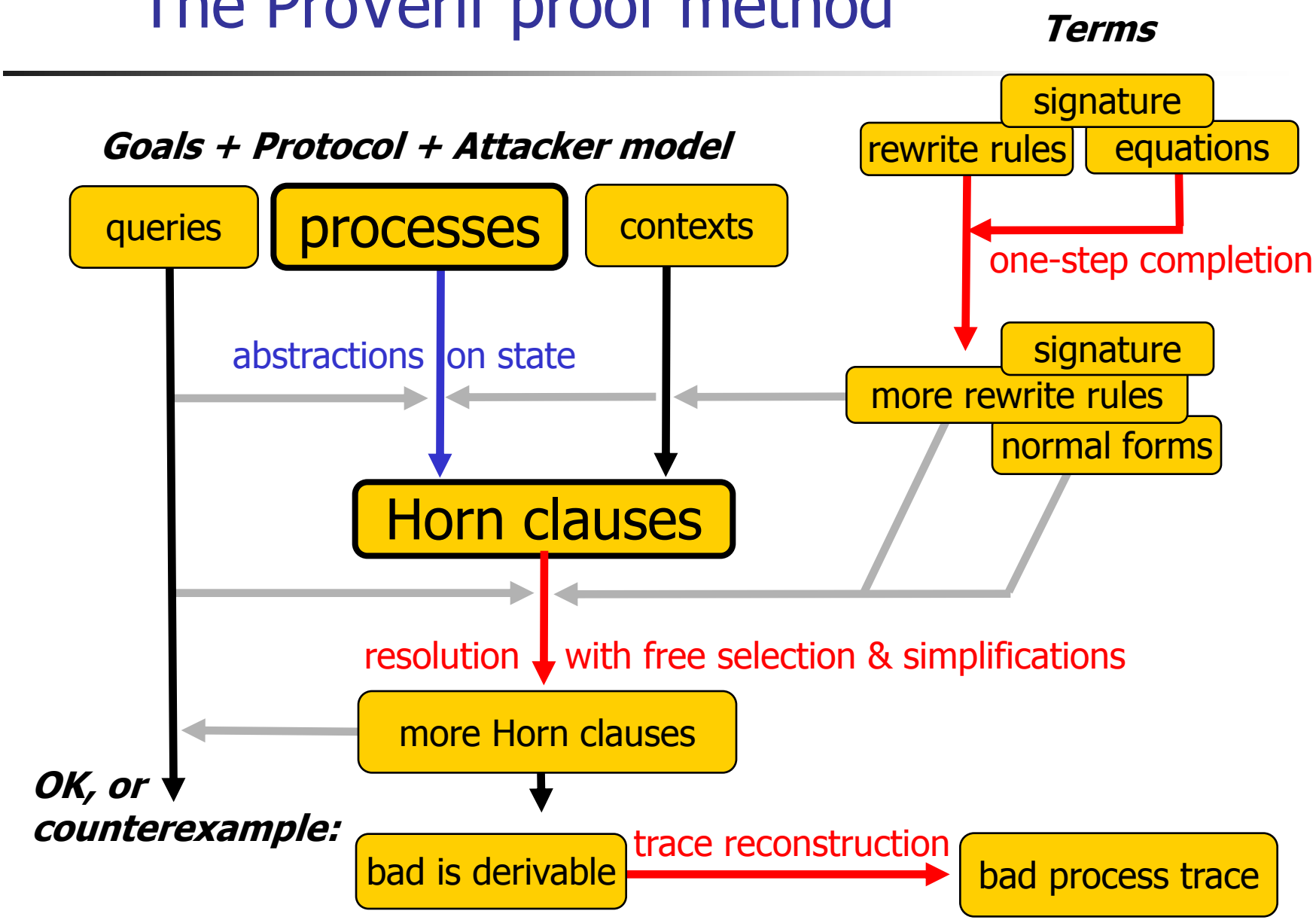
ProVerif: an automated protocol verifier for the applied pi calculus

B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW-14*, pages 82–96, June 2001.

M. Abadi and B. Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. *Journal of the ACM*, 52(1):102–146, Jan. 2005.

B. Blanchet and M. Abadi and C. Fournet Automated Verification of Selected Equivalences for Security Protocols. In *LICS 2005*.

The ProVerif proof method



Diffie-Hellman key exchange

W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

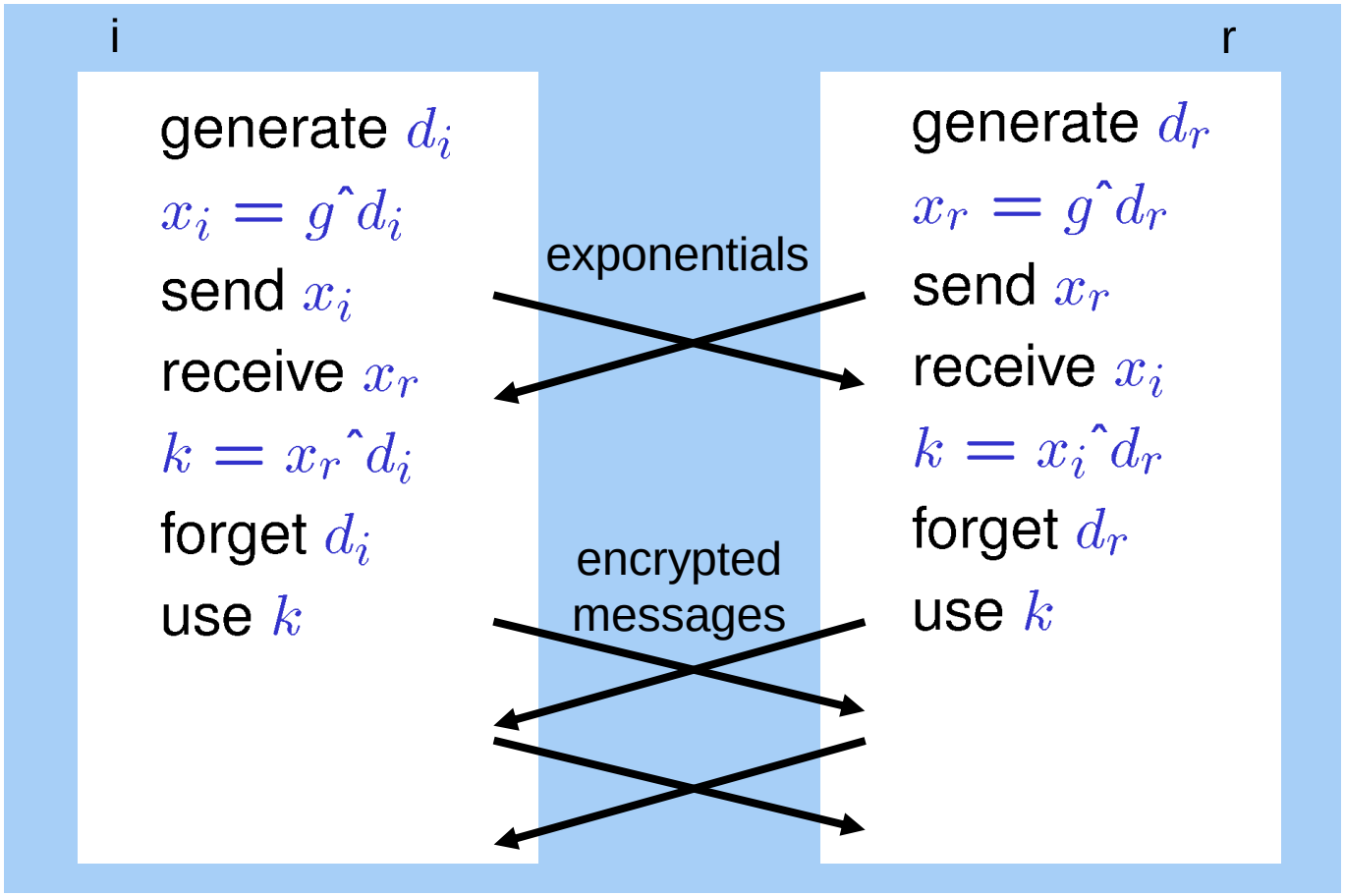
Diffie-Hellman

- A cryptographic protocol for creating a shared secret between two parties, e.g. establishing a session key.
- The two parties communicate over a public network, in the presence of a passive attacker
- The protocol relies on large exponentials, with the commutative equation:

$$(g\hat{}x)\hat{}y \;=\; (g\hat{}y)\hat{}x \quad \text{mod } p$$

Can't extract x from $g\hat{}x$

Diffie-Hellman exchange



We get "perfect forward secrecy":
the values x_i, x_r, k seem unrelated

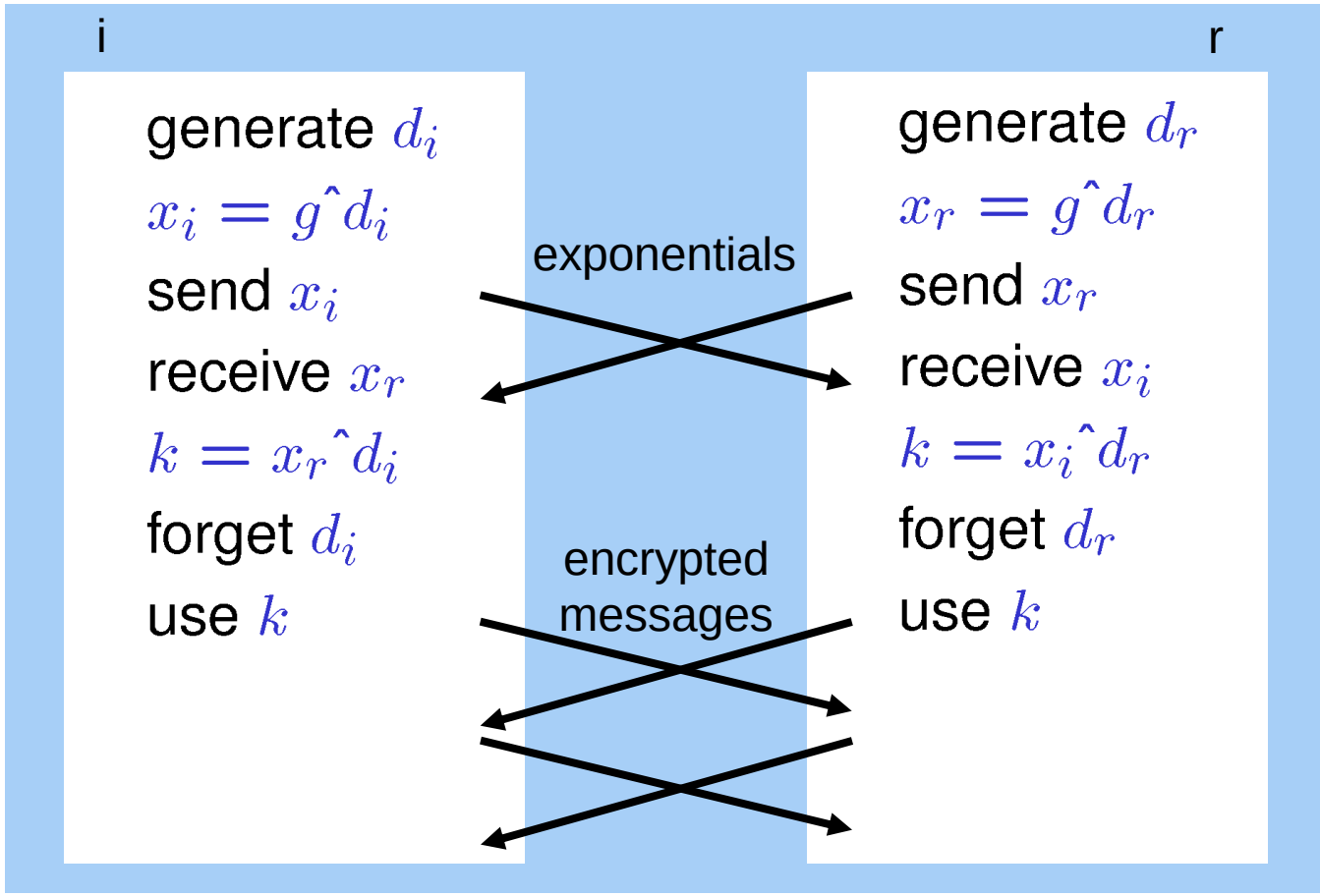
Diffie-Hellman in applied pi

$$A_i \stackrel{\text{def}}{=} \nu d_i.$$

$$\overline{c_i}\langle g\hat{\ }d_i\rangle.$$

$$c_r(x_r).$$

$$P_i\{{}^{x_r\hat{\ }d_i}/_{k}\}$$



A_r is defined symmetrically

Diffie-Hellman in applied pi

- Processes A_i, A_r represent the initial state.
- Processes P_i, P_r represent the final state with free variable z for the shared key.

$$A_i \stackrel{\text{def}}{=} \nu d_i.\left(\overline{c_i}\langle x_i \sigma_i \rangle \mid c_r(x_r).P_i \phi_i\right)$$

- Auxiliary substitutions account for the messages x_i being exchanged and the shared key z.

$$\begin{aligned} \sigma_i &= \{{}^{g\hat{}d_i}/_{x_i}\} \\ \phi_i &= \{{}^{x_r\hat{}d_i}/_{z}\} \end{aligned}$$

Diffie-Hellman in applied pi

- A normal run consists of two reduction steps:

$$A_i \mid A_r \quad \rightarrow\rightarrow \quad \nu z.(P_i \mid P_r \mid \nu x_i, x_r.\ \varphi)$$

- A passive attacker intercepts both messages and forwards those messages unchanged, leading to the final state:

$$\nu z.(P_i \mid P_r \mid \ \varphi).$$

- We used an auxiliary frame to record messages and computations:

$$\varphi \quad \stackrel{\text{def}}{=} \quad (\nu d_i.\ (\phi_i \mid \sigma_i)) \mid (\nu d_r.\ \sigma_r)$$

A correctness property

- Specification:
  1. The final processes share a "pure secret" = a fresh name

$$\nu k.(P_i \mid P_r)\{^k/_z\}$$

  2. Intercepted messages are "pure noise" = fresh names

$$\nu s_i.\{^{s_i}/_{x_i}\} \mid \nu s_r.\{^{s_r}/_{x_r}\}$$

- Theorem:

$$\nu z.(P_i \mid P_r \mid \varphi)$$

$$\approx \quad \nu k.(P_i \mid P_r)\{^k/_z\} \quad \mid \quad \nu s_i.\{^{s_i}/_{x_i}\} \mid \nu s_r.\{^{s_r}/_{x_r}\}$$

Perfect forward secrecy

$$\nu z.(P_i \mid P_r \mid \varphi)$$
$$\approx \quad \nu k.(P_i \mid P_r)\{^k/_z\} \quad \mid \quad \nu s_i.\{^{s_i}/_{x_i}\} \mid \nu s_r.\{^{s_r}/_{x_r}\}$$

- We can forget about the key establishment protocol: the key freshness & secrecy do not depend on its use
- Examples:
  - Send a first message
    $$P_i = \overline{a}\langle\{\text{Toulouse}\}_z\rangle$$
    $$P_r = a(x).Q'$$
  - Reveal the key to the environment
    $$P_i = \overline{a}\langle\{\text{Toulouse}\}_z\rangle$$
    $$P_r = \overline{c}\langle z\rangle$$

Summary on applied pi

- We use a pi calculus parameterized by an equational theory for terms.

- We obtain an expressive and flexible framework for reasoning on security protocols, which typically mix:
  - creations of "fresh" values : "new" & scope extrusions
  - various cryptographic operations : various equational theories
  - communications : pi calculus

- We uniformly build tools to state and prove their properties

Many related works

- Complexity-theoretical analyses, focusing on the cryptographic operations.
- Higher-level presentations with black box cryptography, focusing on their usage in protocols.
  - Logics
  - Term rewriting
  - Strand spaces [Guttman et al.]
  - Process calculi
    - CSP [Lowe], CCS [DeNicola et al.]
    - Pi calculi
      - the spi calculus [Abadi & Gordon]
      - Specific type systems for security
        - Information control flow [Honda]
        - Syntactic containment [Abadi, Blanchet]
        - Correspondence assertions [Gordon, Jeffrey]
        - Authorization [Fournet, Gordon, Maffei].

…