AVISPA

www.avispa-project.org

IST-2001-39252

Automated Validation of Internet Security Protocols and Applications

Deliverable 5.3: Completeness Issue

Abstract

Our backends either consider a finite number of protocol sessions or perform abstractions. In the former case, we cannot be sure that there are no attacks at all; in the latter case some false attacks might be reported. In this deliverable we have investigated two classes of protocols for which we can decide the existence of secrecy flaws. For the first class we have encoded the protocol analysis problem as a logical satisfiability problem that we have decided by a resolution strategy. For the second class we have applied tree automata techniques.

Deliverable details

Deliverable version: *v1.0*
Person-months required: *7*
Date of delivery: *31.05.2005*
Due on: *31.03.2005*
Classification: *public*
Total pages: *20*

Project details

Start date: *January 1st, 2003*
Duration: *30 months*
Project Coordinator: *Alessandro Armando*
Partners: *Università di Genova, INRIA Lorraine, ETH Zürich, Siemens AG*

Information Society Technologies

Project funded by the European Community under the *Information Society Technologies* Programme (1998-2002)

Contents

1 Introduction

The secrecy problem of cryptographic protocols is, in general, undecidable. Thus, known methods of verifying protocols either consider a finite number of protocol sessions or perform some abstractions. In the former case, we cannot be sure that there are no attacks at all, while in the latter case, some false attacks might be reported. In this deliverable, we have investigated two classes of protocols for which we can decide the existence of secrecy flaws.

First, in [7], we propose a resolution strategy for deciding a fragment of first-order logic that allows one to incorporate the prefix property of CBC encryption in our protocol modeling and to decide the existence of attacks exploiting this property. The same fragment applies to abstract properties of blind signature schemes, which are the main alternative to homomorphic encryption in E-voting [11]. The approach follows the line of [4], but requires a refined strategy in order to eliminate the clauses generated by resolution with the prefix properties. The verification algorithm has been applied to Needham-Schroeder Symmetric Key protocol, which is subject to an attack when implemented using CBC encryption. We show how to fix the protocol and we show the correction of the resulting protocol.

Second, in [17], we define a class of cryptographic protocols, called *regular protocols*, such that the knowledge which the intruder can gain during an unbounded number of sessions of a protocol is a regular language. We prove that the secrecy of regular protocols is DEXPTIME-complete. Moreover, if keys are constants, regular protocols can use blind signatures and can be verified taking into account the prefix property of CBC encryption.

Protocols and the Intruder. To describe rewrite-send actions of a principal, we use rewrite rules of the form $t \to s$, where t and s are terms. The meaning of such a rule is that a principal, after receiving any ground instance $t\theta$ of t (for any ground substitution θ), replies with $s\theta$. Note that this formalism does not model nonces. A principal is defined as a sequence of rules.

We will express the ability of the intruder of deriving messages by a set $\mathcal{I}$ of Horn clauses. Moreover, since we are interested in describing the knowledge of the intruder in the case of unbounded number of sessions, each protocol can be presented as a set T of Horn clauses such that the set of messages the intruder can gain is exactly the least Herbrand model of $T \cup \mathcal{I}$. We obtain T by transforming each principal defined by a sequence $(r_i \to s_i)_{i=1}^n$ of rules to the clauses $I(r_1), \ldots, I(r_i) \to I(s_i)$, for each $i = 1, \ldots, n$.

Related work. Decision procedures for various classes of cryptographic protocols were studied by Dolev, Even and Karp [8], Comon and Cortier [4], Amadio and Charatonik [1].

In [2], Blanchet and Podelski showed decidability results for *tagged* protocols without nonces. In [15], Ramanujam and Suresh present a technique of tagging protocols which enforces decidability even for protocols with unbounded number of nonces.

2 Resolution Strategy

In this section we propose a resolution strategy for deciding a fragment of first-order logic that allows one to incorporate the prefix property of CBC encryption in our protocol modeling and to decide the existence of attacks exploiting this property. The same fragment applies to abstract properties of blind signature schemes, which are the main alternative to homomorphic encryption in E-voting [11]. The approach follows the line of [4] but requires a refined strategy in order to eliminate the clauses generated by resolution with the prefix properties. The verification algorithm has been applied to Needham-Schroeder Symmetric Key protocol, which is subject to an attack when implemented using CBC encryption. We show how to fix the protocol, and we show the correctness of the resulting protocol.

In Section 2.1, we explain how protocols can be modeled using Horn clauses, introducing a new fragment of first order clauses. In Section 2.2, we present our resolution strategy and apply it to this fragment, proving that this strategy is both complete and terminating for this class. This leads to our main contribution: the decidability of satisfiability for this fragment. As a consequence, we obtain that secrecy of cryptographic protocols is decidable for an unbounded number of sessions, in the case for example of CBC encryption and blind signatures, when nonces are abstracted by constant terms and at most one copy of some unknown data (represented by a variable) is sent at each transition. We apply this result to the Needham-Schroeder symmetric key authentication protocol: this protocol is flawed when the CBC mode is used [14]. We show (in Section 2.3) how to fix the protocol and we formally prove, using our technique, that the fixed version preserves secrecy. Concluding remarks can be found in Section **??**.

2.1 Modeling protocols

The aims of this section is to introduce some notations and to introduce the class of clauses we consider. We also explain how protocols can be modeled using these clauses.

2.1.1 Notations

Let $\mathcal{F}$ be a finite set of function symbols and $\mathcal{V}$ a set of variables. The set of terms on $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F} \cup \mathcal{V})$. If u is a term, $V(u)$ denotes the set of variables occurring in u. A term is *ground* if it has no variables. If u and v are two terms of $\mathcal{T}(\mathcal{F} \cup \mathcal{V})$, the term $u[v/x]$ is the term u where each occurrence of x has been replaced by v. If x is the only variable of u, then we may write $u(x)$ instead of u, in order to emphasize this property. And also, in this case, we simply write $u[v]$ instead of $u[v/x]$. Let I be a unary predicate. *Atoms* A are of the form $I(u)$ where u is a term. *Literals* L are either positive literals $+A$ (or simply A) or negative literals $-A$ where A is an atom. A *clause* is a finite set of literals. If C_1 and C_2 are clauses, $C_1 \vee C_2$ denotes $C_1 \cup C_2$. A *Horn clause* is a clause that contains at most one positive literal. For Horn clauses we may use the alternative notation $A_1, A_2, \ldots, A_{n-1} \rightarrow A_n$ to denote the clause $-A_1 \vee -A_2 \vee \ldots \vee -A_{n-1} \vee A_n$. A *substitution* is a function $\sigma : \mathcal{V} \rightarrow \mathcal{T}(\mathcal{F} \cup \mathcal{V})$. A *ground substitution* maps every variable to a ground term. If M is a term, literal, clause, substitution or, set of such objects, then $M\sigma$ obtained by applying σ to M is defined as usual. If M and N are terms or literals then a *unifier* of M and N is a substitution σ such that $M\sigma = N\sigma$. If such a unifier exists then there exists a *most general unifier (mgu)*, denoted by $mgu(M, N)$.

If $u \in \mathcal{T}(\mathcal{F} \cup \mathcal{V})$, $|u|$ is the *depth* of u (maximal size of its positions). For $x \in \mathcal{V}$, $|u|_x$ is the maximal depth of an occurrence of x in u. By convention, if $x \in \mathcal{V}$ then $|u|_x = 0$ if $x \notin V(u)$. The definitions of $|\cdot|$ and $|\cdot|_x$ are extended to literals by $|\pm I(u)| = |u|$ and $|\pm I(u)|_x = |u|_x$.

We consider a strict and total order $<_{\mathcal{F}}$ on the function symbols. If u is a term which is not a variable, h_u denotes the head symbol of u. We fix an order on terms. The order is chosen in order to ensure the termination of our resolution procedure.

Definition 1 (order $<$). *Let u and v be two terms. We say that $u < v$ if one of the two following conditions holds:*

- $|u| < |v|$ *and* $|u|_x < |v|_x$*, for every* $x \in V(u) \cup V(v)$*;*
- $|u| \leq |v|$, $|u|_x \leq |v|_x$ *for every* $x \in V(u) \cup V(v)$ *and,* $h_u <_{\mathcal{F}} h_v$.

For example, if u is a strict subterm of v then $u < v$. Variables are incomparable. We have $\langle a, x\rangle < \mathrm{h}(\mathrm{h}(x))$ but $\langle \mathrm{h}(\mathrm{h}(a)), x\rangle \not< \mathrm{h}(\mathrm{h}(x))$, where as usual $\langle \_, \_\rangle$ is a function symbol denoting the pairing operation. Later, we may also omit the $\langle\rangle$. By convention, we assume that terms are parsed from left to right, *i.e.* the term $\langle u_1, u_2, \ldots, u_k\rangle$ means the term $\langle u_1, \langle u_2, \langle \ldots, u_k\rangle \ldots\rangle$.

An order is said to be *liftable* if, for any two terms u, v and for any substitution θ, $u < v$ implies $u\theta < v\theta$. This is a crucial property for the completeness of ordered resolution.

Proposition 1. *The relation $<$ is a strict liftable ordering.*

A term v is said to be *maximal* in a set S if there is no term $u \in S$ such that $v < u$.

All the definitions are extended on literals by $\pm I(u) < \pm' I(v)$ if and only if $u < v$.

We assume for the rest of this deliverable that $\{\_\}_\_$, h(_), sign(_, _) and blind(_, _) are function symbols.

2.1.2 Intruder clauses: the class $\mathcal{C}_I$

The intruder analyzes the messages sent on the network. For example, if he sees an encrypted message and if he knows the encryption key, then he can decrypt the message. This can be easily modeled using a very simple Horn clause: $I(\{x\}_y), I(y) \to I(x)$. The predicate I represents the knowledge of the intruder: $I(m)$ means that the intruder knows the term (or message) m. Thus this clause should be read as "if the intruder knows some message of the form $\{x\}_y$ and if he knows y, then he knows x". Other examples of clauses modeling the intruder power can be found in Figure 1. The set of all the clauses of Figure 1 is denoted by $\mathcal{I}$. Note that we may also consider public encryption by adding for each identity the clauses for encryption and decryption.

Each of these clauses contains at most one function symbol. That is why we consider the following class of clauses.

Definition 2 (class $\mathcal{C}_I$). *Let $\mathcal{C}_I$ be the class of Horn clauses of the form*

$$\pm I(f(x_1, \ldots, x_n)) \vee \bigvee_{j=1}^{m} \pm I(x_{i_j}).$$

where $\{i_1, \ldots, i_m\} \subset \{1, \ldots, n\}$.

The set of clauses $\mathcal{I}$ (defined in Figure 1) corresponding to the intruder capabilities is clearly in the class $\mathcal{C}_I$.

$I(x), I(y) \rightarrow I(\langle x, y\rangle)$	pairing of messages
$I(\langle x, y\rangle) \rightarrow I(x)$	first projection
$I(\langle x, y\rangle) \rightarrow I(y)$	second projection
$I(x), I(y) \rightarrow I(\{x\}_y)$	symmetrical encryption
$I(\{x\}_y), I(y) \rightarrow I(x)$	symmetrical decryption
$I(x) \rightarrow I(\mathrm{h}(x))$	hashing
$I(x), I(y) \rightarrow I(\mathrm{sign}(x, y))$	signing
$I(\mathrm{sign}(x, y)), I(y) \rightarrow I(x)$	verifying the signature
$I(x), I(y) \rightarrow I(\mathrm{blind}(x, y))$	blinding
$I(\mathrm{blind}(x, y)), I(y) \rightarrow I(x)$	undo blinding

Figure 1: Intruder rules: the set $\mathcal{I}$

2.1.3 Protocol clauses: the class $\mathcal{C}_P$

We now show how the rules of a protocol can also be modeled using Horn clauses. We consider a variant of the Needham-Schroeder symmetric key authentication protocol [12]. The original protocol will be analyzed in Section 2.3.

The description of the protocol is as follows:

$$\begin{array}{rl} A \Rightarrow S: & A, B, N_a \\ S \Rightarrow A: & \{N_a, B, K_{ab}\}_{K_{as}} \\ S \Rightarrow B: & \{K_{ab}, A\}_{K_{bs}} \\ B \Rightarrow A: & \{N_b\}_{K_{ab}} \end{array}$$

The notations A, B, and S represent, respectively, the roles Alice, Bob and the server. The intruder is a special role that can impersonate other roles and we will denote by $I(A)$, for example, a spoofed instance of A. The message field N_a is a *nonce*, meaning a random number freshly generated by A just before it is sent in the first message. In this first message Alice tells the server that she wants to communicate with Bob, and puts a nonce in her message. The server replies with a message containing the same nonce, Bob's name and a fresh session key K_{ab}. The bracketed term $\{N_a, B, K_{ab}\}_{K_{as}}$ represents the encryption of the concatenation of N_a, B and K_{ab} using key K_{as} shared by Alice and the server. In the third message, the server forwards to Bob the session key and his name, all encrypted by their common key K_{bs}. Finally, Bob can challenge Alice by sending her a nonce N_b encrypted by the session key. The protocol can be extended with the response to this challenge.

The translation in Horn clauses may be found in Figure 2. The intruder controls all the network communications. He can either build and send new messages or forward messages from other agents. Thus we may assume that any message is sent trough the intruder. We explain here the translation of the second rule: each time the server S receives a message of the form $\langle a, b, x\rangle$ (sent by the intruder) where x can be any term, he answers by the message $\{x, b, \mathrm{k}(a, b)\}_{\mathrm{k}(a,s)}$ (intercepted by the intruder). This is expressed by the rule $I(\langle a, b, x\rangle) \to I(\{x, b, \mathrm{k}(a, b)\}_{\mathrm{k}(a,s)})$. Since every protocol session generates new nonces, for modeling the protocol we have to perform some abstraction by letting the nonces only depend on the agent that has created it and the agent that should receive it. Similarly, a fresh session key will be parameterized by the agents who share the key. Note that this abstraction is correct w.r.t. secrecy properties, *i.e.* if a protocol is deemed secure using these abstractions, then it is secure without them.

$$\begin{aligned}
&\to I(\langle a, b, n_1(a, b)\rangle)\\
I(\langle a, b, x\rangle) &\to I(\{x, b, \mathrm{k}(a, b)\}_{\mathrm{k}(a,s)})\\
&\to I(\{\mathrm{k}(a, b), a\}_{\mathrm{k}(b,s)})\\
I(\{y, a\}_{\mathrm{k}(b,s)}) &\to I(\{n_2(b, a)\}_y)
\end{aligned}$$

Figure 2: Clauses for a variant of the Needham-Schroeder protocol: the set $\mathcal{P}_{\mathsf{ex}}$

We note that each of the clauses has at most one variable. As observed in [4], this is the case of protocols *with single blind copying*, *i.e.* protocols for which, at each step of the protocol, at most one part of the message is blindly copied. For example, in the second rule of the Needham-Schroeder protocol, the only blindly copied part is N_a since the other parts (A and B) are names known to the server. Therefore, the second class of clauses we consider is the class $\mathcal{C}_P$ of Horn clauses that contain at most one variable. We have $\mathcal{P}_{\mathsf{ex}} \in \mathcal{C}_P$.

Then to check whether the secrecy of a message m is preserved, we add a clause $-I(m)$ to the set of clauses modeling the protocols, the intruder activities and the intruder knowledge (if she knows t we add $I(t)$). The satisfiability of the resulting set of clauses will prove the secrecy property.

Comon and Cortier [4] have shown that satisfiability of a set of clauses of $\mathcal{C}_I \cup \mathcal{C}_P$ is decidable in 3-EXPTIME and Seidl and Verma [16] have shown that satisfiability is in fact DEXPTIME-complete. Since secrecy properties can also be modeled using Horn clauses (of the class $\mathcal{C}_P$), this means that the secrecy preservation of protocols (with at most one blind copy and no nonces) is decidable in 3-EXPTIME.

2.1.4 Extending the intruder power

The paper [7] has extended the decidability result of [4] to a larger class of clauses, in order to model an extended power of the intruder. Indeed, the set of clauses $\mathcal{I}$, described in Figure 1, represents the capabilities of an intruder, assuming *perfect cryptography*. In particular, the intruder cannot learn anything from an encrypted message $\{m\}_k$, unless he has the inverse key. However, depending on the implementation of the cryptographic primitives, the intruder may be able to deduce more messages. We consider here CBC encryption and blind signatures.

Prefix property

Depending on the encryption scheme, an intruder may be able to get, from an encrypted message, the encryption of any of its prefixes: from a message $\{x, y\}_z$, he can deduce the message $\{x\}_z$. This is encoded by the clause:

$$C_{pre} \stackrel{\mathrm{def}}{=} -I(\{\langle x, y\rangle\}_z) \vee I(\{x\}_z)$$

This is for example the case for Cipher Block Chaining (CBC) encryption. In such a system, the encryption of the message block sequence $P_1P_2\cdots P_n$ (where some bits may be added to P_n such that every block has the same length) with the key K is $C_0C_1C_2\cdots C_n$ where $C_0 = I$ (initialization block) and $C_i = \{C_{i-1} \oplus P_i\}_K$. The CBC encryption system has the following property: if $C_0C_1C_2\cdots C_iC_{i+1}\cdots C_n = \{P_1P_2\cdots P_iP_{i+1}\cdots P_n\}_K$ then $C_0C_1\cdots C_i = \{P_1P_2\cdots P_i\}_K$, that is to say an intruder can get $\{x\}_z$ from $\{x, y\}_z$ if the length of x is a multiple of the block length used by the cryptographic algorithm. This property can be used to mount attacks on several well-known protocols. For example, we explain in Section 2.3.1 the attack discovered by O. Pereira and J.-J. Quisquater [14] on the Needham-Schroeder symmetric key authentication protocol [12].

This property also holds for homomorphic encryption, *i.e.* encryption schemes that verify that $\{\langle x, y\rangle\}_k = \langle\{x\}_k, \{y\}_k\rangle$. This is the case of the ECB (Electronic Code Book) encryption scheme for example, where the encryption of message block sequence $P_1P_2\cdots P_n$ with the key K is simply the sequence $\{P_1\}_K\{P_2\}_K\cdots\{P_n\}_K$. For such encryption schemes, the clause C_{pre} models only partially the intruder power. Indeed, the intruder is able to recombine messages, which is not modeled by the clause.

Blind signatures

Blind signatures are used in voting protocols like the FOO 92 voting protocol [10, 11]. The idea of the protocol is that the voter first commits its vote v using a blinding function blind and a random blinding factor r: he sends the message $\mathrm{blind}(v, r)$ together with a signature of the message.

The administrator A verifies that the voter has the right to vote and has not voted yet. If it is the case, he signs the message, *i.e.* sends the message $\mathrm{sign}(\mathrm{blind}(v, r), \mathrm{ska})$. Note that the administrator does not have access to the vote since it is blinded. Now, the voter can unblind the message, getting $\mathrm{sign}(v, \mathrm{ska})$, using that $\mathrm{unblind}(\mathrm{sign}(\mathrm{blind}(v, r), \mathrm{ska}), r) = \mathrm{sign}(v, \mathrm{ska})$. Then the voter can send its vote to the collector. The "commutativity" property between blinding and signing can be modeled by the clause:

$$C_{sig} \overset{\mathrm{def}}{=} -I(\mathrm{sign}(\mathrm{blind}(x, y), z)) \vee -I(y) \vee I(\mathrm{sign}(x, z)).$$

Definition of the class $\mathcal{C}_S$

First let us note that the clauses C_{pre} and C_{sig} are neither in the class $\mathcal{C}_I$ nor in the class $\mathcal{C}_P$. Therefore they cannot be treated by [4, 16] techniques.

In order to extend the intruder power to clauses such as C_{pre} or C_{sig}, we consider the class of *special clauses*, denoted by $\mathcal{C}_S$.

We assume that the set of function symbols $\mathcal{F}$ contains a special symbol f_0 and that this symbol is the smallest symbol of $\mathcal{F}$ for the order $<_{\mathcal{F}}$. This special symbol will stand for encryption in the case of the prefix property or will stand for signing in the case of blind signatures.

Definition 3 (class $\mathcal{C}_S$). *Let $\mathcal{C}_S$ be the set of clauses of the form:*

$$\begin{aligned} & I(f_0(y_j, z)) \vee -I(f_0(u[g(y_1, \ldots, y_k)], v)) \vee \\ & \vee \bigvee_{i=1}^{p} -I(w_i[g(y_1, \ldots, y_k)]) \vee \bigvee_{l=1}^{q} -I(y_{i_l}), \end{aligned} \tag{1}$$

where

- $\{j, i_1, \ldots, i_q\} \subseteq \{1, \ldots, k\}$, $p, q \geq 0$, *and*
- $u, v \in \mathcal{T}(\mathcal{F} \cup \{z\})$ *and,*
- $I(f_0(u[g(y_1, \ldots, y_k)], v))$ *is greater than any other literal of the clause.*

For example, the clause C_{pre} is obtained when $u = v = z$, $j = 1$, $p = q = 0$, $f_0 = \{\_\}_\_$ and, $g = \langle \_, \_ \rangle$. The clause C_{sig} is obtained when $u = v = z$, $j = 1$, $p = 0$, $q = 1$, $f_0 = \mathrm{sig}$ and, $g = \mathrm{blind}$. We could also consider for example the clause $-I(\{\langle x, y \rangle\}_z) \vee I(\{y\}_z)$.

Of course, this class could also be used to express more complex protocol clauses.

2.2 Main result

We show that satisfiability of clauses of $\mathcal{C} = \mathcal{C}_I \cup \mathcal{C}_P \cup \mathcal{C}_S$ is still decidable, under a slight semantical assumption. To get this result, we consider a variant of ordered resolution where resolution between clauses of a *saturated* set are forbidden. In Section 2.2.1, we recall the definition of ordered resolution. In Section 2.2.2, we introduce our variant of ordered resolution. We prove our decidability result in Section 2.2.3 and show in Section 2.2.4 that both CBC encryption and blind signatures satisfy the hypotheses of our theorem.

2.2.1 Ordered resolution

We consider a liftable partial ordering $\prec$, total on closed atoms.

Let A and B be two atoms, C, C_1 and C_2 be clauses and, $\sigma = mgu(A, B)$. The *resolution rule* is defined by:

$$\frac{C_1' \stackrel{\text{def}}{=} C_1 \vee A \quad -B \vee C_2 \stackrel{\text{def}}{=} C_2'}{C_1\sigma \vee C_2\sigma}$$

The clause $C_1\sigma \vee C_2\sigma$ is called *resolvent* of the clauses C_1' and C_2'. The atom $A\sigma$ is called the *resolved atom.*

The *ordered resolution* (wrt $\prec$) requires that there is no atom in the resolvent greater than the resolved atom.

If $C_1, C_2, \dots, C_n$ are clauses such that their sets of variables are pairwise disjoint, then we denote the clause $C_1 \vee C_2 \vee \cdots \vee C_n$ by $C_1 \sqcup C_2 \sqcup \cdots \sqcup C_n$, in order to emphasize this property. Considering a set $\mathcal{T}$ whose elements are sets of clauses, the *splitting rule* is defined as follows: $\mathcal{T} \rightarrow_{spl} (\mathcal{T} \setminus \{S\}) \cup \{(S \setminus \{C_1 \sqcup C_2\}) \cup \{C_1\}\} \cup \{(S \setminus \{C_1 \sqcup C_2\}) \cup \{C_2\}\}$, where $S \in \mathcal{T}$ and $C_1 \sqcup C_2 \in S$. We write $\mathcal{T} \Rightarrow_{spl} \mathcal{T}'$ to say that $\mathcal{T} \rightarrow_{spl}^{*} \mathcal{T}'$ and no application of the splitting rule on $\mathcal{T}'$ is possible anymore. For a binary relation ρ, we denoted by ρ^* its reflexive and transitive closure.

2.2.2 Our resolution method

A *partial ordered interpretation* $\mathfrak{I}$ is a set of ground literals such that if $A \in \mathfrak{I}$ then $-A \notin \mathfrak{I}$, and if $\pm A \in \mathfrak{I}$ and $B \prec A$, then $\pm B \in \mathfrak{I}$. A ground clause C is *false* in $\mathfrak{I}$ if, for every literal $\pm A$ in C, the opposite literal $\mp A$ belongs to $\mathfrak{I}$. A clause C is *unsatisfiable* in the partial interpretation $\mathfrak{I}$ if there exists a ground substitution θ such that all atoms of $C\theta$ are among those of $\mathfrak{I}$ and $C\theta$ is false in the interpretation. A set of clauses is *unsatisfiable* in the partial interpretation if there is a clause in the set that is unsatisfiable in the partial interpretation.

Definition 4. *A set S of clauses is* saturated *w.r.t. the order $\prec$ if for every resolvent C obtained by ordered resolution from S and for every partial interpretation $\mathfrak{I}$, if C is unsatisfiable in $\mathfrak{I}$ then S is unsatisfiable in $\mathfrak{I}$.*

Let S be a saturated set of clauses. For a set of clauses T such that $S \subseteq T$, we denote by $Res(T)$ the set of clauses derived by ordered resolution method with the restriction that we do not apply resolution if the two premises are clauses of S. We define $R(T) \stackrel{\text{def}}{=} T \cup Res(T)$. For a class $\mathcal{T}$ of sets of clauses we denote by $R(\mathcal{T}) \stackrel{\text{def}}{=} \{R(T) \mid T \in \mathcal{T}\}$. Also, we write $\mathcal{T} \Rightarrow_{\prec,spl} \mathcal{T}'$ to say that $R(\mathcal{T}) \Rightarrow_{spl} \mathcal{T}'$. Remark that $\mathcal{T}'$ is unique. We denote by $\mathcal{R}_S$ the ordered resolution method with splitting together with the mentioned restriction. The following result states the refutational completeness of this method:

Proposition 2. *For any liftable ordering $\prec$, for any sets S and T of clauses, such that S is saturated w.r.t. $\prec$ and $S \subseteq T$, T is unsatisfiable if and only if $\{T\} \Rightarrow^*_{\prec,spl} \mathcal{T}$, for some $\mathcal{T}$ such that every set of clauses in $\mathcal{T}$ contains the empty clause.*

2.2.3 A decidable class

Our resolution method is still not sufficient to ensure termination for clauses of $\mathcal{C}$. Thus we consider an additional slight syntactic restriction. For a protocol point of view, this restriction does not reduce the expressivity of the fragment of clauses under consideration.

Definition 5. *We say that a term $f_0(u, v)$ is* well-behaved *if the following two implications are true: if $V(u) \neq \emptyset$ then v is a constant; and, if $V(v) \neq \emptyset$ then u is a ground term.*

Definition 6. *We say that a clause of $\mathcal{C}_S$ is* well-behaved *if all subterms $f_0(s, t)$ of the terms u, v and w_i, for all i, (see Definition 3) are well-behaved. We say that a clause C not in $\mathcal{C}_S$ is* well-behaved *if for every literal $\pm I(w)$ of C, every subterm $f_0(s, t)$ of w is well-behaved.*

For example, if $\mathcal{S} = \{C_{pre}\}$ (or $\mathcal{S} = \{C_{sig}\}$) (see previous section) then $\mathcal{S} \cup \mathcal{P}_{\mathsf{ex}}$ is well-behaved.

We are now ready to state our main result.

Theorem 3. *Let $\mathcal{I}, \mathcal{P}, \mathcal{S}$ be finite sets included, respectively, in the classes $\mathcal{C}_I$, $\mathcal{C}_P$ and $\mathcal{C}_S$. If $\mathcal{I} \cup \mathcal{S}$ is saturated and $\mathcal{P} \cup \mathcal{S}$ is well-behaved then the satisfiability of $\mathcal{I} \cup \mathcal{P} \cup \mathcal{S}$ is decidable.*

With regard to the complexity of this decision procedure we can obtain, using an argument similar to that in [6], that the satisfiability of the set $\mathcal{I} \cup \mathcal{P} \cup \mathcal{S}$ is decidable in 3-EXPTIME.

2.2.4 Examples

In this section, we show that the intruder clauses corresponding to our two examples (CBC encryption and blind signatures) are saturated. This means that we can analyze any protocol encoded in $\mathcal{C}_P$ under an extended intruder that has access either to CBC encryption or blind signatures. Comon-Lundh and Cortier [4] have identified these protocols to be protocols for which, at each transition, at most one part of the message is blindly copied or tested.

We assume a fixed basic intruder power, modelled by the set $\mathcal{I}$ presented in Figure 1.

We first consider the case of the CBC encryption.

Proposition 4. *The set $\mathcal{I} \cup \{C_{pre}\}$ is saturated.*

The same property is true in the blind signature case.

Proposition 5. *The set $\mathcal{I} \cup \{C_{sig}\}$ is saturated.*

As a consequence of these two propositions and applying Theorem 3, we get that for any well-behaved set $\mathcal{P}$ (encoding both a protocol and a security property), the satisfiability of $\mathcal{I} \cup \{C_{pre}\} \cup \mathcal{P}$ (resp. of $\mathcal{I} \cup \{C_{sig}\} \cup \mathcal{P}$) is decidable. Since, for example, secrecy can be modeled using a ground clause (for example $-I(\mathrm{n}(a, b))$ to express that the intruder should not learn the nonce between a and b), we obtained a procedure for deciding the secrecy of protocols that use the described prefix property or blind signature.

Corollary 6. *The secrecy problem for cryptographic protocols with single blind copying , with bounded number of nonces but unbounded number of sessions, using as primitives CBC encryption or blind signature is decidable.*

In addition, in the case of other extensions of the intruder power leading to other sets $\mathcal{S}$ of clauses in $\mathcal{C}_S$, the saturation of the set $\mathcal{I} \cup \mathcal{S}$ can be easily verified by hand (like in our examples).

2.3 Application to a cryptographic protocol

2.3.1 Presentation of the protocol

We consider the Needham-Schroeder symmetric key authentication protocol [12] as an example of application of our result. The goal of the protocol is the key exchange between two parties, which we call Alice and Bob, and the mutual conviction of the possession of the key by each other. The key is

$$\begin{array}{lll}
(1).1 & A \Rightarrow S: & A, B, N_a \\
(1).2 & S \Rightarrow A: & \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}} \\
(2).3 & I(B) \Rightarrow A: & \{N_a, B\}_{K_{as}} \\
(2).4 & A \Rightarrow I(B): & \{N'_a\}_{N_a}
\end{array}$$

Figure 3: Attack on the Needham-Schroeder protocol, using the prefix property

created by a trusted server which shares the secret keys K_{as} and K_{bs} with Alice and Bob, respectively. The description of the protocol is as follows:

$$P_{\mathsf{NS}}: \left\{ \begin{array}{ll}
A \Rightarrow S: & A, B, N_a \\
S \Rightarrow A: & \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}} \\
A \Rightarrow B: & \{K_{ab}, A\}_{K_{bs}} \\
B \Rightarrow A: & \{N_b\}_{K_{ab}} \\
A \Rightarrow B: & \{N_b - 1\}_{K_{ab}}
\end{array} \right.$$

Here we concentrate on the key exchange goal, rather than on the authentication of the two parties. The key exchange goal can be expressed as the secrecy of the nonce N_b. Intuitively, if N_b remains secret, it means that the key K_{ab} used by B has also been kept secret.

If the cryptosystem used to implement this protocol uses, for example, CBC then the following attack [14] is possible. In a first session (1), an intruder can listen to the message $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$ and then, using the CBC property, he can compute $\{N_a, B\}_{K_{as}}$. In another session of the protocol, he can send it to Alice in the third round. Alice thinks that Bob has started a session (2) with her: Bob plays the role of the initiator and Alice the role of the second participant. And so Alice would use N_a as the shared key, while it is a publicly known message. This attack is summarized in Figure 3.

The clauses that model the protocol rules are the following ones:

$$\begin{array}{rl}
& \rightarrow I(\langle \alpha, \beta, \mathrm{n}_1(\alpha, \beta)\rangle) \\
I(\langle \alpha, \beta, x\rangle) & \rightarrow I(\{x, \beta, \mathrm{k}(\alpha, \beta), \\
& \qquad \{\mathrm{k}(\alpha, \beta), \alpha\}_{\mathrm{k}(\beta,s)}\}_{\mathrm{k}(\alpha,s)}) \\
I(\{\mathrm{n}_1(\alpha, \beta), \alpha, y, z\}_{\mathrm{k}(\alpha,s)}) & \rightarrow I(z) \\
I(\{y, \alpha\}_{\mathrm{k}(\beta,s)}) & \rightarrow I(\{\mathrm{n}_2(\beta, \alpha)\}_y) \\
I(\{x\}_y) & \rightarrow I(\{\mathrm{pred}(x)\}_y)
\end{array}$$

where α and β range over constants that denote the identities of the involved parties. Comon and Cortier [5] have shown that, for secrecy properties, it is sufficient to verify the correctness of a protocol for only three participants:

two honest and one dishonest. Hence, we consider three agents having their identities represented by the constants a, b and i, where a and b stand for the honest participants while i stands for the dishonest participant. We denote by $C_{\mathsf{NS}_j}(\alpha,\beta)$, for j from 1 to 5, the clauses listed above, corresponding to the rules of the protocol P_{NS} when the initiator is α and the responder is β. The set of clauses that model the rules of the protocol is

$$\mathcal{P}_{\mathsf{NS}} \overset{\text{def}}{=} \bigcup_{1\leq j\leq 5,\ \alpha,\beta\in\{a,b,i\},\alpha\neq\beta} \{C_{\mathsf{NS}_j}(\alpha,\beta)\}.$$

The intruder has also some initial knowledge. He knows the identities of the participating parties, he can create nonces and, he knows the secret key of the compromised agent. This initial knowledge is modeled by the following clauses:

$$\begin{array}{ll} \to I(a) & \to I(\mathrm{k}(i,s)) \\ \to I(b) & \to I(\mathrm{n}_1(i,x)) \\ \to I(i) & \to I(\mathrm{n}_2(i,x)) \end{array}$$

We denote this set of clauses, corresponding to the initial knowledge of the intruder, by $\mathcal{P}_0$. We remark that these clauses are either ground or with a single variable thus belong to $\mathcal{C}_P$.

In addition, we enrich the set $\mathcal{I}$ (defined in Section 2.1.2) with he clause $I(x) \to I(\mathrm{pred}(x))$ that models the ability of the intruder to compute the predecessor of a message (seen as a number).

2.3.2 Correcting the protocol

We remark that the attack comes from the fact that the intruder, using the second rule of the protocol together with the CBC property, can get the encryption of any message by the key K_{as}: replacing the nonce N_a by any plaintext m of its choice in the first message, he obtains a message of the form $\{m,\ldots\}_{K_{as}}$ from the server and using the CBC property he gets $\{m\}_{K_{as}}$.

To avoid this, we interchange the place of N_a and B in the message sent in the second round. But a similar attack is still possible since the intruder can modify the first message of Alice and send $\langle A,B,B\rangle$ to the server. Then the shared key would be the identity B. Such an attack is possible only if identities can be confused with keys.

To avoid such a type flaw attack, we add a hash of the shared key as the first component of the message sent by the server to Alice and then to Bob. Note that this second transformation is not sufficient by itself since the intruder has also the ability to produce hashes. The obtained protocol is described below. We refer to this version as the corrected protocol.

$$P_{\mathsf{NSc}} : \begin{cases} A \Rightarrow S : & A, B, N_a \\ S \Rightarrow A : & \{B, N_a, K_{ab}, \{h(K_{ab}), K_{ab}, A\}_{K_{bs}}\}_{K_{as}} \\ A \Rightarrow B : & \{h(K_{ab}), K_{ab}, A\}_{K_{bs}} \\ B \Rightarrow A : & \{N_b\}_{K_{ab}} \\ A \Rightarrow B : & \{N_b - 1\}_{K_{ab}} \end{cases}$$

The clauses that model the rules of this protocol are the following ones:

$$\begin{aligned} &\to I(\langle \alpha, \beta, \mathrm{n}_1(\alpha, \beta)\rangle) \\ I(\langle \alpha, \beta, x\rangle) &\to I(\{\beta, x, \mathrm{k}(\alpha, \beta), \{\mathrm{h}(\mathrm{k}(\alpha, \beta)), \\ &\qquad \mathrm{k}(\alpha, \beta), \alpha\}_{\mathrm{k}(\beta,s)}\}_{\mathrm{k}(\alpha,s)}) \\ I(\{\beta, \mathrm{n}_1(\alpha, \beta), y, z\}_{\mathrm{k}(\alpha,s)}) &\to I(z) \\ I(\{\mathrm{h}(y), y, \alpha\}_{\mathrm{k}(\beta,s)}) &\to I(\{\mathrm{n}_2(\beta, \alpha)\}_y) \\ I(\{x\}_y) &\to I(\{\mathrm{pred}(x)\}_y) \end{aligned}$$

As for the protocol P_{NS}, we denote by $\mathcal{P}_{\mathsf{NSc}}$ the set of clauses obtained from the ones presented above by instantiating α and β ($\alpha \neq \beta$) by the constants a, b and i.

2.3.3 Secrecy of the corrected protocol

We observe that the clauses corresponding to the third round and fifth round of the protocol P_{NSc} are not in $\mathcal{C}_P$ since they have two variables. Therefore we cannot apply directly our result and we are led to an additional modification of the protocol.

We remark that the server sends to Alice in the second round an encrypted message that Alice cannot decrypt. This message could be directly sent to Bob by the server. In addition, the last rule of the protocol does not seem to be able to compromise the secrecy of N_b, thus we remove it.

Our approach is as follows: we prove that this version is a weaker version than the corrected protocol, *i.e.* that its correctness implies the correctness of the corrected version. Then, since this version fits our class, we apply our resolution method to prove that this version preserves the secrecy of N_b, which allow us to conclude that the corrected version also preserves the secrecy of N_b. For more details about this approach see [7].

3 Regular Protocols

3.1 Introduction

A common practice in automatic verification of cryptographic protocols is approximating the knowledge of the intruder by regular languages of terms. In [17], Tomasz Truderung defines a class of cryptographic protocols, called *regular protocols*, such that the *exact* knowledge which the intruder can gain during an *unbounded number of sessions* of a protocol is a regular language. More precisely, this knowledge can be represented by an alternating tree automaton of polynomial number of states (and thus by a deterministic tree automaton of exponential size). This automaton can be computed in exponential time. The secrecy of regular protocols is DEXPTIME-complete.

Regular protocols are a generalization of regular unary-predicate programs defined in [9]. They are also closely related to a class of monadic Horn theories defined in [18]. Regular protocols are more general than the class H1 defined in [13], and more general than the class of protocols without nonces, and satisfying so called *independence* condition, defined in [1] (the authors showed DEXPTIME-completeness of security problem for protocols in this class).

In the definitions below, we assume that a protocol is given as a set of Horn clauses. A protocol is regular if all its clauses are regular as it is defined in Subsection 3.2. One of the restrictions is that right-hand sides of rules have to be *almost linear*, i.e. each variable can occur at most once in each encrypted subterm. This condition is obviously necessary, if the knowledge of the intruder should be a regular language. The other syntactical conditions, as follows from the comparison to the similar results, is relatively unrestrictive. In fact, when we replace nonces by constants, many simple protocols (for example from the Clark and Jacob library [3] turns out to be regular.

We can model any ability of the intruder to derive new messages which can be expressed as regular clause, as is the case for the standard abilities of the intruder (pairing, encrypting, decryption, and so on). Under the additional restriction that keys used have to be constant, then we can also model blind signatures, and the prefix property of CBC encrytpion.

3.2 The Definition

Definition 7. *A term s* covers *x in a term t, if either $s = x$, or $s = f(s_1, \ldots, s_n)$ and each occurrence of x in t is in the context of (at least) one of $s_1, \ldots, s_n$. Next, let φ be the function which assigns a set of terms to* *a term, defined by the equations $\varphi(t) = \varphi(t_1) \cup \varphi(t_2)$, if $t = \langle t_1, t_2 \rangle$, and $\varphi(t) = \{t\}$, otherwise.*

Example 1. *It is easy to check that the term $s = \{\{x\}_b, y\}_a$ covers x in $t = \{\{x\}_b, \{y, \{x\}_b\}_a\}_a$ (because each occurrence of x in t is in the context of $\{x\}_b$), but s does not cover x in $\{\{x\}_c\}_b$. Note also that any term covers x in $\{y\}_a$.*

Example 2. *We have $\varphi(\langle \{b\}_k, \langle \{b, c\}_k, d \rangle \rangle) = \{\{b\}_k, \{b, c\}_k, d\}$.*

Definition 8. *A rule $r \to s$ is* regular*, if for each $s' \in \varphi(s)$ the following conditions hold: s' is linear and there exists a function $\gamma_{s'}$, which assigns subterms of s' to terms in $\varphi(r)$, such that:*

(i) for each $r' \in \varphi(r)$ and each $x \in Var(s')$, the term $\gamma_{s'}(r')$ covers x in r',

(ii) for each $r', r'' \in \varphi(r)$, if a variable $y \notin Var(s')$ occurs in both r' and r'', then $\gamma_{s'}(r') = \gamma_{s'}(r'')$.

A protocol is regular *if it consists only of regular rules.*

Example 3. *The following rules are regular:*

$$I(\{N_A, x, B, \{x, A\}^p_{K_B}\}^p_{K_A}) \to I(\{x, A\}^p_{K_B}) \tag{2}$$

$$I(\{z, \{\{y\}_a, x\}_b\}_a) \to I(\{\{x, \{y\}_a\}_b, z'\}_c) \tag{3}$$

$$I(\{\{x, y\}_a, z\}_b), I(\{z, z\}_c) \to I(\{\{y, x\}_b, d\}_c), I(\{z, \{x, y\}_a\}_c). \tag{4}$$

The rule $\{\{x\}_b, y\}_a \to \{x, \{y\}_b\}_a$ is not regular.

3.3 The Results

Now, we state the results presented in [17].

Theorem 7. *The knowledge which the intruder can gain during an unbounded number of sessions of a regular protocol, can be described by an alternating tree automaton with the polynomial number of states w.r.t. the size of the protocol. Moreover, such an automaton can be computed in exponential time.*

To prove the theorem we proceed as follows. First, one can translate a regular protocol into a unary logic program in a direct way. Next, clauses describing the ability of the intruder to derive new messages are added to this program. One can show that the resulting logic program describes the knowledge of the intruder, and moreover, that it can be rewritten to an equivalent program having only clauses of the form $p(f(x_1, \ldots, x_n)) \leftarrow p_1(t_1), \ldots, p_n(t_n)$, where $f(x_1, \ldots, x_n)$ is linear. Monadic Horn theories consisting of clauses of this form are considered in [18], where it is shown that they can be finitely saturated by a sort resolution. We can proceed similarly. Roughly speaking, we saturate the program (using a kind of sort resolution), and then remove some redundant clauses. The result is, in essence, an alternating automaton. One can show that the saturation process stops after at most exponential number of steps.

Theorem 8. *Deciding secrecy of a regular protocol is* DEXPTIME*-complete.*

To decide secrecy of a regular protocol, one can build (in exponential time) an alternating tree automaton of polynomial number of states which describes the knowledge of the intruder, and check whether the constant *secret* belongs to the language of this automaton, which again can be done in exponential time. DEXPTIME-hardness can be shown by reduction of the emptiness of the intersection of regular tree languages given by n finite automata.

4 Conclusion

We have presented two techniques for obtaining completeness results w.r.t. an unbounded number of sessions. The first one is based on the refutation completeness of a resolution strategy and it has been applied to the debugging of a protocol under a more realistic threat model than the one usually considered.

The second one is based on the representation of intruder knowledge as a regular language. Extensions are planned for handling more general algebraic properties of cryptoprimitives and also for adressing authentication goals too.

References

[1] R. M. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *CONCUR*, volume 2421 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2002.

[2] B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *FoSSaCS*, 2003.

[3] J. Clark and J. Jacob. A survey of authentication protocol literature : Version 1.0., November 1997.

[4] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *Lecture Notes in Computer Science*, pages 148–164, Valencia (Spain), June 2003. Springer-Verlag.

[5] H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In *Proc. of the 12th European Symposium On Programming (ESOP'03)*, volume 2618 of *Lecture Notes in Computer Science*, pages 99–113, Warsaw (Poland), April 2003. Springer-Verlag.

[6] V. Cortier. *Vérification automatique des protocoles cryptographiques.* PhD thesis, École Normale Supérieure de Cachan, Cachan (France), March 2003.

[7] V. Cortier, M. Rusinowitch, and E. Zălinescu. A Resolution Strategy for Verifying Cryptographic Protocols with CBC Encryption and Blind Signatures. In *Proc. of the 7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'05)*, Lisboa (Portugal), July 2005. ACM Press. To appear.

[8] D. Dolev, S. Even, and R. M. Karp. On the security of ping-pong protocols. In *CRYPTO*, pages 177–186, 1982.

[9] T. W. Frühwirth, E. Y. Shapiro, M. Y. Vardi, and E. Yardeni. Logic programs as types for logic programs. In *LICS*, pages 300–309, 1991.

[10] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology - AUSACRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer-Verlag, 1992.

[11] S. Kremer and M. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi-Calculus. In M. Sagiv, editor, *Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., April 2005. Springer-Verlag.

[12] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, 1978.

[13] F. Nielson, H. R. Nielson, and H. Seidl. Normalizable horn clauses, strongly recognizable relations, and spi. In *SAS*, volume 2477 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2002.

[14] O. Pereira and J.-J. Quisquater. On the perfect encryption assumption. In *Proc. of the 1st Workshop on Issues in the Theory of Security (WITS'00)*, pages 42–45, Geneva (Switzerland), 2000.

[15] R. Ramanujam and S. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In *FSTTCS*, 2003.

[16] H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *Proc. of 11th International Conference on Logic for Programming and Automated Reasoning (LPAR'04)*, volume 3452 of *Lecture Notes in Computer Science*, pages 79–94, Montevideo (Uruguay), 2005. Springer-Verlag.

[17] T. Truderung. Regular protocols and attacks with regular knowledge. In *CADE*, Lecture Notes in Computer Science. Springer, 2005. To appear.

[18] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *CADE*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 1999.