A decision procedure for proving observational equivalence (Work in progress)

Vincent Cheval
LSV, ENS Cachan

Hubert Comon-Lundh
AIST, Tokyo
and INRIA Saclay

Stéphanie Delaune
LSV, ENS Cachan

June 11, 2009

1 Introduction

We need to increase our confidence in security protocols. One of the important directions towards this goal is the formal analysis of protocol specifications. Elaborating strategies and algorithms to build such formal security proofs has been the subject of active research in the past two decades.

Most existing results focus on trace properties like secrecy (expressed as a reachability property) or authentication. There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties and require the notion of observational equivalence. Typical examples are anonymity, privacy related properties or statements closer to security properties used in cryptography.

B. Blanchet, M. Abadi and C. Fournet in [2] show how an approximation of observational equivalence (the so-called "diff-equivalence") can be simulated using bi-processes and included in the automatic tool ProVerif. This allows for instance to get observational equivalence proofs for an unbounded number of sessions and for (some) equational theories.

Another approach consists in bounding the number of sessions (copies of each process) and trying to find decision procedures for the equivalence. In case an attack is found, this is really an attack (which is not always the case in ProVerif). On the other hand, when no attack is found, we only get a security proof for a fixed number of sessions. An advantage of this approach is also the termination of the algorithm. Along this line of research, the most significant result is due to M. Baudet [1]. It is shown there that the diff-equivalence is decidable for a bounded number of sessions and any attacker theory that can be represented by a subterm convergent rewriting system. Furthermore, in a recent paper [5], it is shown that, for finite and deterministic processes without else branches, the observational equivalence of two processes can be reduced to the diff-equivalence of finitely many pairs of processes. Altogether, we get a decision procedure for observational equivalence, for this class of processes.

It is not clear how the results of [1, 5] could be extended to processes that contain else branches or to other attacker's theories. Moreover, the proof of M. Baudet is very complicated. The goal of the present work is to revisit Baudet's result and provide with another decision algorithm that would be hopefully more amenable to extensions and implementation.

Even in the case of a bounded number of sessions, there are infinitely many possible process executions, because of the unbounded possible attacker's actions. It is however possible

to represent all possible traces using a symbolic representation thanks to *deducibility constraints*, that have been introduced in [6]. The observational equivalence of finite deterministic processes can be reduced to the symbolic equivalence of the traces that are represented by deducibility constraints [5] (extended with disequalities in case of negative tests).

The decidability of deducibility constraints is proved to be in NP in [7] for a standard attacker theory. But the existence of a solution is not what we need for the decision of equivalence. Another recent work [4] shows how it is possible to simplify the deducibility constraints into solved forms *without loosing any solution.* In other words, this shows how to get a simplified representation of the set of possible traces. This is used in [4] to decide other trace properties than reachability.

In the present work, we extend this result to equivalence properties: we will preserve not only the set of solutions, but also all witnesses (attacker's recipes) that they are solutions indeed. For instance if the attacker has two ways to get a given message in the first experminents, he must have the very same two ways to obtain the corresponding message in the second experiment.

We do not simply decide the symbolic equivalence of deducibility constraints, but actually show that the problem can be simplified to the equivlence of finitely many solved forms. There is then a hope to extend the procedure to processes that contain negative tests or timing constraints for instance. At least in this respect, we believe that our procedure improved over [3].

We only completed the proofs in case of the classical encryption/decryption and pairing/projection primitives (not assuming atomic keys), but our algorithm is simply reflecting the intruder capabilities and we also hope that it can be extended to more primitives.

2 Deducibility constraints

We consider the set $\mathcal{T}$ of terms, that are built on an unbounded number of constants, the pairing function symbol $\langle \_, \_ \rangle$, and a symmetric encryption symbol $\{\_\}\_$. $\mathcal{T}(X)$ is the set of terms that are built using an additional set X of variable symbols (written $x, y, z, x_1, ...$ in what follows). A *recipe* is a term built with the function symbols $\{\_\}\_$, $\langle \_, \_ \rangle$, $\mathsf{dec}(\_, \_)$, π_1, π_2 and a special set of variable symbols AX, that will be written $ax_1, ax_2, ...$ Recipes typically represent attacker's actions and do not make use of the constants, but only the public function symbols.

We use the standard convergent rewrite system that consists of the three rules:

$$\mathsf{dec}(\{x\}_y, y) \to x \qquad \pi_1(\langle x, y \rangle) \to x \qquad \pi_2(\langle x, y \rangle) \to y$$

The normal form of a term u w.r.t. this system is written $u\downarrow$.

If ζ is a recipe, and T is a finite sequence of terms $u_1, \ldots, u_n$, $\zeta[T]$ is the term ζ in which the variables ax_i have been respectively replaced with u_i. This notations always assumes that any ax_i that occurs in ζ is such that $i \leq |T|$.

A *deducibility constraint* is a sequence $T_1 \Vdash u_1, \ldots, T_n \Vdash u_n$ such that

- each T_i is a sequence of terms in $\mathcal{T}(X)$ and T_i is a prefix of T_{i+1}
- each u_i is a term in $\mathcal{T}(X)$.
- if x is a variable occurring in some T_i, there is an index $j < i$ such that x occurs in u_j.

A constraint system is a deducibility constraints system, together with a set of equations (and possibly disequations).

A *solution* of a constraint system $(T_1 \Vdash u_1, \ldots, T_n \Vdash u_n, E)$ is a tuple $(\sigma, \zeta_1, \ldots, \zeta_n)$ such that

- σ is a substitution that maps every variable of the constraint system to a term in $\mathcal{T}$, and that is a solution of E
- for every i, and every j, if ax_j occurs in ζ_i, then $|T_i| \geq j$
- for every i, $\zeta_i[T_i\sigma] \downarrow = u_i\sigma$.

Two pairs $(C_1, S_1), (C_2, S_2)$ where C_i is a constraint system and S_i is a sequence of terms, are *symbolically equivalent* if

1. $|S_1| = |S_2|$
2. For every solution $(\sigma_1, \zeta_1, \ldots, \zeta_n)$ of C_1, there is a solution $(\sigma_2, \zeta_1, \ldots, \zeta_n)$ of C_2 such that, for every recipes ξ_1, ξ_2,
$$\xi_1[S_1\sigma_1] \downarrow = \xi_2[S_1\sigma_1] \downarrow \Leftrightarrow \xi_1[S_2\sigma_2] \downarrow = \xi_2[S_2\sigma_2] \downarrow$$
3. For every solution $(\sigma_2, \zeta_1, \ldots, \zeta_n)$ of C_2, there is a solution $(\sigma_1, \zeta_1, \ldots, \zeta_n)$ of C_1 such that, for every recipes ξ_1, ξ_2,
$$\xi_1[S_1\sigma_1] \downarrow = \xi_2[S_1\sigma_1] \downarrow \Leftrightarrow \xi_1[S_2\sigma_2] \downarrow = \xi_2[S_2\sigma_2] \downarrow$$

In words, for any attackers computations (the recipes $\zeta_1, \ldots, \zeta_n$) that yield a feasible trace (the substitution σ_1) there is a feasible trace in the second system (the substitution σ_2) corresponding to the same computations and such that any observation that can be performed on the first sequence of messages (the recipes ξ_1, ξ_2 and the first equality) can also be performed on the second sequence of messages, and conversely.

Example 1 *Consider the two systems*

$a, \langle a, b\rangle$	$\Vdash$	x	a, b	$\Vdash$	x
$S_1 = \{x\}$			$S_2 = \{x\}$		

They are not symbolically equivalent since, choosing the substitution $x \mapsto b$ and the recipe $\zeta_1 = \pi_1(ax_2)$, we get a solution of the first system ($\zeta_2[a, \langle a, b\rangle] \downarrow = b$), while there is no substitution σ' that maps x to a term in $\mathcal{T}$ and such that $\zeta_2[a, b] \downarrow = x\sigma'$.

Example 2

a	$\Vdash$	x	b	$\Vdash$	b
$a, \{x\}_b$	$\Vdash$	$\{a\}_b$	$b, \{b\}_c$	$\Vdash$	$\{x\}_c$
$S_1 = (x, \{a\}_b)$			$S_2 = (b, \{x\}_c)$		

are two equivalent systems since the only possible solution of the first deducibility constraint is $(\{x \mapsto a\}, ax_1, ax_2)$ and the only possible solution of the second system is $(\{x \mapsto b\}, ax_1, ax_2)$ and there is no observable (non-trivial) equality on $(a, \{a\}_b)$ as well as on $(b, \{b\}_c)$.

A *solved form* is a pair of a constraint system $(T_1 \Vdash x_1, \ldots, T_n \Vdash x_n, E)$ and a sequence of terms S such that $x_1, \ldots, x_n$ are distinct variables and E is a set of equations $y = u$ such that y has no other occurrence in the constraint (E may also contain in addition disequations, timing constraints... provided they do not entail any equation).

3 The main result

Theorem 1 *There is an algorithm that, given two pairs $(C, S), (C', S')$ of a constraint system and a sequence of terms outputs either "fail" or a finite sequence of pairs of solved forms $((C_1, S_1), (C'_1, S'_1)), \ldots, ((C_n, S_n), (C'_n, S'n))$ such that:*

- *When the output is "fail", then (C, S) and (C', S') are not symbolically equivalent*
- *Otherwise, (C, S) and (C', S') are symbolically equivalent if and only if, for every i, (C_i, S_i) and (C'_i, S'_i) are symbolically equivalent.*

We have no space for the proof of the theorem, but we display now an example showing how the algorithm works: it simplifies successively the pairs of constraints, possibly by splitting the constraints into two constraints, always keeping the same solutions of both constraints. (We omit the sequence of terms for simplicity).

d, e	$\Vdash$	$< x, y >$	a, b	$\Vdash$	$< x, y >$
d, e	$\Vdash$	z	a, b	$\Vdash$	z
d, e, z	$\Vdash$	w	$a, b, \{a\}_b$	$\Vdash$	w
$d, e, z, \{c\}_{<d,e>}$	$\Vdash$	c	$a, b, \{a\}_b, \{c\}_{<a,b>}$	$\Vdash$	c
$d, e, z, \{c\}_{<d,e>}, \{z\}_h$	$\Vdash$	$\{\{d\}_e\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, \{z\}_f$	$\Vdash$	$\{\{a\}_b\}_f$
$d, e, z, \{c\}_{<d,e>}, \{z\}_h, \{e\}_h$	$\Vdash$	$\{w\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, \{z\}_f, \{b\}_f$	$\Vdash$	$\{w\}_f$

First split the pairs on both sides, yielding:

d, e	$\Vdash$	x	a, b	$\Vdash$	x
d, e	$\Vdash$	y	a, b	$\Vdash$	y
d, e	$\Vdash$	z	a, b	$\Vdash$	z
d, e, z	$\Vdash$	w	$a, b, \{a\}_b$	$\Vdash$	w
$d, e, z, \{c\}_{<d,e>}$	$\Vdash$	c	$a, b, \{a\}_b, \{c\}_{<a,b>}$	$\Vdash$	c
$d, e, z, \{c\}_{<d,e>}, \{z\}_h$	$\Vdash$	$\{\{d\}_e\}\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, \{z\}_f$	$\Vdash$	$\{\{a\}_b\}_f$
$d, e, z, \{c\}_{<d,e>}, \{z\}_h, \{e\}_h$	$\Vdash$	$\{w\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, \{z\}_f, \{b\}_f$	$\Vdash$	$\{w\}_f$

We now split into two cases, depending on whether the key $\langle d, e\rangle$ (resp. $\langle a, b\rangle$) is deducible. The other branch, that yields a failure in both systems, is not displayed. From now on, we also omit the two first deducibility constraints, that remain always unchanged in both systems.

d, e	$\Vdash$	z	a, b	$\Vdash$	z
d, e, z	$\Vdash$	w	$a, b, \{a\}_b$	$\Vdash$	w
$d, e, z, \{c\}_{<d,e>}$	$\Vdash$	$< d, e >$	$a, b, \{a\}_b, \{c\}_{<a,b>}$	$\Vdash$	$< a, b >$
$d, e, z, \{c\}_{<d,e>}, c$	$\Vdash$	c	$a, b, \{a\}_b, \{c\}_{<a,b>}, c$	$\Vdash$	c
$d, e, z, \{c\}_{<d,e>}, c, \{z\}_h$	$\Vdash$	$\{\{d\}_e\}\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{z\}_f$	$\Vdash$	$\{\{a\}_b\}_f$
$d, e, z, \{c\}_{<d,e>}, c, \{z\}_h, \{e\}_h$	$\Vdash$	$\{w\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{z\}_f, \{b\}_f$	$\Vdash$	$\{w\}_f$

We observe now an identity in the first system (the two occurrences of z) that must hold in the second system (unless there exists a solution of the second contraints system where $z \neq \{a\}_b$, which is considered in another branch, that is not displayed here because it yields

a failure due to the fifth constraint):

d, e	$\Vdash$	z	a, b	$\Vdash$	$\{a\}_b$
d, e, z	$\Vdash$	w	$a, b, \{a\}_b$	$\Vdash$	w
$d, e, z, \{c\}_{<d,e>}$	$\Vdash$	$< d, e >$	$a, b, \{a\}_b, \{c\}_{<a,b>}$	$\Vdash$	$< a, b >$
$d, e, z, \{c\}_{<d,e>}, c$	$\Vdash$	c	$a, b, \{a\}_b, \{c\}_{<a,b>}, c$	$\Vdash$	c
$d, e, z, \{c\}_{<d,e>}, c, \{z\}_h$	$\Vdash$	$\{\{d\}_e\}\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f$	$\Vdash$	$\{\{a\}_b\}_f$
$d, e, z, \{c\}_{<d,e>}, c, \{z\}_h, \{e\}_h$	$\Vdash$	$\{w\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f, \{b\}_f$	$\Vdash$	$\{w\}_f$

Now, we observe an identity on the last but one constraint on the right, that must hold on the left, yielding:

d, e	$\Vdash$	$\{d\}_e$	a, b	$\Vdash$	$\{a\}_b$
$d, e, \{d\}_e$	$\Vdash$	w	$a, b, \{a\}_b$	$\Vdash$	w
$d, e, \{d\}_e, \{c\}_{<d,e>}$	$\Vdash$	$< d, e >$	$a, b, \{a\}_b, \{c\}_{<a,b>}$	$\Vdash$	$< a, b >$
$d, e, \{d\}_e, \{c\}_{<d,e>}, c$	$\Vdash$	c	$a, b, \{a\}_b, \{c\}_{<a,b>}, c$	$\Vdash$	c
$d, e, \{d\}_e, \{c\}_{<d,e>}, c, \{\{d\}_e\}\}_h$	$\Vdash$	$\{\{d\}_e\}\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f$	$\Vdash$	$\{\{a\}_b\}_f$
$d, e, \{d\}_e, \{c\}_{<d,e>}, c, \{\{d\}_e\}\}_h, \{e\}_h$	$\Vdash$	$\{w\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f, \{b\}_f$	$\Vdash$	$\{w\}_f$

We now split into two cases, depending on whether we deduce $\{w\}_h$ (resp. $\{w\}_f$) from $\{e\}_h$ (resp. $\{b\}_f$), yielding :

d, e	$\Vdash$	$\{d\}_e$	a, b	$\Vdash$	$\{a\}_b$
$d, e, \{d\}_e$	$\Vdash$	e	$a, b, \{a\}_b$	$\Vdash$	b
$d, e, \{d\}_e, \{c\}_{<d,e>}$	$\Vdash$	$< d, e >$	$a, b, \{a\}_b, \{c\}_{<a,b>}$	$\Vdash$	$< a, b >$
$d, e, \{d\}_e, \{c\}_{<d,e>}, c$	$\Vdash$	c	$a, b, \{a\}_b, \{c\}_{<a,b>}, c$	$\Vdash$	c
$d, e, \{d\}_e, \{c\}_{<d,e>}, c, \{\{d\}_e\}\}_h$	$\Vdash$	$\{\{d\}_e\}\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f$	$\Vdash$	$\{\{a\}_b\}_f$
$d, e, \{d\}_e, \{c\}_{<d,e>}, c, \{\{d\}_e\}\}_h, \{e\}_h$	$\Vdash$	$\{e\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f, \{b\}_f$	$\Vdash$	$\{b\}_f$

Those two constraints systems are equal through renaming and so equivalent. We now have to check on the second branch where $\{w\}_h$ (resp. $\{w\}_f$) isn't deduce from $\{e\}_h$ (resp. $\{b\}_f$). In this case, the only choice we have is to deduce $\{w\}_h$ (resp. $\{w\}_f$) from $\{\{d\}_e\}_h$ (resp. $\{\{a\}_b\}_f$), yielding :

d, e	$\Vdash$	$\{d\}_e$	a, b	$\Vdash$	$\{a\}_b$
$d, e, \{d\}_e$	$\Vdash$	$\{d\}_e$	$a, b, \{a\}_b$	$\Vdash$	$\{a\}_b$
$d, e, \{d\}_e, \{c\}_{<d,e>}$	$\Vdash$	$< d, e >$	$a, b, \{a\}_b, \{c\}_{<a,b>}$	$\Vdash$	$< a, b >$
$d, e, \{d\}_e, \{c\}_{<d,e>}, c$	$\Vdash$	c	$a, b, \{a\}_b, \{c\}_{<a,b>}, c$	$\Vdash$	c
$d, e, \{d\}_e, \{c\}_{<d,e>}, c, \{\{d\}_e\}\}_h$	$\Vdash$	$\{\{d\}_e\}\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f$	$\Vdash$	$\{\{a\}_b\}_f$
$d, e, \{d\}_e, \{c\}_{<d,e>}, c, \{\{d\}_e\}\}_h, \{e\}_h$	$\Vdash$	$\{\{d\}_e\}\}_h$	$a, b, \{a\}_b, \{c\}_{<a,b>}, c, \{\{a\}_b\}_f, \{b\}_f$	$\Vdash$	$\{\{a\}_b\}_f$

Again those two constraints systems are equivalent. Now because all the branches lead to a couple of equivalent constraints systems, the algorithm will output "success".

References

[1] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25, Alexandria, Virginia, USA, November 2005. ACM Press.

[2] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, February–March 2008.

[3] Yannick Chevalier and Michael Rusinowitch. Decidability of symbolic equivalence of derivations. Unpublished draft, 2009.

[4] Hubert Comon-Lundh, Véronique Cortier, and Eugen Zlinescu. Deciding security properties of cryptographic protocols. application to key cycles. *Transaction on Computational Logic*, 2009. To appear. A preminary verion is available at `http://arxiv.org/abs/0708.3564`.

[5] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, Port Jefferson, NY, USA, July 2009. IEEE Computer Society Press. To appear.

[6] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security*, 2001.

[7] Michael Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is np-complete. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001.