YAPA: A generic tool for computing intruder knowledge [⋆]

Mathieu Baudet[1], Véronique Cortier[2], and Stéphanie Delaune[3]

[1] DCSSI, France
[2] LORIA, CNRS & INRIA project Cassis, France
[3] LSV, ENS Cachan & CNRS & INRIA, France

Abstract. Reasoning about the knowledge of an attacker is a necessary step in many formal analyses of security protocols. In the framework of the applied pi calculus, as in similar languages based on equational logics, knowledge is typically expressed by two relations: deducibility and static equivalence. Several decision procedures have been proposed for these relations under a variety of equational theories. However, each theory has its particular algorithm, and none has been implemented so far.

We provide a generic procedure for deducibility and static equivalence that takes as input any convergent rewrite system. We show that our algorithm covers all the existing decision procedures for convergent theories. We also provide an efficient implementation, and compare it briefly with the more general tool ProVerif.

1 Introduction

Understanding security protocols often requires reasoning about the information accessible to an online attacker. Accordingly, many formal approaches to security rely on a notion of *deducibility* [18, 19] that models whether a piece of data, typically a secret, is retrievable from a finite set of messages. Deducibility, however, does not always suffice to reflect the knowledge of an attacker. Consider for instance a protocol sending an encrypted Boolean value, say, a vote in an electronic voting protocol. Rather than deducibility, the key idea to express confidentiality of the plaintext is that an attacker should not be able to *distinguish* between the sequences of messages corresponding to each possible value.

In the framework of the applied pi-calculus [3], as in similar languages based on equational logics [10], indistinguishability corresponds to a relation called *static equivalence*: roughly, two sequences of messages are *statically equivalent* when they satisfy the same algebraic relations from the attacker's point of view. Static equivalence plays an important role in the study of guessing attacks (e.g. [13, 5, 1]), as well as for anonymity properties and electronic voting protocols (e.g. [17]). In several cases, this notion has also been shown to imply the more complex and precise notion of cryptographic indistinguishability [8, 1], related to probabilistic polynomial-time Turing machines.

[⋆] Kindly supported by ANR-07-SESU-002 AVOTÉ and ARA SSIA FormaCrypt.

We emphasize that both deducibility and static equivalence apply to observations on finite sets of messages, and do not take into account the dynamic behavior of protocols. Nevertheless, deducibility is used as a subroutine by many general decision procedures [12, 11]. Besides, it has been shown that observational equivalence in the applied pi-calculus coincides with labeled bisimulation [3], that is, corresponds to checking a number of static equivalences and some standard bisimulation conditions.

Deducibility and static equivalence rely on an underlying equational theory for axiomatizing the properties of cryptographic functions. Many decision procedures [2, 14] have been proposed to compute these relations under a variety of equational theories, including symmetric and asymmetric encryptions, signatures, exclusive OR, and homomorphic operators. However, except for the class of subterm convergent theories [2], which covers the standard flavors of encryption and signature, each of these decision results introduces a new procedure, devoted to a particular theory. Even in the case of the general decidability criterion given in [2], we note that the algorithm underlying the proof has to be adapted for each theory, depending on how the criterion is fulfilled.

Perhaps as a consequence of this fact, none of these decision procedures has been implemented so far. Up to our knowledge the only tool able to verify static equivalence is ProVerif [9, 10]. This general tool can handle various equational theories and analyze security protocols under active adversaries. However termination of the verifier is not guaranteed in general, and protocols are subject to (safe) approximations.

The present work aims to fill this gap between theory and implementation and propose an efficient tool for deciding deducibility and static equivalence in a uniform way. It is initially inspired from a procedure for solving more general constraint systems related to active adversaries and equivalence of finite processes, presented in [5], with corrected extended version in [6] (in French). However, due to the complexity of the constraint systems, this decision procedure was only studied for subterm convergent theories, and remains too complex to enable an efficient implementation.

Our first contribution is to provide and study a generic procedure for checking deducibility and static equivalence, taking as input any convergent theory (that is, any equational theory described by a finite convergent rewrite system). We prove the algorithm sound and complete, up to explicit failure cases. Note that (unfailing) termination cannot be guaranteed in general since the problem of checking deducibility and static equivalence is undecidable, even for convergent theories [2]. To address this issue and turn our algorithm into a decision procedure for a given convergent theory, we provide two criteria. First, we define a syntactic criterion on the rewrite rules that ensures that the algorithm never fails. This criterion is enjoyed in particular by any convergent subterm theory, as well as the theories of blind signature and homomorphic encryption. Termination often follows from a simple analysis of the rules of the algorithm: as a proof of concept, we obtain a new decidability result for deducibility and static equivalence for the prefix theory, representing encryption in CBC mode.

Second, we provide a termination criterion based on deducibility: provided that failure cannot occur, termination on a given input is equivalent to the existence of some natural finite representation of deducible terms. As a consequence, we obtain that our algorithm can decide deducibility and static equivalence for all the convergent theories previously known to be decidable [2].

Our second contribution is an efficient implementation of this generic procedure, called YAPA. After describing the main features of the implementation, we report several experiments suggesting that our tool computes static equivalence faster and for more convergent theories than the general tool ProVerif [9, 10]. Because of space constraints, proofs are in an extended version of this paper [7].

2 Preliminaries

2.1 Term algebra

We start by introducing the necessary notions to describe cryptographic messages in a symbolical way. For modeling cryptographic primitives, we assume a given set of *function symbols* $\mathcal{F}$ together with an arity function $\mathrm{ar} : \mathcal{F} \to \mathbb{N}$. Symbols in $\mathcal{F}$ of arity 0 are called *constants*. We consider a set of *variables* $\mathcal{X}$ and a set of additional constants $\mathcal{W}$ called *parameters*. The (usual, first-order) term algebra generated by $\mathcal{F}$ over $\mathcal{W}$ and $\mathcal{X}$ is written $\mathcal{F}[\mathcal{W} \cup \mathcal{X}]$ with elements denoted by $T, U, T_1 \ldots$ More generally, we write $\mathcal{F}'[A]$ for the least set of terms containing a set A and stable by application of symbols in $\mathcal{F}' \subseteq \mathcal{F}$.

We write $\mathrm{var}(T)$ (resp. $\mathrm{par}(T)$) for the set of variables (resp. parameters) that occur in a term T. These notations are extended to tuples and sets of terms in the usual way. The set of positions (resp. subterms) of a term T is written $\mathrm{pos}(T) \subseteq \mathbb{N}^*$ (resp. $\mathrm{st}(T)$). The subterm of T at position $p \in \mathrm{pos}(T)$ is written $T|_p$. The term obtained by replacing $T|_p$ with a term U in T is denoted $T[U]_p$.

A *(finite, partial) substitution* σ is a mapping from a finite subset of variables, called its *domain* and written $\mathrm{dom}(\sigma)$, to terms. The *image* of a substitution is its image as a mapping $\mathrm{im}(\sigma) = \{\sigma(x) \mid x \in \mathrm{dom}(\sigma)\}$. Substitutions are extended to endomorphisms of $\mathcal{F}[\mathcal{X} \cup \mathcal{W}]$ as usual. We use a postfix notation for their application. A term T (resp. a substitution σ) is *ground* iff $\mathrm{var}(T) = \emptyset$ (resp. $\mathrm{var}(\mathrm{im}(\sigma)) = \emptyset$).

For our cryptographic purposes, it is useful to distinguish a subset $\mathcal{F}_{\mathsf{pub}}$ of $\mathcal{F}$, made of *public function symbols*, that is, intuitively, the symbols made available to the attacker. A *recipe* (or *second-order term*) M, N, $M_1 \ldots$ is a term in $\mathcal{F}_{\mathsf{pub}}[\mathcal{W} \cup \mathcal{X}]$, that is, a term containing no *private* (non-public) function symbols. A *plain term* (or *first-order term*) t, r, s, $t_1 \ldots$ is a term in $\mathcal{F}[\mathcal{X}]$, that is, containing no parameters. A *(public, ground, non-necessarily linear) n-ary context* C is a recipe in $\mathcal{F}_{\mathsf{pub}}[\mathsf{w}_1, \ldots, \mathsf{w}_n]$, where we assume a fixed countable subset of parameters $\{\mathsf{w}_1, \ldots, \mathsf{w}_n, \ldots\} \subseteq \mathcal{W}$. If C is a n-ary context, $C[T_1, \ldots, T_n]$ denotes the term obtained by replacing each occurrence of w_i with T_i in C.

2.2 Rewriting

A *rewrite system* $\mathcal{R}$ is a finite set of *rewrite rules* $l \to r$ where $l, r \in \mathcal{F}[\mathcal{X}]$ and $\mathrm{var}(r) \subseteq \mathrm{var}(l)$. A term S *rewrites* to T by $\mathcal{R}$, denoted $S \to_{\mathcal{R}} T$, if there exist

$l \to r$ in $\mathcal{R}$, $p \in \mathrm{pos}(S)$ and a substitution σ such that $S|_p = l\sigma$ and $T = S[r\sigma]_p$. We write $\to^+_{\mathcal{R}}$ for the transitive closure of $\to_{\mathcal{R}}$, $\to^*_{\mathcal{R}}$ for its reflexive and transitive closure, and $=_{\mathcal{R}}$ for its reflexive, symmetric and transitive closure.

A rewrite system $\mathcal{R}$ is *convergent* if is *terminating*, i.e. there is no infinite chains $T_1 \to_{\mathcal{R}} T_2 \to_{\mathcal{R}} \ldots$, and *confluent*, i.e. for every terms S, T such that $S =_{\mathcal{R}} T$, there exists U such that $S \to^*_{\mathcal{R}} U$ and $T \to^*_{\mathcal{R}} U$.

A term T is *$\mathcal{R}$-reduced* if there is no term S such that $T \to_{\mathcal{R}} S$. If $T \to^*_{\mathcal{R}} S$ and S is $\mathcal{R}$-reduced then S is *a $\mathcal{R}$-reduced form of* T. When this reduced form is unique (in particular if $\mathcal{R}$ is convergent), we write $S = T{\downarrow_{\mathcal{R}}}$.

2.3 Equational theories

We equip the signature $\mathcal{F}$ with an equational theory represented by a set of equations $\mathcal{E}$ of the form $s = t$ with $s, t \in \mathcal{F}[\mathcal{X}]$. The equational theory E generated by $\mathcal{E}$ is the least set of equations containing $\mathcal{E}$ that is stable under the axioms of congruence (reflexivity, symmetry, transitivity, application of function symbols) and under application of substitutions. We write $=_{\mathsf{E}}$ for the corresponding relation on terms. Equational theories have proved very useful for modeling algebraic properties of cryptographic primitives [15, 2].

We are particularly interested in theories E that can be represented by a convergent rewrite system $\mathcal{R}$, i.e. theories for which there exists a convergent rewrite system $\mathcal{R}$ such that the two relations $=_{\mathcal{R}}$ and $=_{\mathsf{E}}$ coincide. The rewrite system $\mathcal{R}$ —and by extension the equational theory E— is *subterm convergent* if, in addition, we have that for every rule $l \to r \in \mathcal{R}$, r is either a subterm of l or a ground $\mathcal{R}$-reduced term. This class encompasses the one of the same name used in [2], the class of dwindling theories used in [4], and the class of public-collapsing theories introduced in [16].

Example 1. Consider the signature $\mathcal{F}_{\mathsf{enc}} = \{\mathsf{dec}, \mathsf{enc}, \langle \_, \_ \rangle, \pi_1, \pi_2\}$. The symbols dec, enc and $\langle \_, \_ \rangle$ are functional symbols of arity 2 that represent respectively the decryption, encryption and pairing functions, whereas π_1 and π_2 are functional symbols of arity 1 that represent the projection function on the first and the second component of a pair, respectively. The equational theory of pairing and symmetric (deterministic) encryption, denoted by $\mathsf{E}_{\mathsf{enc}}$, is generated by the equations $\mathcal{E}_{\mathsf{enc}} = \{\mathsf{dec}(\mathsf{enc}(x, y), y) = x,\ \ \pi_1(\langle x, y\rangle) = x,\ \ \pi_2(\langle x, y\rangle) = y\}$.

Motivated by the modeling of the ECB mode of encryption, we may also consider an encryption symbol that is homomorphic with respect to pairing:

$$\mathcal{E}_{\mathsf{hom}} = \mathcal{E}_{\mathsf{enc}} \cup \left\{ \begin{array}{l} \mathsf{enc}(\langle x, y\rangle, z) = \langle \mathsf{enc}(x, z), \mathsf{enc}(y, z)\rangle \\ \mathsf{dec}(\langle x, y\rangle, z) = \langle \mathsf{dec}(x, z), \mathsf{dec}(y, z)\rangle \end{array} \right\}.$$

If we orient the equations from left to right, we obtain two rewrite systems $\mathcal{R}_{\mathsf{enc}}$ and $\mathcal{R}_{\mathsf{hom}}$. Both rewrite systems are convergent, only $\mathcal{R}_{\mathsf{enc}}$ is subterm convergent.

From now on, we assume a given equational theory E represented by a convergent rewrite system $\mathcal{R}$. A symbol f is *free* if f does not occur in $\mathcal{R}$. In order to model (an unbounded number of) random values possibly generated by the attacker, we assume that $\mathcal{F}_{\mathsf{pub}}$ contains infinitely many free public constants. We will use free private constants to model secrets, for instance the secret keys used to encrypt a message. Private (resp. public) free constants are closely related to bound (resp. free) *names* in the framework of the applied pi calculus [3]. Our formalism also allows one to consider non-constant private symbols.

3 Deducibility and static equivalence

In order to describe the cryptographic messages observed or inferred by an attacker, we introduce the following notions of deduction facts and frames.

A *deduction fact* is a pair, written $M \rhd t$, made of a recipe $M \in \mathcal{F}_{\mathsf{pub}}[\mathcal{W} \cup \mathcal{X}]$ and a plain term $t \in \mathcal{F}[\mathcal{X}]$. Such a deduction fact is *ground* if $\mathrm{var}(M,t) = \emptyset$. A *frame*, denoted by letters φ, Φ, $\Phi_0 \dots$, is a finite set of ground deduction facts. The *image* of a frame is defined by $\mathrm{im}(\Phi) = \{t \mid M \rhd t \in \Phi\}$. A frame Φ is *one-to-one* if $M_1 \rhd t$, $M_2 \rhd t \in \Phi$ implies $M_1 = M_2$.

A frame φ is *initial* if it is of the form $\varphi = \{w_1 \rhd t_1, \dots, w_\ell \rhd t_\ell\}$ for some distinct parameters $w_1, \dots, w_\ell \in \mathcal{W}$. Initial frames are closely related to the notion of frames in the applied pi-calculus [3]. The parameters w_i can be seen as labels that refer to the messages observed by an attacker. Given such an initial frame φ, we denote by $\mathrm{dom}(\varphi)$ its *domain* $\mathrm{dom}(\varphi) = \{w_1, \dots, w_\ell\}$. If $\mathrm{par}(M) \subseteq \mathrm{dom}(\varphi)$, we write $M\varphi$ for the term obtained by replacing each w_i by t_i in M. If in addition M is ground then $t = M\varphi$ is a ground plain term.

3.1 Deducibility, recipes

Classically (see e.g. [2]), a ground term t is *deducible* modulo E from an initial frame φ if there exists $M \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi)]$ such that $M\varphi =_{\mathsf{E}} t$. This corresponds to the intuition that the attacker may compute (infer) t from φ. For the purpose of our study, we generalize this notion to arbitrary frames, and even sets of (non-necessarily ground) deduction facts ϕ, using the notations $\rhd_\phi$ and $\rhd_\phi^{\mathsf{E}}$.

Definition 1 (deducibility). *Let ϕ be finite set of deductions facts, for instance a frame. We say that M* is a recipe of t in ϕ, *written $M \rhd_\phi t$, iff there exist a (public, ground, non-necessarily linear) n-ary context C and some deduction facts $M_1 \rhd t_1, \dots, M_n \rhd t_n$ in ϕ such that $M = C[M_1, \dots, M_n]$ and $t = C[t_1, \dots, t_n]$. In that case, we say that t is* syntactically deducible *from ϕ, also written $\phi \vdash t$.*

We say that M is a recipe of t in ϕ modulo E, *written $M \rhd_\phi^{\mathsf{E}} t$, iff there exists a term t' such that $M \rhd_\phi t'$ and $t' =_{\mathsf{E}} t$. In that case, we say that t is* deducible from ϕ modulo E, *written $\phi \vdash_{\mathsf{E}} t$.*

We note that $M \rhd_\varphi t$ is equivalent to $M\varphi = t$ when φ is an initial frame and when t (or equivalently M) is ground.

Example 2. Consider the equational theory $\mathsf{E}_{\mathsf{enc}}$ given in Example 1. Let $\varphi = \{\mathsf{w}_1 \rhd \langle \mathsf{enc}(\mathsf{s}_1, \mathsf{k}), \mathsf{enc}(\mathsf{s}_2, \mathsf{k})\rangle, \mathsf{w}_2 \rhd \mathsf{k}\}$ where s_1, s_2 and k are private constant symbols. We have that $\langle \mathsf{w}_2, \mathsf{w}_2\rangle \rhd_\varphi \langle \mathsf{k}, \mathsf{k}\rangle$, and $\mathsf{dec}(\mathsf{proj}_1(\mathsf{w}_1), \mathsf{w}_2) \rhd_\varphi^{\mathsf{E}_{\mathsf{enc}}} \mathsf{s}_1$.

3.2 Static equivalence, visible equations

Deducibility does not always suffice for expressing the knowledge of an attacker. In particular, it does not account for the partial information that an attacker may obtain about secrets. This issue motivates the study of visible equations and static equivalence [3], defined as follows.

Definition 2 (static equivalence). *Let φ be an initial frame. The set of* visible equations of φ modulo E *is defined as*

$$\mathrm{eq}_{\mathsf{E}}(\varphi) = \{M \bowtie N \mid M, N \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi)],\ M\varphi =_{\mathsf{E}} N\varphi\}$$

where $\bowtie$ is a dedicated commutative symbol. Two initial frames φ_1 and φ_2 with the same domain are statically equivalent *modulo E, written $\varphi_1 \approx_{\mathsf{E}} \varphi_2$, if their sets of visible equations are equal, i.e.* $\mathrm{eq}_{\mathsf{E}}(\varphi_1) = \mathrm{eq}_{\mathsf{E}}(\varphi_2)$.

This definition is in line with static equivalence in the applied pi calculus [3]. For the purpose of finitely describing the set of visible equations $\mathrm{eq}_{\mathsf{E}}(\varphi)$ of an initial frame, we introduce *quantified equations* of the form $\forall z_1, \ldots, z_q.M \bowtie N$ where $z_1, \ldots, z_q \in \mathcal{X}$, $q \geq 0$ and $\mathrm{var}(M, N) \subseteq \{z_1, \ldots, z_q\}$. In the following, finite sets of quantified equations are denoted Ψ, Ψ_0,... We write $\Psi \models M \bowtie N$ when the ground equation $M \bowtie N$ is a consequence of Ψ in the usual, first-order logics with equality axioms for the relation $\bowtie$ (that is, reflexivity, symmetry, transitivity and compatibility with symbols in $\mathcal{F}_{\mathsf{pub}}$). When no confusion arises, we may refer to quantified equations simply as *equations*. As usual, quantified equations are considered up to renaming of bound variables.

Example 3. Consider again the equational theory $\mathsf{E}_{\mathsf{enc}}$ given in Example 1. Let $\varphi_1 = \{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_0, \mathsf{k}),\ \mathsf{w}_2 \rhd \mathsf{k}\}$ and $\varphi_2 = \{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_1, \mathsf{k}),\ \mathsf{w}_2 \rhd \mathsf{k}\}$ where c_0, c_1 are public constants and k is a private constant. Let $\Psi_1 = \{\mathsf{enc}(\mathsf{c}_0, \mathsf{w}_2) \bowtie \mathsf{w}_1\}$ and $\Psi_2 = \{\mathsf{enc}(\mathsf{c}_1, \mathsf{w}_2) \bowtie \mathsf{w}_1\}$. We have that $\Psi_i \models \mathrm{eq}_{\mathsf{E}_{\mathsf{enc}}}(\varphi_i)$ for $i = 1, 2$. Hence, $\mathrm{eq}_{\mathsf{E}_{\mathsf{enc}}}(\varphi_1) \neq \mathrm{eq}_{\mathsf{E}_{\mathsf{enc}}}(\varphi_2)$ and the two frames φ_1 and φ_2 are not statically equivalent. However, it can be shown that $\{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_0, \mathsf{k})\} \approx_{\mathsf{E}_{\mathsf{enc}}} \{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_1, \mathsf{k})\}$.

4 Main procedure

In this section, we describe our algorithms for checking deducibility and static equivalence on convergent rewrite systems. After some additional notations, we present the core of the procedure, which consists of a set of transformation rules used to saturate a frame and a finite set of quantified equations. We then show how to use this procedure to decide deducibility and static equivalence, provided that saturation succeeds.

Soundness and completeness of the saturation procedure are detailed in Section 5. We provide sufficient conditions on the rewrite systems to ensure success of saturation in Section 6.

4.1 Decompositions of rewrite rules

Before stating the procedure, we introduce the following notion of *decomposition* to account for the possible superpositions of an attacker's context with a left-hand side of rewrite rule.

Definition 3 (decomposition). *Let n, p, q be non-negative integers. A (n, p, q)-*decomposition *of a term l (and by an extension of any rewrite rule $l \to r$) is a (public, ground, non-necessarily linear) context $D \in \mathcal{F}_{\mathsf{pub}}[\mathcal{W}]$ such that* $\mathrm{par}(D) = \{\mathsf{w}_1, \ldots, \mathsf{w}_{n+p+q}\}$ *and $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ where*

- *$l_1, \ldots, l_n$ are mutually-distinct non-variable terms,*
- *$y_1, \ldots, y_p$ and $z_1, \ldots, z_q$ are mutually-distinct variables, and*
- *$y_1, \ldots, y_p \in \mathrm{var}(l_1, \ldots, l_n)$ whereas $z_1, \ldots, z_q \notin \mathrm{var}(l_1, \ldots, l_n)$.*

A decomposition D is proper *if it is not a parameter (i.e. $D \neq \mathsf{w}_1$).*

Example 4. Consider the rewrite rule $\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$. This rule admits two proper decompositions up to permutation of parameters:

- $D_1 = \mathsf{dec}(\mathsf{enc}(\mathsf{w}_1, \mathsf{w}_2), \mathsf{w}_2)$ where $n = 0$, $p = 0$, $q = 2$, $z_1 = x$, $z_2 = y$;
- $D_2 = \mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2)$ where $n = 1$, $p = 1$, $q = 0$, $l_1 = \mathsf{enc}(x, y)$ and $y_1 = y$.

4.2 Transformation rules

To check deducibility and static equivalence, we proceed by saturating an initial frame, adding some deduction facts and equations satisfied by the frame. We consider *states* that are either the failure state $\perp$ or a couple (Φ, Ψ) formed by a one-to-one frame Φ in $\mathcal{R}$-reduced form and a finite set of quantified equations Ψ.

Given an initial frame φ, our procedure starts from an initial state associated to φ, denoted by $\mathrm{Init}(\varphi)$, obtained by reducing φ and replacing duplicated terms by equations. Formally, $\mathrm{Init}(\varphi)$ is the result of a procedure recursively defined as follows: $\mathrm{Init}(\emptyset) = (\emptyset, \emptyset)$, and assuming $\mathrm{Init}(\varphi) = (\Phi, \Psi)$, we have

$$\mathrm{Init}(\varphi \uplus \{w \rhd t\}) = \begin{cases} (\Phi, \Psi \cup \{w \bowtie w'\}) & \text{if there exists some } w' \rhd t{\downarrow_{\mathcal{R}}} \in \Phi \\ (\Phi \cup \{w \rhd t{\downarrow_{\mathcal{R}}}\}, \Psi) & \text{otherwise.} \end{cases}$$

The main part of our procedure consists in saturating a state (Φ, Ψ) by means of the transformation rules described in Figure 1. The **A** rules are designed for applying a rewrite step on top of existing deduction facts. If the resulting term is already syntactically deducible then a corresponding equation is added (rule **A.1**); or else if it is ground, the corresponding deduction fact is added to the state (rule **A.2**); otherwise, the procedure may fail (rule **A.3**). The **B** rules are meant to add syntactically deducible subterms (rule **B.2**) or related equations (rule **B.1**). For technical reasons, rule **A.1** is parametrized by a function Ctx with values of the form M or $\perp$, and satisfying the following properties:

(a) if $\phi \vdash t{\downarrow_{\mathcal{R}}}$, then for any Ψ and α, $\mathrm{Ctx}(\phi \vdash^{?}_{\mathcal{R}} t, \Psi, \alpha) \neq \perp$;

A. Inferring deduction facts and equations by context reduction
Assume that

$l = D[l_1, \dots, l_n, y_1, \dots, y_p, z_1, \dots, z_q]$ is a proper decomposition of $(l \to r) \in \mathcal{R}$
$M_1 \rhd t_1, \dots, M_{n+p} \rhd t_{n+p} \in \Phi$
$(l_1, \dots, l_n, y_1, \dots, y_p)\,\sigma = (t_1, \dots, t_{n+p})$

1. If there exists $M = \mathrm{Ctx}(\Phi \cup \{z_1 \rhd z_1, \dots, z_q \rhd z_q\} \vdash^?_{\mathcal{R}} r\sigma,\ \Psi,\ (l, r, D, \sigma))$, then

$$(\Phi, \Psi) \Longrightarrow (\Phi, \Psi \cup \{\forall z_1, \dots, z_q . D[M_1, \dots, M_{n+p}, z_1 \dots, z_q] \bowtie M\}) \qquad (\mathbf{A.1})$$

2. Else, if $(r\sigma)\downarrow_{\mathcal{R}}$ is ground, then

$$\begin{aligned}(\Phi, \Psi) \Longrightarrow (&\Phi \cup \{M_0 \rhd (r\sigma)\downarrow_{\mathcal{R}}\},\\ &\Psi \cup \{\forall z_1, \dots, z_q . D[M_1, \dots, M_{n+p}, z_1 \dots, z_q] \bowtie M_0\})\end{aligned} \qquad (\mathbf{A.2})$$

where $M_0 = D[M_1, \dots, M_{n+p}, \mathsf{a}, \dots, \mathsf{a}]$ for some fixed public constant a.

3. Otherwise, $(\Phi, \Psi) \Longrightarrow \bot$ $\qquad (\mathbf{A.3})$

B. Inferring deduction facts and equations syntactically
Assume that $M_0 \rhd t_0, \dots, M_n \rhd t_n \in \Phi \qquad t = f(t_1, \dots, t_n) \in \mathrm{st}(t_0) \qquad f \in \mathcal{F}_{\mathsf{pub}}$

1. If there exists M such that $(M \rhd t) \in \Phi$,

$$(\Phi, \Psi) \Longrightarrow (\Phi, \Psi \cup \{f(M_1, \dots, M_n) \bowtie M\}) \qquad (\mathbf{B.1})$$

2. Otherwise, $(\Phi, \Psi) \Longrightarrow (\Phi \cup \{f(M_1, \dots, M_n) \rhd t\}, \Psi)$ $\qquad (\mathbf{B.2})$

Fig. 1. Transformation rules

(b) if $M = \mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha)$ then there exist M' and s such that $\Psi \models M \bowtie M'$, $M' \rhd_\phi s$ and $t \to^*_{\mathcal{R}} s$. (This justifies the notation $\phi \vdash^?_{\mathcal{R}} t$ used to denote a specific deducibility problem.)

Note that a simple choice for $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha)$ is to solve the deducibility problem $\phi \vdash^? t\downarrow_{\mathcal{R}}$ in the empty equational theory, and then return a corresponding recipe M, if any. (This problem is easily solved by induction on $t\downarrow_{\mathcal{R}}$.) Yet, optimizing the function Ctx is a nontrivial task: on the one hand, letting $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha) \neq \bot$ for more values ϕ, t, Ψ, α makes the procedure more likely to succeed; on the other hand, it is computationally more demanding. We explain in Section 6.1 the choice of Ctx made in our implementation.

We write $\Longrightarrow^*$ for the transitive and reflexive closure of $\Longrightarrow$. The definitions of Ctx and of the transformation rules ensure that whenever $S \Longrightarrow^* S'$ and S is a state, then S' is also a state, with the same parameters unless $S' = \bot$.

Example 5. Consider the frame φ_1 previously described in Example 3. We can apply rule **A.1** as follows. Consider the rewrite rule $\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$, the decomposition D_2 given in Example 4 and $t_1 = \mathsf{enc}(\mathsf{c}_0, \mathsf{k})$. We have $\mathrm{Init}(\varphi_1) = (\varphi_1, \emptyset) \Longrightarrow (\varphi_1, \{\mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2) \bowtie \mathsf{c}_0\})$. In other words, since we know the key k through w_2, we can check that the decryption of w_1 by w_2 leads to the public constant c_0. Next we apply rule **B.1** as follows: $(\varphi_1, \{\mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2) \bowtie \mathsf{c}_0\}) \Longrightarrow$

$(\varphi_1, \{\mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2) \bowtie \mathsf{c}_0, \mathsf{enc}(\mathsf{c}_0, \mathsf{w}_2) \bowtie \mathsf{w}_1\})$. No more rules can then modify the state.

Main theorem. We now state the soundness and the completeness of the transformation rules provided that a *saturated state* is reached, that is, a state $S \neq \bot$ such that $S \Longrightarrow S'$ implies $S' = S$. The technical lemmas involved in the proof are detailed in Section 5.

Theorem 1 (soundness and completeness). *Let* E *be an equational theory generated by a convergent rewrite system* $\mathcal{R}$*. Let* φ *be an initial frame and* (Φ, Ψ) *be a saturated state such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$*.*

1. *For all* $M \in \mathcal{F}_{\mathsf{pub}}[\mathrm{par}(\varphi)]$ *and* $t \in \mathcal{F}[\emptyset]$*, we have*
$$M\varphi =_{\mathsf{E}} t \quad \Leftrightarrow \quad \exists N,\ \Psi \models M \bowtie N \text{ and } N \rhd_{\Phi} t{\downarrow_{\mathcal{R}}}$$
2. *For all* $M, N \in \mathcal{F}_{\mathsf{pub}}[\mathrm{par}(\varphi) \cup \mathcal{X}]$*, we have that* $M\varphi =_{\mathsf{E}} N\varphi \Leftrightarrow \Psi \models M \bowtie N$*.*

While the saturation procedure is sound and complete, it may not terminate, or *fail* if rule **A.3** becomes the only applicable rule. In Section 6, we explore several sufficient conditions to prevent failure and ensure termination.

4.3 Application to deduction and static equivalence

Decision procedures for deduction and static equivalence follow from Theorem 1.

Algorithm for deduction. Let φ be an initial frame and t be a ground term. The procedure for checking $\varphi \vdash_{\mathsf{E}} t$ runs as follows:

1. Apply the transformation rules to obtain (if any) a saturated state (Φ, Ψ) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$;
2. Return *yes* if there exists N such that $N \rhd_{\Phi} t{\downarrow_{\mathcal{R}}}$ (that is, the $\mathcal{R}$-reduced form of t is syntactically deducible from Φ); otherwise return *no*.

Algorithm for static equivalence. Let φ_1 and φ_2 be two initial frames. The procedure for checking $\varphi_1 \approx_{\mathsf{E}} \varphi_2$ runs as follows:

1. Apply the transformation rules to obtain (if possible) two saturated states (Φ_1, Ψ_1) and (Φ_2, Ψ_2) such that $\mathrm{Init}(\varphi_i) \Longrightarrow^* (\Phi_i, \Psi_i)$, $i = 1, 2$;
2. For $\{i, j\} = \{1, 2\}$, for every equation $(\forall z_1, \ldots, z_\ell . M \bowtie N)$ in Ψ_i, check that $M\varphi_j =_{\mathsf{E}} N\varphi_j$ — that is, in other words, $(M\varphi_j){\downarrow_{\mathcal{R}}} = (N\varphi_j){\downarrow_{\mathcal{R}}}$;
3. If so return *yes*; otherwise return *no*.

5 Soundness and completeness of the saturation

The proof of Theorem 1 is based on three main lemmas. First, the transformation rules are sound in the sense that, along the saturation process, we add only deducible terms and valid equations with respect to the initial frame.

Lemma 1 (soundness). *Let φ be an initial frame and (Φ, Ψ) be a state such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$. *Then, we have that*

1. $M \rhd_\Phi t \;\Rightarrow\; M\varphi =_{\mathsf{E}} t$ *for all* $M \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi)]$ *and* $t \in \mathcal{F}[\emptyset]$;
2. $\Psi \models M \bowtie N \;\Rightarrow\; M\varphi =_{\mathsf{E}} N\varphi$ *for all* $M, N \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi) \cup \mathcal{X}]$.

The next two lemmas are dedicated to the completeness of **B** and **A** rules, respectively. Lemma 2 ensures that saturated states account for all the syntactic equations possibly visible. Lemma 3 deals with the reduction of a deducible term along the rewrite system $\mathcal{R}$. Using that $\mathcal{R}$ is convergent, this allows us to prove that every deducible term from a saturated frame is syntactically deducible.

Lemma 2 (completeness, syntactic equations). *Let (Φ, Ψ) be a state, and M, N be two terms such that $M \rhd_\Phi t$ and $N \rhd_\Phi t$ for some term t. Then there exists (Φ', Ψ') such that $(\Phi, \Psi) \Longrightarrow^* (\Phi', \Psi')$ using **B** rules and $\Psi' \models M \bowtie N$.*

Lemma 3 (completeness, context reduction). *Let (Φ, Ψ) be a state and M, t, t' be three terms such that $M \rhd_\Phi t$ and $t \to_{\mathcal{R}} t'$. Then, either $(\Phi, \Psi) \Longrightarrow^* \bot$ or there exist (Φ', Ψ'), M' and t'' such that $(\Phi, \Psi) \Longrightarrow^* (\Phi', \Psi')$, $M' \rhd_{\Phi'} t''$ with $t' \to^*_{\mathcal{R}} t''$, and $\Psi' \models M \bowtie M'$.*

*Besides, in both cases, the corresponding derivation from (Φ, Ψ) can be chosen to consist of a number of **B** rules, possibly followed by one instance of **A** rule involving the same rewrite rule $l \to r$ as the rewrite step $t \to_{\mathcal{R}} t'$.*

6 Termination and non-failure

In the previous section, we proved that saturated frames yield sound and complete characterizations of deducible terms and visible equations of their initial frames. Yet, the saturation procedure may still not terminate, or fail due to rule **A.3**. In this section, we study different conditions on the rewrite system $\mathcal{R}$ so that failure never happens and/or termination is ensured.

6.1 A syntactic criterion to prevent failure

Our first criterion is syntactic and ensures that the algorithm never fails. It is enjoyed by a large class of equational theories, called *layered convergent.*

Definition 4 (layered rewrite system). *A rewrite system $\mathcal{R}$, and by extension its equational theory E, are* layered *if there exists an ascending chain of subsets $\emptyset = \mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \ldots \subseteq \mathcal{R}_{N+1} = \mathcal{R}$ ($N \geq 0$), such that for every $0 \leq i \leq N$, for every rule $l \to r$ in $\mathcal{R}_{i+1} - \mathcal{R}_i$, for every (n,p,q)-decomposition $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, one of the following two conditions holds:*

(i) $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_n)$;
(ii) there exist $C_0, C_1, \ldots, C_k$ and $s_1, \ldots, s_k$ such that
- $r = C_0[s_1, \ldots, s_k]$;

– *for each $1 \leq i \leq k$, $C_i[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ rewrites to s_i in zero or one step of rewrite rule in head position along $\mathcal{R}_i$.*

In the latter case, we say that the context $C = C_0[C_1, \ldots, C_k]$ is associated *to the decomposition D of $l \to r$. Note that $C[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q] \to^*_{\mathcal{R}_i} r$.*

Proposition 1. *Assume that the function* Ctx *in use is* maximal*: for every ϕ and t, if there exists s such that $\phi \vdash s$ and $t \to^*_{\mathcal{R}} s$, then for any Ψ, α, $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha) \neq \bot$. Then, provided that $\mathcal{R}$ is layered convergent, there exists no state (Φ, Ψ) from which $(\Phi, \Psi) \Longrightarrow \bot$ is the only applicable derivation.*

Practical considerations. Unfortunately, such a maximal Ctx is too inefficient in practice as one has to consider the syntactic deducibility problem $\phi \vdash^? s$ for every $t \to^*_{\mathcal{R}} s$. This is why we rather use the following lighter implementation:

– for every index $0 \leq i \leq N$, and every rule $l \to r$ in $\mathcal{R}_{i+1} - \mathcal{R}_i$, if $l = D[l_1, \ldots, l_n, y_1, \ldots, y_{p+q}]$ is a (n, p, q)-decomposition satisfying condition (ii) above for some (arbitrarily chosen) associated context C, then, for every ϕ and σ such that $\phi \vdash l\sigma$, we let

$$\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} r\sigma,\, \Psi,\, (l, r, D, \sigma)) \;=\; C[M_1, \ldots, M_{n+p+q}]$$

where the M_k are fixed recipes such that $(M_i \rhd l_i\sigma) \in \phi$ for $1 \leq i \leq n$ and $(M_{n+j} \rhd y_j\sigma) \in \phi$ for $1 \leq j \leq p + q$;

– otherwise, if $\phi \vdash t\!\downarrow_{\mathcal{R}}$, we let $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha)$ be some fixed M such that $M \rhd_\phi t\!\downarrow_{\mathcal{R}}$;

– in any other case, we let $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha) \;=\; \bot$.

Using similar ideas as for the proof of Proposition 1, we can show that, for any convergent rewrite system $\mathcal{R}$, this choice of Ctx is compatible with property (b) of Subsection 4.2, and more generally with completeness, as long as, during saturation, the transformation rules **A** involve the rewrite rules of $\mathcal{R}_i$ with greater priority than those of $\mathcal{R}_j$, $i < j$. Moreover, when $\mathcal{R}$ is additionally layered, this definition ensures that the procedure never fails. Indeed, using the notations of Figure 1, $\mathrm{Ctx}(\Phi \cup \{z_1 \rhd z_1, \ldots, z_q \rhd z_q\} \vdash^?_{\mathcal{R}} r\sigma,\, \Psi,\, (l, r, D, \sigma)) \;=\; \bot$ implies that (ii) is false on D, thus (i) $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_n)$ holds and $(r\sigma)\!\downarrow_{\mathcal{R}}$ is ground.

Example 6. Any convergent subterm rewrite system $\mathcal{R}$ is layered convergent. Indeed, let $N = 0$ and $\mathcal{R}_1 = \mathcal{R}$. For any $l \to r$ in $\mathcal{R}$ and for every decomposition $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, the term r is a subterm of l, thus either $r = C[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ for some context C, or r is a subterm of some l_i thus $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_n)$.

Example 7. Other examples are provided by the theory of homomorphism $\mathsf{E}_{\mathsf{hom}}$ defined in Section 2.3 as well as the convergent theories of blind signatures $\mathsf{E}_{\mathsf{blind}}$ and prefix encryption $\mathsf{E}_{\mathsf{pref}}$ defined by the following sets of equations.

$$\mathcal{E}_{\mathsf{blind}} = \mathcal{E}_{\mathsf{enc}} \cup \left\{ \begin{array}{c} \mathsf{unblind}(\mathsf{blind}(x, y), y) = x \\ \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x, y), z), y) = \mathsf{sign}(x, z) \end{array} \right\}$$

$$\mathcal{E}_{\mathsf{pref}} = \mathcal{E}_{\mathsf{enc}} \cup \left\{ \mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z)) = \mathsf{enc}(x, z) \right\}$$

The theory $\mathsf{E}_{\mathsf{blind}}$ models primitives used in e-voting protocols [17]. The prefix theory represents the property of many chained modes of encryption (e.g. CBC) where an attacker can retrieve any encrypted prefix out of a ciphertext.

Let us check for instance that the prefix theory $\mathsf{E}_{\mathsf{pref}}$ is layered. Let $N = 1$, $\mathcal{R}_1$ be the rewrite system obtained from $\mathcal{E}_{\mathsf{enc}}$ by orienting the equations from left to right, and $\mathcal{R}_2 = \mathcal{R}_1 \cup \{\mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z)) \to \mathsf{enc}(x, z)\}$. The rewrite rules of $\mathcal{R}_1$ satisfy the assumptions since $\mathcal{R}_1$ forms a convergent subterm rewrite system. The additional rule $\mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z)) \to \mathsf{enc}(x, z)$ admits three decompositions up to permutation of parameters:

- $l = \mathsf{pref}(l_1)$, in which case $\mathrm{var}(r) \subseteq \mathrm{var}(l_1)$;
- $l = \mathsf{pref}(\mathsf{enc}(l_1, z))$, in which case $\mathsf{enc}(\pi_1(l_1), z) \to_{\mathcal{R}_1} r$;
- $l = \mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z))$, in which case $r = \mathsf{enc}(x, z)$.

Verifying that the convergent theories $\mathsf{E}_{\mathsf{hom}}$ and $\mathsf{E}_{\mathsf{blind}}$ are layered is similar.

6.2 Termination

In the previous subsection, we described a sufficient criterion for non-failure. To obtain decidability for a given layered convergent theory, there remains only to provide a termination argument. Such an argument is generally easy to develop by hand as we illustrate on the example of the prefix theory. For the case of existing decidability results from [2], such as the theories of blind signature and homomorphic encryption, we also provide a semantic criterion that allows us to directly conclude termination of the procedure.

Proving termination by hand. To begin with, we note that **B** rules always terminate after a polynomial number of steps. Let us write $\overset{\centerdot}{\Longrightarrow}^n$ for the relation made of exactly n *strict applications* of rules ($S \overset{\centerdot}{\Longrightarrow} S'$ iff $S \Longrightarrow S'$ and $S \neq S'$).

Proposition 2. *For every states $S = (\Phi, \Psi)$ and S' such that $S \overset{\centerdot}{\Longrightarrow}^n S'$ using only **B** rules, n is polynomially bounded in the size of* $\mathrm{im}(\Phi)$.

This is due to the fact that frames are one-to-one and that the rule **B.2** only adds deduction facts $M \rhd t$ such that t is a subterm of an existing term in Φ. Hence, for proving termination, we observe that it is sufficient to provide a function s mapping each frame Φ to a finite set of terms $s(\Phi)$ including the subterms of $\mathrm{im}(\Phi)$ and such that rule **A.2** only adds deduction facts $M \rhd t$ satisfying $t \in s(\Phi)$.

For subterm theories, we obtain polynomial termination by choosing $s(\Phi)$ to be the subterms of $\mathrm{im}(\Phi)$ together with the ground right-hand sides of $\mathcal{R}$.

Proposition 3. *Let E be a convergent subterm theory. For every $S = (\Phi, \Psi)$ and S' such that $S \overset{\centerdot}{\Longrightarrow}^n S'$, n is polynomially bounded in the size of* $\mathrm{im}(\Phi)$.

To conclude that deduction and static equivalence are decidable in polynomial time [2], we need to show that the deduction facts and the equations are of polynomial size. This requires a DAG representation for terms and visible equations. For our implementation, we have chosen not to use DAGs for the sake of simplicity (and perhaps efficiency) since DAGs require much heavier data structures. However, similar techniques as those described in [2] would apply to implement our procedure using DAGs.

For proving termination of the prefix theory, we let $s(\Phi)$ be the minimal set containing Φ, closed by subterm and such that $\mathsf{enc}(t_1, k) \in s(\Phi)$ whenever $\mathsf{enc}(\langle t_1, t_2\rangle, k) \in s(\Phi)$. We then deduce that deduction and static equivalence are decidable for the equational theory $\mathsf{E}_{\mathsf{pref}}$, which is a new decidability result.

A criterion to ensure termination. We now provide a semantic criterion that more generally explains why our procedure succeeds on theories previously known to be decidable [2]. This criterion intuitively states that the set of deducible terms from any initial frame φ should be equivalent to a set of *syntactically* deducible terms. Provided that failures are prevented and assuming a *fair* strategy for rule application, we prove that this criterion is a necessary and sufficient condition for our procedure to terminate.

Definition 5 (fair derivation). *An infinite derivation* $(\Phi_0, \Psi_0) \Longrightarrow \ldots \Longrightarrow (\Phi_n, \Psi_n) \Longrightarrow \ldots$ *is* fair *iff along this derivation,*

(a) ***B*** *rules are applied with greatest priority, and*
(b) whenever a ***A*** *rule is applicable for some instance* $(l \to r, D, t_1, \ldots, t_n, \ldots)$*, eventually the same instance of rule is applied during the derivation.*

Fairness implies that any deducible term is eventually syntactically deducible.

Lemma 4. *Let* $S_0 = (\Phi_0, \Psi_0) \Longrightarrow \ldots \Longrightarrow (\Phi_n, \Psi_n) \Longrightarrow \ldots$ *be an infinite fair derivation from a state* S_0*. For every ground term* t *such that* $\Phi_0 \vdash_{\mathsf{E}} t$*, either* $(\Phi_0, \Psi_0) \Longrightarrow^* \bot$ *or there exists* i *such that* $\Phi_i \vdash t{\downarrow_{\mathcal{R}}}$*.*

Proposition 4 (criterion for saturation). *Let* φ *be an initial frame such that* $\mathrm{Init}(\varphi) \not\Longrightarrow^* \bot$*. The following conditions are equivalent:*

(i) There exists a saturated couple (Φ, Ψ) *such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$*.*
(ii) There exists a (finite) initial frame φ_s *such that for every term* t*,* t *is deducible from* φ *modulo* E *iff* $t{\downarrow_{\mathcal{R}}}$ *is syntactically deducible from* φ_s*.*
(iii) There exists no fair infinite derivation starting from $\mathrm{Init}(\varphi)$*.*

Together with the syntactic criterion described in Section 6.1, this criterion (Property *(ii)*) allows us to prove decidability of deduction and static equivalence for layered convergent theories that belong to the class of *locally stable* theories defined in [2]. As a consequence, our procedure always saturates for the theories of blind signatures and homomorphic encryption since those theories are layered and have been proved locally stable [2]. Other examples of layered convergent theories enjoying this criterion can be found in [2] (e.g. a theory of addition).

7 Implementation: the YAPA tool

YAPA is an Ocaml implementation[4] of the saturation procedure presented in Section 4, using by default the optimized function Ctx defined in Section 6, and a fair strategy of rule application (see Definition 5).

The tool takes as input an equational theory described by a finite convergent rewrite system, as well as frame definitions and queries. A few optimizations may be activated for subterm theories, e.g. to accelerate normalization. The procedure starts by computing the decompositions of the rewrite system. Provided that the rewrite rules are given in an order compatible with the sets $\mathcal{R}_0 \subseteq \ldots \subseteq \mathcal{R}_{N+1}$ of Definition 4, it is able to recognize (fully or partially) layered theories and to pre-compute the associated contexts C related to condition (ii) of this definition, and exploited by the function Ctx in use for eliminating failure cases.

We have conducted several experiments on a PC Intel Core 2 Duo at 2.4 GHz with 2 Go RAM for various equational theories (see below) and found that YAPA provides an efficient way to check static equivalence and deducibility.

Equational theory	$\mathsf{E_{enc}}$ $n = 10$	$\mathsf{E_{enc}}$ $n = 14$	$\mathsf{E_{enc}}$ $n = 16$	$\mathsf{E_{enc}}$ $n = 18$	$\mathsf{E_{enc}}$ $n = 20$	$\mathsf{E_{blind}}$	$\mathsf{E_{pref}}$	$\mathsf{E_{hom}}$	$\mathsf{E_{add}}$
Execution time	< 1s	1,7s	8s	30s	< 3min	< 1s	< 1s	< 1s	< 1s

For the case of $\mathsf{E_{enc}}$, we have run YAPA on the frames $\varphi_n = \{\mathsf{w}_1 \triangleright t_n^0, \mathsf{w}_2 \triangleright \mathsf{c}_0, \mathsf{w}_3 \triangleright \mathsf{c}_1\}$ and $\varphi'_n = \{\mathsf{w}_1 \triangleright t_n^1, \mathsf{w}_2 \triangleright \mathsf{c}_0, \mathsf{w}_3 \triangleright \mathsf{c}_1\}$, where $t_0^i = \mathsf{c}_i$ and $t_{n+1}^i = \langle \mathsf{enc}(t_n^i, \mathsf{k}_n^i), \mathsf{k}_n^i \rangle$, $i \in \{0, 1\}$. These examples allow us to increase the (tree, non-DAG) size of the distinguishing tests exponentially, while the sizes of the frames grow linearly. Despite the size of the output, we have observed satisfactory performances for the tool. We have also experimented YAPA on several convergent theories, e.g. $\mathsf{E_{blind}}$, $\mathsf{E_{hom}}$, $\mathsf{E_{pref}}$ and the theory of addition $\mathsf{E_{add}}$ defined in [2].

In comparison with the tool ProVerif [9, 10], here instrumented to check static equivalences, our test samples suggest a running time between one and two orders of magnitude faster for YAPA. Also we did not succeed in making ProVerif terminate on the two theories $\mathsf{E_{hom}}$ and $\mathsf{E_{add}}$. Of course, these results are not entirely surprising given that ProVerif is tailored for the more general (and difficult) problem of protocol (in)security under active adversaries. In particular ProVerif's initial preprocessing of the rewrite system appears more substantial than ours and does not terminate on the theories $\mathsf{E_{hom}}$ and $\mathsf{E_{add}}$ (although termination is guaranteed for linear or subterm-convergent theories [10]).

Altogether, these results suggest that YAPA significantly improves the state of the art for checking deducibility and static equivalence under convergent theories, both from practical and theoretical perspectives.

References

1. M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static equivalence. In *Foundations of Software Science and Computation Structures (FOSSACS'06)*, pages 398–412, 2006.

[4] Freely available at `http://www.lsv.ens-cachan.fr/~baudet/yapa/`

2. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.
3. 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. ACM, 2001.
4. S. Anantharaman, P. Narendran, and M. Rusinowitch. Intruders with caps. In *18th International Conference on Term Rewriting and Applications (RTA'07)*, volume 4533 of *LNCS*. Springer, 2007.
5. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.
6. M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires.* Thèse de doctorat, LSV, ENS Cachan, France, 2007.
7. M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. Research Report LSV-09-03, Laboratoire Spécification et Vérification, ENS Cachan, France, Feb. 2009. 28 pages.
8. M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 652–663. Springer, 2005.
9. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.
10. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.
11. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *18th IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.
12. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.
13. R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. In *2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04)*, ENTCS, 2004.
14. V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, LNAI. Springer, 2007.
15. V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.
16. S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, 2004.
17. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2008. To appear.
18. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.
19. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.