Automatic Verification of Time Sensitive Cryptographic Protocols*

Giorgio Delzanno[1] and Pierre Ganty[2]

[1] Dip. di Informatica e Scienze dell'Informazione - Università di Genova
via Dodecaneso 35, 16146 Genova, Italy
giorgio@disi.unige.it
[2] Département d'Informatique - Université Libre de Bruxelles
Boulevard du Triomphe, 1050 Bruxelles, Belgium
pganty@ulb.ac.be

Abstract. We investigate the applicability of symbolic exploration to the automatic verification of secrecy and authentication properties for time sensitive cryptographic protocols. Our formal specifications are given in *multiset rewriting over first order atomic formulas* enriched with *constraints* so as to uniformly model fresh name generation and validity condition of time stamps. Our verification approach is based on data structures for symbolically representing sets of configurations of an arbitrary number of parallel protocol sessions. As a case study we discuss the verification of timed authentication for the Wide Mouth Frog protocol.

1 Introduction

Several authentication and key-establishment protocols make use of *time-stamps* to avoid possible replay attacks of malicious intruders. However, time-stamps are often abstracted away in formal models of cryptographic protocols (see e.g. [4, 6,19]). One reason for the use of this abstraction is that all known decidability results for verification of crypto-protocols are given for untimed models (see e.g. [14,18]). Powerful theorem provers like Isabelle in [2], Spass in [8], and PVS in [9] have been applied to verify timed dependent security properties.

In this paper we will present an automatic procedure for proving secrecy and authentication properties of time sensitive cryptographic protocols. Our procedure is an extension of the symbolic exploration method devised for the specification language MSR($\mathcal{L}$) of [5]. By combining multiset rewriting over first order atomic formulas [7] and linear arithmetic constraints, MSR($\mathcal{L}$) can be naturally applied to specify *unbounded* models of cryptographic protocols with *fresh name generation* and validity conditions for *time-stamps*. Our extension to the technique of [5] is based on the following points.

* This work was partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-39252 AVISPA project.

K. Jensen and A. Podelski (Eds.): TACAS 2004, LNCS 2988, pp. 342–356, 2004.

As a first technical contribution, we will define a new *data structure* for representing infinite set of configurations of (unbounded) MSR($\mathcal{L}$) specifications. Our data structure combines two logics: the existential zones over n-ary predicates with linear arithmetic constraints of [5], and the first order term language enriched with the **sup** operator of [6]. Since existential zones are given an upward closed semantics with respect to inclusion of configurations, our data structure can be used to finitely represent snapshots of executions of an unbounded number of parallel protocol sessions. In this setting first order terms can be used to represent structured and partially specified data, whereas the special operator **sup** can be used to represent patterns occurring at arbitrary depth in a message. The term $\mathbf{sup}(t)$ represents in fact any term containing t as a subterm. Finally, linear arithmetic constraints are used to model freshness and validity conditions for nonces and time-stamps. To exploit the upward closed semantics of our symbolic representation, we incorporate the new data structure within a backward search scheme. The resulting *symbolic backward exploration* procedure is based on effective *entailment* and *pre-image* operators. These operators are defined via unification and subsumption for first order terms extended with the **sup** operator, and constraint operations like satisfiability and variable elimination.

Since verification of (control) reachability problems for unbounded cryptographic protocols is undecidable [7], termination of state exploration cannot be guaranteed in general. However, when terminating, our search procedure returns a *symbolic reachability graph* that finitely represents an infinite set of protocol traces. As a second technical contribution, we establish conditions under which the symbolic reachability graph can be used to verify security properties like *secrecy* and *authentication* in presence of validity conditions for time-stamps. The conditions we propose here can be verified by using a visit of the graph. Specifically, in order to verify that secrecy holds, we have to check that the set of nodes of the constructed graph does not contain initial protocol configurations. For authentication properties, we have to search for specific pairs of principals states along paths in the graph starting from an initial protocol configuration.

By using the term manipulation library, the graph package, and the linear constraint solver provided by Sicstus Prolog, we have extended the symbolic verification procedure defined for verification of *parametric data consistency protocols* in [5] to cope with the new class of protocols, properties, and symbolic data structures we will discuss in the present paper. As a practical contribution, we present here a detailed analysis of a timed model of the Wide Mouth Frog protocol of [3]. As shown in Section 5, using our method we have automatically proved *non-injective timed agreement* for the fixed version proposed by Lowe in [11]. Our results are obtained for an unbounded protocol model and for *parametric delays* in the validity conditions of time-stamps. In Section 6 we will compare the results of this analysis with two other methods (based on semi-automated [9] and finite-state verification [13]) applied to this protocol.

We believe that the main novelty of the resulting method with respect to existing ones is that it allows us to handle unbound parallelism, complex term structures, time-stamps, and freshness of names both at the level of specification and analy-

sis in a uniform and effective way. Furthermore, the symbolic reachability graph allows us to verify security and authentication properties or to extract potential attacks (in form of a symbolic trace), a feature often not so easy in other verification methods based on theorem proving like TAPS [8].

2 Time Sensitive Cryptographic Protocols in MSR($\mathcal{L}$)

Let us first recall the main definitions of the language MSR($\mathcal{L}$)[5]. In this paper we will use $\cdot|\cdot$ and ϵ as multiset constructors and $\oplus$ to denote multiset union. Furthermore, we will use $Fv(t)$ to denote the set of free variables of a term or formula t. An MSR($\mathcal{L}$) specification $\mathcal{S}$ is a tuple $\langle \mathcal{P}, \Sigma, \mathcal{V}, \mathcal{I}, \mathcal{R} \rangle$, where $\mathcal{P}$ is a set of predicate symbols, Σ a first order signature, $\mathcal{V} = \mathcal{V}_t \cup \mathcal{V}_i$, $\mathcal{V}_t$ is a set of *term* variables, $\mathcal{V}_t$ is a set of *integer* variables, $\mathcal{V}_t \cap \mathcal{V}_i = \emptyset$, $\mathcal{I}$ is a set of initial configurations, and $\mathcal{R}$ is a finite set of labelled rules. A labelled rule has the form $\alpha : \mathcal{M} \longrightarrow \mathcal{N} : \varphi$, where α is a label, $\mathcal{M}$ and $\mathcal{N}$ are two (possibly empty) multisets of atomic formulas built on $\mathcal{P}$ and Σ, and φ is a *linear arithmetic constraint* such that $Fv(\varphi) \subseteq Fv(\mathcal{M} \oplus \mathcal{M}') \cap \mathcal{V}_i$. In this paper we will always embed *integer variables* within unary casting symbols working as *types*. For instance, in the rule $\alpha : p(ts(x), h(u)) \mid q(ts(y), h(z)) \rightarrow p(ts(y)) \mid q(ts(y), h(u, z)) : x > y$ the template $ts(\cdot)$ is used to encapsulate the integer variables x and y within a term.

The operational semantics of $\mathcal{S}$ is defined via the rewriting relation $\Rightarrow_{\mathcal{R}}$ defined over *configurations*. A configuration is a multiset of ground atomic formulas like $p(ts(3)) \mid q(ts(4), h(5))$. Given two configurations $\mathcal{M}_1$ and $\mathcal{M}_2$, $\mathcal{M}_1 \Rightarrow_{\mathcal{R}} \mathcal{M}_2$ if and only if there exists a configuration $\mathcal{Q}$ s.t. $\mathcal{M}_1 = \mathcal{N}_1 \oplus \mathcal{Q}$, $\mathcal{M}_2 = \mathcal{N}_2 \oplus \mathcal{Q}$, and $\mathcal{N}_1 \longrightarrow \mathcal{N}_2$ is a ground instance of a rule in $\mathcal{R}$. A ground instance of a rule $\alpha : \mathcal{M} \longrightarrow \mathcal{M}' : \varphi$ is obtained by extending a substitution in the set of solutions $Sol(\varphi)$ of the constraint φ to a grounding substitution (i.e. with ground terms in its range) for $Fv(\mathcal{M}, \mathcal{M}')$. Given a set of MSR($\mathcal{L}$) configurations S, the *predecessor* operator is defined as

$$Pre_{\mathcal{R}}(S) = \{\mathcal{M} \mid \exists \mathcal{M}' \in S \ s.t. \ \mathcal{M} \Rightarrow_{\mathcal{R}} \mathcal{M}'\}.$$

A configuration $\mathcal{M}$ is *reachable* if $\mathcal{M}_0 \in Pre^*_{\mathcal{R}}(\{M\})$ for some $\mathcal{M}_0 \in \mathcal{I}$. $Pre^*_{\mathcal{R}}$ represents here the transitive closure of the predecessor relation.

2.1 Time Sensitive Crypto-protocols

In the following we will assume the reader to be familiar with the use of multiset rewriting for the specification of cryptographic protocols (for more details, see e.g., [7]). Under the *perfect cryptography* assumption, time sensitive cryptographic protocols can be specified in MSR($\mathcal{L}$) as follows. Identifiers of honest principals are represented as integer numbers greater than zero. The intruder (Trudy) has the special identifier 0. The current state of honest principals is represented via atomic formulas like $r_i(id(n), \langle t_1, \ldots, t_n \rangle)$, meaning that the honest

agent $n > 0$ has executed the i-th step of the protocol in the role r (and he/she is ready for next step); $t_1, \ldots, t_n$ are terms representing his/her current knowledge. Data will be represented here by means of first order terms like $id(\cdot)$ (principal identifier), $sk(\cdot)$ (secret key), $sk(\cdot,\cdot)$ (shared key), $ts(\cdot)$ (time-stamp), $enc(\cdot,\cdot)$ (encrypted message), and $\langle \cdot, \ldots, \cdot \rangle$ (tuple constructor).

The knowledge of the attacker (Trudy) is represented by a collection of atomic formulas of the shape $mem(t)$ for some term t. Trudy's behavior can be described by using a Dolev-Yao model as in [7]. We will come back to this point in Section 5. A *message* sent on a directional channel is represented by means of an atomic formula $net(sender, receiver, message)$. Encrypted data is represented as the term $enc(key, t)$, where key is a term like $sk(a, b)$ denoting a key shared between a and b, and t is a term.

To model time-stamps, we use a *global clock* whose current value is represented via an atomic formula $clock(\cdot)$. The value of the clock is non-deterministically updated via the following rule:

$$\mathbf{time}: \ clock(ts(now)) \ \longrightarrow \ clock(ts(next)) \ : \ next > now$$

This rule introduces arbitrary *delays* between principals actions. Variables shared between the clock formula and a message can be used to represent generation of a time-stamp. The behavior of honest principals is defined by rule having the following shape:

$$\alpha: \ clock(\mathbf{ts}) \mid fresh(sk(y)) \mid r(id(a), \mathbf{k}) \mid net(id(b), id(a), \mathbf{m}) \ \longrightarrow \\ clock(\mathbf{ts'}) \mid fresh(sk(y')) \mid r'(id(a), \mathbf{k'}) \mid net(id(a), id(b'), \mathbf{m'}) \ : \ \varphi$$

This template represents the following transition: at instant $\mathbf{ts}$ principal a receives message $\mathbf{m}$ apparently from b, updates his/her state from r to r' and its knowledge from $\mathbf{k}$ to $\mathbf{k'}$, and, at instant $\mathbf{ts'}$, he/she sends a new message $\mathbf{m'}$. Freshness of a value k (key, nonce) can be ensured by adding the constraint $y' > k > y$ in the constraint φ. If we assume that a single principal step requires negligible time, then $\mathbf{ts} = \mathbf{ts'} = ts(now)$ for some integer variable now representing the current instant. *Parameters* in the validity conditions of time-stamps (e.g. delays) can be specified by including atomic formulas working as global memory in the initial protocol configuration as explained in the example of the next section.

2.2 The Wide Mouth Frog (WMF) Protocol

The goal of the WMF protocol [3] is to allow two principals to exchange a secret key via a trusted server. In the first step the initiator A sends to the server S the message $A, \{B, K, T_1\}_{K_{AS}}$ containing his/her identity together with a message encrypted with the key shared between A and the server S containing the identity B of the principal the initiator would like to talk to, a secret key K, and a time-stamp T_1. In the second step the server S forwards to principal B the message $\{A, K, T_2\}_{K_{BS}}$ containing A's identity, the secret key K, and an

createkey :
$clock(ts(n)) \mid fresh(sk(y)) \mid init_1(id(a))$
$\longrightarrow$
$clock(ts(n)) \mid fresh(sk(y')) \mid init_2(id(a), \langle id(b), id(s), sk(k), ts(n)\rangle)$
: $a > 0, b > 0, y' > k, k > y$

init :
$clock(ts(n)) \mid server(s) \mid init_2(id(a), \langle id(b), id(s), sk(k), ts(n)\rangle)$
$\longrightarrow$
$clock(ts(n)) \mid server(s) \mid init_3(id(a), \langle id(b), id(s), sk(k), ts(n)\rangle) \mid$
$\mid net(a, s, enc(sk(a, s), \langle r(0), id(b), sk(k), ts(n), ts(n)\rangle))$
: $a > 0, b > 0$

server :
$clock(ts(n)) \mid server(s) \mid delay(d_1, d_2) \mid$
$\mid net(a, s, enc(sk(a, s), \langle r(0), id(b), sk(k), ts(t), ts(st)\rangle))$
$\longrightarrow$
$clock(ts(n)) \mid server(s) \mid delay(d_1, d_2) \mid$
$\mid net(s, b, enc(sk(b, s), \langle r(1), id(a), sk(k), ts(n), ts(st)\rangle)$
: $a > 0, b > 0, n - d_1 \leq t, t \leq n$

resp :
$clock(ts(n)) \mid delay(d_1, d_2) \mid resp_1(id(b)) \mid server(id(s)) \mid$
$\mid net(s, b, enc(sk(b, s), \langle r(1), id(b), sk(k), ts(t), ts(st)\rangle))$
$\longrightarrow$
$clock(ts(n)) \mid delay(d_1, d_2) \mid server(id(s)) \mid resp_2(id(b), \langle id(a), sk(k), ts(n), ts(st)\rangle)$
: $a > 0, b > 0, n - d_2 \leq t, t \leq n$

Fig. 1. Wide-Mouth Frog: core protocol rules

updated time-stamp T_2. The message is encrypted with the key K_{BS} shared between B and the server. On receipt of a message the server and the responder have to check the validity of the time-stamps.

This protocol should ensure *timed-authentication*, i.e., the responder should authenticate a message coming from the initiator with a delay not greater than the (estimated a priori) network delay. In this protocol messages sent by and received from the server have the same structure. In [1] Anderson and Needham have discovered an interesting type of attacks due to this property (see also Section 5). Specifically, if Trudy repeatedly replays messages sent by the server, the server will maintain the time-stamp in the replayed message up-to-date. This way, Trudy can enforce an arbitrary delay between the time A starts the protocol and the time B receives the key, thus violating *timed authentication*. This attack can be prevented by distinguishing the two messages used in the protocol, e.g., by inserting an extra field 0/1.

2.3 Formal Specification of WMF

To specify all these aspects in a rigorous way, we specify the protocol in as in Fig. 1. In our model, at a given instant time t, the initiator a creates a new key

k (rule **createkey**) and (non-deterministically) selects a server s and a partner b. Then, a starts the protocol (rule **init**) by sending the message containing the tag $r(0)$, b, k, a time-stamp, and the message creation time (they coincide with the current time). The message is encrypted with the secret key shared between a and s represented as $sk(a, s)$. The rules **createkey** and **init** are executed within the same instant. This is achieved by storing the current time in the initiator knowledge in the former rule and by using it as a precondition in the latter. This example gives us an example of independence between state- and time-transitions in MSR($\mathcal{L}$): we can either specify several transitions within the same instant or introduce fixed/arbitrary delays within a given state transition. In rule *server* the server checks if the time-stamp is recent and then forwards the message (tagged with $r(1)$) to b. This test is modelled in a natural way by using the constraints $n - d_1 \leq t, t \leq n$, where d_1 represents a parametric delay; parameters are stored in the atomic formula $delay(d_1, d_2)$ as part of the initial configuration. In rule *resp b* performs a similar validity test for the time-stamp. Note that the *actual* creation time of the key is always copied from one message to the other. Every honest principal is allowed to play the role of both initiator and responder and to run several role instances in parallel.

In this protocol the responder should authenticate a message coming from the initiator with a delay of at most $d_1 + d_2$ time units (timed-authentication). Our goal is to show that timed-authentication holds for any $d_1, d_2 \geq 0$.

3 Verification via Symbolic Exploration

In order to define a symbolic representation of sets of configurations of multiple parallel protocol sessions we proceed in two steps. Following [6], we first introduce a special term constructor $\mathbf{sup}(t)$ that can be used to finitely represent all terms containing t as a subterm. Then, we incorporate the resulting *extended terms* in a symbolic representation of collections of protocol sessions.

3.1 Symbolic Data Structure

The set of extended terms over the variables $\mathcal{V}$ and the signature Σ is built by induction as follows: constants in Σ and variables in $\mathcal{V}$ are extended terms; if t is an extended term, $\mathbf{sup}(t)$ is an extended term; if $\boldsymbol{t}$ is a list of extended terms and f is in Σ then $f(\boldsymbol{t})$ is an extended term. Given a *ground* extended term t, its *denotation* is defined by induction as follows:

$$\begin{array}{l}[\![c]\!] = \{c\} \text{ if } c \text{ is a constant;}\\ [\![\mathbf{sup}(t)]\!] = \{s \mid s \text{ is a ground term, t occurs in } s\};\\ [\![f(t_1, \ldots, t_n)]\!] = \{f(s_1, \ldots, s_n) \mid s_1 \in [\![t_1]\!], \ldots, s_n \in [\![t_n]\!]\}.\end{array}$$

A *symbolic configuration* is defined then as a formula $p_1(\boldsymbol{t}_1) \mid \ldots \mid p_n(\boldsymbol{t}_n) : \varphi$, where $p_1, \ldots, p_n$ are predicate symbols, $\boldsymbol{t}_i$ is a tuple of *extended* terms for any $i : 1, \ldots n$, and the satisfiable constraint φ is such that $Fv(\varphi) \subseteq Fv(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n)$.

In the rest of the paper we will use $M, N, \ldots$ to indicate symbolic configurations. The *ground denotation* of $M \doteq A_1 \mid \ldots \mid A_n : \varphi$ is defined as follows:

$$[\![M]\!] = \{\ B_1 \mid \ldots \mid B_n \ s.t.\ \exists\ \sigma \text{ grounding for } A_1 \mid \ldots \mid A_n \\ B_i \in [\![A_i\sigma]\!],\ i : 1, \ldots, n, \text{ and } \sigma_{|Fv(\varphi)} \in Sol(\varphi)\ \}$$

This definition is extended to sets of symbolic configurations in the natural way. Finally, in order to reason about configurations consisting of an arbitrary number of parallel protocol sessions, we define the *upward closed denotation* of a set of symbolic configurations as follows

$$\langle\!\langle \mathbf{S} \rangle\!\rangle = \{\mathcal{N} \mid \text{there exists } \mathcal{M} \in [\![\mathbf{S}]\!] \text{ s.t. } \mathcal{M} \text{ is contained in } \mathcal{N}\}$$

This extended semantics is particularly useful for locally representing *violations* to properties like *secrecy*, where, independently from the number of sessions and principals, disclosed secrets are shared between the intruder and a *finite* number of agents. The use of extended terms gives us a further level of parameterization, e.g., we can locally require that a piece of data occurs at some depth inside a message. As an example the upward closed denotation of the singleton with

$$p(ts(x)) \mid r(ts(y), \mathbf{sup}(f(ts(z), \mathbf{sup}(ts(x))))) \ :\ x > y, y > z$$

contains all configurations $p(ts(v_1)) \mid r(ts(v_2), \mathbf{T_1}[f(ts(v_3), \mathbf{T_2}[ts(v_1)])]) \ \oplus\ Q$ for some $v_1 > v_2 > v_3$, two ground terms with a hole $\mathbf{T_1}[\cdot]$ and $\mathbf{T_2}[\cdot]$, and some configuration Q.

The previous semantics is well-suited for a backward analysis of a protocol model in which we do not impose restrictions on the number of parallel sessions, range of time-stamps and generated nonces.

3.2 Symbolic Operations

Contrary to ordinary unification, given two extended terms t and t' the extended unification algorithm of [6] computes the *set* of *maximal general unifiers* $max.g.u.(t, t')$. The interesting case in the extension of the unification algorithm occurs when a term like $\mathbf{sup}(t)$ must be unified with t': we have to search for a subterm of t' that can be unified with t. E.g. $x \mapsto \mathbf{sup}(a)$ is a max.g.u. for $g(\mathbf{sup}(a))$ and $\mathbf{sup}(g(x))$. We can avoid bindings between integer variables and terms by restricting the unification algorithm in such a way that it cannot descend the term structure of casting terms like $ts(\cdot)$. E.g., $\sigma = \{x \mapsto a\}$ is a max.g.u. for $\mathbf{sup}(a)$ and $ts(x)$ only if ts is not a casting symbol. If ts is a casting symbol, we are forced to unify a with $ts(x)$ (we cannot select other subterms), and, thus, unification fails. When combined with constraint satisfiability and simplification, the previous operations can be used to extend the unification algorithm to symbolic configurations (i.e. multisets of atomic formulas defined over extended terms annotated with constraints). Specifically, we call the pair $\langle\sigma, \gamma\rangle$ a *maximal constrained unifier* **m.c.u.** for $M : \varphi$ and $N : \psi$, whenever $\sigma \in max.g.u.(M, N)$, and, assuming that σ' is the maximal sub-constraint of σ involving only integer variables, $\gamma \equiv \varphi, \psi, \sigma'$ is satisfiable.

Let L be a global variable, initially $= \emptyset$
$subs_t(x,t,t)$ if x is a variable and $(x \mapsto t) \in L$;
$subs_t(x,t,t)$ if x is a variable and $\nexists s.\ (x \mapsto s) \in L; (x \mapsto t)$ is added to L
$subs_t(a,a,a)$ if a is a constant;
$subs_t(\mathbf{sup}(t), \mathbf{sup}(s), \mathbf{sup}(s'))$ if $subs_t(\mathbf{sup}(t), s, s')$;
$subs_t(\mathbf{sup}(t), s, \mathbf{sup}(s''))$ if $subs_t(t, s', s'')$ for some subterm s' *of* s;
$subs_t(f(t_1,\ldots,t_n), f(t'_1,\ldots,t'_n), f(s_1,\ldots,s_n))$ if $subs_t(t_1,t'_1,s_1),\ldots,subs_t(t_n,t'_n,s_n)$.

Fig. 2. Subsumption relation over symbolic configurations and extended terms.

Similarly to unification we can define a subsumption relation $subs_t$ over extended terms such that $subs_t(t,t')$ implies $[\![t']\!] \subseteq [\![t]\!]$. The algorithm is described in Fig. 2. Actually, we use a ternary relation $subs_t(t,t',s)$ that also computes a witness extended term s that is built by taking the extended subterms of t' that are needed to establish subsumption. When combined with constraint entailment, extended term subsumption allows us to define a comparison relation $\sqsubseteq$ between symbolic configurations such that $M \sqsubseteq N$ implies $\langle\!\langle N \rangle\!\rangle \subseteq \langle\!\langle M \rangle\!\rangle$. The procedure for testing $M \sqsubseteq N$ is defined as follows:

$$(\mathcal{M} : \varphi) \sqsubseteq (\mathcal{N} : \psi) \text{ iff } \begin{cases} subs_t(\mathcal{M}', \mathcal{N}, \mathcal{S}) \text{ for some } \mathcal{M}' = \mathcal{M} \oplus \mathcal{Q} \text{ for some } \mathcal{Q}, \\ \sigma = m.g.u.(\mathcal{M}', \mathcal{S}),\ \sigma' = \sigma_{|\mathcal{V}_i}, \\ \exists \boldsymbol{x}.\ (\psi, \sigma') \text{ entails } \varphi, \text{ where } \boldsymbol{x} = Fv(\psi, \sigma') \setminus Fv(\mathcal{S}\sigma) \end{cases}$$

Intuitively, we first select a submultiset $\mathcal{M}'$ of $\mathcal{M}$ that is subsumed by $\mathcal{N}$ with witness $\mathcal{S}$. Then, we unify $\mathcal{S}$ with $\mathcal{M}'$ (using ordinary term unification considering the symbol **sup** as any other symbol in Σ) in order to have formulas defined over the same set of variables (variables in $\mathcal{N}$ but not in $\mathcal{S}$ are projected away using existential quantification). Finally, we check entailment of the corresponding constraints.

Example 1. Consider the following symbolic configurations:

$$M_1 = p(ts(x)) \mid r(ts(y), \mathbf{sup}(f(ts(z), \mathbf{sup}(ts(x))))) : x > y, y > z$$
$$M_2 = p(ts(u)) \mid r(ts(v), f(ts(w), h(g(ts(u)))) \mid p(ts(m)) : u > v, v > w, w > m$$

Given $\sigma = \{u \mapsto x, v \mapsto y, w \mapsto z\}$ and $\gamma \equiv x > y, y > z, x = u, y = v, z = w$, $\langle \sigma, \gamma \rangle$ is an **m.c.u** for M_1 and the submultiset obtained by removing $p(ts(m))$ from M_2. Furthermore, by applying the definition, we have that $subs_t(M_1, M_2, S)$ holds with witness $S = p(ts(u)) \mid r(ts(v), \mathbf{sup}(f(ts(w), \mathbf{sup}(ts(u)))))$. The most general unifier (using ordinary first order unification) for S and M_1 is $\theta = \{x \mapsto u, y \mapsto v, z \mapsto w\}$. Thus, since $\exists u, v, w, m. u > v, v > w, w > m, x = u, y = v, z = w$ is equivalent to $x > y, y > z$, it follows that $M_1 \sqsubseteq M_2$.

3.3 Symbolic Predecessor Operator

Let **S** be a set of symbolic configurations with distinct variables each other. Symbolic backward exploration is based on the pre-image operator $\mathbf{Pre}_R$ defined over

a rule R, and to its natural extension $\mathbf{Pre}_{\mathcal{R}}$ to a set $\mathcal{R}$ of rules. These operators take into account the upward closed semantics of symbolic configurations by applying rewriting rules as follows: submultisets of symbolic configurations are matched against submultisets of right-hand side of rules (we reason modulo any possible *context*). Namely, given $R = \alpha : \ \mathcal{A} \longrightarrow \mathcal{B} : \psi$, the operator $\mathbf{Pre}_R$ is defined as follows

$$\mathbf{Pre}_R(\mathbf{S}) = \left\{ \mathcal{A}\sigma \oplus Q\sigma \ : \ \xi \ \middle| \begin{array}{l} \exists\ (M : \varphi)\ in\ \mathbf{S},\ M = M' \oplus Q,\ \mathcal{B} = \mathcal{B}' \oplus \mathcal{D} \\ s.t.\ \langle \sigma, \gamma \rangle\ is\ an\ \mathbf{m.c.u.}\ for\ M' : \varphi\ and\ \mathcal{B}' : \psi \\ \xi \equiv \exists \boldsymbol{x}.\gamma\ where\ \boldsymbol{x} = Fv(\gamma) \setminus Fv(\mathcal{A}\sigma \oplus Q\sigma) \end{array} \right\}$$

The following property holds.

Proposition 1. *Let $\mathcal{R}$ be a set of MSR($\mathcal{L}$) rules and $\mathbf{S}$ a set of symbolic configurations, then $\langle\!\langle \mathbf{Pre}_{\mathcal{R}}(\mathbf{S}) \rangle\!\rangle = Pre_{\mathcal{R}}(\langle\!\langle \mathbf{S} \rangle\!\rangle)$.*

3.4 Symbolic Backward Reachability Graph

Let $\mathbf{U}$ be a set of symbolic configurations. Our symbolic operations can be used to build a *symbolic backward reachability graph* $G = \langle \mathbf{N}, \mathbf{E} \rangle$ as follows. The set of nodes $\mathbf{N}$ is initially set to $\mathbf{U}$. At each step, for any $M \in \mathbf{N}$ and $Q \in \mathbf{Pre}_{\mathcal{R}}(\{M\})$ if there exists a *visited* symbolic configurations $O \in \mathbf{N}$ such that $O \sqsubseteq Q$, then we discharge Q and add an edge $O \rhd M$ to $\mathbf{E}$. If there are no *visited* symbolic configurations that subsumes Q (i.e. Q brings new information), then Q is added to the set of nodes $\mathbf{N}$, and an edge $Q \rhd M$ is added to the graph. If $\mathbf{Pre}_{\mathcal{R}}(\{M\})$ is empty, then the node M has no incoming edges. Also note that if $M \rhd N$, then there exists R such that either $M = \mathbf{Pre}_R(\{N\})$ or $M \sqsubseteq \mathbf{Pre}_R(\{N\})$.

4 Verification of Secrecy and Authentication

Violations of secrecy properties can often be generated by only looking at one honest principal and at part of the knowledge of Trudy, they are denoted by (according to the upward closed semantics) of symbolic configurations like

$$mem(\mathbf{secret}) \mid r(id(a), \langle \ldots, \mathbf{secret}, \ldots \rangle) \ : \ true$$

Under this assumption, the following property holds.

Theorem 1 (Secrecy). Let $\mathcal{S}$ be an MSR($\mathcal{L}$) protocol and attacker specification, and $\mathbf{U}$ be the set of symbolic configurations representing violations to a secrecy property. If the symbolic backward reachability terminates when invoked on $\mathcal{S}$ and $\mathbf{U}$, and the set $\mathbf{N}$ of nodes of the resulting *graph* does not contain initial configurations, then the protocol is not subject to secrecy attacks. The property holds for any number of nonces, principals, and parallel sessions.

Since we work on a *concrete protocol model* and a *symbolic semantics* without approximations (unless we apply widening operators during search), our procedure with an on-the-fly check for the initial configurations can be also used

for *falsification*, i.e, to extract finite-length symbolic trajectories that represent attacks.

Let us now consider authentication properties. In this section we will focus our attention on *(non-injective) agreement* [12]. A protocol guarantees *non-injective agreement*, e.g., for a responder B and for certain data d (e.g. keys/nonces), if each time the principal B completes a run of the protocol as a *responder* using data d, apparently with A, then there exists a run of the protocol with the principal A as an initiator with data d, apparently with B. *Agreement* requires a one-to-one correspondence between the runs of A and those of B. It can proved after proving non-injective agreement if freshness of part of the data in d ensures that there cannot be two different sessions with the same data.

Violations of agreement properties *cannot* be represented as upward closed sets of configurations. They are sets of *bad traces*. However, we can still exploit our symbolic search by using the following idea. We first define the two symbolic configurations

$$M_I \doteq r_i(id(a), \langle \ldots, id(b), \ldots \rangle) : \varphi \qquad M_R \doteq r'_f(id(b), \langle \ldots, id(a), \ldots \rangle) : \psi$$

M_I represents all configurations containing *at least* the initial state of the initiator a in a session, apparently with b, and M_R represents all configurations containing *at least* the final state of the responder b in a session, apparently with a. We select two generic values a and b for the identifiers of Alice and Bob, and generic nonces and keys for the data we want to observe (we will show how this can be done in practice in Section 5). Data may appear in M_I and M_R; φ and ψ can be used to define constraints over them. Our verification method is based then on the following property (formulated for the responder).

Theorem 2 (Non-injective agreement). Let $\mathcal{S}$ be an MSR($\mathcal{L}$) protocol and attacker specification, M_I and M_R be as described before. Suppose that the symbolic backward search terminates when invoked on $\mathbf{U} = \{M_R\}$ and $\mathcal{S}$. Then, if for every path $M_0 \rhd \ldots \rhd M_k \rhd M_R$ in the resulting graph going from an *initial (symbolic) configuration* M_0 to M_R, there exists i such that $M_I \sqsubseteq M_i$, then non-injective agreement property holds for any number of parallel sessions, principals, and nonces.

The previous property can be verified by checking that every *acyclic* path π going from an initial symbolic configuration to M_R *contains* at least a node N_π such that $M_I \sqsubseteq N_\pi$. Being formulated over a time sensitive specification logic, the previous properties can be used to verify time-dependent secrecy and authentication properties. In other words, we reason about time *inside* our specification logic. We will illustrate this point in Section 5.

Symbolic configurations can be used in other interesting ways. As an example, we might be interested in checking whether a key k stored in the state r_i of a principal will or will not be used to encrypt messages. This property can be used to formally justify the use of specialized theories for the intruder. We can then search for traces leading to $net(a, b, \mathbf{sup}(enc(sk(k), msg))) \mid r_j(x, \mathbf{sup}(sk(k))) \ : \ true$

Prop	Model	Description	Seed	Steps	Time	True
P_1	CoreWMF	Generated keys are never used	$\mathbf{U}_1$	6	260s	Yes
P_2	CoreWMF	Nested encryption is never used	$\mathbf{U}_2$	1	0.1s	Yes
P_3	WMF+Trudy	Non-injective agreement	$\mathbf{U}_3$	6	24s	Yes
P_4	WMF+Trudy	Non-injective timed agreement	$\mathbf{U}_4$	7	41s	Yes
P_5	WMF+Trudy	Uniqueness of gen. keys (init.)	$\mathbf{U}_5$	2	0.4s	Yes
P_6	WMF+Trudy	Uniqueness of gen. keys (resp.)	$\mathbf{U}_6$	11	3093s	No
P_7	OrigWMF+Trudy	Non-injective timed agreement	$\mathbf{U}_4$	11	88s	No

Fig. 3. List of properties, and results of the analysis: Seeds are described in Fig. 4.

for some $j \geq i$. Furthermore, following [6], when dealing with attacker theories in which we allow arbitrary use of rules like *decryption with Trudy's key* it is often possible to apply dynamic widening operators that approximate sets of terms like $enc(sk(0), \ldots, enc(sk(0), msg)\ldots)$ with $enc(sk(0), \mathbf{sup}(msg))$. This kind of accelerations may return overapproximations.

5 Detailed Analysis of Wide Mouth Frog

In Fig. 3 we show the list of properties of WMF we have checked using our method. The *seed* $\mathbf{U}_i$ of symbolic exploration id described in Fig. 4, **Steps** denotes the number of iterations before reaching a fixpoint/error trace, **Time** denotes the execution time. The initial configurations of the WMF model of Fig. 1 are those containing (any number of) initial states of honest principals, servers, and auxiliary information like $fresh$ and $delay$. The first two properties are related to the structure of the core protocol rules (without attacker): generated keys are never used to encrypt messages, and nested encryption is never used. We proved these two properties automatically by checking that the set of states described by the violations $\mathbf{U}_1$ and $\mathbf{U}_2$ are never reached via the core protocol rules. This preliminary step allows us to consider the simplified attacker theory defined in Fig. 5: we do not need rules to handle nested encryption, we assume that Trudy only uses the key she shared with the server, and that Trudy never learns new keys. This is the theory typically used for the analysis of WMF [9, 13]. In our protocol the data exchanged by the honest principals are *typed* by casting symbols. Thus, the intruder has only to deal with messages of the shape $enc(sk(a,b), \langle m_1, \ldots, m_n \rangle)$ where m_i is built using one of sk, id, ts, r. Among the typical attacker's capabilities like interception and duplication, we add the capability of *guessing* any time-stamp. This can be refined depending on the assumption taken on the intruder (e.g. guess only in the past, etc.).

Our next task is to prove *timed non-injective agreement*, i.e., non injective agreement within the maximal allowed delay of $d_1 + d_2$ time units for any value of the parameters d_1 and d_2. To prove this property, we first check *non-injective agreement* forgetting about time (property P_3), and then prove that there cannot be violations of the time-conditions (property P_4). We prove automatically P_3 starting symbolic search from the seed $\mathbf{U}_3$ of Fig. 3. The atomic formula

Seed	Symbolic configurations
$\mathbf{U}_1$	$init_2(x_1, \langle x_2, sk(k), x_3, x_4\rangle) \mid net(x_5, x_6, \mathbf{sup}(enc(sk(k, x_7), x_8))) : true,$ $init_2(x_1, \langle x_2, sk(k), x_3, x_4\rangle) \mid net(x_5, x_6, \mathbf{sup}(enc(sk(x_7, k), x_8))) : true$
$\mathbf{U}_2$	$net(x_1, x_2, \mathbf{sup}(enc(x_3, \mathbf{sup}(enc(x_4, x_5))))) : true$
$\mathbf{U}_3$	$par(id(a), id(b), sk(k)) \mid resp_2(id(b), \langle id(a), sk(k), ts(e), ts(s)\rangle) \mid$ $\mid delay(ts(d_1), ts(d_2)) \ : \ e, s, d_1, d_2 \geq 0$
$\mathbf{U}_4$	$par(id(a), id(b), sk(k)) \mid resp_2(id(b), \langle id(a), sk(k), ts(e), ts(s)\rangle) \mid$ $\mid delay(ts(d_1), ts(d_2)) \ : \ e > s + d_1 + d_2$
$\mathbf{U}_5$	$init_2(x, \mathbf{sup}(sk(k))) \mid init_2(y, \mathbf{sup}(sk(k))) \ : \ true$
$\mathbf{U}_6$	$resp_2(id(a), \mathbf{sup}(sk(k))) \mid resp_2(id(b), \mathbf{sup}(sk(k))) \ : \ true$

Fig. 4. Violations/seed of symbolic exploration.

$par(id(a), id(b), sk(k))$ is used here to make the analysis parametric on principals id's (a, b, k are all variables). In the resulting fixpoint we detect only a single initial configuration. Furthermore, all paths going to the seed configuration pass through a symbolic configuration containing

$$init_2(id(a), \langle id(b), id(s), sk(k), ts(t)\rangle) \mid par(id(a), id(b), sk(k))$$

that represents the state of the *initiator* right after the creation of the key. The binding of variables a, b, and k with the parameters $par(id(a), id(b), sk(k))$ maintains the proof independent from specific identifiers and keys.

In order to check property P_4, we start our search from the violations specified by means of $\mathbf{U}_4$ (the responder terminates its part of the protocol outside the window $s + d_1 + d_2$). In the resulting fixpoint there are no initial (symbolic) configurations. Thus, there exist no path reaching the previously described violations. This proves non-injective timed-agreement for any number of parallel protocol sessions and principals, and values of d_1 and d_2.

To check agreement, we still have to check that there is a one-to-one correspondence between runs of responder and initiator. We first check if different initiators can share the same secret key. When invoked on the seed $\mathbf{U}_5$ of Fig. 3, the procedure stops without detecting error traces. Then, we try to check whether the responder can receive multiple copies of the same key by selecting the violations $\mathbf{U}_6$ of Fig. 3. Our procedure returns the following counterexample trace: Alice creates a key, starts the protocol, the server forwards the message, Trudy intercepts it, Bob receives the message, Trudy forges a copy, Bob accepts the second copy. Thus, the protocol still suffers from possible replay attacks. As suggested by Lowe in [12], this flaw can be corrected by adding an handshaking part at the end the protocol in which the principals exchange a nonce encrypted with the shared key.

Finally, in order to compare the use of our method for *falsification* with [9,13], we have considered a model (called OrigWMF in Fig 3) in which we have removed the tags distinguishing the two messages, thus going back to the original WMF

$$
\begin{array}{lll}
\textbf{remove}: & net(x,y,m) \longrightarrow \epsilon & : true \\
\textbf{intercept}: & net(x,y,m) \longrightarrow mem(m) & : true \\
\textbf{fresh}: & fresh(sk(x)) \longrightarrow fresh(sk(y)) \mid mem(sk(n)) & : y > n, n > x \\
\textbf{guess}: & \epsilon \longrightarrow mem(ts(t)) & : t \geq 0 \\
\textbf{replay}: & net(x,y,m) \longrightarrow net(y,x,m) & : true \\
\textbf{dup}: & mem(m) \longrightarrow mem(m) \mid mem(m) & : true \\
\textbf{forge}: & mem(m) \longrightarrow net(id(a), id(b), m) & : a \geq 0, b \geq 0 \\
\textbf{dec}: & mem(enc(sk(0,s), \langle f_1(m_1), \ldots, f_n(m_n)\rangle)) \longrightarrow & \\
 & \quad mem(f_1(m_1)) \mid \ldots \mid mem(f_n(m_n)) & : s \geq 0 \\
\textbf{comp}: & mem(f_1(m_1)) \mid \ldots \mid mem(f_n(m_n)) \longrightarrow & \\
 & \quad mem(enc(sk(0,s), \langle f_1(m_1), \ldots, f_n(m_n)\rangle)) & : s \geq 0 \\
\multicolumn{3}{l}{\text{where } f_i \in \{sk/1, id/1, ts/1, r/1\} \; for \; i : 1, \ldots, n, \; n \leq 5}
\end{array}
$$

Fig. 5. Restricted theory of the intruder.

protocol. In this model we impose that server, initiator, and responder have different identifiers by adding constraints like $s > a, a > b$ in the core protocol rules, and we set $d_1 = d_2$. Starting from the violation $\mathbf{U}_4$, we have automatically computed the following error trace: Alice sends M to the Server, the Server forwards M to Bob, time advances, Trudy impersonates Bob and replays M, the Server forwards M to Alice, time advances, Trudy impersonate Alice and replays M, the Server forwards M to Bob, time advances, Bob receives M. This trace falls in the class of attacks discovered by [1].

6 Related Work

The use of *constraint programming* combined with *symbolic exploration* over unbounded models distinguishes our approach from other methods based on theorem proving or model checking used for the analysis of protocols with timestamps like [2,8,9,13]. In [9] Evans and Schneider perform a *semi-automated* analysis of a CSP model (with event-based time) of WMF. PVS is used to discharge proof obligations needed to find an invariant property (a rank function according to Schneider's verification method). During this process the attack with parametric validity period arises in form of a contradiction to the proof of an obligation. In [13] Lowe applies finite-state model checking to find the timed-authentication attack on finite protocol instances (an initiator, a responder, and a server), discrete time, fixed values for the validity period (namely $d_1 = d_2 = 1$) and with an upper bound on the time window observed during the state exploration. Differently from [9,13], our method can be used for both *automatic* verification and falsification in presence of parallel multiple sessions, without need of upper bounds on the time window considered during execution, and with parametric delays.

Time-stamps can be handled by the TAPS verifier [8]. The TAPS verifier generates state-dependent invariants from a protocol model. The first order theorem prover SPASS is used then to discharge the proof obligations produced by a proof system used to simplify the invariants according to the capability of the intruder. While TAPS is very fast and effective, it does not always produce readable counterexamples, an advantage of symbolic exploration methods like ours.

The uniform treatment of time-stamps and fresh name generation via constraint programming also distinguishes our method from other symbolic methods like NRLPA [16], Athena [19], and the methods of Blanchet [4], Bozga, Lakhnech and Périn [6], and Genet and Klay [10]. Furthermore, our use of the **sup** operator differs from [6]. In [6] **sup** is used to build a widening operator for computing secrecy invariants for *abstract* protocol models. On the contrary, our use of **sup** is within a symbolic representation of infinite collection of configurations of *concrete* protocol specifications. When terminating, our procedure returns accurate results that are useful for obtaining error traces. Widening operators similar to that of [6] can be included in our setting and applied to compute conservative approximations.

Our symbolic exploration method is complementary to approached based on model checking where abstractions or data independence techniques are used to extract a finite model from the original specification [13,15,18].

Finally, differently from constraint-based approaches used for*untimed* falsification of bounded number of protocol sessions [17], where *ad hoc* constraints relate a message expected by a principal to the current intruder knowledge, we use constraints for designing a symbolic *verification* method.

7 Conclusions

In this paper we have studied the applicability of symbolic exploration for the automatic verification of time sensitive cryptographic protocols. Our verification method is based on the combination of *constraint solving technology* and *first order term manipulation*. This combination provides a uniform framework for the specification of an arbitrary number of protocol sessions with fresh name generation and time-stamps.

In this paper we have described the main ideas behind this technology and discussed the analysis of the Wide Mouth Frog protocol. We have performed several other experiments on timed and untimed models (e.g. secrecy and agreement properties for Lowe's fix to the Needham-Schroeder public key protocol). The prototype procedure and the experimental results (with execution times) we obtained so far are available on the web page [20].

The verification approach presented in this paper can be coupled with a *symbolic forward exploration* method for a fixed number of sessions suitable to discover potential attacks (see [20]). Forward exploration often finds attacks faster than backward exploration. Currently, we are studying an extension of forward exploration with the lazy intruder technique of [17] to cope with MSR($\mathcal{L}$) specification and unrestricted attacker models.

References

1. R. Anderson, and R. Needham. Programming Satan's computer. Computer Science Today: 426–440, LNCS 1000, 1995.
2. G. Bella and L. C. Paulson. Kerberos version IV: inductive analysis of the secrecy goals. ESORICS '98: 361–375, 1998.
3. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Trans. on Computer Systems*, 8(1):18–36, 1990.
4. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. CSFW '01: 82–96, 2001.
5. M. Bozzano and G. Delzanno. Beyond Parameterized Verification. TACAS '02, LNCS 2280: 221–235, 2002.
6. L. Bozga, Y. Lakhnech, and M. Périn. Pattern-based Abstraction for Verifying Secrecy in Protocols. TACAS '03, LNCS 2619: 299–314, 2003.
7. I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for Protocol Analysis. CSFW '99: 55–69, 1999.
8. E. Cohen. TAPS: A First-Order Verifier for Cryptographic Protocols. CSFW '00: 144–158, 2000.
9. N. Evans and S. Schneider. Analysing Time Dependent Security Properties in CSP Using PVS. ESORICS '00, LNCS 1895: 222–237, 2000.
10. T. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. CADE '00, LNCS 1831: 271–290, 2000.
11. G. Lowe. A Family of Attacks upon Authentication Protocols. Technical Report 1997/5, University of Leicester, 1997.
12. G. Lowe. A Hierarchy of Authentication Specifications. CSFW '97: 31-44, 1997.
13. G. Lowe. Casper: A compiler for the analysis of security protocols. CSFW '97: 18–30, 1997.
14. G. Lowe. Towards a completeness result for model checking of security protocols. *J. of Computer Security*, 7(2-3):89–146, 1998.
15. W. Marrero, E. Clarke, and S. Jha. Verifying Security Protocols with Brutus. ACM Trans. Softw. Eng. Methodol. 9(4): 443-487, 2000.
16. C. Meadows. The NRL protocol analyzer: An overview. *J. of Logic Programming*, 26(2):113–131, 1996.
17. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. CCS '01: 166–175, 2001.
18. A. W. Roscoe and P. J. Broadfoot. Proving Security Protocols with Model Checkers by Data Independence Techniques. *J. of Computer Security*, 7(2,3):147-190, 1999.
19. D. X. Song. Athena. A New Efficient Automatic Checker for Security Protocol Analysis. CSFW '99: 192–202, 1999.
20. The MSR($\mathcal{C}$) home page: http://www.disi.unige.it/person/DelzannoG/MSR/