Deciding Security of Protocols against Off-line Guessing Attacks

Mathieu Baudet
LSV – CNRS & INRIA Futurs Projet SECSI & ENS Cachan
61, avenue du Président Wilson, 94235 Cachan Cedex, France
mathieu.baudet@lsv.ens-cachan.fr

ABSTRACT

We provide an effective procedure for deciding the existence of off-line guessing attacks on security protocols, for a bounded number of sessions.

The procedure consists of a constraint solving algorithm for determining satisfiability and equivalence of a class of second-order E-unification problems, where the equational theory E is presented by a convergent subterm rewriting system.

To the best of our knowledge, this is the first decidability result to use the generic definition of off-line guessing attacks due to Corin *et al.* based on static equivalence in the applied pi calculus.

Categories and Subject Descriptors: C.2.2 [Network Protocol]: Protocol Verification, D.2.4 [Software/Program Verification]: Formal Methods, F.4.2 [Grammars and Other Rewriting Systems]: Decision Problems

General Terms: Security, Theory, Verification

Keywords: Security Protocols, Formal Methods, Guessing Attacks, Second-Order E-Unification

1. INTRODUCTION

Guessing attacks, also known as dictionary or brute-force attacks, occur in cryptography when an attacker is able to guess a secret by just trying every possible value for it. Practically, this is feasible only when the number of those values (the "dictionary") is small—say 2^{32} for a (poor) password or even less for a vote in an election protocol. In this case the secret is called a *weak secret*.

Fortunately not all weak secrets can be broken. As pointed out by Lowe [18] the attacker must still be able to test whether one of his guesses is correct or not, typically by exploiting redundancy between messages.

Among guessing attacks, by definition, *off-line* guessing attacks are those for which the attacker does not need to participate in any communication during the guessing phase (but he may have interacted with the honest agents before). When the size of the dictionary becomes significant, off-line guessing attacks are more feasible in practice than on-line ones and thus are more crucial to detect. On-line guessing attacks may need to send as many requests to the network as the size of the dictionary.

Several attempts have been made, based on the initial work of Lowe [18], to characterize off-line guessing attacks from a logical point of view and derive formal methods for detecting them [9, 11, 13]. The attempts for a "good" definition currently culminate with Corin *et al.* [10]. Using the notion of static equivalence from the applied pi calculus [3], they give a simple and intuitive definition of off-line guessing attacks for an arbitrary set of primitives, modeled by an equational theory. However no automatic procedure is given in [10] and the mentioned examples only deal with passive adversaries.

In this paper we address the question whether the general definition of Corin *et al.* [10] is suitable for automatic reasoning. We show that the problem of finding off-line guessing attacks is decidable in the case of a bounded number of sessions, for protocols whose set of primitives is described by a convergent subterm rewriting system (see Section 2 for a definition). In particular our class of protocols encompasses the initial Dolev-Yao model [14] and subsequent extensions as in [19, 12, 3, 6, 1] with symmetric encryption, ciphers, compound keys, signatures and hashes.

Our algorithm is based on a procedure for solving a class of second-order E-unification problems. We show that the satisfiability of those constraint systems is decidable, and more remarkably, that the equivalence between two systems (that is, the equality of their sets of solutions) is decidable.

The purpose of our constraint systems is here to represent symbolic traces of protocols (see for instance [19, 20, 8, 12]). We use the equivalence of systems to specify the absence of off-line guessing attacks, by expressing the fact that the intruder cannot distinguish between two versions of the same symbolic trace: one corresponding to a good guess and the other one to a bad guess of the secret.

To our knowledge, this is the first decision procedure for finding such off-line guessing attacks. It is fair to mention, though, that recent releases of Blanchet's tool Proverif [5] do support off-line guessing attacks based on Corin *et al.*'s definition. In case of success, Blanchet's (non-terminating, approximate) procedure for proving strong secrecy [6], later refined by [7], rules out the possibility of on-line/off-line guessing attacks, for an unbounded number of sessions. Our concern is different here, as we look for an exact answer, given by a terminating algorithm, to the security problem for a bounded number of sessions.

This work appeared in the proceedings of the ACM CCS'05 conference, November 7–11, 2005, Alexandria, Virginia, USA. This is a revised version, dated January 6, 2006.

FURTHER RELATED WORK. Delaune and Jacquemard [12] give an NP procedure for verifying trace properties, for a bounded number of sessions, on protocols with explicit destructors. Our algorithm deals with a more general class of security properties, including resistance to off-line guessing attacks. We also relax (and somewhat simplify) the syntactic conditions on the equational theory, by considering convergent subterm rewriting systems instead of convergent public-collapsing rewriting ones (see Section 2).

Abadi and Cortier [1] present an algorithm for deciding static equivalence between closed frames in polynomial time, when the equational theory is given by a convergent subterm rewriting system. Recently, this result has been extended to a more general class of equational theories allowing associative and commutative symbols [2]. This work corresponds to passive adversaries, that is, pure eavesdroppers. Our procedure can be seen as an extension of [1] to the general case of active adversaries.

OUTLINE OF THE PAPER. Section 2 begins with preliminary definitions. We describe our class of constraint systems, called *(standard) intruder constraint systems*. These correspond to second-order E-unification problems with a number of regularity conditions. Compared to usual unification, though, we allow distinguishing between *private* and *public symbols*. Private symbols typically account for secret values not directly available to the intruder. Our main theorem is the following: if the equational theory E is presented by a convergent subterm rewriting system and the signature contains an infinite number of public constants, the satisfiability of intruder constraint systems is decidable, and so is the equivalence between standard intruder constraint systems.

We apply this result in Section 3 for deciding trace properties and, more remarkably, the existence of off-line guessing attacks on cryptographic protocols, for a bounded number of sessions.

Section 4 is devoted to the proof of the main theorem. The heart of our decision procedure is a sound and complete set of transformation rules together with a (nondeterministic) terminating strategy, so as to bring any intruder constraint system into a solved form. As such solved forms are always satisfiable, this gives a procedure to decide satisfiability. In order to decide the equivalence of systems, we provide a criterion for testing whether the set of solutions corresponding to a solved system is included in that of a standard intruder constraint system. We conclude in Section 5.

2. PRELIMINARIES

2.1 Syntax and Basic Definitions

A signature is a set of symbols $\mathcal{F}$ together with non-negative arities. Given an additional set of variables $\mathcal{X}$, we write $\mathcal{T}(\mathcal{F}, \mathcal{X})$ for the set of (usual first-order) terms built upon $\mathcal{X}$ using the symbols in $\mathcal{F}$.

We assume a given set of symbols $\mathcal{F}$, with elements denoted by $f, g \ldots$ Symbols of arity 0 are called *constants*. $\mathcal{F}$ is partitioned into *public symbols* $\mathcal{F}_{\text{pub}}$ and *private symbols* $\mathcal{F}_{\text{priv}}$. We also use an additional set of constants, called *parameters* $\mathcal{W} = \{w_1, w_2 \ldots w_k \ldots\}$.

We fix a set of *variables* $\mathcal{X}$, partitioned into *first-order variables* $\mathcal{X}^1$, with elements denoted by $x, y \ldots$ and *second-order variables* $\mathcal{X}^2$, written $X, Y \ldots$ Second-order variables X are given with arities, denoted $\text{ar}(X)$. In the following we assume that infinite numbers of public constants, first-order variables and second-order variables of each arity, are available.

Elements of $\mathcal{T}(\mathcal{F}, \mathcal{X}^1)$ are called *first-order terms*, and denoted by letters $t, s \ldots$ Elements of $\mathcal{T}(\mathcal{F}_{\text{pub}} \cup \mathcal{W}, \mathcal{X}^2)$ are called *second-order terms*, and denoted by $M, N \ldots$ More generally we use letters $T, S \ldots$ for terms in $\mathcal{T}(\mathcal{F} \cup \mathcal{W}, \mathcal{X})$ and letter v for variables in $\mathcal{X}$.

We write $\text{var}(T)$ and $\text{par}(T)$ for the sets of variables and parameters, respectively, occurring in T. A term is *closed* iff it has no variable, *public* iff is uses no private symbol. Note that our second-order terms are always public. We extend the notations $\text{var}(.)$ and $\text{par}(.)$ to tuples and sets of terms in the obvious way.

Substitutions are written $\sigma = \{v_1 \mapsto T_1, \ldots, v_n \mapsto T_n\}$. We let $\text{dom}(\sigma) = \{v_1 \ldots v_n\} \subseteq \mathcal{X}$ if $\forall i, v_i \neq T_i$. We write $T\sigma = \sigma(T)$ for the (usual, first-order) application of σ to T, $\sigma\mu = \mu \circ \sigma$ for the composition of substitutions. σ is *closed* iff all the T_i are closed terms, *idempotent* iff $\sigma\sigma = \sigma$, that is, $\forall i, \text{var}(T_i) \cap \text{dom}(\sigma) = \emptyset$. We let $\text{var}(\sigma) = \{v_1 \ldots v_n\} \cup \text{var}(T_1 \ldots T_n)$. $T|_p$ denotes the subterm of T at position p, whereas $T[p := T']$ is the result of replacing the subterm at position p in T with T'. This notation is extended to equations: for instance $(T =^? T')|_p$ and $(T =^? T')[p := T'']$.

For each pair of terms (t_1, t_2), $\text{mgu}(t_1, t_2)$ denotes a most general unifier of t_1 and t_2, that is, an idempotent substitution μ such that $t_1\mu = t_2\mu$ and for every substitution μ', $t_1\mu' = t_2\mu'$ implies $\mu' = \mu\mu'$.

In the rest of this paper, unless stated otherwise, we only consider *well-formed* substitutions, that is, substitutions σ which assign first-order variables to first-order terms, and second-order variables X to second-order terms of *correct arity*: $\forall w_i \in \text{par}(X\sigma)$, $i \leq \text{ar}(X)$ and $\forall Y \in \text{var}(X\sigma)$, $\text{ar}(Y) \leq \text{ar}(X)$. Thus, arity represents the maximal index of eligible parameters for substituting a second-order variable.

A (closed, public, non necessarily linear) *context* is a closed second-order term C given with an arity $n \geq 0$ such that $par(C) \subseteq \{w_1 \ldots w_n\}$. Each parameter may occur zero, one or several times. If C is n-ary, $C[T_1 \ldots T_n]$ denotes the result of replacing each w_k by T_k in C. We extend the notation to second-order terms and may write $M[T_1 \ldots T_n]$ provided that $\text{par}(M) \subseteq \{w_1 \ldots w_n\}$.

A *term rewriting system*, or simply *rewriting system*, is a finite set $\mathcal{R}$ of *rewriting rules* $l \to r$, where l, r are two first-order terms. A term T *reduces* to S by $\mathcal{R}$, written $T \to_{\mathcal{R}} S$, iff there exists a rule $l \to r$ in $\mathcal{R}$, a position p and a non-necessarily well-formed substitution σ such that $T|_p = l\sigma$ and $S = T[p := r\sigma]$. We write $\to^*_{\mathcal{R}}$ for the reflexive and transitive closure of the binary relation $\to_{\mathcal{R}}$, $=_{\mathcal{R}}$ for its reflexive, symmetric and transitive closure. A term T is $\mathcal{R}$-*reduced*, or equivalently, is in $\mathcal{R}$-*normal form* iff there is no T' such that $T \to_{\mathcal{R}} T'$.

A rewriting system $\mathcal{R}$ is *terminating* iff it admits no infinite sequence of reductions; *confluent* iff for every T_1, T_2, T_3 such that $T_1 \to^*_{\mathcal{R}} T_2$, $T_1 \to^*_{\mathcal{R}} T_3$, there exists T_4 such that $T_2 \to^*_{\mathcal{R}} T_4$, $T_3 \to^*_{\mathcal{R}} T_4$; *convergent* iff it is confluent and terminating.

2.2 Intruder Constraint Systems

Constraint solving is by now a standard method for deciding reachability properties in cryptographic protocols for a finite number of sessions [19, 20, 8, 12]. Our purpose is to extend this method to properties that *compare* the behavior of two systems from the intruder's point of view. This leads us to introduce the following constraint systems and the corresponding notion of satisfiability and equivalence. We show in Section 3 how to use these notions to express the security of protocols against off-line guessing attacks.

Definition 1. Let $\mathcal{R}$ be a rewriting system and $\mathcal{Y}$ a set of m pairwise distinct second-order variables $X_1 \dots X_m$. Define $a_i = \mathrm{ar}(X_i)$ and assume $0 \leq a_1 \leq a_2 \leq \dots \leq a_m$. An $(\mathcal{R}, \mathcal{Y})$-*intruder constraint system* is a system of equations Σ of the form

$$\exists x_1 \dots x_m, \quad \begin{cases} X_1[t_1 \dots t_{a_1}] & =^? & x_1 \\ & \dots & \\ X_m[t_1 \dots t_{a_m}] & =^? & x_m \\ s_1 & =^?_{\mathcal{R}} & s'_1 \\ & \dots & \\ s_n & =^?_{\mathcal{R}} & s'_n \end{cases}$$

such that the following *regularity conditions* hold:

(1) $\mathrm{var}(s_1 \dots s_n, s'_1 \dots s'_n) \subseteq \{x_1 \dots x_m\}$ and

(2) for all $1 \leq i \leq m$ and $1 \leq j \leq a_i$, $\mathrm{var}(t_j) \subseteq \{x_1 \dots x_{i-1}\}$.

As suggested by the notation above, a *solution to* Σ is a closed (well-formed) substitution θ with domain $\mathrm{dom}(\theta) = \{X_1 \dots X_m\}$ such that there exists a closed (well-formed) substitution θ' with $\mathrm{dom}(\theta') = \{x_1 \dots x_m\}$ satisfying the two conditions

(i) for all $1 \leq i \leq m$, $(X_i\,\theta)[t_1\theta' \dots t_{a_i}\theta'] = x_i\,\theta'$, and

(ii) for all $1 \leq j \leq n$, $s_j\,\theta' =_{\mathcal{R}} s'_j\,\theta'$.

Note that the regularity condition (2) implies that for each closed substitution θ with $\mathrm{dom}(\theta) = \{X_1 \dots X_m\}$, there exists a unique θ' satisfying condition (i). Indeed, $x_i\theta'$ is computed from (i) by induction on i. Thus, given condition (1), and provided that the word problem for $=_{\mathcal{R}}$ is decidable, it is easy to check whether a given θ is a solution.

An intruder constraint system is *satisfiable* iff it admits at least one solution. Two $(\mathcal{R}, \mathcal{Y})$-intruder constraint systems are *equivalent* iff they have the same set of solutions. We emphasize that, given our definition of solutions, equivalence of systems does not depend on the exact values of first-order variables. This is crucial for the applications described in Section 3.

Using the same notations as above, an $(\mathcal{R}, \mathcal{Y})$-intruder constraint system Σ is (X, Y)-*standard* iff $X = X_{m-1}$ and $Y = X_m$ are two fixed distinct variables of maximal arity in $\mathcal{Y}$ (that is, $a_{m-1} = a_m$), Σ contains the equation $x_{m-1} =^?_{\mathcal{R}} x_m$ and x_{m-1}, x_m occur in no other equation $s_j =^?_{\mathcal{R}} s'_j$ of Σ.

A rewriting system $\mathcal{R}$ is *subterm* iff: for each rule $l \to r$ in $\mathcal{R}$, r is either a proper subterm of l or a closed public $\mathcal{R}$-reduced term. In Section 4 we prove the following result:

THEOREM 1. *Let $\mathcal{R}$ be a convergent subterm rewriting system and $\mathcal{Y}$ a finite set of second-order variables. The satisfiability of $(\mathcal{R}, \mathcal{Y})$-intruder constraint systems is decidable. Assume that X, Y are two distinct variables of maximal arity in $\mathcal{Y}$. The equivalence between (X, Y)-standard $(\mathcal{R}, \mathcal{Y})$-intruder constraint systems is decidable.*

We leave open the question if it is decidable whether two (non necessarily standard) intruder constraint systems are equivalent. On the other hand, the equivalence problems useful for the applications (Section 3) only involve standard constraint systems.

Following Delaune and Jacquemard [12], a rewriting system $\mathcal{R}$ is *public-collapsing* iff for every rule $l \to r \in \mathcal{R}$, the following two conditions hold:

1. $r \neq l$ and either $r \in \mathrm{var}(l)$ or r is a public $\mathcal{R}$-reduced term;
2. if $l = f(l_1 \dots l_n)$ and $f \in \mathcal{F}_{\mathrm{pub}}$, then for all proper subterms of l of the form $g(t_1 \dots t_m)$ with $g \in \mathcal{F}_{\mathrm{pub}}$, we have that either $g(t_1 \dots t_m)$ is a closed public $\mathcal{R}$-reduced term, or there exists j such that $t_j = r$.

Thus in particular, every public-collapsing rewriting system is subterm according to our definition.

3. APPLICATION TO SECURITY

In this section we illustrate how to use the intruder constraint systems of Section 2 for analyzing cryptographic protocols with respect to trace properties and, more remarkably, resistance to off-line guessing attacks.

Let us consider the following example, called the Handshake protocol [15, 13]:

$$\begin{array}{lll} 0. & A \to B: & \{N\}_{k_{AB}} \\ 1. & B \to A: & \{f(N)\}_{k_{AB}} \end{array}$$

The goal of these two messages is to authenticate B from A's point of view, provided that they share an initial secret k_{AB}. This is done by a simple challenge-response transaction: A sends a random number (a *nonce*) encrypted with the secret key k_{AB}; B decrypts this number, applies a given function (for instance $f(N) = N+1$) to it, and sends the result back, also encrypted with k_{AB}; finally A checks the validity of the result, for instance by decrypting the message and checking the decryption against $f(N)$.

We model this protocol using the sets of public symbols $\mathcal{F}_{\mathrm{pub}} = \{\mathrm{senc}(2), \mathrm{sdec}(2), \mathrm{f}(1), \mathrm{a}, \mathrm{b}, \mathrm{i}, \mathrm{c}_1, \mathrm{c}_2 \dots\}$ and private symbols $\mathcal{F}_{\mathrm{priv}} = \{\mathrm{k}(2), \mathrm{n}, \mathrm{n}_1, \mathrm{n}_2 \dots\}$ where the numbers in parentheses denote the arities of non-constant symbols. We may write $\{x\}_y$ or $\mathrm{senc}(x, y)$ equivalently. The symbols $\mathrm{c}_1, \mathrm{c}_2 \dots$ and $\mathrm{n}, \mathrm{n}_1, \mathrm{n}_2 \dots$ are respectively pools of public and private constants used to model nonces. We equip terms with the following convergent subterm rewriting system $\mathcal{R}$, so as to model a symmetric, deterministic, length-preserving encryption scheme (that is, a *cipher*):

$$\mathrm{sdec}(\mathrm{senc}(x, y), y) \to x \qquad \mathrm{senc}(\mathrm{sdec}(x, y), y) \to x$$

The second rule states that any message x is a valid ciphertext for any key y. This characteristic property of ciphers is useful for preventing the trivial guessing attack on any message encrypted by a weak key, which arises when decryption fails whenever it is given a wrong decryption key.

More generally, we refer the reader to previous work [3, 6, 1, 10] for classical examples on how to model pairs, public-key encryption, hash functions, signatures... using convergent subterm rewriting systems.

3.1 Symbolic Traces

Informally, a *symbolic trace* (*e.g.* [19, 20, 8, 12]) is an execution trace of the protocol where the messages sent by the intruder (the sizes of which are unbounded) are replaced by fresh variables x_i. Each symbolic trace is associated to a constraint system which accounts for the conditions that the messages x_i must satisfy for the trace to be feasible. Importantly, we also keep track of the *computations* X_i done by the attacker to compute the x_i. This makes it possible to define a suitable notion of equivalence between traces.

Suppose that we want to prove the authentication property for one session on our example protocol. This boils down to asking whether the intruder I can emulate B in the normal session. The intruder constraint system corresponding to this problem is:

$$\exists x_1, \quad X_1[\{\mathrm{n}\}_{\mathrm{k(a,b)}}] =^? x_1 \quad \mathrm{sdec}(x_1, \mathrm{k(a,b)}) =^?_{\mathcal{R}} \mathrm{f(n)}$$

The first equation means that x_1, the answer of the intruder, must be computable from the message sent by A using a (public) context X_1. The second one is the test done by A upon receiving the second message. This system is easily showed unsatisfiable, either manually or using the procedure of Section 4. (Recall that k is a private symbol so it may not be used in X_1.) Hence, there exists no attack on authentication using only one session.

More generally, trace properties on security protocols are verified by checking that no symbolic trace corresponding to an attack is satisfiable. This is possible indeed because a bounded number of sessions of a protocol may only generate finitely many symbolic traces [19, 20, 8, 12].

3.2 Off-line Guessing Attacks

A more interesting problem arises if the key $\mathrm{k(a,b)}$ is a weak secret, that is, vulnerable to brute-force off-line testing. In [10], Corin *et al.* give a general definition of off-line guessing attacks using *static equivalence* [3, 1].

Static equivalence usually relates *frames* [3, 1], meant to represent sequences of messages sent on the network. In our setting, a *frame* is a tuple of first-order terms written $\Phi = \{w_1 \rhd t_1 \dots w_n \rhd t_n\}$ (this notation will prove useful in the next sections). Φ is *closed* iff all the t_i are closed terms. Two closed frames $\Phi = \{w_1 \rhd t_1 \dots w_n \rhd t_n\}$ and $\Phi' = \{w_1 \rhd t'_1 \dots w_n \rhd t'_n\}$ are *statically equivalent*, written $\Phi \approx_{\mathcal{R}} \Phi'$, iff for all n-ary closed (public) contexts C_1 and C_2, $C_1[t_1 \dots t_n] =_{\mathcal{R}} C_2[t_1 \dots t_n] \Leftrightarrow C_1[t'_1 \dots t'_n] =_{\mathcal{R}} C_2[t'_1 \dots t'_n]$.

Let Σ be an intruder constraint system modeling the satisfiability of a given symbolic trace τ of the protocol, involving a weak secret s. Using the same notation as before, Σ is written

$$\exists x_1 \dots x_m, \quad \left\{ \begin{array}{rcl} X_1[t_1 \dots t_{a_1}] & =^? & x_1 \\ & \dots & \\ X_m[t_1 \dots t_{a_m}] & =^? & x_m \\ s_1 & =^?_{\mathcal{R}} & s'_1 \\ & \dots & \\ s_n & =^?_{\mathcal{R}} & s'_n \end{array} \right.$$

Let $\Phi = \{w_1 \rhd t_1, \dots, w_{a_m} \rhd t_{a_m}\}$.

The idea behind Corin *et al.* [10]'s definition for off-line guessing attacks is the following. Assume that the intruder is given an additional message $t_{a_m+1} \in \{\mathrm{s}, \mathrm{s}'\}$ where s' is a fresh private constant. Let θ be a solution to Σ and θ' its (unique) extension to first-order variables as before. There is an off-line guessing attack on s at the end of the concrete trace $\tau\theta$ iff intuitively it is possible for the intruder to distinguish (off-line) whichever $t_{a_m+1} = \mathrm{s}$ (correct guess) or $t_{a_m+1} = \mathrm{s}'$ (wrong guess), that is, in terms of static equivalence:

$$\Phi\theta' \cup \{w_{a_m+1} \rhd \mathrm{s}\} \not\approx_{\mathcal{R}} \Phi\theta' \cup \{w_{a_m+1} \rhd \mathrm{s}'\} \tag{1}$$

In terms of intruder constraint systems, we model off-line guessing attacks as follows. Let X, Y be fresh second-order variables of arity $a_m + 1$ and x, y fresh first-order variables. For any term t, we define the (X, Y)-standard intruder constraint system $\Sigma[t]$:

$$\exists x_1 \dots x_m, x, y, \quad \left\{ \begin{array}{rcl} X_1[t_1 \dots t_{a_1}] & =^? & x_1 \\ & \dots & \\ X_m[t_1 \dots t_{a_m}] & =^? & x_m \\ X[t_1 \dots t_{a_m}, t] & =^? & x \\ Y[t_1 \dots t_{a_m}, t] & =^? & y \\ s_1 & =^?_{\mathcal{R}} & s'_1 \\ & \dots & \\ s_n & =^?_{\mathcal{R}} & s'_n \\ x & =^?_{\mathcal{R}} & y \end{array} \right.$$

Let s' be a fresh private constant.

FACT 1. *There exists a solution θ to Σ such that θ' fulfills equation (1) iff the two systems $\Sigma[\mathrm{s}]$ and $\Sigma[\mathrm{s}']$ are* not *equivalent.*

PROOF. Assume for instance that θ_1 is a solution to $\Sigma[\mathrm{s}]$ but not to $\Sigma[\mathrm{s}']$. (The other case is similar.) We let θ be the restriction of θ_1 to $\{X_1 \dots X_m\}$, $C_1 = X\theta_1$, and $C_2 = Y\theta_1$. Let θ' be the extension of θ to the first-order variables $x_1 \dots x_m$ in Σ as before. In the case of $\Sigma[\mathrm{s}]$, the extension of θ_1 is written $\theta'_1 = \theta' \{x \mapsto C_1[t_1\theta' \dots t_n\theta', \mathrm{s}],\ y \mapsto C_2[t_1\theta' \dots t_n\theta', \mathrm{s}]\}$, whereas in the case of $\Sigma[\mathrm{s}']$, it is written $\theta''_1 = \theta' \{x \mapsto C_1[t_1\theta' \dots t_n\theta', \mathrm{s}'],\ y \mapsto C_2[t_1\theta' \dots t_n\theta', \mathrm{s}']\}$. Since θ_1 is a solution to $\Sigma[\mathrm{s}]$, θ is a solution to Σ and we have $C_1[t_1\theta' \dots t_n\theta', \mathrm{s}] =_{\mathcal{R}} C_2[t_1\theta' \dots t_n\theta', \mathrm{s}]$. Besides, θ_1 may only fail on the last equation of $\Sigma[\mathrm{s}']$, thus we deduce $C_1[t_1\theta' \dots t_n\theta', \mathrm{s}'] \neq_{\mathcal{R}} C_2[t_1 \dots t_n, \mathrm{s}']$.

Conversely, let θ be a solution to Σ such that θ' satisfies (1). By assumption, there exist two contexts C_1 and C_2 such that *e.g.* $C_1[t_1\theta' \dots t_n\theta', \mathrm{s}] =_{\mathcal{R}} C_2[t_1\theta' \dots t_n\theta', \mathrm{s}]$ but $C_1[t_1\theta' \dots t_n\theta', \mathrm{s}'] \neq_{\mathcal{R}} C_2[t_1\theta' \dots t_n\theta', \mathrm{s}']$. Let $\theta_1 = \theta\{X \mapsto C_1,\ Y \mapsto C_2\}$. Then, θ_1 is a solution to $\Sigma[\mathrm{s}]$ but not to $\Sigma[\mathrm{s}']$. □

As an application, the intruder constraint system Σ corresponding to the main session of our example protocol is written:

$$\exists x_1, x_2, \quad \left\{ \begin{array}{rcl} X_1[\{\mathrm{n}\}_{\mathrm{k(a,b)}}] & =^? & x_1 \\ X_2[\{\mathrm{n}\}_{\mathrm{k(a,b)}},\ \{\mathrm{f}(\mathrm{sdec}(x_1, \mathrm{k(a,b)}))\}_{\mathrm{k(a,b)}}] & =^? & x_2 \\ \mathrm{sdec}(x_2, \mathrm{k(a,b)}) & =^?_{\mathcal{R}} & \mathrm{f(n)} \end{array} \right.$$

with $\mathrm{ar}(X_1) = 1$, $\mathrm{ar}(X_2) = 2$. One solution θ to this system is given by the normal run of the protocol[1]:

$$X_1\theta = w_1 \text{ thus } x_1\theta' = \{\mathrm{n}\}_{\mathrm{k(a,b)}}$$
$$X_2\theta = w_2 \text{ thus } x_2\theta' = \{\mathrm{f}(\mathrm{sdec}(\{\mathrm{n}\}_{\mathrm{k(a,b)}}, \mathrm{k(a,b)}))\}_{\mathrm{k(a,b)}} =_{\mathcal{R}} \{\mathrm{f(n)}\}_{\mathrm{k(a,b)}}$$

Let us define the two systems $\Sigma[\mathrm{k(a,b)}]$ and $\Sigma[\mathrm{s}']$ as above and extend θ by

$$X\theta = \mathrm{f}(\mathrm{sdec}(w_1, w_3)) \text{ and } Y\theta = \mathrm{sdec}(w_2, w_3).$$

[1] In this sense, the simple example presented here only involves a passive attacker. More complex off-line guessing attacks may be found in [15, 13].

Then θ is a solution to $\Sigma[\mathrm{k(a,b)}]$ but not to $\Sigma[\mathrm{s}']$. This corresponds to the classical guessing attack on the Handshake protocol [15]: by decrypting both messages of the protocols with the guess x and checking the relation

$$\mathrm{f}(\mathrm{sdec}(\{\mathrm{n}\}_{\mathrm{k(a,b)}}, x)) = \mathrm{sdec}(\{\mathrm{f(n)}\}_{\mathrm{k(a,b)}}, x)$$

it is possible to test whether $x = \mathrm{k(a,b)}$ and thus to recover the weak secret $\mathrm{k(a,b)}$ by brute-force testing.

4. DECISION PROCEDURE

We now describe a decision procedure for the satisfiability and the equivalence of (standard) intruder constraint systems. We begin by introducing *extended constraint systems.* These can be seen as some syntax for representing (generally infinite) sets of solutions to the initial problem. We then describe a set of transformation rules on the extended systems that is sound and complete for every convergent rewriting system. Finally we show how to enforce termination in the case of convergent subterm rewriting systems, and conclude the proof of Theorem 1.

Detailed proofs are available in an extended version [4].

4.1 Extended Constraint Systems

Let $\mathcal{R}$ be a convergent rewriting system and $\mathcal{Y}$ a finite set of second-order variables. An $(\mathcal{R}, \mathcal{Y})$-*extended constraint system* (or simply *constraint system* in this section) is a tuple $\Sigma = \Phi; \Psi; \mathcal{C}; \sigma$ where

- Φ is a finite set of expressions $\forall\beta.M \rhd t$, called *frame rules* (or simply *rules*) of Σ, where β ranges over finite sets of second-order variables;
- Ψ is a finite set of expressions $\forall\beta.M \bowtie N$, called *equations* of Σ;
- $\mathcal{C}$ is a finite set of *constraints* of the form $t_1 \stackrel{?}{=}_{\mathcal{R}} t_2$ and $X \rhd^? t$, where in the latter case, X may occur only once in $\mathcal{C}$;
- σ is an idempotent substitution satisfying $\mathrm{dom}(\sigma) \cap \mathrm{var}(\Phi; \Psi; \mathcal{C}) = \emptyset$.

We introduce quantifiers on second-order variables $\forall\beta$ for technical reasons regarding the termination of the procedure on convergent subterm rewriting systems. Those will be used in Section 4.3 and may be safely ignored for the moment. We apply the usual conventions on binders $\forall\beta$. Notably, equality between quantified expressions is understood modulo (arity-preserving) renaming of bound variables and deletion of useless ones (that is, $\forall\beta.M = \forall(\beta \cap \mathrm{var}(M)).M$). We let $\mathrm{var}(\forall\beta.M \rhd t) = \mathrm{var}(M) \cup \mathrm{var}(t) - \beta$, $\mathrm{var}(\forall\beta.M \bowtie N) = \mathrm{var}(M) \cup \mathrm{var}(N) - \beta$. Substitutions are applied accordingly: $(\forall\beta.M \rhd t)\sigma = \forall\beta.M\sigma \rhd t\sigma$ if $\beta \cap \mathrm{var}(\sigma) = \emptyset$. Besides we see $\bowtie$ as a commutative symbol.

The cryptographic intuition behind the four sets $\Phi; \Psi; \mathcal{C}; \sigma$ of each constraint system is the following. A frame rule $\forall\beta.M \rhd t$ in Φ records the fact that a term t is *computable* (or *deducible* [1]) by the intruder using any *computation* (or *recipe*) $M\mu$, with $\mathrm{dom}(\mu) \subseteq \beta$. Initially, recipes are simply parameters $w_1 \ldots w_{a_m}$ and the set of rules is simply a frame $\{w_1 \rhd t_1, \ldots, w_{a_m} \rhd t_{a_m}\}$. During the procedure, new facts are inferred. For instance, if $w_1 \rhd \mathrm{enc(a,b)}$ and $w_2 \rhd \mathrm{b}$ belong to Φ, we may add $\mathrm{dec}(w_1, w_2) \rhd \mathrm{a}$.

Equations $\forall\beta.M \bowtie N$ in Ψ correspond to relations that are visible to the intruder: intuitively any two computations $M\mu$ and $N\mu$, $\mathrm{dom}(\mu) \subseteq \beta$, yield the same result when parameters are substituted with their actual values. For instance, given the two rules $w_1 \rhd \mathrm{h(n)}$ and $w_2 \rhd \mathrm{n}$, a visible equation is $w_1 \bowtie \mathrm{h}(w_2)$.

$\mathcal{C}$ is a set of constraints: either *deducibility constraints* $X \rhd^? t$, meaning that t must be computable by the intruder using a (yet) unknown recipe X, or *equality constraints* $t_1 \stackrel{?}{=}_{\mathcal{R}} t_2$, so as to account for tests done by honest participants. Finally, σ is used to record the *solved variables* of a system.

A closed (well-formed) substitution θ with $\mathrm{dom}(\theta) = \mathcal{Y}$ is a *solution* to $\Sigma = \Phi; \Psi; \mathcal{C}; \sigma$, written $\theta \models \Sigma$, iff there exists a closed (well-formed) substitution λ with $dom(\lambda) \supseteq \mathrm{var}(\Sigma)$ such that:

- for every constraint $X \rhd^? t$ in $\mathcal{C}$, there exist an m-ary (public, closed) context C, some rules $\forall\beta_1.M_1 \rhd t_1 \ldots \forall\beta_m.M_m \rhd t_m \in \Phi$, some closed substitutions $\mu_1 \ldots \mu_m$ with $\mathrm{dom}(\mu_i) \subseteq \beta_i$ such that

  $$X\lambda = C[M_1\mu_1\lambda \ldots M_m\mu_m\lambda] \text{ and } t\lambda = C[t_1\lambda \ldots t_m\lambda];$$
- for every equation $t_1 \stackrel{?}{=}_{\mathcal{R}} t_2$ in $\mathcal{C}$, $t_1\lambda =_{\mathcal{R}} t_2\lambda$;
- λ *extends* σ, in the sense that $\sigma\lambda = \lambda$;
- λ is *related* to θ, meaning that for all $X \in \mathcal{Y}$, we have $X\theta =_{\mathcal{R}\cup\Psi\lambda} X\lambda$.

We have used $=_{\mathcal{R}\cup\Psi\lambda}$ to denote the equivalence relation associated to the rewriting rules in $\mathcal{R}$ and the equations in $\Psi\lambda$—seen as pairs of rewriting rules, one for each direction. We may write $\theta, \lambda \models \Sigma$ to specify a λ associated to a solution θ.

Intruder constraint systems of Section 2 are seen as extended constraint systems $\Sigma = \Phi; \emptyset; \mathcal{C}; \emptyset$ where, using the notations of Section 2, we let

$$\Phi = \{w_1 \rhd t_1, \ldots, w_{a_m} \rhd t_{a_m}\} \quad \text{and}$$
$$\mathcal{C} = \{X_1 \rhd^? x_1, \ldots, X_m \rhd^? x_m,\ s_1 \stackrel{?}{=}_{\mathcal{R}} s'_1, \ldots, s_n \stackrel{?}{=}_{\mathcal{R}} s'_n\}.$$

Solutions to such a Σ are defined equivalently using the definition of intruder constraint systems or that of extended constraint systems. Notice that, due to the regularity conditions on intruder constraint systems (Section 2), for every solution θ, there exists a unique λ (up to $=_{\mathcal{R}}$, once restricted to $\mathrm{var}(\Sigma)$) such that $\theta, \lambda \models \Sigma$.

A constraint system $\Sigma = \Phi; \Psi; \mathcal{C}; \sigma$ is *pre-solved* iff $\mathcal{C}$ is of the form above, that is, the right-hand sides of deducibility constraints $X \rhd^? t$ in $\mathcal{C}$ are pairwise distinct variables. It is *solved* iff besides $\mathcal{C}$ contains no equality constraints $s_i \stackrel{?}{=}_{\mathcal{R}} s'_i$.

FACT 2. *Every solved constraint system Σ is satisfiable.*

Indeed, let λ_0 assign fresh public constants to every unsolved variable in Σ, ensuring that $X\lambda_0 = x\lambda_0$ for each $X \rhd^? x$ in Σ, but for any other pair of unsolved variables v_1, v_2, $v_1\lambda \neq v_2\lambda$. Let $\lambda = \sigma\lambda_0$ and define θ as the restriction of λ to $\mathcal{Y}$. Then θ is a solution to Σ. In the following, we call such a θ a *principal solution* of Σ.

Our goal in the next subsection is to describe a set of transformation rules that is sound and complete in the following sense: for all intruder constraint system $\Sigma = \Phi; \emptyset; \mathcal{C}; \emptyset$ with Φ and $\mathcal{C}$ written as above,

- **(soundness)** for every Σ', if $\Sigma \Longrightarrow^* \Sigma'$ and $\theta \models \Sigma'$ then $\theta \models \Sigma$; moreover the set of equations Ψ' of Σ' is *sound* with respect to Σ: for all $\forall\beta.M \bowtie N$ in Ψ', if $\theta, \lambda \models \Sigma$ and $\theta, \lambda' \models \Sigma'$, we have that $(M\lambda')[t_1\lambda \dots t_{a_m}\lambda] =_{\mathcal{R}} (N\lambda')[t_1\lambda \dots t_{a_m}\lambda]$;
- **(completeness)** if $\theta \models \Sigma$ then there exists a solved constraint system Σ' such that $\Sigma \Longrightarrow^* \Sigma'$ and $\theta \models \Sigma'$.

All these notions are motivated by the following criterion for the satisfiability and the equivalence of intruder constraint systems.

Proposition 1. *Let $\Longrightarrow$ be a sound and complete set of transformation rules.*
1. An intruder constraint system Σ is satisfiable iff there exists a solved system Σ' such that $\Sigma \Longrightarrow^* \Sigma'$.
2. Let Σ_1 and Σ_2 be two (X, Y)-standard $(\mathcal{R}, \mathcal{Y})$-intruder constraint systems, with $\Sigma_2 = \Phi_2; \emptyset; \mathcal{C}_2; \emptyset$ and $\Phi_2 = \{w_1 \rhd t_1, \dots, w_{a_m} \rhd t_{a_m}\}$. The following conditions are equivalent:

(a) Every solution to Σ_1 is a solution to Σ_2.

*(b) For every solved constraint system Σ such that $\Sigma_1 \Longrightarrow^* \Sigma$, every (*resp. *at least one) principal solution θ of Σ satisfies (i) $\theta \models \Sigma_2$ and (ii) for every (*resp. *at least one) λ_2 such that $\theta, \lambda_2 \models \Sigma_2$, for every equation $\forall\beta.M \bowtie N$ of Σ, we have that $M[t_1\lambda_2 \dots t_{a_m}\lambda_2] =_{\mathcal{R}} N[t_1\lambda_2 \dots t_{a_m}\lambda_2]$.*

Provided that $\Longrightarrow$ is effective, this entails a semi-decision procedure for testing (non-)inclusion of sets of solutions: enumerate all the solved constraints systems reachable from Σ_1 and check conditions (i) and (ii) on each of them. Moreover, if $\Longrightarrow$ is finitely-branching and terminates, then by König's Lemma, the number of reachable solved constraint systems is finite, so we obtain a decision procedure.

4.2 Transformation Rules for Convergent Rewriting Systems

We now describe a set of transformation rules that is sound and complete for any convergent rewriting system. Let $a_m = \max\{\mathrm{ar}(Y) \,|\, Y \in \mathcal{Y}\}$ be the maximal arity of the second-order variables in $\mathcal{Y}$. We consider the two groups of transformation rules presented in Table 1.

The first three rules, **Project**, **Imitate** and **Coalesce**, aim to simplify deducibility constraints and bring constraint systems into a pre-solved form. The other rules in this paper only apply to constraint systems that are already pre-solved.

Specifically, rule **Project** uses a frame rule of Φ to solve a deducibility constraint in $\mathcal{C}$. By *fresh renaming*, we mean that ρ may substitute variables in M with distinct variables not occurring in the system yet. This is useful for lowering arities of second-variables so as to keep σ well-formed. As usual, we require the bound variables in β to be fresh, that is, $\beta \cap \mathrm{var}(\Phi; \Psi; \mathcal{C}; \sigma) = \emptyset$. Rule **Imitate** decomposes a deducibility constraint into smaller constraints by applying a public symbol in head position. Rule **Coalesce** merges deducibility constraints which deal with the same first-order variables.

It is not hard to prove that this set of three rules terminates. Indeed, **Imitate** and **Coalesce** create no first-order variables and decrease the total size of right-hand sides of deducibility constraints. **Project** either reduces the number of unsolved first-order variables (if $\mathrm{dom}(\mu) \neq \emptyset$) or decreases the size of deducibility constraints as well.

The next five rules constitute the main loop of the procedure. We discuss later their termination on convergent subterm rewriting systems. As already mentioned, these rules are restricted to pre-solved constraint systems. Hence, an application of any of those is generally followed by a number of **Project**, **Imitate** and **Coalesce** steps.

Rules **Narrowing-1**, **Narrowing-2** and **Constrain** are classical (see for instance [16, 12]). By "$l \to r$ fresh from $\mathcal{R}$", we mean that the rewriting rule $l \to r$ is obtained by renaming the variables of some rule in $\mathcal{R}$ so that they do not occur in the left-hand constraint system. Rules **Narrowing-1**, **Narrowing-2** aim to guess possible reductions, respectively in computable terms and in equality constraints. Rule **Constrain** tries to solve an equality constraint by syntactic unification.

Rule **Context** accounts for reductions that occur at the top of computable terms. An example of application of this rule is the following. Assume that $M_1 \rhd \mathrm{enc}(\mathrm{a}, x_0)$ is in Φ and $X_0 \rhd^? x_0$ in $\mathcal{C}$. Assume a fresh rewriting rule $l \to r = \mathrm{dec}(\mathrm{enc}(x, y), y) \to x$ from $\mathcal{R}$ and fresh variables X, X_1, X_2 of maximal arity. If $\Sigma = \Phi; \Psi; \mathcal{C}; \sigma$ is pre-solved then we have the following sequence of reductions:

$$\Phi; \Psi; \mathcal{C}; \sigma$$

$$\Longrightarrow_{\mathrm{Context}} \Phi \cup \{X \rhd x\};\ \Psi;\ \mathcal{C} \cup \{X \rhd^? \mathrm{dec}(\mathrm{enc}(x, y), y)\};\ \sigma$$

$$\Longrightarrow_{\mathrm{Imitate}} \Phi \cup \{\mathrm{dec}(X_1, X_2) \rhd x\};\ \Psi;\ \mathcal{C} \cup \{X_1 \rhd^? \mathrm{enc}(x, y),\ X_2 \rhd^? y)\};\ \sigma\{X \mapsto \mathrm{dec}(X_1, X_2)\}$$

$$\Longrightarrow_{\mathrm{Project}} \Phi \cup \{\mathrm{dec}(M_1, X_2) \rhd \mathrm{a}\};\ \Psi;\ \mathcal{C} \cup \{X_2 \rhd^? x_0\};\ \dots \{X_1 \mapsto M_1,\ x \mapsto \mathrm{a},\ y \mapsto x_0\}$$

$$\Longrightarrow_{\mathrm{Coalesce}} \Phi \cup \{\mathrm{dec}(M_1, X_0) \rhd \mathrm{a}\};\ \Psi;\ \mathcal{C};\ \dots \{X_2 \mapsto X_0\}$$

where the dots $(\dots)$ stand for the previous substitutions. Thus, we have inferred the new fact $\mathrm{dec}(M_1, X_0) \rhd \mathrm{a}$, by applying decryption at the top of computable terms (or supposedly computable terms in the case of x_0).

Rule **Relate** is needed for the completeness of **Coalesce**. It tries to find new visible equations, that is, different ways to obtain a same computable term.

We now state the soundness and the completeness of the transformation rules for any convergent rewriting system $\mathcal{R}$.

Proposition 2 (Soundness). *Let $\Sigma_0 = \Phi_0; \emptyset; \mathcal{C}_0; \emptyset$ be an intruder constraint system with $\Phi_0 = \{w_1 \rhd t_1, \dots, t_{a_m} \rhd t_{a_m}\}$. Let $\Sigma_0 \Longrightarrow^* \Sigma$ be a derivation using the rules of Table 1. If $\theta \models \Sigma$ then $\theta \models \Sigma_0$. Moreover, the sets of rules Φ and equations Ψ of Σ are* sound *with respect to Σ_0: for all $\theta, \lambda, \lambda_0$ such that $\theta, \lambda_0 \models \Sigma_0$ and $\theta, \lambda \models \Sigma$,*

1. *for every $\forall\beta.M \rhd t$ in Φ, $(M\lambda)[t_1\lambda_0 \dots t_{a_m}\lambda_0] =_{\mathcal{R}} t\lambda$;*
2. *for every $\forall\beta.M \bowtie N$ in Ψ, $(M\lambda)[t_1\lambda_0 \dots t_{a_m}\lambda_0] =_{\mathcal{R}} (N\lambda)[t_1\lambda_0 \dots t_{a_m}\lambda_0]$.*

The proof of soundness is done by induction on the derivation. Our proof for **Narrowing-1**, **Context** and **Relate** relies on a number of syntactic invariants. Notably, we establish the following important invariant, originating from the regularity condition (2) on intruder constraint systems:

> For every $\forall\beta.M \rhd t$ in Φ and $x \in \mathrm{var}(t)$, there exists $X \rhd^? t'$ in $\mathcal{C}$ such that $x \in \mathrm{var}(t')$ and either $X \in \mathrm{var}(\forall\beta.M)$ or $\mathrm{ar}(X) < \max\{i \,|\, w_i \in \mathrm{par}(M)\}$.

Intuitively each variable occurring in a deducible term t with $\forall\beta.M \rhd t \in \Phi$ is constrained by a second-order variable at a *lower level* than M.

PROPOSITION 3 (COMPLETENESS). *Let Σ_0 be an intruder constraint system. If $\theta \models \Sigma_0$, then there exists a solved constraint system Σ and a derivation $\Sigma_0 \Longrightarrow^* \Sigma$ using the rules of Table 1 such that $\theta \models \Sigma$.*

Completeness is shown by instrumenting the rules of Table 1 with the considered solution θ, that is, intuitively by defining transformations rules of the form $(\theta, \lambda \models \Sigma) \Longrightarrow (\theta, \lambda' \models \Sigma')$. We successively prove the correctness of the instrumented rules (that is, the symbol $\models$ is actually preserved), progress (if the system is not solved, at least one rule applies) and termination.

4.3 Enforcing Termination on Convergent Subterm Rewriting Systems

We now assume a convergent *subterm* rewriting system $\mathcal{R}$ and show how to enforce the termination of the transformation rules.

We have already proved the termination of rules **Project**, **Imitate** and **Coalesce**. Concerning the narrowing rules, we enforce termination by using a variant of the *basic narrowing* strategy (see for instance [12]). Specifically, we augment constraint systems with a fifth component $\mathcal{N}$, standing for a set of first-order terms. The set $\mathcal{N}$ is meant to store terms known to be $\mathcal{R}$-reduced. Semantically, θ is a solution to $\Sigma = \Phi; \Psi; \mathcal{C}; \sigma; \mathcal{N}$ for some λ iff $\theta, \lambda \models \Psi; \Psi; \mathcal{C}; \sigma$ in the previous sense *and* for all $t \in \mathcal{N}$, $t\lambda$ is $\mathcal{R}$-reduced. Hence, terms which appear in $\mathcal{N}$ need not be narrowed anymore. Initially, that is, for intruder constraint systems, $\mathcal{N}$ is set to $\emptyset$. We write $\mathrm{st}(\mathcal{N})$ for the set of subterms of terms in $\mathcal{N}$.

The case of rule **Context** is more problematic as it may introduce new variables indefinitely. We address this issue by introducing four additional transformation rules, meant to be applied eagerly on pre-solved constraint systems.

The new set of transformation rules is presented in Table 2. We have omitted rules **Imitate** and **Coalesce** which are the same as in Table 1, except for the additional component $\mathcal{N}$ which is left unchanged.

As suggested, rules **Narrowing-{1,2}** now require that the narrowed term $t|_p$ (*resp.* $(t_1 =_\mathcal{R} t_2)|_p$) do not belong to $\mathrm{st}(\mathcal{N})$. Rules **Project** and **Relate** add their argument t to $\mathcal{N}$ in order to prevent further narrowing inside t. Similarly, rule **Constrain** records the unified term $t_1\mu$ as being $\mathcal{R}$-reduced.

Interestingly, rules **Narrowing-{1,2}** and **Context** tag their result $r\sigma$ as being $\mathcal{R}$-reduced as well. This is crucial for the termination of the algorithm as it entails that the number of positions p eligible for narrowing steps does not increase with rule **Context** and strictly decreases with rules **Narrowing-{1,2}**.

As for the new rules, the two simple rules **Clean-1** and **Clean-2** delete useless variables in the system. In the same vein, rule **Generalize** adds universal quantifiers on second-order variables X which appear free in frame rules and equations but nowhere else. This aims to reduce the number of rules and equations, since these are considered modulo renaming of bound variables.

Rule **Discard** is a variant of **Relate** used to remove frame rules $\forall\beta.M \rhd t$ that are intuitively subsumed by existing rules $\forall\beta_i.M_i \rhd t_i$ and existing constraints $X_j \rhd^? x_j$, provided that a new equation is added to Ψ. The sets of bound variables $\beta, \beta_1 \dots \beta_n$ are required to be fresh and mutually disjoint. The last two technical conditions in the premisses ensure the completeness of this rule when it is applied eagerly.

We now make precise the control on transformation rules, that is, which sequences of rules need to be considered by the algorithm. For simplicity, we assume that the unification procedure tries to match its second argument against the first one whenever possible. In other words, $\mathrm{mgu}(t, l) = \sigma$ whenever $\mathrm{var}(t) \cap \mathrm{var}(l) = \emptyset$ and $t = l\sigma$. In rule **Project**, we also require that ρ does not rename second-order variables in $\mathrm{var}(M)$ more than necessary, that is formally, for all $Y \in \mathrm{dom}(\rho)$, $\mathrm{ar}(Y) > \mathrm{ar}(X)$ and $\mathrm{ar}(Y\rho) = \mathrm{ar}(X)$. The choice of arity $a_m + 1$ rather than a_m for the fresh second-order variables introduced by **Context** and **Related** is to ensure that rule **Coalesce** substitutes these variables in priority.

Let Σ be an intruder constraint system. A sequence of transformations (*derivation*) $\Sigma \Longrightarrow^* \Sigma'$ by the rules of Table 2 is *standard* iff it has the following structure:

- If Σ_1 occurs before Σ_2 in the derivation, with $\Sigma_i = \Phi_i; \Psi_i; \mathcal{C}_i; \sigma_i; \mathcal{N}_i$, and both constraint systems are pre-solved and saturated for the last four rules, **Discard**, **Clean-{1,2}** and **Generalize**, then $\Phi_1; \Psi_1; \mathcal{C}_1; \sigma_1 \neq \Phi_2; \Psi_2; \mathcal{C}_2; \sigma_2$.
- Each rule **Narrowing-{1,2}**, **Constrain**, **Context**, **Relate** is followed by a maximal sequence of **Project**, **Imitate**, **Coalesce**, and then, if a pre-solved form is reached, by a maximal sequence of **Discard**, **Clean-1**, **Clean-2** and **Generalize**, in this order of priority.
- Rule **Discard** is applied in priority to the most recently created frame rules $\forall\beta.M \rhd t$. (Formally, each frame rule created by **Context** is labeled with the value of a global counter, incremented each time. Rules with the highest labels are discarded in priority.)

Finally, we state our main theorem.

THEOREM 2. *Standard derivations form a sound, complete, effective and finitely branching (up to renaming) transformation system. Moreover there exists no infinite standard derivation.*

We deduce the decision result of Theorem 1 using König's Lemma and Proposition 1. The proofs of soundness and completeness follow the same structure as previously. For rules **Narrowing-{1,2}** and **Context**, the fact that $r\sigma$ may be added to $\mathcal{N}$ is justified by the following property of subterm rewriting systems: if $f(t_1 \dots t_n) = l\mu$ for some rewriting rule $l \to r \in \mathcal{R}$, and all the $t_1 \dots t_n$ are $\mathcal{R}$-reduced then $r\mu$ is reduced. This property together with the convergence of $\mathcal{R}$ turns out to be sufficient to imply the completeness of standard derivations.

As for the proof of termination, due to the constraints on narrowing positions, the rules **Narrowing-{1,2}** terminate independently from the other rules. So does rule **Constrain**, as well as the last four rules of Table 2. Concerning **Context** and **Relate**, we prove that in any reachable system Σ that is saturated for **Discard**, the number of right-hand sides t of frame rules in Σ cannot exceed the number of subterms of right-hand sides t_0 in Σ_0—or more exactly the number of subterms of such t_0, once they have been narrowed and added to $\mathcal{N}$. We then bound the number of frame rules created by **Context**, and deduce termination for the whole set of rules.

5. CONCLUSION

In this work we described a class of second-order E-unification problems and provided a terminating procedure to decide their satisfiability and their equivalence, in the case where the equational theory E is presented by a convergent subterm rewriting system. This decision result is interesting by itself as it is not implied by previous work in the area, for instance [16, 21, 17].

A major application, for which these constraint systems were intended, is the security of cryptographic protocols against off-line guessing attacks. No previous decision results existed for such properties, in any case, not using the recent general definition of Corin *et al.* [10]. Using our main result, we recovered the decidability of trace properties, and proved the decidability of security against of off-line guessing attacks for a bounded number of sessions.

As future work, we foresee to apply our notion of equivalence between symbolic traces to other security properties, such as strong secrecy and resistance to on-line guessing attacks. On the long term an interesting avenue would be to extend our result to equational theories involving algebraic properties such as associativity-commutativity, XOR or homomorphism.

Acknowledgments

We are very grateful to Véronique Cortier, Florent Jacquemard and Stéphanie Delaune for helpful discussions, to Jean Goubault-Larrecq and Steve Kremer for useful comments. This work was partially supported by the RNTL project PROUVÉ and the ACI-SI Rossignol.

6. REFERENCES

[1] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *LNCS*, pages 46–58, 2004.

[2] M. Abadi and V. Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 62–76, 2005.

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

[4] M. Baudet. Deciding security of protocols against off-line guessing attacks (extended version), 2005. Manuscript.

[5] B. Blanchet. Personal web page. `http://www.di.ens.fr/~blanchet`.

[6] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Proc. 25th IEEE Symposium on Security and Privacy (SSP'04)*, pages 86–100, 2004.

[7] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *Proc. 20th IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 331–340, 2005.

[8] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST–TCS'03)*, volume 2914 of *LNCS*, pages 124–135, 2003.

[9] E. Cohen. Proving protocols safe from guessing. In *Proc. Foundations of Computer Security (FCS'02)*, pages 85–92, 2002.

[10] R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. In *Proc. 2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04)*, volume 121 of *ENTCS*, pages 47–63, 2005.

[11] R. Corin, S. Malladi, J. Alves-Foss, and S. Etalle. Guess what? Here is a new tool that finds some new guessing attacks. In *Proc. Workshop on Issues in the Theory of Security (WITS'03)*, pages 62–71, 2003.

[12] S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proc. 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, 2004.

[13] S. Delaune and F. Jacquemard. A theory of dictionary attacks and its complexity. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 2–15, 2004.

[14] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, 1983.

[15] L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[16] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 257–321. MIT Press, 1991.

[17] J. Levy and M. Veanes. On the undecidability of second-order unification. *Information and Computation*, 159(1-2):125–150, 2000.

[18] G. Lowe. Analysing protocols subject to guessing attacks. *Journal of Computer Security*, 12(1):83–98, 2004.

[19] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, pages 166–175, 2001.

[20] J. K. Millen and V. Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proc. 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 47–61, 2003.

[21] W. Snyder and J. H. Gallier. Higher-order unification revisited. *Journal of Symbolic Computations*, 8:101–140, 1989.

Project

$$\frac{\begin{array}{c}\mu = \mathrm{mgu}(t, f(t_1 \ldots t_n)) \qquad X \notin \mathrm{var}(M) \qquad \forall w_i \in \mathrm{par}(M), i \leq \mathrm{ar}(X) \\ \rho \text{ fresh renaming such that } \mathrm{dom}(\rho) \subseteq \mathrm{var}(M) \text{ and } \forall Y \in \mathrm{var}(M\rho), \mathrm{ar}(Y) \leq \mathrm{ar}(X)\end{array}}{\Phi \cup \{\forall \beta. M \rhd t\};\ \Psi;\ \mathcal{C} \uplus \{X \rhd^? f(t_1 \ldots t_n)\};\ \sigma \Longrightarrow (\ \Phi \cup \{\forall \beta. M \rhd t\};\ \Psi;\ \mathcal{C};\ \sigma\ )\ \{X \mapsto M\rho\}\rho\mu}$$

Imitate

$$\frac{f \in \mathcal{F}_{\mathrm{pub}} \qquad X_1 \ldots X_n \text{ fresh second-order variables with } \mathrm{ar}(X_i) = \mathrm{ar}(X)}{\begin{array}{l}\Phi;\ \Psi;\ \mathcal{C} \uplus \{X \rhd^? f(t_1 \ldots t_n)\};\ \sigma \\ \Longrightarrow (\ \Phi;\ \Psi;\ \mathcal{C} \cup \{X_1 \rhd^? t_1, \ldots, X_n \rhd^? t_n\};\ \sigma\ )\ \{X \mapsto f(X_1 \ldots X_n)\}\end{array}}$$

Coalesce

$$\frac{\mathrm{ar}(X_1) \leq \mathrm{ar}(X_2)}{\Phi;\ \Psi;\ \mathcal{C} \uplus \{X_1 \rhd^? x,\ X_2 \rhd^? x\};\ \sigma \Longrightarrow (\ \Phi;\ \Psi;\ \mathcal{C} \cup \{X_1 \rhd^? x\};\ \sigma\ )\ \{X_2 \mapsto X_1\}}$$

Narrowing-1

$$\frac{l \to r \text{ fresh rule from } \mathcal{R} \qquad t|_p \notin \mathcal{X} \qquad \mu = \mathrm{mgu}(t|_p, l)}{\Phi \uplus \{\forall \beta. M \rhd t\};\ \Psi;\ \mathcal{C};\ \sigma \Longrightarrow (\ \Phi \cup \{\forall \beta. M \rhd t[p := r]\};\ \Psi;\ \mathcal{C};\ \sigma\ )\ \mu}$$

Narrowing-2

$$\frac{l \to r \text{ fresh rule from } \mathcal{R} \qquad (t_1 =^?_{\mathcal{R}} t_2)|_p \notin \mathcal{X} \qquad \mu = \mathrm{mgu}((t_1 =^?_{\mathcal{R}} t_2)|_p, l)}{\Phi;\ \Psi;\ \mathcal{C} \uplus \{t_1 =^?_{\mathcal{R}} t_2\};\ \sigma \Longrightarrow (\ \Phi;\ \Psi;\ \mathcal{C} \uplus \{(t_1 =^?_{\mathcal{R}} t_2)[p := r]\};\ \sigma\ )\ \mu}$$

Constrain

$$\frac{\mu = \mathrm{mgu}(t_1, t_2)}{\Phi;\ \Psi;\ \mathcal{C} \uplus \{t_1 =^?_{\mathcal{R}} t_2\};\ \sigma \Longrightarrow (\ \Phi;\ \Psi;\ \mathcal{C};\ \sigma\ )\ \mu}$$

Context

$$\frac{l \to r \text{ fresh rule from } \mathcal{R} \qquad X \text{ fresh second-order variable of arity } a_m}{\Phi;\ \Psi;\ \mathcal{C};\ \sigma \Longrightarrow \Phi \cup \{X \rhd r\};\ \Psi;\ \mathcal{C} \cup \{X \rhd^? l\};\ \sigma}$$

Relate

$$\frac{X \text{ fresh second-order variable of arity } a_m}{\Phi \cup \{\forall \beta. M \rhd t\};\ \Psi;\ \mathcal{C};\ \sigma \Longrightarrow \Phi \cup \{\forall \beta. M \rhd t\};\ \Psi \cup \{\forall \beta. X \bowtie M\};\ \mathcal{C} \cup \{X \rhd^? t\};\ \sigma}$$

Table 1: Transformation rules for convergent rewriting systems

Each of the last five rules additionally requires its left-hand constraint system to be pre-solved, that is, saturated for the first three rules.

Project

$$\frac{\begin{array}{l}\mu = \mathrm{mgu}(t, f(t_1 \dots t_n)) \qquad X \notin \mathrm{var}(M) \qquad \forall w_i \in \mathrm{par}(M), i \leq \mathrm{ar}(X) \\ \rho \text{ fresh renaming such that } \mathrm{dom}(\rho) \subseteq \mathrm{var}(M) \text{ and } \forall Y \in \mathrm{var}(M\rho), \mathrm{ar}(Y) \leq \mathrm{ar}(X)\end{array}}{\Phi \cup \{\forall\beta.M \rhd t\};\ \Psi;\ \mathcal{C} \uplus \{X \rhd^? f(t_1 \dots t_n)\};\ \sigma;\ \mathcal{N} \Longrightarrow (\,\Phi \cup \{\forall\beta.M \rhd t\};\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \cup \{t\}\,)\,\{X \mapsto M\rho\}\rho\mu}$$

Narrowing-1

$$\frac{l \to r \text{ fresh rule from } \mathcal{R} \qquad t|_p \notin \mathcal{X} \cup \mathrm{st}(\mathcal{N}) \qquad \mu = \mathrm{mgu}(t|_p, l)}{\Phi \uplus \{\forall\beta.M \rhd t\};\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \Longrightarrow (\,\Phi \cup \{\forall\beta.M \rhd t[p := r]\};\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \cup \{r\}\,)\,\mu}$$

Narrowing-2

$$\frac{l \to r \text{ fresh rule from } \mathcal{R} \qquad (t_1 =^?_{\mathcal{R}} t_2)|_p \notin \mathcal{X} \cup \mathrm{st}(\mathcal{N}) \qquad \mu = \mathrm{mgu}((t_1 =^?_{\mathcal{R}} t_2)|_p, l)}{\Phi;\ \Psi;\ \mathcal{C} \uplus \{t_1 =^?_{\mathcal{R}} t_2\};\ \sigma;\ \mathcal{N} \Longrightarrow (\,\Phi;\ \Psi;\ \mathcal{C} \uplus \{(t_1 =^?_{\mathcal{R}} t_2)[p := r]\};\ \sigma;\ \mathcal{N} \cup \{r\}\,)\,\mu}$$

Constrain

$$\frac{\mu = \mathrm{mgu}(t_1, t_2)}{\Phi;\ \Psi;\ \mathcal{C} \uplus \{t_1 =^?_{\mathcal{R}} t_2\};\ \sigma;\ \mathcal{N} \Longrightarrow (\,\Phi;\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \cup \{t_1\}\,)\,\mu}$$

Context

$$\frac{l \to r \text{ fresh rule from } \mathcal{R} \qquad X \text{ fresh second-order variable of arity } a_m + 1}{\Phi;\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \Longrightarrow \Phi \cup \{X \rhd r\};\ \Psi;\ \mathcal{C} \cup \{X \rhd^? l\};\ \sigma;\ \mathcal{N} \cup \{r\}}$$

Relate

$$\frac{X \text{ fresh second-order variable of arity } a_m + 1}{\Phi \cup \{\forall\beta.M \rhd t\};\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \Longrightarrow \Phi \cup \{\forall\beta.M \rhd t\};\ \Psi \cup \{\forall\beta.X \bowtie M\};\ \mathcal{C} \cup \{X \rhd^? t\};\ \sigma;\ \mathcal{N} \cup \{t\}}$$

Discard

$$\frac{\begin{array}{l} t = C[t_1 \dots t_n, x_1 \dots x_m] \qquad t, t_1 \dots t_n \in \mathrm{st}(\mathcal{N}) \\ \forall\beta_1.M_1 \rhd t_1, \dots, \forall\beta_n.M_n \rhd t_n \in \Phi \qquad X_1 \rhd^? x_1, \dots, X_m \rhd^? x_m \in \mathcal{C} \\ \max\{j \mid w_j \in \mathrm{par}(M_1 \dots M_n)\} \leq \max\{j \mid w_j \in \mathrm{par}(M)\} \\ \forall Y \in \mathrm{var}(\forall\beta_1 \dots \beta_n.C[M_1 \dots M_n, X_1 \dots X_m]), \begin{cases} \text{either} & Y \in \mathrm{var}(\forall\beta.M) \\ \text{or} & \mathrm{ar}(Y) < \max\{j \mid w_j \in \mathrm{par}(M)\} \end{cases} \end{array}}{\Phi \uplus \{\forall\beta.M \rhd t\};\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \Longrightarrow \Phi;\ \Psi \cup \{\forall\beta_1 \dots \beta_n, \beta.C[M_1 \dots M_n, X_1 \dots X_m] \bowtie M\};\ \mathcal{C};\ \sigma;\ \mathcal{N}}$$

Clean-1

$$\frac{v \notin \mathcal{Y}}{\Phi;\ \Psi;\ \mathcal{C};\ \sigma \uplus \{v \mapsto T\};\ \mathcal{N} \Longrightarrow \Phi;\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N}}$$

Clean-2

$$\frac{x \notin \mathrm{var}(\Phi;\ \mathcal{C};\ \sigma)}{\Phi;\ \Psi;\ \mathcal{C} \uplus \{X \rhd^? x\};\ \sigma;\ \mathcal{N} \Longrightarrow \Phi;\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N}}$$

Generalize

$$\frac{X \in \mathrm{var}(\Phi) \cup \mathrm{var}(\Psi) - \mathrm{var}(\mathcal{C}) - \mathrm{var}(\sigma) - \mathcal{Y}}{\Phi;\ \Psi;\ \mathcal{C};\ \sigma;\ \mathcal{N} \Longrightarrow \{\forall X.\forall\beta.M \rhd t\}_{(\forall\beta.M \rhd t) \in \Phi};\ \{\forall X.\forall\beta.M \bowtie N\}_{(\forall\beta.M \bowtie N) \in \Psi};\ \mathcal{C};\ \sigma;\ \mathcal{N}}$$

Table 2: Transformation rules for convergent subterm rewriting systems

Rules **Imitate** and **Coalesce** are similar to those of Table 1 and thus omitted.
The five rules from **Narrowing-1** to **Relate** additionally require their left-hand constraint systems to be pre-solved and saturated for the last four rules.
Each of the last four rules requires its left-hand constraint system to be pre-solved and saturated for the higher rules in this group.