Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

Decidability Of Opacity With Non-Atomic Keys

Laurent Mazaré

Report n^{o} 23

October 27, 2004

Decidability Of Opacity With Non-Atomic Keys

Laurent Mazaré

October 27, 2004

Abstract

The most studied property, secrecy, is not always sufficient to prove the security of a protocol. Other properties such as anonymity, privacy or opacity could be useful. Here, we use a simple definition of opacity which works by looking at the possible traces of the protocol using a new property over messages called similarity. The opacity property becomes a logical constraint involving both similarities and syntactic equalities. The main theorem proves that satisfiability of these constraints and thus opacity are decidable without having to make the hypothesis of atomic keys. Moreover, we use syntactic equalities to model some deductions an intruder could make by performing bit-to-bit comparisons (i.e. known-ciphertext attack).

Keywords: Opacity, Security, Formal Verification, Dolev-Yao Constraints, Rewriting Systems, Decidability.

Reviewers:

Notes:

How to cite this report:

@techreport { verimag-TR-2004-23,
title = { Decidability Of Opacity With Non-Atomic Keys},
authors = { Laurent Mazaré},
institution = { Verimag Technical Report },
number = {23},
year = { 2004},
note = { }
}

1 Introduction

During the last decade, verification of security protocols has been widely investigated. The majority of the studies focussed on demonstrating secrecy properties using formal methods (see for example [5], [7], [6] or [11]). These methods have lead to effective algorithms and so to concrete tools for verifying secrecy such as these proposed by the EVA project [4] or the Avispa project [2]. However, checking security protocols requires studying other properties such as anonymity or opacity: hiding a piece of information from an intruder. For example, in a voting protocol, whereas the intruder is able to infer the possible values of the vote (yes or no), it should be impossible for him to guess which vote was expressed, only by observing a session of this protocol. Checking a protocol should include a way of formalizing the informations that were leaked and that the intruder can guess. In the last few years, attempts have been made to properly define opacity properties, to prove their decidability in certain cases and to propose some verification algorithms. As far as we know, other versions of opacity ([3], [12]) have been given in the literature but none of these criterion were implemented. Our notion of opacity is very close to the one introduced in [12] except that we use a formalism dedicated to protocols studies when they use a more general functional approach.

In this paper, we adopt a simple definition for opacity. The intruder C has a passive view of a protocol session involving two agents A and B. He is able to read any exchanged messages but he cannot modify, block or create a message. A property will be called *opaque* if there are two possible sessions of the protocol such that: in one of these, the property is true whereas it is not in the other, and it is impossible for the intruder to differentiate the messages from these two sessions from the messages exchanged in the original session. The starting point is the notion of similarity. This binary relation noted $\sim$ is an equivalence relation between messages. Two messages are similar if it is not feasible for the intruder to differentiate them. A typical example is two different messages encoded by a key that the intruder cannot infer. From the point of view of the intruder, these messages will be said similar. This notion is of course dependent of the knowledge of the intruder given by Dolev-Yao theory [9]: if the intruder is able to infer any of the used keys, then similarity will be equivalent to syntactic equality. This notion of similarity will allow us to express opacity properties as constraints. These constraints will also include syntactic equality. Equalities are used to show that the intruder can perform bit-to-bit comparisons between some messages (this is best known in the literature as known-ciphertext attacks). Let us give a simple example that will be useful throughout this paper. A simple electronic voting protocol is given by the transmission between A (the voter) and S (the authority counting votes). Variable v is the expressed vote chosen among the possible values yes or no.

$$A \to S : \{v\}_{pub(S)}$$

If the intruder intercepts the value of $\{v\}_{pub(S)}$, then as he can compare it to $\{yes\}_{pub(S)}$ and $\{no\}_{pub(S)}$, the intruder is able to deduce the value of v. We will say that the intruder performed a bit-to-bit comparison between $\{v\}_{pub(S)}$ and $\{yes\}_{pub(S)}$ (or $\{no\}_{pub(S)}$).

The aim of this paper is to formalize opacity as two constraints involving similarities and equalities taking into account possible bit-to-bit comparisons. The main result is decidability of satisfiability for such constraints using a finite-model property.

The remainder of this paper is organized as follows. In section 2, we recall usual definition for messages and protocols. Similarity over messages is introduced in section 3 and some useful properties are given. This section also formalizes the opacity property and translates it to a constraint. Both section 2 and 3 are very close to sections appearing in [15], they give the necessary basis to formalize the following sections. Then, section 4 proves that satisfiability for such constraints is decidable. Section 5 introduces the bit-to-bit comparisons in our constraints. Eventually, section 6 shows how to use this technique on a simple example, and section 7 concludes this paper.

2 Cryptographic Protocols

Let $Atoms$ and X be two infinite countable disjoint sets. $Atoms$ is the set of atomic messages a, b,... Set X contains variables called "protocol variables" x, y,...

Definition 1 (Message) *Let Σ be the signature $Atoms \cup \{pair, encrypt\}$ where $pair$ and $encrypt$ are two binary functions. The atomic messages are considered constant functions. Then a* message *is a first order term over Σ and the set of variables X, namely an element of $T(\Sigma, X)$. A message is said to be* closed *if it is a closed term of $T(\Sigma, X)$, i.e. a term of $T(\Sigma)$.*

In the rest of this paper, we will use the following notations:

$$\langle m_1, m_2 \rangle = pair(m_1, m_2)$$

$$\{m_1\}_{m_2} = encrypt(m_1, m_2)$$

Height of message m can be easily defined recursively and will be noted $|m|$. Substitutions σ from X to $T(\Sigma, X)$ are defined as usual. Application of substitution σ to message m will be noted $m\sigma$. If σ is defined by $x\sigma = n$ and $y\sigma = y$ for any other variable y, then we could write $m[x\backslash n]$ instead of $m\sigma$. The domain of a substitution σ is the set $dom(\sigma)$ of variables x such that $x\sigma \neq x$.

The set of variables used in a message m is noted $var(m)$ (or $free(m)$).

Definition 2 (Protocol) *Let $Actors$ be a finite set of participants called* actors*. The set of* programs $Progs$ *is given by the following syntax where B is in $Actors$, m_1, m_2 and m are messages.*

$$\begin{array}{rcl} G & ::= & \epsilon \\ & | & !_B m.G \\ & | & ?m.G \\ & | & if\ m_1 = m_2\ then\ G\ else\ G\ fi \end{array}$$

A protocol over the set of actors $Actors$ is a function from $Actors$ to $Progs$ associating a program to each actor.

The intuitive semantic of programs is the usual one: signification of $!_B m$ is to send the message m to agent B, signification of $?m$ is to receive a message and using pattern matching, to replace the variables learnt from m in the rest of the program.

For the following, the set of actors is fixed to $Actors$. Let $free$ be the function giving free variables of a program. It is easy to define $free$ in the usual recursive way. Then, $free$ can be extended over protocols. An instance of the protocol P is a protocol $P\sigma$ where σ instantiates exactly the free variables of P with closed messages. For that purpose, it is possible to rename every bound variable with a fresh variable such that bound variables are distinct and not in the free variables set. The substitution σ is called a *session* of the protocol P. When there is no risk of confusion on the protocol, its name will be omitted. Thus, we will talk about a session σ.

Definition 3 (Protocol Semantic) *The semantic of a protocol is the transition system over protocols defined by the following rules:*

- *If m is a closed message and σ is the most general unifier of m and m' (m' is called the proto-message of m),*

  $$\frac{Prog(A) = !_B m.P_A \quad Prog(B) = ?m'.P_B}{Prog \xrightarrow{m} Prog[A \to P_A; B \to P_B]\sigma}$$

  Note that, if σ does not exist, the protocol can be blocked. The transition is from the protocol $Prog$ to the protocol $Prog[A \to P_A; B \to P_B]$, i.e. the protocol linking A to program P_A, B to program P_B and other actors D to program $Prog(D)$.

- *If m_1 and m_2 are the same closed message,*

  $$\frac{Prog(A) = if\ m_1 = m_2\ then\ P_A\ else\ G\ fi}{Prog \to Prog[A \to P_A]}$$

- *If m_1 and m_2 are two distinct closed messages,*

$$\frac{Prog(A) = if\ m_1 = m_2\ then\ G\ else\ P_A\ fi}{Prog \rightarrow Prog[A \rightarrow P_A]}$$

A protocol P *terminates* iff for any Q such that $P \rightarrow^* Q$, it is possible to reach the state ϵ: $Q \rightarrow^* \epsilon$. Note that only closed protocols could terminate. A *run* of a session σ for a protocol P is an ordered set of messages $r = r_1.r_2...r_n$ such that

$$P\sigma \xrightarrow{r_1} ... \xrightarrow{r_n} \epsilon$$

A protocol session is *deterministic* if it has exactly only one possible run. This run will be noted $run(P\sigma)$. In the following sections, protocols will always be considered deterministic, i.e. each of their sessions is deterministic.

Eventually, to simplify notations, instead of writing:

$$Prog(A) = !_S\{v\}_{pub(S)}$$

$$Prog(S) = ?\{v\}_{pub(S)}$$

We will shorten this to:

$$A \rightarrow S : \{v\}_{pub(S)}$$

This paper will make an extensive use of Dolev-Yao theory [9]. Let E be a set of messages and m be a message, then we will note $E \vdash m$ if m is deducible from E using Dolev-Yao inferences.

3 Similarity and Opacity

3.1 Similarity

The intuitive definition of opacity is that an intruder is not able to distinguish a run where the property is satisfied from a run where it is not. To distinguish two messages, the intruder can decompose them, according to his knowledge but if he does not know the key k for example, he will not be able to make the difference between two different messages encoded by this key k. Two such messages will be called similar messages. This definition will be formalized using inference rules.

An *environment* is a finite set of closed messages. Usually, it will denote the set of messages known by the intruder. This definition will suppose that we only use symmetric key cryptography. However, all the following results can easily be generalized to public key cryptography.

Definition 4 (Similar Messages) *Two closed messages m_1 and m_2 are said to be* similar *for the environment env iff $env \vdash m_1 \sim m_2$ where $\sim$ is the smallest (w.r.t set inclusion) binary relation satisfying:*

$$\frac{a \in Atoms}{a \sim a}$$

$$\frac{u_1 \sim u_2 \quad v_1 \sim v_2}{\langle u_1, v_1 \rangle \sim \langle u_2, v_2 \rangle}$$

$$\frac{env \vdash k \quad u \sim v}{\{u\}_k \sim \{v\}_k}$$

$$\frac{\neg env \vdash k \quad \neg env \vdash k'}{\{u\}_k \sim \{v\}_{k'}}$$

Intuitively, this means that an intruder with the knowledge env will not be able to distinguish two similar messages. The environment name will be omitted as soon as it is not relevant for comprehension. Our definition of $\sim$ is very closed to the $\equiv$ operator introduced by Abadi and Rogaway in [1], except that we have an explicit environment to tell which keys are compromised instead of using directly the messages linked by the $\equiv$ operator.

Moreover, the definition of $\sim$ could easily be extended to non-closed environments and messages by adding this inference:

$$\frac{x \in X}{x \sim x}$$

This can also be achieved by defining $m \sim n$ for non-closed messages as $m\sigma \sim n\sigma$ for each σ such that $m\sigma$ and $n\sigma$ are closed.

Property 1 *The binary relation $\sim$ is an equivalence relation: let m_1, m_2 and m_3 be three messages.*

$$m_1 \sim m_1$$

$$m_1 \sim m_2 \Rightarrow m_2 \sim m_1$$

$$m_1 \sim m_2 \wedge m_2 \sim m_3 \Rightarrow m_1 \sim m_3$$

Proof 1 *These three properties can easily be proved using an induction on the structure of message m_1.* ∎

To prove that the $\sim$ relation is compatible with the context operation, we will have to suppose that only atomic keys are allowed. This hypothesis is only required for the following property.

Property 2 (Context) *Let m_1, m_2, m_3 and m_4 be four messages. If m_3 and m_4 have only one free variable x,*

$$m_1 \sim m_2 \wedge m_3 \sim m_4 \Rightarrow m_3[x \backslash m_1] \sim m_4[x \backslash m_2]$$

And in particular,

$$m_1 \sim m_2 \Rightarrow m_3[x \backslash m_1] \sim m_3[x \backslash m_2]$$

Let m and n be two messages and x a variable. Let σ be a substitution such that $x\sigma \sim n\sigma$. Then

$$m\sigma \sim m[x \backslash n]\sigma$$

Proof 2 *An induction on m_3 allows to deduce the first formulae. Then, using reflexivity of $\sim$, the second formulae is true. Eventually, the third formulae could be proved quickly with an induction on message m.* ∎

When considering similarity, an important problem is: given an environment env and a closed message m, what is the set of closed message n such that

$$env \vdash n \text{ and } env \vdash m \sim n$$

Note that the main difficulty is that we do not necesseraly have $env \vdash m$.

For that purpose, the $fresh$ function will be introduced. It is inductively defined over messages by the following equalities where all the variables y have to be instantiated with different fresh variables (i.e. variables that do not appear anywhere else).

$$\begin{array}{rcl} fresh(a) & = & a \\ fresh(x) & = & x \\ fresh(\langle m, m' \rangle) & = & \langle fresh(m), fresh(m') \rangle \\ fresh(\{m\}_k) & = & \{fresh(m)\}_k \text{ if } env \vdash k \\ fresh(\{m\}_k) & = & \{y\}_k \text{ if } env \nvdash k, \quad y \text{ is a fresh variable} \end{array}$$

Property 3 *For every substitution σ, we have*

$$m\sigma \sim fresh(m)\sigma$$

Proof 3 *This can be proved using an induction on the structure of m.*

- *If m is an atom or a variable then $m\sigma = fresh(m)\sigma$.*
- *If m is a pair $\langle m_1, m_2 \rangle$, then we have $m_1\sigma \sim fresh(m_1)\sigma$ and $m_2\sigma \sim fresh(m_2)\sigma$. So, we have:*

$$\begin{array}{rcl} m\sigma & = & \langle m_1\sigma, m_2\sigma \rangle \\ & \sim & \langle fresh(m_1)\sigma, fresh(m_2)\sigma \rangle \\ & \sim & fresh(m)\sigma \end{array}$$

- *If m is an encoding $\{m_1\}_k$ and $env \vdash k\sigma$, then $m_1\sigma \sim fresh(m_1)\sigma$. So $m\sigma \sim \{fresh(m_1)\}_{k\sigma} \sim fresh(m)\sigma$.*
- *If m is an encoding $\{m_1\}_k$ and $env \nvdash k\sigma$, then $m\sigma \sim \{y\sigma\}_{k\sigma} \sim fresh(m)\sigma$.*

∎

The reciprocal of this property is that if m is similar to n, then n is an instance of $fresh(m)$, i.e. $fresh(m)$ where all free variables are instantiated by closed messages.

Property 4 *Let m and n be two closed messages. If $m \sim n$, then there exists a substitution σ that acts over the free variables of $fresh(m)$ such that $n = fresh(m)\sigma$.*

Proof 4 *This proof can also be achieved using a simple induction on the structure of m.* ∎

3.2 The Opacity Problem

Let us consider a protocol PR and one of its session σ. We will be interested in predicates over σ, namely properties ψ that act over variables instantiated by σ. Such properties may express the identity of an agent, or the value of a vote, for instance. The opacity problem considered here relies on several hypothesis:

- The intruder C has a passive view of protocol session σ involving two agents A and B. Passive means that the intruder can intercept and view any messages exchanged by A and B but is not able to block, modify nor to send any message.
- The intruder knows the protocol used PR.
- The intruder has an initial knowledge c_0, which is a predicate (for example, $c_0 = (k_1 = k_2)$ means that C knows that the keys that will instantiate k_1 and k_2 are the same).

If we consider the witness run $run(P\sigma) = m_1.m_2...m_n$, property ψ will be opaque if there exist two possible sessions σ_1 and σ_2 of the protocol giving messages similar to the witness messages (m_1 to m_n) where for example, $\psi\sigma_1$ is true and $\psi\sigma_2$ is false. In this case, the intruder will not be able to deduce any knowledge on $\psi\sigma$. Of course, there is no need to find both σ_1 and σ_2: if $\psi\sigma$ is true, then we could use σ instead of σ_1, as exchanged messages are the same, they are similar. But we will keep this notation with three substitutions to show the symmetry of this problem.

Definition 5 (Opacity) *A property ψ is said to be* opaque *for a protocol session σ of P iff there exist two sessions of the protocol σ_1 and σ_2 such that*

$$c_0\sigma_1 \wedge p_1 \sim m_1 \wedge ... \wedge p_n \sim m_n \wedge \psi\sigma_1$$

$$c_0\sigma_2 \wedge q_1 \sim m_1 \wedge ... \wedge q_n \sim m_n \wedge \neg\psi\sigma_2$$

Where $p_1.p_2...p_n$ is the run of the protocol P related to σ_1, $q_1.q_2...q_n$ is related to σ_2 and $m_1.m_2...m_n$ is related to σ. Note that the three runs must have the same length n.

The environment env used in the previous conjunctions is $\{m_1, ..., m_n, p_1, ..., p_n, q_1, ..., q_n\}$ and could be augmented with an initial knowledge of the intruder env_0.

We defined opacity for a protocol session, this can be extended to protocols by saying that a property is opaque in a protocol if it is opaque for all its session. The problem is that the number of possible sessions (and their size) is unbounded. This leads to an unbounded number of possible behaviors for a protocol. In the following, we give a method to check opacity for a given session but we lack the method to extend it to a whole protocol.

For instance, let us consider the simple electronic voting protocol. Suppose that the session observed by the intruder is $\sigma = [v \backslash yes]$. Then, the environment will be $env = \{yes, no, pub(S), \{yes\}_{pub(S)}\}$. The predicates expressing the opacity of the vote value will be:

$$\{v\}_{pub(S)} \sim \{yes\}_{pub(S)} \wedge v = yes$$

$$\{v\}_{pub(S)} \sim \{yes\}_{pub(S)} \wedge \neg v = yes$$

As both predicates are satisfiable, the vote value is opaque in this case.

Our property of opacity can also be used to check anonymity. For example, if we take a definition of anonymity closed to the one given in [16], we just have to add a "restricted view" for the intruder, i.e. the intruder only intercepts some of the exchanged messages (for example, when considering a system with both secure and insecure channels). Then, opacity for property "identity of such actor" will be similar to what is defined as anonymity.

4 Initial Predicates and Satisfiability

In this section, the environment env is a finite set of closed messages. We will first define a class of predicates called *initial predicates*. Then, we will show that satisfiability for such predicates is decidable.

Definition 6 (Initial Predicates) *The set IP of* initial predicates *is given by the following formulas:*

$$P ::= P_A | P \wedge P$$

$$P_A ::= m \sim n | m = n | \bot | \top$$

Where m and n are two messages.

It is easy to extend initial predicates with the logical or operator $\vee$. This can be done by adding to the previous grammar the line:

$$P ::= P \vee P$$

Then, using usual rules on boolean operations, every predicate is equivalent to a predicate P' of the form:

$$P' = \bigvee \left(\bigwedge P_A \right)$$

If satisfiability for these predicates is decidable, then satisfiability over initial predicates is, of course, decidable too. Reciprocally, let us suppose that satisfiability over initial predicates is decidable, then satisfiability of any $\bigwedge P_A$ in P' is decidable and so is satisfiability of P'. Thus, adding the $\vee$ operator is cost-less and will not be done here.

The atomic predicate $\neg \vdash m$ can be added as syntactic sugar for $\{a\}_m \sim \{b\}_m$ where a and b are two distinct atoms. Indeed, these two messages could only be similar if message m is not deducible.

In order to prove satisfiability, we will first need the notion of model, that will be introduced in a very classical way.

Definition 7 (Model) *A substitution σ such that $P\sigma$ is closed is a* model *of P iff $\sigma \models P$ where $\models$ is defined by:*

$$\frac{}{\sigma \models \top} \qquad \frac{\sigma \models P_1 \quad \sigma \models P_2}{\sigma \models P_1 \wedge P_2} \qquad \frac{m\sigma \sim n\sigma}{\sigma \models m \sim n} \qquad \frac{m\sigma = n\sigma}{\sigma \models m = n}$$

The satisfiability problem is the usual one: given a predicate, we want to know wether there exists a model of this predicate or not. To study satisfiability, we will prove a finite model property: if a predicate is satisfiable, then there exists a valid model whose size is lower than a constant. If this holds, then to check satisfiability of a predicate, it is possible to try any models with a size lower than the constant: this gives us an immediate satisfiability checking algorithm (made of a finite number of checking steps which can all be achieved in finite time).

For that purpose, we will fix the notations. Let P be an initial predicate that is satisfiable and σ be a model of P. The set of messages $keys$ is defined as the set of keys that appear in P: a message m is in $keys$ iff there exists a message n such that $\{n\}_m$ is a sub-term of P. The set $keys$ is of course finite and does not depend on the chosen model σ. Eventually, the set $keys$ will be separated in two sets $keys^+$ and $keys^-$ defined by:

$$k \in keys^+ \Leftrightarrow env \vdash k\sigma$$

Note that there only exists a finite number of possibilities for $keys^+$ and $keys^-$ independently of σ. A substitution σ' will be said *adapted* to $keys^+, keys^-$ if the two following assertions are true:

$$k \in keys^+ \Rightarrow env \vdash k\sigma'$$

$$k \in keys^- \Rightarrow \neg env \vdash k\sigma'$$

The set of substitutions adapted to $keys^+, keys^-$ will be noted $ad(keys^+, keys^-)$. The idea is now to use these two sets of keys in order to formulate a rewriting system that will be correct for adapted models. Then, this rewriting system will terminate, giving us a simple normal form. With this normal form, we will be able to describe the set of models of the initial predicate. Eventually, we will show that there exists a model that is adapted (to $keys^+$ and $keys^-$) and whose size can be bounded. Model σ is supposed fixed and is, of course, a valid model of P.

Definition 8 (Rewriting System) *The rewriting system R_I is defined over the set of initial predicates by the following rules:*

- $Dec_\sim\{\}_1$ *: If there exist $m_2', n_2' \in keys^-$ such that $m_2\sigma = m_2'\sigma$ and $n_2\sigma = n_2'\sigma$,*

$$\{m_1\}_{m_2} \sim \{n_1\}_{n_2} \to \top$$

- $Dec_\sim\{\}_2$ *: Else, if there exist $m_2', n_2' \in keys^+$ such that $m_2\sigma = m_2'\sigma$ and $n_2\sigma = n_2'\sigma$,*

$$\{m_1\}_{m_2} \sim \{n_1\}_{n_2} \to m_1 \sim n_1 \wedge m_2 = n_2$$

- $Dec_\sim\{\}_3$ *: Else,*

$$\{m_1\}_{m_2} \sim \{n_1\}_{n_2} \to \bot$$

- $Dec_\sim\langle\rangle$ *:*

$$\langle m_1, m_2\rangle \sim \langle n_1, n_2\rangle \to m_1 \sim n_1 \wedge m_2 \sim n_2$$

- $Single$ *: If x is not in $var(m_1)$ nor in $var(m_2)$ and $|m_1| \leq |m_2|$,*

$$x \sim m_1 \wedge x \sim m_2 \to x \sim m_1 \wedge m_1 \sim m_2$$

- $Atom_\sim$ *: For atomic messages,*

$$a \sim a \to \top$$

- $Type_{\sim 1}$ *to* $Type_{\sim 4}$ *: If a and b are two distinct atoms,*

$$a \sim b \to \bot$$

$$a \sim \langle m_1, m_2\rangle \to \bot$$

$$a \sim \{m_1\}_{m_2} \to \bot$$

$$\{m_1\}_{m_2} \sim \langle n_1, n_2\rangle \to \bot$$

For the syntactic equality part, we will use classical unification rules noted $Dec_=\{\}$, $Dec_=\langle\rangle$, $Atom_=$ and $Type_{=i}$. The resolution rules will be noted Res_1 and Res_2. Res_1 occurs if x does not appear in m and does appear in P:

$$x = m \wedge P \rightarrow x = m \wedge P[x\backslash m]$$

Res_2 occurs in the case where x appears in m (and $m \neq x$):

$$x = m \wedge P \rightarrow \bot$$

We do not have a resolution-like rule for similarity, because that kind of rule would not be correct. Consider three messages m, n and p. Having m similar to n does not imply that $p[x\backslash m]$ is similar to $p[x\backslash n]$. In fact, using messages $m = \{a\}_k$, $n = \{b\}_k$ and $p = \{c\}_x$ with environment $\{m, n\}$ will prove that the former proposition is false when using non-atomic keys.

Now, we will state some basic results over R_I in the following properties.

Property 5 *The rewriting system R_I terminates. Moreover, R_I is correct and complete for models in $ad(keys^+, keys^-)$.*

Proof 5 *To prove termination, we will define some measure related to an initial predicate Q. Let nb_r be the number of variables that are not solved in Q. The definition of solved used here is the standard one used in unification. For details, see for example [13]. Let sd be the quantity defined by:*

$$sd = \sum_{m_1 \sim m_2} max(3^{|m_1|}, 3^{|m_2|}) + \sum_{m_1 = m_2} max(3^{|m_1|}, 3^{|m_2|})$$

Where $m_1 \sim m_2$ and $m_1 = m_2$ range over the atomic predicates (P_A) composing Q. And let nb_v be the number of occurrences in Q of similarities that have the form $x \sim m$.

Then, the order $<$ will be defined over initial predicates as the lexicographic order over (nb_r, sd, nb_v). This order will decrease strictly when applying $\rightarrow$ as shown in this array :

Rule	nb_r	sd	nb_v
Res_1	$<$		
Res_2	$<$		
$Dec_=\{\}$	$=$	$<$	
$Dec_=\langle\rangle$	$=$	$<$	
$Atom_=$	$=$	$<$	
$Type_{=i}$	$\leq$	$<$	
$Dec_\sim\{\}$	$=$	$<$	
$Dec_\sim\langle\rangle$	$=$	$<$	
$Atom_\sim$	$=$	$<$	
$Type_{\sim i}$	$\leq$	$<$	
$Single$	$=$	$=$	$<$

And so, this proves that R_I terminates.

Correctness and completeness are trivial for rewriting rules borrowed to unification. For the Single rule, transitivity of similarity ensures correctness and completeness. For the $Dec_\sim\{\}$ rules, this comes directly from the definition of $\sim$ and these rules are the reason why this is only correct for adapted models. ∎

Now that R_I does terminate, we can focus on the normal forms obtained using R_I on a predicate Q such that Q is adapted to $keys$, i.e. the set of keys related to Q is included in $keys$ (of course, we will usually take $Q = P$). It is easy to view that the normal form could be:

- $\top$, then Q is satisfied by any model (for example, Q was $a \sim a$).
- $\bot$, then Q cannot be satisfied by any model (for example, Q was $a \sim b$).

- $x_1 \sim m_1 \wedge ... \wedge x_a \sim m_a \wedge y_1 = n_1 \wedge ... \wedge y_b = n_b$ where the variables x_i and y_j are distinct, and the y_j do not appear in any m_i nor in any n_k.

Let us consider the third case. The length of messages in the normal form can be majored independently from σ. The only rule that could make the maximal length of messages in Q grow is Res. In that case, the maximal length could be multiplied by two. But this rule can only occur one time for each variable in Q. So, if n is the number of variables in P, m is the maximal length of messages in Q, then the maximal length of messages in the normal form is lower than $m.2^n$. The worst case is to produce a message by piling up all the initial messages.

To compute the set of models satisfying Q, we will proceed in three steps. These steps are close to those used in the key quantification method proposed in [15].

- Replace m_i by $fresh(m_i)$ using $keys^+$ and $keys^-$.
- If there is a loop $x_{i_1} \sim m_{i_1}$,...,$x_{i_\alpha} \sim m_{i_\alpha}$ and x_{i_2} occurs in m_{i_1},... x_{i_1} occurs in m_{i_α}, then there are no models satisfying Q.
- Else, there exists $x_i \sim m_i$ such that no variable x_j (for $j \neq i$) occurs in m_i. The possible values for x_i are given by instantiating the free variables of m_i with any possible messages. Like in unification, this can be done recursively for every x_j and then for every y_j.

Property 6 *There exists a message f_x for each variable x that is free in Q such that σ' models Q iff there exists a message v_i for each free variable z_i in f_x such that:*

$$x\sigma' = f_x[z_1 \backslash v_1, ..., z_c \backslash v_c]$$

The substitution $[z_1 \backslash v_1, ..., z_c \backslash v_c]$ related to a given σ' will be noted σ'_e.

Proof 6 *This is a direct consequence of the structure of the normal forms.* ∎

An important thing to notice is that the length of each f_x can be bounded by a constant that only depends on P and Q. This comes from the bound over the length of messages appearing in the normal form. For the rest of this paper, we will consider $Q = P$ and this bound will be noted $C_f(P)$.

We now have given all the definition needed to formulate the finite model property.

Definition 9 *The* potential complexity *of a predicate P, $C(P)$, is defined by:*

$$C(P) = C_f(P) + max(|m|/m \in env).c$$

Of course, the potential complexity depends only on P.

Theorem 1 (Finite Model) *If a predicate P is satisfiable, then it admits a model σ' such that*

$$\sum_{x \in var(P)} |x\sigma'| \leq C(P).card(var(P))$$

Corollary 2 *Satisfiability of predicates in IP is decidable.*

The finite model property is not enough to prove the corollary: the model is finite but there are an infinite number of possibilities for the atoms. This is not really a problem because we can restrict the possible atoms to those used in P as long as there are enough distinct fresh atoms such that all the leaf in any $x\sigma$ can be distinct (as the length of $x\sigma$ is small, this is possible).

5 Syntactic Equality

The addition of syntactic equality in predicates will be used to model the intruder performing bit-to-bit comparison between two deducible messages. For example, if the intruder has intercepted $\{v\}_{pub(S)}$ and has a bit-to-bit value equal to $\{yes\}_{pub(S)}$, then he can deduce that the value of v is yes. Two such messages will be called *identifiable messages*. We will show in this section that the knowledge brought by identifiable messages is computable. A kind of attack close to this one has already been studied with different techniques under the name of *guessing attacks* in [14], [10] or [8]. But these studies do not precisely link the values to the proto-messages as will be done here. In the previous section, we used the Dolev-Yao model: the only way to obtain some information from a ciphertext was to find the right key and to decode it. Here, we use a model where the intruder has stronger deduction capacities. This hypothesis holds for some encryption schemes but most of them use random bits so that these attacks are impossible to perform.

Definition 10 *Let $M = \{m_1, ..., m_n\}$ be a finite set of messages and σ be a substitution such that $M\sigma$ is closed. We define $\vdash_\sigma$ as follows.*

$$\frac{}{M\vdash_\sigma m_i}(1) \qquad \frac{M\vdash_\sigma m \quad M\vdash_\sigma n}{M\vdash_\sigma \langle m,n\rangle}(2) \qquad \frac{M\vdash_\sigma m \quad M\vdash_\sigma n}{M\vdash_\sigma \{m\}_n}(3)$$

$$\frac{M\vdash_\sigma \langle m,n\rangle}{M\vdash_\sigma m}(4) \qquad \frac{M\vdash_\sigma \langle m,n\rangle}{M\vdash_\sigma n}(5) \qquad \frac{M\vdash_\sigma \{m\}_n \quad M\sigma\vdash n\sigma}{M\vdash_\sigma m}(6)$$

The meaning of $M \vdash_\sigma m$ is that an intruder knowing M and looking at $M\sigma$ can link the prototype m to its value $m\sigma$. The intruder can add to its knowledge $m = m\sigma$ but is not allowed to discompose m and $m\sigma$ as soon as some keys could be not deducible. But, if the intruder can find two times the same message linked to two different prototypes m and n, then he will be able to deduce the syntactic equality $m = n$.

Note that $M \vdash_\sigma m$ implies $M\sigma \vdash m\sigma$. The inverse is of course false (take for example $m = x$, $M = \{y\}$ and $x\sigma = y\sigma = a$). First, we will prove a general property on our new theory: its locality.

Property 7 (Locality) *The theory $\vdash_\sigma$ is local: if $M \vdash_\sigma m$, then there exists a proof of $M \vdash_\sigma m$ such that for any intermediate occurrence of $M \vdash_\sigma m'$ in the proof, m' is either a sub-message of m, or a sub-message of a message present in M.*

Proof 7 *Let us consider a minimal proof of $M \vdash_\sigma m$. The idea is to use an induction on the structure of the proof in order to demonstrate the property as long as another one: if the last rule is among (1), (4), (5) or (6), then any intermediate occurrence of $M \vdash_\sigma m'$ in the proof, m' is a sub-message of a message present in M.* ∎

A direct application of this property is that if the last rule used in a minimal proof of $M \vdash_\sigma m$ is a decomposition, then m is a sub-message of a message of M.

Definition 11 (Identifiable Messages) *Two messages m and n are said to be identifiable for σ and M iff*

- $m \neq n$
- $M \vdash_\sigma m$ *and* $M \vdash_\sigma n$
- $m\sigma = n\sigma$

m and n are minimal *iff there does not exist a non initial position p such that $m_{|p}$ and $n_{|p}$ are identifiable ($m_{|p}$ is the sub-term of m occuring at position p).*

If two messages m and n are identifiable, then the intruder can add $m = n$ to its knowledge. Now, we want to be able to add all the knowledge that can be inferred to the intruder's knowledge without testing any possible couple of messages. We want to show that this knowledge is computable in a finite time. For that purpose, we will state two distinct properties.

Property 8 *The set of minimal identifiable messages is computable.*

To prove this property, we will have to use the locality of our theory. This property will lead us to a decision algorithm capable of producing every possible pair of minimal identifiable messages. Let m and n be two minimal identifiable messages. Consider minimal proofs for $M \vdash_\sigma m$ and $M \vdash_\sigma n$, using the symmetry between m and n, only three combinations of final rules for $M \vdash_\sigma m$ and $M \vdash_\sigma n$ are possible.

- These rules are both 2 or 3: then either $m = \langle m_1, m_2 \rangle$ and $n = \langle n_1, n_2 \rangle$, either $m = \{m_1\}_{m_2}$ and $n = \{n_1\}_{n_2}$. Let us consider the case of pairs. As we have $m \neq n$, we have $m_1 \neq n_1$ *or* $m_2 \neq n_2$. To fix the idea, we will consider $m_1 \neq n_1$. We have, of course, $m_1\sigma = n_1\sigma$ and $M \vdash_\sigma m_1$ and $M \vdash_\sigma n_1$. So m_1 and n_1 are identifiable, there is a contradiction with the minimality of the pair m, n. That is why, these rules cannot occur at the end of minimal proofs.
- If both rules are in 1, 4, 5 and 6. Then m and n are sub-messages from M. These messages are only in finite number.
- If the rule concerning m is in 1, 4, 5 and 6 and the rule concerning n is in 2 and 3. Then m is a sub-message of M. As $m\sigma = n\sigma$, we have $n\sigma$ in $SM(M\sigma)$ (sub-messages, i.e. sub-terms of $M\sigma$). So we have
$$|n\sigma| \leq max(|p|, p \in SM(M\sigma))$$
And we obtain a bound of the length of n:
$$|n| \leq max(|p|, p \in SM(M\sigma))$$
The atoms occurring in n have to occur in M and the variables occurring in n have to be instantiated by σ. So there are only a finite number of possible messages for n.

To find all the minimal similar messages, we have to test all the messages whose lengths are below $max(|p|, p \in SM(M\sigma))$. Messages from $SM(M)$ are in that set. These messages can only use atoms used in M and variables instantiated by σ. That is why the set of messages to be tested is finite. Moreover, checking that two messages are identifiable and minimal can be done in a finite time too, and so all the minimal identifiable messages can be found in a finite time.

Property 9 *If m and n are identifiable and p_1,...,p_k are the positions such that $m_{|p_i}$ and $n_{|p_i}$ are identifiable and minimal, then for any model σ,*

$$\sigma \models \left(m = n \Leftrightarrow m_{|p_1} = n_{|p_1} \wedge ... \wedge m_{|p_k} = n_{|p_k}\right)$$

Using properties stated in this section, we now have a method to model what an intruder can guess using bit-to-bit comparisons. Our method is easy to apply but inefficient as it tests any couple of message whose lengths are below a fixed bound. It produces constraints of the form $m = n$, they can be added to the opacity predicates. As we show in the previous section, satisfiability of the resulting predicates will remain decidable. More formally, M will be the awaited trace of the protocol in terms of proto-messages augmented with the initial knowledge of the intruder and env will be the set of intercepted messages as long as the initial knowledge, so we will usually take $env = M\sigma$. Then, opacity of a property P will be checked as satisfiability of to predicates $S \wedge P$ and $S \wedge \neg P$ where S contains similarities that occur in the opacity constraint. We will have to compute the set of minimal identifiable messages m_1, n_1 to m_k, n_k. This gives use another predicate E defined by:

$$E = \left(m_1 = n_1 \wedge ... \wedge m_k = n_k\right)$$

And so, we will study satisfiability of two predicates: $E \wedge S \wedge P$ and $E \wedge S \wedge \neg P$. This will be applied in the next section on two very simple electronic voting protocols.

6 Example: A Simple Electronic Vote Protocol

6.1 Simple Does Not Mean Secure

Let us consider the most simple electronic voting protocol. A is the voter and S the authority that will count the different votes. The possible votes are yes and no. Of course, one of the objective is that the expressed vote remains opaque. In a first version, the vote will just be sent from A to S encoded using the public key of S. The protocols is written:

$$A \rightarrow S : \{v\}_{pub(S)}$$

Where v is chosen among the values yes and no. Let us suppose that the expressed vote is yes, so the substitution σ is defined by $\sigma = [v \backslash yes]$. Then M is the set $\{\{v\}_{pub(S)}, yes, no, pub(S)\}$. The environment env is the set $M\sigma$. The value of $max(|p|, p \in SM(M\sigma))$ is 2 ($pub(S)$ is considered as an atomic message). We easily obtain the set of minimal identifiable messages:

$$\{(\{v\}_{pub(S)}, \{yes\}_{pub(S)})\}$$

So $E = \left(\{v\}_{pub(S)} = \{yes\}_{pub(S)}\right)$, the constraints of opacity related to the value of v are:

$$\{v\}_{pub(S)} = \{yes\}_{pub(S)} \wedge \{v\}_{pub(S)} \sim \{yes\}_{pub(S)} \wedge v = yes$$

$$\{v\}_{pub(S)} = \{yes\}_{pub(S)} \wedge \{v\}_{pub(S)} \sim \{yes\}_{pub(S)} \wedge v = no$$

By using our rewriting system on the second predicate, we have

$$v = yes \wedge \{v\}_{pub(S)} \sim \{yes\}_{pub(S)} \wedge v = no$$

And so this constraint can be rewrited to $\perp$, the second constraint is not satisfiable. The value of v is not opaque: the intruder can guess the vote. Intuitively, we already shew how the intruder could guess the vote in the introduction. Now, we want to fix this opacity flaw by modifying the protocol.

6.2 Adding Complexity

The technique used by the intruder is to guess which message can be encrypted and to compare the result with the intercepted message. As we do not want the intruder to guess which message can be encrypted, we add a nonce in the protocol:

$$A \rightarrow S : \{\langle v, n\rangle\}_{pub(S)}$$

As in the previous example, let us suppose that the expressed vote is yes and the nonce is instantiated by a fresh atom N, so the substitution σ is defined by $\sigma = [v \backslash yes, n \backslash N]$. Then M is the set $\{\{\langle v, n\rangle\}_{pub(S)}, yes, no, pub(S)\}$. The environment env is the set $M\sigma$. The value of $max(|p|, p \in SM(M\sigma))$ is 3. But now, the set of minimal identifiable messages is empty, so the constraints of opacity concerning the value of v are:

$$\{\langle v, n\rangle\}_{pub(S)} \sim \{\langle yes, N\rangle\}_{pub(S)} \wedge v = yes$$

$$\{\langle v, n\rangle\}_{pub(S)} \sim \{\langle yes, N\rangle\}_{pub(S)} \wedge v = no$$

It is easy to see that both constraints are satisfiable, so we now have that the expressed vote is opaque. This protocol can be used without fearing that an intruder can guess the value of the vote using bit-to-bit comparisons.

7 Conclusion

In this paper, we extended the notions presented in [15]: opacity is also defined as satisfiability of two constraints, but we do not need the hypothesis that keys are atomic anymore. However, the method introduced in [15] can be applicated to real case whereas our new method to decide satisfiability of constraints is far more complex. The set of predicates which satisfiability is decidable has been extended to predicates using syntactic equalities and non-atomic keys. Moreover, we introduced a new technique to determine what an intruder can guess using bit-to-bit comparisons. Now, we have two distinct theories: on one side Dolev-Yao $\vdash$ and on the other side $\vdash_\sigma$. Even if the two theories are closely linked, an idea for future work would be to produce a single theory modeling both kind of attacks. Another interesting extension would be to make the intruder active. If C can intercept and modify the messages, could he find the right messages to alter such that the property is not opaque any more ?

Acknowledgements

The author wishes to thank the anonymous reviewers for their very helpful and constructive comments.

References

[1] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany. 3.1

[2] The Avispa Project. http://www.avispa-project.org/, 1999. 1

[3] A. Boisseau. *Abstractions pour la vérification de propriétés de sécurité de protocoles cryptographiques*. PhD thesis, Laboratoire Spécification et Vérification (LSV), ENS de Cachan, 2003. 1

[4] L. Bozga, Y. Lakhnech, and M. Périn. Abstract interpretation for secrecy using patterns. Technical report, EVA : http://www-eva.imag.fr/, 2002. 1

[5] E.M. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *IFIP Working Conference on Programming Concepts and Methods*, 1998. 1

[6] H. Comon-Lundh. and V. Cortier. Security properties: Two agents are sufficient. Technical report, LSV, 2002. 1

[7] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *14th Int. Conf. Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, 2003. 1

[8] S. Delaune. Intruder deduction problem in presence of guessing attacks. Workshop on Security Protocols Verification (SPV'03), co-located with the 14th International Conference on Concurrency Theory (CONCUR'03), 2003. 5

[9] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. 1, 2

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

[11] Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *International Workshop on Formal Methods for Parallel Programming: Theory and Applications*, volume 1800 of *LNCS*, 2000. 1

[12] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2004. 1

[13] Claude Kirchner and Hélène Kirchner. Rewriting, solving, proving. A preliminary version of a book available at http://www.loria.fr/~ckirchne/rsp.ps.gz, 1999. 5

[14] Gavin Lowe. Analysing protocols subject to guessing attacks. In *Proc. of the Workshop on Issues in the Theory of Security (WITS'02)*, 2002. 5

[15] L. Mazaré. Using unification for opacity properties. In *Proc. of the Workshop on Issues in the Theory of Security (WITS'04)*. To appear, 2004. 1, 4, 7

[16] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *ESORICS*, pages 198–218, 1996. 3.2