

# Efficient Decision Procedures for Message Deducibility and Static Equivalence

Bruno Conchinha<sup>1</sup>, David Basin<sup>1</sup>, and Carlos Caleiro<sup>2</sup>

<sup>1</sup> Information Security Group, ETH Zürich, Zürich, Switzerland  
brunoco@inf.ethz.ch, basin@inf.ethz.ch

<sup>2</sup> SQIG - Instituto de Telecomunicações, Department of Mathematics,  
IST, TU Lisbon, Portugal  
ccal@math.ist.utl.pt

**Abstract.** We consider two standard notions in formal security protocol analysis: message deducibility and static equivalence under equational theories. We present new polynomial-time algorithms for deciding both notions under subterm convergent equational theories and under a theory representing symmetric encryption with the prefix property. For these equational theories, polynomial-time algorithms for the decision problems associated to both notions are well-known (although this has not been proven for static equivalence under the prefix theory). However, our algorithms have a significantly better asymptotic complexity than existing approaches.

As an application, we use our algorithm for static equivalence to discover off-line guessing attacks on the Kerberos protocol when implemented using a symmetric encryption scheme for which the prefix property holds.

**Keywords:** security protocols, equational theories, deducibility, static equivalence.

## 1 Introduction

Formal methods and associated tools are now well established for analyzing security protocols. In symbolic approaches, the messages exchanged by agents are represented by terms in an algebra and properties of cryptographic operators are formalized equationally. This sets the scene for different analysis approaches.

Message deducibility and static equivalence are two relations, formulated in the equational setting, with direct applications to security protocol analysis. Procedures for analyzing the security of cryptographic protocols use algorithms for deduction [6, 5, 17, 14], and static equivalence has been used to study cryptographic indistinguishability [9] and to model and reason about off-line guessing attacks [20, 7, 1].

*Contributions.* In this paper, we give new algorithms for deciding both deduction and static equivalence in two practically relevant cases: subterm convergent

theories and theories for symmetric encryption with the prefix property. Our algorithms have better asymptotic complexity than existing approaches. Our algorithms for the prefix theory are a simple generalization of those for subterm convergent theories.

As an application, we analyze the security of the Kerberos protocol against off-line guessing attacks. Although it was previously known that Kerberos admits an off-line guessing attack [24], we are able to find numerous new attacks by considering an implementation with a symmetric encryption scheme for which the prefix property holds, *e.g.*, if ECB or CBC modes of operation are used. Identifying such attacks highlights design and implementation issues (such as message ordering, modes, etc.) that may not appear important. Automating such analysis requires static equivalence procedures for different theories and highlights the importance of efficient, automated methods for this task.

*Background and Related Work.* The notions of deduction and static equivalence that we consider were originally introduced in the context of the pi-calculus [4].

Considerable work has been devoted to proving the decidability of both deduction and static equivalence for a wide range of equational theories. [2] gives polynomial-time algorithms for both decision problems under equational theories generated by subterm convergent rewriting systems. For other equational theories there are few polynomial-time results; however, decidability results exist under fairly general conditions, *e.g.*, [3, 14, 15, 16, 17, 18, 22, 23, 25, 26].

Despite the considerable prior work in this area, there have been few implementations of these algorithms, particularly for static equivalence. For some time, ProVerif [10, 11] was the only tool capable of deciding static equivalence. General algorithms for deduction and static equivalence have more recently been implemented by the YAPA [8] and KISS [16] tools. Although the precise set of equational theories and conditions under which these algorithms terminate is unclear, they seem able to handle most theories previously studied. We provide a detailed comparison of our algorithms with those implemented by these tools.

Even less attention has been devoted to improving the efficiency of algorithms for these problems. Given the interest in tools for solving these problems, complexity issues are not only theoretically interesting but also practically relevant.

*Organization.* In Section 2, we introduce basic definitions and notation. In Section 3, we present our algorithms and their properties. In Section 4, we extend the methods of Section 3 to handle symmetric encryption with the prefix property. As a case study, we analyze the security of the Kerberos protocol against off-line guessing attacks. We draw conclusions in Section 5. For reasons of space and readability, full proofs are given in the technical report [19].

## 2 Background and Basic Definitions

Given a function  $g$ , we denote by  $dom(g)$  and by  $ran(g)$  its domain and range, respectively. When  $X \subseteq dom(g)$ , we write  $g[X]$  for the image of  $X$  under  $g$ .

We consider signatures  $\Sigma = \biguplus_{n \in \mathbb{N}} \Sigma_n$  consisting of finitely many function symbols, where  $\Sigma_i$  contains the functions symbols of arity  $i$ . For each  $f \in \Sigma$ , the function  $\text{ar}: \Sigma \rightarrow \mathbb{N}$  returns the arity  $\text{ar}(f)$  of  $f$ . Function symbols are used to represent publicly known functions over messages, including cryptographic operators. Constant function symbols (*i.e.*, symbols in  $\Sigma_0$ ) represent publicly known constants. We also fix infinite, disjoint sets  $\mathbf{Var}$  and  $\mathbf{Name}$  of variables and names, and assume that  $x, y, z \in \mathbf{Var}$  and that  $\{x_i \mid i \in \mathbb{N}\} \subseteq \mathbf{Var}$ .

*Example 1.* The signature  $\Sigma^{\mathcal{DY}}$ , representing a Dolev-Yao model with an hash function  $\mathbf{h}$ , a pairing function  $\mathbf{pair}$ , the projections  $\pi_1$  and  $\pi_2$ , and symmetric and asymmetric encryption and decryption, is given by  $\Sigma^{\mathcal{DY}} = \Sigma_1^{\mathcal{DY}} \cup \Sigma_2^{\mathcal{DY}}$ , where  $\Sigma_1^{\mathcal{DY}} = \{\mathbf{h}, \pi_1, \pi_2, \mathbf{pub}, \mathbf{priv}\}$  and  $\Sigma_2^{\mathcal{DY}} = \{\mathbf{enc}_{\text{sym}}, \mathbf{dec}_{\text{sym}}, \mathbf{pair}, \mathbf{enc}_{\text{asym}}, \mathbf{dec}_{\text{asym}}\}$ .

The functions  $\mathbf{pub}$  and  $\mathbf{priv}$  represent the generation of public and private keys. We will use the following abbreviations:  $x_{\text{pub}}$  for  $\mathbf{pub}(x)$ ;  $x_{\text{priv}}$  for  $\mathbf{priv}(x)$ ;  $\langle x, y \rangle$  for  $\mathbf{pair}(x, y)$ ;  $\{P\}_K^s$  for  $\mathbf{enc}_{\text{sym}}(P, K)$ ;  $\{C\}_K^{s,-1}$  for  $\mathbf{dec}_{\text{sym}}(C, K)$ ;  $\{P\}_K$  for  $\mathbf{enc}_{\text{asym}}(P, K)$ ;  $\{C\}_K^{-1}$  for  $\mathbf{dec}_{\text{asym}}(C, K)$ ; and  $\langle x_1, \dots, x_n \rangle$  for  $\langle \dots \langle x_1, x_2 \rangle \dots, x_n \rangle$ .

As usual, given a set  $X$ ,  $T(\Sigma, X)$  is the set of  $\Sigma$ -terms over  $X$ , *i.e.*, the smallest set such that  $X \subseteq T(\Sigma, X)$  and  $f(t_1, \dots, t_n) \in T(\Sigma, X)$  for all  $t_1, \dots, t_n \in T(\Sigma, X)$  and all  $f \in \Sigma_n$ . We use the symbol  $=$  to denote syntactic equality. Given  $t \in T(\Sigma, X)$ , we define the set  $\text{sub}(t)$  of subterms of  $t$  as usual: if  $t \in X$ , then  $\text{sub}(t) = \{t\}$ ; if  $t = f(t_1, \dots, t_n)$  for some  $f \in \Sigma_n$  and  $t_1, \dots, t_n \in T(\Sigma, X)$ , then  $\text{sub}(t) = \{t\} \cup \bigcup_{i=1}^n \text{sub}(t_i)$ . We denote by  $\text{vars}(t) = \text{sub}(t) \cap \mathbf{Var}$  the set of variables occurring in  $t$ .

We use the standard notion of substitution as a partial function  $\sigma: \mathbf{Var} \dashrightarrow T(\Sigma, X)$ . We abuse notation by using the same symbol  $\sigma$  for a substitution and its homomorphic extension to  $T(\Sigma, X)$ , where  $\text{dom}(\sigma) \subseteq X$ . As usual, we write  $t\sigma$  instead of  $\sigma(t)$ .

A frame is a pair  $(\tilde{n}, \sigma)$ , written  $v\tilde{n}.\sigma$ , where  $\tilde{n} \subseteq \mathbf{Name}$  is a finite set of names and  $\sigma: \mathbf{Var} \dashrightarrow T(\Sigma, \mathbf{Name})$  is a substitution with finite domain. Intuitively, names in  $\tilde{n}$  represent fresh data generated by agents and thus unavailable to the attacker, while  $\sigma$  represents the messages learned by the attacker by eavesdropping on the network. Given a frame  $\phi = v\tilde{n}.\sigma$ , we define  $T_\phi = T(\Sigma, (\mathbf{Name} \setminus \tilde{n}) \cup \text{dom}(\sigma))$ . We say that terms in  $T_\phi$  are  $\phi$ -recipes. The terms in  $\sigma[T_\phi]$  are the concrete terms that the attacker can obtain and we refer to them as terms *constructible* from  $\phi$ .

A rewriting system  $R$  over  $\Sigma$  is a set of rewrite rules of the form  $l \rightarrow r$ , where  $l, r \in T(\Sigma, \mathbf{Var})$ . We assume that rewriting systems have only finitely many rules. Given a rewriting system  $R$ , we define the relation  $\rightarrow_R \subseteq T(\Sigma, \mathbf{Name}) \times T(\Sigma, \mathbf{Name})$  as the smallest relation such that:

- if  $(l \rightarrow r) \in R$  and  $\sigma: \text{vars}(l) \rightarrow T(\Sigma, \mathbf{Name})$  is a substitution, then  $l\sigma \rightarrow_R r\sigma$ , and
- if  $t_1, \dots, t_n, t'_i \in T(\Sigma, \mathbf{Name})$ ,  $t_i \rightarrow_R t'_i$ , and  $f \in \Sigma_n$ , then  $f(t_1, \dots, t_i, \dots, t_n) \rightarrow_R f(t_1, \dots, t'_i, \dots, t_n)$ .

We drop the  $R$  from  $\rightarrow_R$  when it is clear from context.

A rewriting system  $R$  is convergent if it is terminating and confluent. In this case, each term  $t$  has a unique normal form  $t \downarrow_R \in T(\Sigma, \text{Name})$ . Given a convergent rewriting system  $R$ , we define  $\approx_R \subseteq T(\Sigma, \text{Name}) \times T(\Sigma, \text{Name})$  as the relation such that  $t \approx_R t'$  if and only if  $t \downarrow_R = t' \downarrow_R$ . Note that we adopt the usual convention of writing  $t \approx_R t'$  instead of  $(t, t') \in \approx_R$ . It is simple to check that  $\approx_R$  is an equational theory (i.e., an equivalence relation closed under the application of contexts). We call  $\approx_R$  the *equational theory generated by  $R$* . A rewriting system  $R$  is subterm convergent if it is convergent and, for each  $(l \rightarrow r) \in R$ , either  $r \in \text{sub}(l)$  or  $r \in T(\Sigma, \emptyset)$  is a term in normal form. Permitting terms in  $T(\Sigma, \emptyset)$  on the right-hand side follows [8].

*Example 2.* The rewriting system  $R_{\mathcal{DY}}$  over  $\Sigma^{\mathcal{DY}}$ , formalizing the standard capabilities of the Dolev-Yao intruder, is given by

$$R_{\mathcal{DY}} = \left\{ \pi_1(\langle x, y \rangle) \rightarrow x, \pi_2(\langle x, y \rangle) \rightarrow y, \left\{ \{x\}_y^s \right\}_y^{s,-1} \rightarrow x, \left\{ \{x\}_{y_{\text{pub}}} \right\}_{y_{\text{priv}}}^{-1} \rightarrow x \right\}.$$

The rewriting system  $R_{\mathfrak{p}}$ , given by  $R_{\mathfrak{p}} = R_{\mathcal{DY}} \cup \{ \pi_1(\langle \{x, y\}_z^s \rangle) \rightarrow \{x\}_z^s \}$ , represents a Dolev-Yao attacker in the presence of symmetric encryption satisfying the prefix property.  $R_{\mathcal{DY}}$  and  $R_{\mathfrak{p}}$  are convergent rewriting systems. However, only  $R_{\mathcal{DY}}$  is subterm convergent. For readability, we write  $\approx_{\mathcal{DY}}$  and  $\approx_{\mathfrak{p}}$  instead of  $\approx_{R_{\mathcal{DY}}}$  and  $\approx_{R_{\mathfrak{p}}}$ , respectively.

Our definitions of deduction and static equivalence differ slightly from those introduced in [4] and used, e.g., in [2, 3, 20]. However, they are equivalent to the original ones and fit our methods better.

**Definition 1.** *Given a frame  $\phi$ , a term  $t \in T(\Sigma, \text{Name})$ , and an equational theory  $\approx$ , we say that  $t$  is deducible from  $\phi$  under  $\approx$ , and write  $\phi \vdash_{\approx} t$ , if there is a  $t' \in \sigma[T_{\phi}]$  such that  $t' \approx t$ .*

The equational theories  $\approx$  that we use are those generated by rewriting systems  $R$ ; thus, we write  $\phi \vdash_R t$  instead of  $\phi \vdash_{\approx_R} t$ .

**Definition 2.** *Given two frames  $\phi = v\tilde{n}.\sigma$  and  $\phi' = v\tilde{n}'.\sigma'$  and an equational theory  $\approx$ , we say that  $\phi$  and  $\phi'$  are statically equivalent under  $\approx$ , and write  $\phi \approx^s \phi'$ , if  $T_{\phi} = T_{\phi'}$  (i.e.,  $\tilde{n} = \tilde{n}'$  and  $\text{dom}(\sigma) = \text{dom}(\sigma')$ ) and, for all  $t, t' \in T_{\phi}$ ,  $t\sigma \approx t'\sigma'$  if and only if  $t\sigma \approx t'\sigma'$ .*

The corresponding decision problems are defined as expected.

**Definition 3 (Deduction Problem).** *Given a frame  $\phi$ , an equational theory  $\approx$ , and a term  $t$ , does  $\phi \vdash_{\approx} t$  hold?*

**Definition 4 (Static Equivalence Problem).** *Given frames  $\phi$  and  $\phi'$  and an equational theory  $\approx$ , does  $\phi \approx^s \phi'$  hold?*

Static equivalence is well-suited for modeling off-line guessing attacks [1, 7, 20]. Suppose that a nonce  $g$  has low entropy: for example,  $g$  is a human-picked password. Then, an attacker may choose a small set of bitstrings with a high

probability of containing the bitstring represented by  $g$ . The attacker can then use each of these bitstrings as his guess for the password. The attack is successful if he can verify which of these guesses is correct. The following definition, in the spirit of [7, 20], captures this intuition.

**Definition 5.** Let  $\approx$  be an equational theory,  $\phi = v\tilde{n}.\sigma$  be a frame, and  $g \in \text{Name}$  be a name. Fix some fresh name  $w \in \text{Name} \setminus (\text{sub}(\text{ran}(\sigma)) \cup \{g\})$  and define  $\phi_g$  and  $\phi_w$  by

$$\begin{aligned}\phi_g &= v(\tilde{n} \cup \{w\}).(\sigma \cup \{x_{n+1} \mapsto g\}), \\ \phi_w &= v(\tilde{n} \cup \{w\}).(\sigma \cup \{x_{n+1} \mapsto w\}).\end{aligned}$$

We say that  $\phi$  allows an off-line guessing attack of  $g$  under  $\approx$  if  $\phi_g \not\approx^s \phi_w$ .

Note that this definition is independent of the particular choice of the name  $w$ . Section 4.1 presents an application of static equivalence to the study of off-line guessing attacks.

In order to obtain polynomial complexity bounds for our algorithms, we will work with DAG (directed acyclic graph) representations of terms, as in [2].

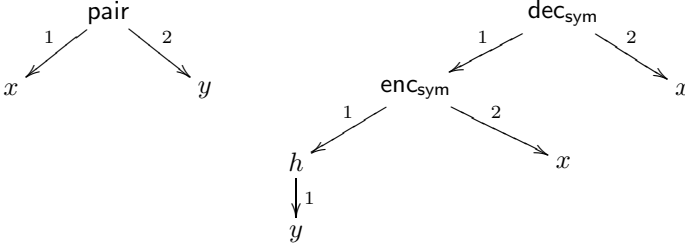
**Definition 6.** Let  $t \in T(\Sigma, X)$  be a term. Let  $V$  be a set of vertices,  $\lambda: V \rightarrow \Sigma \cup X$  a labeling function,  $E \subseteq V \times V \times \mathbb{N}$  a set of edges, and  $v \in V$  a vertex.

If  $t \in X$ , then  $(V, \lambda, E, v)$  is a DAG-representation of  $t$  if  $\lambda(v) = t$  and  $(v, v', n) \notin E$  for all  $v' \in V$  and all  $n \in \mathbb{N}$ . If  $t = f(t_1, \dots, t_n)$ , then  $(V, \lambda, E, v)$  is a DAG-representation of  $t$  if  $\lambda(v) = f$ , there are edges  $(v, v_1, 1), \dots, (v, v_n, n) \in E$  such that, for each  $i \in \{1, \dots, n\}$ ,  $(V, \lambda, E, v_i)$  is a DAG-representation of  $t_i$ , and there are no other edges  $(v, v', m) \in E$  for any  $v' \in V$  and any  $m \in \mathbb{N}$ .

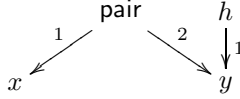
A tuple  $\mathcal{T} = (V, \lambda, E)$  is a DAG-forest if, for all  $v \in V$ ,  $(V, \lambda, E, v)$  represents some term  $t \in T(\Sigma, X)$ . If  $\mathcal{T} = (V, \lambda, E)$  is a DAG-forest and  $v \in V$ , we use the following notions:  $\text{term}_{\mathcal{T}}(v)$  is the (unique) term represented by  $(V, \lambda, E, v)$ ;  $e_{i, \mathcal{T}}(v)$  is the (only)  $v' \in V$  such that  $(v, v', i) \in E$ ;  $\text{in}_{\mathcal{T}}(v) = \{w \in V \mid (w, v, i) \in E, \text{ for some } i\}$ ;  $\text{out}_{\mathcal{T}}(v) = \{w \in V \mid (v, w, i) \in E, \text{ for some } i\}$ ;  $\text{leaves}(\mathcal{T}) = \{v \in V \mid \text{out}_{\mathcal{T}}(v) = \emptyset\}$ ;  $\text{roots}(\mathcal{T}) = \{v \in V \mid \text{in}_{\mathcal{T}}(v) = \emptyset\}$ . If  $\mathcal{T}$  has only one root, we may refer to it as a DAG-tree.

Let  $\mathcal{T} = (V, \lambda, E)$  be a DAG-forest. If  $\text{roots}(\mathcal{T}) = \{v\}$ , we say that  $\mathcal{T}$  is a DAG-representation of the term  $\text{term}_{\mathcal{T}}(v)$ . When no confusion can arise, we may abuse notation and use the same symbol for such a DAG-tree and the term represented by it.  $\mathcal{T}$  is *minimal* if there are no distinct vertices  $v, v' \in V$  such that  $\text{term}_{\mathcal{T}}(v) = \text{term}_{\mathcal{T}}(v')$ .  $\mathcal{T}$  is in normal form if, for each vertex  $v$  in  $\mathcal{T}$ ,  $\text{term}_{\mathcal{T}}(v)$  is in normal form. A *normal form* of  $\mathcal{T}$  is a DAG-forest  $\mathcal{T}_{\text{nf}}$  such that, for all  $v \in \text{roots}(\mathcal{T})$ , there is a vertex  $v_{\text{nf}}$  in  $\mathcal{T}_{\text{nf}}$  such that  $\text{term}_{\mathcal{T}}(v) \downarrow = \text{term}_{\mathcal{T}_{\text{nf}}}(v_{\text{nf}})$ , and for each  $v_{\text{nf}} \in \text{roots}(\mathcal{T}_{\text{nf}})$ , there is a  $v \in \text{roots}(\mathcal{T})$  such that  $\text{term}_{\mathcal{T}}(v) \downarrow = \text{term}_{\mathcal{T}_{\text{nf}}}(v_{\text{nf}})$ . The minimal normal form of a DAG-forest is unique up to renaming of vertices.

*Example 3.* The diagram



depicts a DAG-forest containing DAG-representations of the terms  $\langle x, y \rangle$  and  $\{\{h(y)\}_x^s\}_x^{s,-1}$ . Its minimal normal form is shown below.



Our complexity results depend on the rewriting system and are stated in terms of the size of terms and frames. If  $t \in T(\Sigma, \text{Name})$  is a term, then the *size*  $|t|$  of  $t$  is 1 if  $t \in X$  and  $1 + \sum_{i=1}^n |t_i|$  if  $t = f(t_1, \dots, t_n)$ . If  $\phi = v\tilde{n}.\sigma$  is a frame, then the *size*  $|\phi|$  of  $\phi$  is given by  $|\phi| = \sum_{x \in \text{dom}(\sigma)} |x\sigma|$ . If  $\mathcal{T} = (V, \lambda, E)$  is a DAG-forest, we define  $|\mathcal{T}| = |V|$ . If  $R$  is a rewriting system, we define  $nvars(R) = \max_{(l \rightarrow r) \in R} |vars(l)|$ . In order to take advantage of DAGs, we always assume a random-access machine model in our complexity analysis.

### 3 Decision Procedures for Subterm Convergent Rewriting Systems

Throughout this section we assume fixed a subterm convergent rewriting system  $R$ , such that  $nvars(R) \geq 1$ , and a frame  $\phi = v\tilde{n}.\sigma$ , such that  $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$  and the terms  $t_i$  are all in normal form. We also fix a set  $\mathcal{Y} = \{\tau_1, \dots, \tau_{nvars(R)}\}$  of fresh names, *i.e.*,  $\mathcal{Y} \cap (\tilde{n} \cup \text{sub}(\text{ran}(\sigma))) = \emptyset$ .

#### 3.1 Frame Saturation

In this section, we present our frame saturation algorithm. Frame saturation is an established technique for deciding both deduction and static equivalence [2,8,16]. Our procedure is less general than those implemented in [8,16], but it is more efficient for subterm convergent equational theories.

**Definition 7.** *We say that  $t$  is a  $\phi$ -recipe (under  $R$ ) for  $t'$  if  $t$  is a  $\phi$ -recipe and  $t\sigma \approx_R t'$ . We say that a frame  $\phi_s = v\tilde{n}.\sigma_s$  is a saturation of  $\phi$  (under  $R$ ) if there is a  $\phi$ -recipe for each  $t \in \text{ran}(\sigma_s)$  and, for all  $t \in T_\phi$ ,  $(t\sigma)\downarrow \in \sigma_s[T_{\phi_s}]$ .*

The following simple lemma justifies the usefulness of saturation.

**Lemma 1.** *Let  $\phi_s$  be a saturation of  $\phi$  and  $t \in T(\Sigma, \text{Name})$  be a term. Then,  $\phi \vdash_R t$  if and only if  $t \downarrow \in \sigma_s[T_{\phi_s}]$ .*

The first step in our saturation algorithm is to build a forest  $\mathcal{T}_{\phi,R} = (V_{\phi,R}, \lambda_{\phi,R}, E_{\phi,R})$  (line 1 in Algorithm 1) and compute functions  $\zeta$  and  $\text{rw}$ . Algorithms for computing  $\mathcal{T}_{\phi,R}$ ,  $\text{rw}$  and  $\zeta$  are given in the technical report [19]. Lemma 2 summarizes their relevant properties.

**Lemma 2.** *The forest  $\mathcal{T}_{\phi,R} = (V_{\phi,R}, \lambda_{\phi,R}, E_{\phi,R})$  and the functions  $\text{rw}$  and  $\zeta$  are such that:*

- (1)  $\mathcal{T}_{\phi,R}$  is minimal, can be obtained in time  $\mathcal{O}(|\phi|^{n\text{vars}(R)} \log^2 |\phi|)$ , and  $|\mathcal{T}_{\phi,R}| \in \mathcal{O}(|\phi|^{n\text{vars}(R)})$ ;
- (2)  $\text{rw}$  can be computed in time  $\mathcal{O}(\log |\phi|)$ ;
- (3)  $\zeta$  can be computed in time  $\mathcal{O}(\log |\phi|)$ ; for each  $v \in \text{dom}(\zeta)$ ,  $|\zeta(v)| = 1$ ;
- (4) for each  $s \in \text{sub}(\text{ran}(\sigma)) \cup \mathcal{Y}$ , there is a unique  $v$  such that  $\text{term}_{\mathcal{T}_{\phi,R}}(v) = s$ ;
- (5) for each  $v \in \text{dom}(\text{rw})$ ,  $\text{term}_{\mathcal{T}_{\phi,R}}(v) \rightarrow_R \text{term}_{\mathcal{T}_{\phi,R}}(\text{rw}(v))$ ;
- (6) for each  $t \in \text{ran}(\sigma) \cup \mathcal{Y}$ , there is a  $v$  such that  $\text{term}_{\mathcal{T}_{\phi,R}}(v) = t$  and  $v \in \text{dom}(\zeta)$ ;
- (7) for each  $v \in \text{dom}(\zeta)$ ,  $\text{term}_{\zeta(v)}(v)$  is a  $\phi$ -recipe for  $\text{term}_{\mathcal{T}_{\phi,R}}(v)$ ;
- (8) for each rule  $(l \rightarrow r) \in R$  and each substitution  $\sigma_l: \text{vars}(l) \rightarrow \text{sub}(\text{ran}(\sigma)) \cup \mathcal{Y}$ , there is a unique  $v \in V_{\phi,R} \cap \text{dom}(\text{rw})$  such that  $\text{term}_{\mathcal{T}_{\phi,R}}(v) = l\sigma_l$  and  $\text{term}_{\mathcal{T}_{\phi,R}}(\text{rw}(v)) = r\sigma_l$ .

Our saturation algorithm traverses the forest  $\mathcal{T}_{\phi,R}$  bottom-up. Whenever a recipe for a term  $\text{term}_{\mathcal{T}_{\phi,R}}(v)$  is found,  $v$  is added to the range of  $\zeta$  (with  $\zeta(v)$  being a DAG-representation of a  $\phi$ -recipe for that term), and all vertices  $v'$  with an outgoing edge  $(v', v, i)$  to  $v$  are added to the list of vertices to visit in the next iteration of the visiting loop. At the end of the process, a term  $t \in \text{sub}(\text{ran}(\sigma))$  is deducible from  $\phi$  if and only if the (unique) vertex representing that term is in the domain of  $\zeta$ .

The algorithm also stores the functions  $\sigma_s$  and  $\zeta_s$ .  $\sigma_s$  is such that  $\phi_s = v\tilde{n}.\sigma_s$  is a saturation of  $\phi$ , and  $\zeta_s$  is such that  $\text{dom}(\zeta_s) = \text{dom}(\sigma_s)$  and, for each  $x \in \text{dom}(\sigma_s)$ ,  $\zeta_s(x)$  is a DAG-representation of a  $\phi$ -recipe for  $x\sigma_s$ .

Furthering our abuse of notation, we use the symbol  $\zeta_s$  as the substitution that assigns, to each  $x \in \text{dom}(\zeta_s)$ , the term represented by (the DAG-forest)  $\zeta_s(x)$ . In this case, we use postfix notation and write  $x\zeta_s$ .

The tree  $\mathcal{T}_{\phi,R}$  has at most  $\mathcal{O}(|\phi|^{n\text{vars}(R)})$  vertices, and each vertex  $v \in V_{\phi,R}$  is visited at most  $|\text{in}_{\mathcal{T}_{\phi,R}}(v)|$  times. Thus, the total number of visits to vertices is at most  $\mathcal{O}(|\phi|^{n\text{vars}(R)})$ . By using suitable data structures, we can ensure that each visit takes at most time  $\mathcal{O}(\log^2 |\phi|)$ . We thus obtain an asymptotic complexity of  $\mathcal{O}(|\phi|^{n\text{vars}(R)} \log^2 |\phi|)$ .

We state the algorithm's properties and complexity in Lemma 3.

**Algorithm 1 (Saturating a Frame)****Input:** a frame  $\phi = v\tilde{n}.\sigma$ , with  $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ **Output:** a saturation  $\phi_s = v\tilde{n}.\sigma_s$  of  $\phi$  and a function  $\zeta_s$ 

```

1: compute  $\mathcal{T}_{\phi,R} = (V_{\phi,R}, \lambda_{\phi,R}, E_{\phi,R}), \text{rw}, \zeta$ 
2:  $\zeta_s \leftarrow \{x \mapsto (\{v_x\}, \{v_x \mapsto x\}, \emptyset) \mid x \in \text{dom}(\sigma)\}$ ,
   where the  $v_x$  are such that  $\text{term}_{\mathcal{T}_{\phi,R}}(v_x) = x\sigma$ 
3:  $\sigma_s \leftarrow \sigma$ 
4:  $\text{visitnow} \leftarrow \text{leaves}(\mathcal{T}_{\phi,R}) \cup (\bigcup_{v \in \text{dom}(\zeta)} \text{in}_{\mathcal{T}_{\phi,R}}(v))$ ,  $\text{visitnext} \leftarrow \emptyset$ 
5: while  $\text{visitnow} \neq \emptyset$ 
6:   for all  $v \in \text{visitnow}$ 
7:     if  $\lambda(v) \in X \setminus \tilde{n}$  and  $v \notin \text{dom}(\zeta)$  then
8:        $\zeta \leftarrow \zeta \cup \{v \mapsto (v, \{v \mapsto \lambda(v)\}, \emptyset)\}$ 
9:        $\text{visitnext} \leftarrow \text{visitnext} \cup \text{in}_{\mathcal{T}_{\phi,R}}(v)$ 
10:    if  $\text{out}_{\mathcal{T}_{\phi,R}}(v) \subseteq \text{dom}(\zeta)$  and  $v \notin \text{dom}(\zeta)$  then
11:      let  $(\tilde{V}_i, \lambda_i, E_i) = \zeta(e_{i,\mathcal{T}_{\phi,R}}(v))$  for  $i \in \{1, \dots, \text{ar}(\lambda_{\phi,R}(v))\}$ 
       $\zeta \leftarrow \zeta \cup \{v \mapsto (v \cup \bigcup_{i=1}^{\text{ar}(\lambda_{\phi,R}(v))} \tilde{V}_i,$ 
12:         $\{v \mapsto \lambda(v)\} \cup \bigcup_{i=1}^{\text{ar}(\lambda_{\phi,R}(v))} \lambda_i,$ 
         $\bigcup_{i=1}^{\text{ar}(\lambda_{\phi,R}(v))} \{(v, e_{i,\mathcal{T}_{\phi,R}}(v), i)\} \cup \bigcup_{i=1}^{\text{ar}(\lambda_{\phi,R}(v))} E_i)\}$ 
13:    if  $v \in \text{dom}(\text{rw})$  and  $\text{rw}(v) \notin \text{dom}(\zeta)$  then
14:       $\zeta \leftarrow \zeta \cup \{\text{rw}(v) \mapsto \zeta(v)\}$ 
15:      if  $\text{term}_{\mathcal{T}_{\phi,R}}(\text{rw}(v)) \in \text{sub}(\text{ran}(\sigma))$ 
16:        then choose  $x \in \text{Var} \setminus \text{dom}(\sigma_s)$ 
17:           $\sigma_s \leftarrow \sigma_s \cup \{x \mapsto \text{term}_{\mathcal{T}_{\phi,R}}(\text{rw}(v))\}$ 
18:           $\zeta_s \leftarrow \zeta_s \cup \{x \mapsto \zeta(\text{rw}(v))\}$ 
19:           $\text{visitnext} \leftarrow \text{visitnext} \cup \text{in}_{\mathcal{T}_{\phi,R}}(\text{rw}(v))$ 
20:        else  $\text{visitnext} \leftarrow \text{visitnext} \cup \text{in}_{\mathcal{T}_{\phi,R}}(v)$ 
21: return  $\zeta_s, \phi_s = v\tilde{n}.\sigma_s$ 

```

**Lemma 3.** Algorithm 1 terminates in time  $\mathcal{O}(|\phi|^{n\text{vars}(R)} \log^2 |\phi|)$ .

$\phi_s$  is a saturation of  $\phi$  (under  $R$ ),  $\text{dom}(\zeta_s) = \text{dom}(\sigma_s)$ , and, for each  $x \in \text{dom}(\sigma_s)$ ,  $\zeta_s(x) \in T_\phi$  and  $\zeta_s(x)$  is a DAG-representation of a  $\phi$ -recipe for  $x\sigma_s$  with size  $|\zeta_s(x)| \in \mathcal{O}(|\phi|)$ .

For each  $v \in \text{dom}(\zeta)$ , there is a  $\phi_s$ -recipe  $t$  for  $\text{term}_{\mathcal{T}_{\phi,R}}(v)$  such that  $\zeta(v) = t\zeta_s$  is a  $\phi$ -recipe for  $\text{term}_{\mathcal{T}_{\phi,R}}(v)$ . If  $\text{term}_{\mathcal{T}_{\phi,R}}(v) \in \sigma_s[T_{\phi_s}]$ , then  $v \in \text{dom}(\zeta)$ .

**3.2 Decision Procedure for Deduction**

In light of Lemma 1, to solve the deduction problem under  $R$  for a frame  $\phi$  and a term  $t$ , it suffices to (1) compute  $t \downarrow_R$ , (2) compute the saturated frame  $\phi_s = v\tilde{n}.\sigma_s$  (using Algorithm 1), and (3) decide whether  $t \downarrow_R \in \sigma_s[T_{\phi_s}]$ . In the technical report [19] we show that the complexities of these three computations are, respectively,  $\mathcal{O}(|t| \log^2 |t|)$ ,  $\mathcal{O}(|\phi|^{n\text{vars}(R)} \log^2 |\phi|)$ , and  $\mathcal{O}(|t| |\phi|^2)$ . Theorem 1 sums up these observations.

**Theorem 1.** Given a frame  $\phi$  and a term  $t$ , the complexity of deciding whether  $\phi \vdash_R t$  is at most  $\mathcal{O}(|t| \log^2 |t| + |t| |\phi|^2 + |\phi|^{n\text{vars}(R)} \log^2 |\phi|)$ .



### 3.3 Decision Procedure for Static Equivalence

Throughout this section,  $\mathcal{T}_{\phi,R}$  is as described in the previous sections,  $\phi_s$  and  $\zeta_s$  are as output by Algorithm 1, and  $\phi' = \nu\tilde{n}.\sigma'$  is a frame such that  $\text{dom}(\sigma') = \text{dom}(\sigma) = \{x_1, \dots, x_n\}$ . We assume that  $\text{dom}(\sigma_s) = \{x_1, \dots, x_{m_s}\}$  and that  $\sigma_s$  is an extension of  $\sigma$ .  $\mathcal{Y}$  will be used as in the previous section.

Algorithm 2 summarizes our procedure for deciding static equivalence. Note that some of the operations performed by this algorithm must use the DAG-representation of terms to ensure polynomial-time complexity. We refer to the technical report [19] for these details.

#### Algorithm 2 (Decision Procedure for $\approx_R^s$ )

**Input:** two frames  $\phi = \nu\tilde{n}.\sigma$  and  $\phi' = \nu\tilde{n}.\sigma'$   
such that  $\text{dom}(\sigma) = \text{dom}(\sigma') = \{x_1, \dots, x_n\}$

**Output:** **true** if  $\phi \approx_R^s \phi'$  and **false** otherwise

- 1: compute  $\mathcal{T}_{\phi,R}$ ,  $\zeta$ ,  $\text{rw}$ ,  $\zeta_s$  and  $\phi_s$
- 2: choose a permutation  $\pi: \{1, \dots, m_s\} \rightarrow \{1, \dots, m_s\}$   
such that  $1 \leq i < j \leq m_s \Rightarrow |x_{\pi(i)}\sigma_s| \leq |x_{\pi(j)}\sigma_s|$
- 3: for each  $k \in \{1, \dots, m_s\}$ , let  
 $\phi_{s,k} = \nu\tilde{n}. \{x_{\pi(1)} \mapsto x_{\pi(1)}\sigma_s, \dots, x_{\pi(k)} \mapsto x_{\pi(k)}\sigma_s\}$
- 4: **for all**  $k \in \{1, \dots, m_s\}$
- 5:     **if**  $x_{\pi(k)}\sigma_s \in \sigma_s[T_{\phi_s, k-1}]$  **then**
- 6:         choose  $t \in T_{\phi_s, k-1}$  such that  $x_{\pi(k)}\sigma_s = t\sigma_s$
- 7:         **if**  $x_{\pi(k)}\zeta_s\sigma' \not\approx_R t\zeta_s\sigma'$  **then return false**
- 8: **for all**  $v \in \text{dom}(\text{rw})$
- 9:     **then if**  $(\zeta(v))\sigma' \not\approx_R (\zeta(\text{rw}(v)))\sigma'$
- 10:         **then return false**
- 11: Repeat once lines 1–10, exchanging  $\phi$  and  $\phi'$
- 12: **return true**

The first loop (lines 4–7) tests whether syntactical equality between terms yielded by two distinct  $\phi$ -recipes under  $\phi$  implies that these two recipes yield equationally equal terms under  $\phi'$ . The condition in lines 8–10 tests whether there are two recipes representing instances of the left and right-hand sides of a rule under  $\phi$  but that do not represent equal terms (modulo  $R$ ) under  $\phi'$ . If either of the two loops outputs **false** then the two frames are not statically equivalent. Otherwise, we conclude that all equalities (between recipes, modulo  $R$ ) satisfied by  $\phi$  are also satisfied by  $\phi'$ . Repeating the procedure, exchanging the roles of  $\phi$  and  $\phi'$ , allows one to decide whether  $\phi \approx_R^s \phi'$ . The correctness of this procedure and its complexity analysis are provided by Theorem 2.

**Theorem 2.** *Algorithm 2 decides whether  $\phi \approx_R^s \phi'$  in time*

$$\mathcal{O}((|\phi| + |\phi'|)^3 \log^2(|\phi| + |\phi'|) + (|\phi| + |\phi'|)^{\text{nvars}(R)+1} \log^2(|\phi| + |\phi'|)).$$

$\mathcal{O}((|\phi| + |\phi'|)^3)$  is an upper bound for the complexity of the first loop (lines 4–7) of Algorithm 2;  $\mathcal{O}((|\phi| + |\phi'|)^{\text{nvars}(R)+1} \log^2(|\phi| + |\phi'|))$  is an upper bound for the complexity of the second (lines 8–10).

## 4 The Prefix Theory

We now investigate how our methods can be extended to deal with theories resulting from other convergent rewriting systems. In particular, we examine the system  $R_p$  presented in Example 2, which represents symmetric encryption with the prefix property. Encryption modes designed to encrypt large messages using block ciphers often have the prefix property, namely ECB and CBC. The decidability of deduction for this theory has been studied in [14].

As before, we assume here that  $\phi = v\tilde{n}.\sigma$  is a frame, with  $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ , and fix a set  $\mathcal{Y} = \{\tau_1, \tau_2, \tau_3\}$  of fresh names (note that  $nvars(R_p) = 3$ , since  $|vars(\pi_1(\{\langle x, y \rangle_z^s\}))| = 3$ ).

We begin by defining  $p$ -subterms, which extend the usual notion of subterm.

**Definition 8.** *Let  $t \in T(\Sigma^{\mathcal{D}\mathcal{Y}}, \text{Name})$ . We define the set of  $p$ -subterms of  $t$  inductively as follows:*

- if  $t \in \text{Name}$ , then  $sub_p(t) = \{t\}$ ;
- if  $t = \{\langle t_1, t_2 \rangle_{t_3}^s\}$  for some  $t_1, t_2, t_3 \in T(\Sigma, \text{Name})$ , then  $sub_p(t) = \{t, t_3\} \cup sub_p(\{t_1\}_{t_3}^s) \cup sub_p(\langle t_1, t_2 \rangle)$ ;
- if  $t = f(t_1, \dots, t_n)$  for some  $f$  and some  $t_1, \dots, t_n$  and  $t \neq \{\langle t'_1, t'_2 \rangle_{t'_3}^s\}$  for all  $t'_1, t'_2, t'_3 \in T(\Sigma, \text{Name})$ , then  $sub_p(t) = \{t\} \cup \bigcup_{i=1}^n sub_p(t_i)$ .

We extend this definition to sets of terms as usual.

Our algorithms for the prefix theory are similar to the algorithms for subterm convergent rewriting systems. As before, we start by building a minimal DAG-forest  $\mathcal{T}_{\phi,p}$  and computing functions  $rw_p$  and  $\zeta_p$ .  $\mathcal{T}_{\phi,p}$ ,  $rw_p$ , and  $\zeta_p$  satisfy properties analogous to those given in Lemma 2 for  $\mathcal{T}_{\phi,R}$ ,  $rw$ , and  $\zeta$ , after replacing some critical instances of  $sub$  by  $sub_p$ . A summary of these properties and details of the construction can be found in the technical report [19].

Let Algorithm 1<sub>p</sub> be Algorithm 1 after replacing  $R$  by  $R_p$ ,  $\mathcal{T}_{\phi,R}$  by  $\mathcal{T}_{\phi,p}$ , and  $sub$  by  $sub_p$  in line 15. Algorithm 1<sub>p</sub> computes a saturated frame under  $R_p$ . Our algorithms for deduction and static equivalence also work for the prefix theory. We summarize these results in the following theorems. Note, in our complexity analysis, that  $nvars(R_p) = 3$ . Hence, the complexity of our algorithms is not affected by the fact that  $R_p$  is not subterm convergent.

**Lemma 4.** *Algorithm 1<sub>p</sub> terminates in time  $\mathcal{O}(|\phi|^3 \log^2 |\phi|)$ .*

*$\phi_s$  is a saturation of  $\phi$  (under  $R_p$ ),  $dom(\zeta_s) = dom(\sigma_s)$ , and, if  $x \in dom(\zeta_s)$ , then  $x\zeta_s$  is a DAG-representation of a  $\phi$ -recipe for  $x\sigma_s$  with size  $|x\zeta_s| \in \mathcal{O}(|\phi|)$ .*

*For each  $v \in dom(\zeta)$ , there is a  $\phi_s$ -recipe  $t$  for term  $\mathcal{T}_{\phi,p}(v)$  such that  $\zeta(v) = t\zeta_s$  is a  $\phi$ -recipe for term  $\mathcal{T}_{\phi,p}(v)$ . If  $\text{term}_{\mathcal{T}_{\phi,p}}(v) \in \sigma_s[\mathcal{T}_{\phi_s}]$ , then  $v \in dom(\zeta)$ .*

**Theorem 3.** *Given a frame  $\phi$  and a term  $t$ , the complexity of deciding whether  $\phi \vdash_{R_p} t$  is at most  $\mathcal{O}(|t| \log^2 |t| + |t||\phi|^2 + |\phi|^3 \log^2 |\phi|)$ .*

**Theorem 4.** *Algorithm 2 decides whether  $\phi \approx_{\mathbb{R}_p}^s \phi'$  in time*

$$\mathcal{O}((|\phi| + |\phi'|)^4 \log^2(|\phi| + |\phi'|)).$$

#### 4.1 Off-Line Guessing Attacks on a Version of Kerberos

We now present multiple off-line guessing attacks on a version of Kerberos. Kerberos is known to be insecure against off-line guessing attacks [24]. However, our formal analysis of its security when implemented with a symmetric encryption scheme satisfying the prefix property is novel.

**Kerberos Variant.** The version of Kerberos we consider is adapted from [12]. We present a short description of the protocol, in standard notation.

1.  $A \rightarrow \text{KAS}: \langle A, \text{KAS}, T_1 \rangle$
2.  $\text{KAS} \rightarrow A: \left\{ \left\{ \langle A, \text{TGS}, K_{A,\text{TGS}}, T_2 \rangle \right\}_{K_{\text{KAS},\text{TGS}}}^s, K_{A,\text{TGS}}, \text{TGS}, T_2 \right\}_{K_{A,\text{KAS}}}^s$
3.  $A \rightarrow \text{TGS}: \left\{ \left\{ \langle A, \text{TGS}, K_{A,\text{TGS}}, T_2 \rangle \right\}_{K_{\text{KAS},\text{TGS}}}^s, \left\{ \langle A, T_3 \rangle \right\}_{K_{A,\text{TGS}}}^s, B \right\}$
4.  $\text{TGS} \rightarrow A: \left\{ \left\{ K_{A,B}, B, T_4, \left\{ \langle A, B, K_{A,B}, T_4 \rangle \right\}_{K_{B,\text{TGS}}}^s \right\}_{K_{A,\text{TGS}}}^s \right\}$
5.  $A \rightarrow B: \left\{ \left\{ \langle A, B, K_{A,B}, T_4 \rangle \right\}_{K_{B,\text{TGS}}}^s, \left\{ \langle A, T_5 \rangle \right\}_{K_{A,B}}^s \right\}$
6.  $B \rightarrow A: \left\{ T_5 \right\}_{K_{A,B}}^s$

$A$  is a name (*e.g.*, of a client of a single-sign-on service).  $\text{KAS}$  is the Kerberos authentication server.  $\text{TGS}$  is the ticket-granting server.  $B$  is some service provider.  $K_{A,\text{KAS}}$  (respectively  $K_{\text{KAS},\text{TGS}}$ ,  $K_{B,\text{TGS}}$ ) is a long-term key shared between  $A$  and  $\text{KAS}$  (respectively between  $\text{KAS}$  and  $\text{TGS}$  and between  $B$  and  $\text{TGS}$ ).  $K_{A,\text{TGS}}$  (respectively  $K_{A,B}$ ) is a short-term key shared between  $A$  and  $\text{TGS}$  (respectively between  $A$  and  $B$ ), freshly generated for each session by  $\text{KAS}$  (respectively by  $\text{TGS}$ ). Finally,  $T_1, \dots, T_5$  are timestamps. Note that for our purposes, it suffices to view them as freshly generated nonces, even if this representation is imprecise.

The only difference between the protocol presented and the model of Kerberos (version IV) presented in [12] is that, in the term sent in step 2, the encryption  $\left\{ \langle A, \text{TGS}, K_{A,\text{TGS}}, T_2 \rangle \right\}_{K_{\text{KAS},\text{TGS}}}^s$  is the first (instead of the last) term in the encrypted tuple. Although this may appear to be a small and harmless change, it gives rise to guessing attacks if a symmetric encryption scheme with the prefix property is used. One of our attacks relies on the double encryption used in version IV of the protocol and is thus prevented by Version V.

The signature  $\Sigma^{\text{ker}}$  we use is obtained by simply adding the set  $\Sigma_0 = \{A, B, \text{KAS}, \text{TGS}\}$  of agent names to the signature  $\Sigma^{\mathcal{D}\mathcal{Y}}$ . We assume that  $\{T_i \mid i \in \mathbb{N}\} \subseteq \text{Name}$ . We also represent the long-term keys  $K_{A,\text{KAS}}, K_{\text{KAS},\text{TGS}} \in \text{Name}$  and the short-term keys  $K_{A,\text{TGS}}, K'_{A,\text{TGS}} \in \text{Name}$  (corresponding to the keys generated by  $\text{KAS}$  for use between  $A$  and  $\text{TGS}$  in two distinct sessions) as names.

**The Attacker's Knowledge.** We consider an attacker who eavesdrops on two different protocol sessions, both executed between an agent  $A$  and the server  $\text{KAS}$ . For simplicity, we assume that the attacker stores only the second

message of each session. This is enough to present our off-line guessing attacks. We represent the attacker's knowledge by the frame  $\phi = v\tilde{n}.\sigma$ , where  $\tilde{n} = \left\{ K_{A,\text{TGS}}, K'_{A,\text{TGS}}, K_{A,\text{KAS}}, K_{KAS,\text{TGS}}, T_1, T_2 \right\}$  and

$$\sigma = \left\{ \begin{array}{l} x_0 \mapsto \left\{ \left\langle \langle A, \text{TGS}, K_{A,\text{TGS}}, T_1 \rangle \right\rangle_{K_{KAS,\text{TGS}}}^s, K_{A,\text{TGS}}, \text{TGS}, T_1 \right\rangle_{K_{A,\text{KAS}}}^s, \\ x_1 \mapsto \left\{ \left\langle \langle A, \text{TGS}, K'_{A,\text{TGS}}, T_2 \rangle \right\rangle_{K_{KAS,\text{TGS}}}^s, K'_{A,\text{TGS}}, \text{TGS}, T_2 \right\rangle_{K_{A,\text{KAS}}}^s \end{array} \right\}.$$

We are interested in determining whether this frame allows an off-line guessing attack of  $K_{A,\text{KAS}}$ . This is a (potentially) weak key, since it is often chosen by human users or derived from such a key. We take  $g = K_{A,\text{KAS}}$  and  $w = a_0$  and, throughout our example, we work with the frames  $\phi_g = v\tilde{n}'.\sigma_g$  and  $\phi_w = v\tilde{n}'.\sigma_w$ , where  $\tilde{n}' = \tilde{n} \cup \{a_0\}$ ,  $\sigma_g = \sigma \cup \{x_2 \mapsto K_{A,\text{KAS}}\}$ , and  $\sigma_w = \sigma \cup \{x_2 \mapsto a_0\}$ .

**Saturation of  $\phi_w$  and  $\phi_g$ .** In [19] we present tables with the results of saturating the frames.

**Table 1.** (Sample) Off-line Guessing Attacks to Kerberos

Attack	$t$	$t'$
1	$\pi_2(\pi_1(\{x_0\}_{x_2}^{s,-1}))$	TGS
2	$\left\{ \left\langle \langle \pi_1(x_0) \rangle_{x_2}^{s,-1}, \pi_2(\{x_0\}_{x_2}^{s,-1}) \right\rangle \right\}_{x_2}^s$	$x_0$
3	$\langle \langle \pi_1(\pi_1(x_0)) \rangle_{x_2}^{s,-1}, \text{TGS} \rangle$	$\pi_1(\{x_0\}_{x_2}^{s,-1})$
4	$\pi_1(\pi_1(\pi_1(\pi_1(\pi_1(\{x_0\}_{x_2}^{s,-1}))))$	$\pi_1(\pi_1(\pi_1(\pi_1(\{x_1\}_{x_2}^{s,-1}))))$

**Off-line Guessing Attacks to Kerberos.** It is clear that  $\phi_w \not\approx_{R_p}^s \phi_g$ . Thus, Kerberos allows an off-line-guessing attack of  $K_{A,\text{KAS}}$ . In fact, an attacker has multiple pairs of recipes  $t, t' \in T_{\phi_g}$  that he can use to validate his guess; we present a few illustrative examples in Table 1. Note that, of the four attacks presented, all but Attack 1 rely on the prefix property of the encryption scheme. Only Attack 4 relies on the fact that we use version IV instead of version V and exchange the order of the messages of the original Kerberos protocol.

How feasible are these attacks in practice? First of all, CBC encryption mode uses a random initialization vector. To prevent Attack 4 it is enough that the initialization vectors used in the two messages are different (they may even be public). We can reason about initialization vectors in this model by representing symmetric encryption as a function  $\text{enc}_{\text{sym}}$  with three arguments (an initialization vector, the message, and the key) and adapting the rewriting rules accordingly.

Furthermore, consider the recipes  $t$  and  $t'$  given in Attack 4. We have  $t\sigma_g = t'\sigma_g = \langle A, \text{TGS} \rangle_{K_{KAS,\text{TGS}}}^s$ . However, in practice, the attacker can only obtain an encryption of the first block of the plaintext. Thus, this attack is only feasible if the first encrypted block is equal in both messages. This depends on the length of the encryption blocks and the names, as well as the padding techniques used.

Note also that Attack 2 (respectively 3) is only feasible if the encryption of the first three (respectively two) elements of the tuple occupy disjoint encryption blocks from the encryption of the remaining elements. Modeling such details would fall outside of the scope of the theories we consider.

The relevance of such implementation details in the study of off-line guessing attacks has been pointed out in [13]. We believe that reasoning about these kinds of possible weaknesses is an important challenge for symbolic methods.

## 5 Related Work and Conclusion

Our algorithms quite compare favorably to previously existing algorithms. [2] presents the first proof that the problems we address can be decided in polynomial-time. However, efficiency is not a main concern and thus it is not surprising that our algorithms have a better asymptotic complexity. For example, for the theory  $\approx_{\mathcal{DY}}$ , the complexities of our algorithms are  $\mathcal{O}(|t| \log |t| + |\phi|^2 |t| + |\phi|^2 \log^2 |\phi|)$  and  $\mathcal{O}((|\phi| + |\phi'|)^3 \log^2(|\phi| + |\phi'|))$  for deduction and static equivalence (respectively), whereas our best estimates for the complexity of the algorithms in [2] are  $\mathcal{O}(|\phi|^{10} + |\phi|^2 |t| + |t|^4)$  and  $\mathcal{O}((|\phi| + |\phi'|)^{15})$ .

The complexity of the YAPA tool [8] is not polynomial, as it uses a straightforward representation of terms, as opposed to DAGs. Furthermore, our saturation technique is also more efficient: in YAPA, for each  $(n, p, q)$ -decomposition of the left-hand side of a rewrite rule and each assignment of the first  $n + p$  parameters to recipes in the frame, it may be necessary to compute the normal form of a term  $t$ . We are not aware of any general algorithms for this task that have a complexity better than  $\mathcal{O}(|t|^4)$  (see discussion below). If we denote by  $Y(R)$  the greatest value of  $n + p$  for all  $(n, p, q)$ -decompositions of rewriting rules in  $R$ , we obtain a complexity of  $\mathcal{O}(|\phi|^{Y(R)+4})$  for YAPA's saturation procedure; the complexity of our saturation procedure is  $\mathcal{O}(|\phi|^{nvars(R)} \log^2 |\phi|)$  (note that  $nvars(R) \leq Y(R)$  in general). For the rewriting system  $R_{\mathcal{DY}}$ , this amounts to a complexity of  $\mathcal{O}(|\phi|^7)$  for YAPA and  $\mathcal{O}(|\phi|^2 \log |\phi|)$  for our algorithm. Note that these estimates assume that DAGs are implemented; the exact implementation of DAGs and the algorithms to compute normal forms may affect the complexity of the procedure. It may also be possible to provide better bounds, *e.g.*, on the number of recipes for which we need to perform this reduction.

Our saturation procedure is also more efficient than that implemented by the KISS tool. In this tool, the rule **Narrowing** generates a number of deduction facts for each rewriting rule in  $R$ . If we denote by  $K(R)$  the maximum number of side conditions in deduction facts generated by any rewriting rule in  $R$ , we again have  $nvars(R) \leq K(R)$  in general: for example,  $K(R_{\mathcal{DY}}) = 3$ . The terms in these side-conditions must then be instantiated (via the rule **F – Solving**) with terms that are heads of some deduction fact. There are at least  $\mathcal{O}(|\phi|)$  such terms, which implies that the saturated frame may have  $\mathcal{O}(|\phi|^{K(R)})$  deduction facts. Testing the premise of the rules **F – Solving**, **E-Solving**, and **Unifying** requires selecting two deduction facts from the frame and performing an operation with linear-time complexity. Since there are  $\mathcal{O}(|\phi|^{2K(R)})$  such pairs, we obtain a complexity

of at least  $\mathcal{O}(|\phi|^{2K(R)+1})$ . For the rewriting system  $R_{\mathcal{D}\mathcal{Y}}$ , this amounts to a complexity of  $\mathcal{O}(|\phi|^7)$  for KISS, in contrast to the complexity of  $\mathcal{O}(|\phi|^2 \log^2 |\phi|)$  for our algorithms. Here it may also be possible to improve this complexity bound, for example by limiting the number of pairs of rules that must be tested.

Finally, we note that all the algorithms discussed here require deciding the equality of two terms  $t$  and  $t'$  under the given equational theory. Our algorithms rely on the subterm convergence of the rewriting system to perform this task with complexity  $\mathcal{O}((|t| + |t'|) \log(|t| + |t'|))$ . This constitutes a marked advantage over algorithms for more general rewriting systems, for which we are not aware of any algorithm improving the complexity of  $\mathcal{O}((|t| + |t'|)^4)$  achieved in [2].

As future work, we plan to implement our algorithms. We also plan to study how our approach can be adapted to other equational theories. The simplicity of extending our methods to the prefix theory suggests that it may be possible to generalize our approach for a much larger class of equational theories, possibly improving upon existing complexity results for the two decision problems, when such results exist.

## Acknowledgements

This work was partly supported by FCT and EU FEDER, namely via the project KLog PTDC/MAT/68723/2006 of SQIG-IT and the project UTAustin/MAT/0057/2008 AMDSC of IST. The first author acknowledges the support of FCT via the PhD grant SFRH/BD/44204/2008.

## References

1. Abadi, M., Baudet, M., Warinschi, B.: Guessing Attacks and the Computational Soundness of Static Equivalence. In: Aceto, L., Ingólfssdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 398–412. Springer, Heidelberg (2006)
2. Abadi, M., Cortier, V.: Deciding Knowledge in Security Protocols Under Equational Theories. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 46–58. Springer, Heidelberg (2004)
3. Abadi, M., Cortier, V.: Deciding Knowledge in Security Protocols Under (Many More) Equational Theories. In: Proc. Workshop on Computer Security Foundations (CSFW 2005), pp. 62–76 (2005)
4. Abadi, M., Fournet, C.: Mobile Values, New Names and Secure Communications. ACM SIGPLAN Notices 36, 104–115 (2001)
5. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P., Heám, P., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Eteessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
6. Basin, D., Mödersheim, S., Viganò, L.: OFMC: A Symbolic Model Checker for Security Protocols. Int. Journal of Information Security 4(3), 181–208 (2005)

7. Baudet, M.: Deciding Security of Protocols against Off-line Guessing Attacks. In: Proc. 12th ACM Conf. on Computer and Communications Security, pp. 16–25 (2005)
8. Baudet, M., Cortier, V., Delaune, S.: YAPA: A Generic Tool for Computing Intruder Knowledge. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 148–163. Springer, Heidelberg (2009)
9. Baudet, M., Cortier, V., Kremer, S.: Computationally Sound Implementations of Equational Theories Against Passive Adversaries. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 652–663. Springer, Heidelberg (2005)
10. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th Computer Security Foundations Workshop (CSFW 2001), pp. 82–96. IEEE Computer Society, Los Alamitos (2001)
11. Blanchet, B., Abadi, M., Fournet, C.: Automated Verification of Selected Equivalences for Security Protocols. In: Symposium on Logic in Computer Science (2005); *Journal of Logic and Algebraic Programming* 75(1), 3–51
12. Bella, G., Paulson, L.C.: Using Isabelle to Prove Properties of the Kerberos Authentication System. In: DIMACS Workshop on Design and Formal Verification of Security Protocols (1997)
13. Bellare, S.M., Merritt, M.: Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In: IEEE Symposium on Research in Security and Privacy, pp. 72–84 (1992)
14. Chevalier, Y., Küsters, R., Rusinowitch, M., Turuani, M.: An NP Decision Procedure for Protocol Insecurity with XOR. In: Proc. 18th Annual IEEE Symposium on Logic in Computer Science (LICS 2003), pp. 261–270 (2003)
15. Chevalier, Y., Küsters, R., Rusinowitch, M., Turuani, M.: Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 124–135. Springer, Heidelberg (2003)
16. Ciobăcă, Ș., Delaune, S., Kremer, S.: Computing Knowledge in Security Protocols under Convergent Equational Theories. In: Schmidt, R.A. (ed.) CADE-22. LNAI, vol. 5663, pp. 355–370. Springer, Heidelberg (2009)
17. Comon-Lundh, H., Shmatikov, V.: Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive Or. In: 18th Annual IEEE Symposium on Logic in Computer Science (LICS 2003), pp. 271–280 (2003)
18. Comon-Lundh, H., Treinen, R.: Easy Intruder Deductions. In: Dershowitz, N. (ed.) *Verification: Theory and Practice*. LNCS, vol. 2772, pp. 225–242. Springer, Heidelberg (2004)
19. Conchinha, B., Basin, D., Caleiro, C.: Efficient Algorithms for Deciding Deduction and Static Equivalence, Technical Reports 680 ETH Zürich, Information Security Group D-INFK (2010),  
<ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/6xx/680.pdf>
20. Corin, R., Doumen, J., Etalle, S.: Analyzing Password Protocol Security Against Off-line Dictionary Attacks. In: Proc. of the 12th ACM Conf. on Computer and Communications Security (CCS 2005). ENTCS, vol. 121, pp. 47–63 (2005)
21. Cortier, V., Delaune, S., Lafourcade, P.: A Survey of Algebraic Properties Used in Cryptographic Protocols. *Journal of Computer Security* 14, 1–43 (2006)
22. Cortier, V., Delaune, S.: Deciding Knowledge in Security Protocols for Monoidal Equational Theories. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 196–210. Springer, Heidelberg (2007)

23. Delaune, S., Lafourcade, P., Lugiez, D., Treinen, R.: Symbolic Protocol Analysis for Monoidal Equational Theories. *Information and Computation* 206, 312–351 (2009)
24. Gong, L., Lomas, M.A., Needham, R.M., Saltzer, J.H.: Protecting Poorly Chosen Secrets From Guessing Attacks. *IEEE Journal on Selected Areas in Communications* 11, 648–656 (1993)
25. Lafourcade, P.: Intruder Deduction for the Equational Theory of Exclusive-or with Commutative and Distributive Encryption. In: Proc. 1st International Workshop on Security and Rewriting Techniques (SecReT 2006). ENTCS, vol. 171, pp. 37–57 (2007)
26. Millen, J., Shmatikov, V.: Symbolic Protocol Analysis With an Abelian Group Operator or Diffie–Hellman Exponentiation. *Journal of Computer Security* 13, 515–564 (2005)