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

Using Unification For Opacity Properties

Laurent Mazaré

Report n^o 24

October 28, 2004

Using Unification For Opacity Properties

Laurent Mazaré

October 28, 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 give a simple definition of opacity by looking at the possible traces of the protocol. Our approach draws on a new property over messages called similarity. Then, using rewriting methods close to those used in unification, we demonstrate the decidability of our opacity property. This is only achieved in the case of atomic keys using a method called Key Quantification.

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

Reviewers:

Notes: A conference version has been published in Proc. of the Workshop on Issues in the Theory of Security (WITS'04) 2004

How to cite this report:

@techreport { verimag-TR-2004-24,
title = { Using Unification For Opacity Properties},
authors = { Laurent Mazaré},
institution = { Verimag Technical Report },
number = {24},
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 [3], [5], [4] or [7]). These methods have lead to concrete tools for verifying secrecy such as these proposed by the EVA project [2]. However, checking security protocols requires studying other properties such as anonymity or opacity : hiding a piece of information from an intruder. For instance, in a vote 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 information that were leaked and that the intruder could 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.

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 could not 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 [6] : 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. We will then use rewriting techniques to find the set of solutions for such constraints. The rewriting rules are mainly inspired by the rules used in the unification algorithm. The problem is very similar to unification except that atomic constraints make use of similarity instead of syntactic equality. That is why, the rewriting rules are similar but not exactly the same. This technique will give the same result as for unification : we will be able to express the set of solutions for any constraint. As the constraint satisfiability is exactly related to opacity, this gives the main result of this paper : decidability of the opacity property in our case. With some hypothesis (passive intruder, atomic keys), the opacity of a given property is decidable and there is an immediate algorithm to perform this checking.

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. Section 4 formalizes the opacity hypothesis and translate the opacity property to a constraint. Then, it provides the method to check satisfiability for such constraints. Eventually, section 5 shows the use of this technique on a simple example, and section 6 concludes this paper.

2 Cryptographic Protocols

Let $Atoms$ and X be two infinite countable disjoint sets. $Atoms$ is the set of atomic messages a. X is a set of variables called "protocols variables" x.

Definition 1 (Message) *Let Σ be the signature $Atoms \cup \{pair, encrypt\}$ where $pair$ and $encrypt$ are two binary functions. The atomic messages are supposed 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 :

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

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

The substitutions σ from X to $T(\Sigma, X)$ are defined as usual. Its application to the message m will be noted $m\sigma$. If σ is defined by $x\sigma = n$ and $y\sigma = y$ for any other variables y, then we could write $m[x\backslash n]$ instead of $m\sigma$. The set of variables used in a message m is called $var(m)$.

Definition 2 (Protocol) *Let $Actors$ be a finite set of participants called* actors*. The set of* programs $Prog$ *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.

For the following, the set of actors is fixed to $Actors$. Let $free(P)$ be the set of free variables in the protocol P. The function $free$ will easily be defined over programs by induction and could be extended over protocols. An instance of the protocol P is a protocol $P\sigma$ where σ instantiates exactly the variables in 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. Such protocols are *closed*, i.e. $free(P\sigma) = \emptyset$.

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 smallest unifier of m and 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 could be blocked.

- *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,*

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

A protocol *terminates* iff for any Q such that $P \to^* Q$, it is possible to reach the state $\epsilon : Q \to^* \epsilon$. 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, the protocols will always be supposed deterministic.

This paper will make an extensive use of the Dolev-Yao theory. 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 the Dolev-Yao's inferences.

3 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 could discompose them, according to his knowledge but if he does not know the key k for example, he won't 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.

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 binary relation satisfying :*

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

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

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

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

Intuitively, this means that an intruder with the knowledge env will not be able to differentiate two similar messages. The environment name will be omitted as soon as it is not relevant for the comprehension. The same thing will be done for Dolev-Yao theory, i.e. $env \vdash m$ will be noted m. Moreover, the definition of $\sim$ could easily be extended to non-closed environments and messages by adding this inference :

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

Property 1 *The binary relation $\sim$ is an equivalence relation : for every messages m_1, m_2 and m_3 :*

$$m_1 \sim m_1$$

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

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

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 will hold for the rest of the document.

Property 2 (Context) *For every messages m_1, m_2, m_3 and m_4, 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

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

An important problem with similarity is : knowing an environment env and a closed message m, is it possible to find a closed message n such that

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

For that purpose, the $fresh$ function will be introduced. It is inductively defined over messages by the following lines where all the variables y have to be instantiated with different fresh variables (i.e. variables

that do not occur in env, m or n). We then call $Keys^+$ the set of keys such that $env \vdash Keys^+$ and $Keys^-$ the set of keys such that $\neg env \vdash Keys^-$.

$$\begin{aligned} fresh(a) &= a \\ fresh(x) &= x \\ fresh(\langle m, m' \rangle) &= \langle fresh(m), fresh(m') \rangle \\ fresh(\{m\}_k) &= \{fresh(m)\}_k \text{ if } k \in Keys^+ \\ fresh(\{m\}_k) &= \{y\}_k \text{ if } k \in Keys^- \end{aligned}$$

Property 3 *For every substitution σ, we have*

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

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

4 Predicates Using Similarity and Dolev-Yao Theory

We will use classical predicates over messages using the binary relations $=$ (syntactic equality) and $\sim$ (similarity), and the atomic formulae $E \vdash m$ where m is a message and E a set of messages. The set of these predicates will be called $Pred$. Satisfiability over $Pred$ is defined as usual. The set of models satisfying $E \vdash m$ is the set of substitutions σ such that $E\sigma$ and $m\sigma$ are closed and $E\sigma \vdash m\sigma : m\sigma$ is deducible from $E\sigma$ using Dolev-Yao theory. Models for $\sim$ and $=$ are defined in the same way. If a substitution σ is a model for a predicate P, we will write $\sigma \models P$. If all the atomic formulas in P use the same environment E, then E could be omitted in the predicate P but we will note $\sigma \models_E P$.

4.1 The Opacity Problem

Let us consider a protocol P and a session σ. The opacity problem considered here needs some hypothesis :

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

The session σ defines a witness run $run(P\sigma) = m_1.m_2...m_n$. A property ψ will be said *opaque* for this session σ if it is impossible to tell according to the knowledge of C if ψ is true or false. This means that there exist two possible sessions σ_1 and σ_2 of the protocol giving messages similar to the witness messages 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 ψ.

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 p, q and m must have the same length n.

The environment used in the precedent 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.

Our property of opacity could also be used to check anonymity. For example, if we take a definition of anonymity closed to the one given in [10], we just have to add a "restricted view" for the intruder, i.e. the intruder only intercepts some of the exchanged messages. Then the opacity of the property "identity of such actor" will be similar to what is defined as anonymity.

4.2 A Decidable Fragment : Global Key Quantification

A *similarity conjunction* is a predicate of the form :

$$P = \bigwedge_{i=1}^{n} m_i \sim n_i$$

Namely, it is a conjunction of similarities. The set of such predicates will be called $Conj$. The purpose of this section is to show that satisfiability over $Conj$ is decidable. The decision algorithm will be based on rewriting rules inspired by these used in unification (see for example [9]). This will transform any conjunction to a solved form, and we will show that, for such forms, the set of solutions is computable. The idea, as in unification, is to reverse the inferences giving $\sim$. An intuitive rule for decoding message would be :

$$\{m_1\}_{k_1} \sim \{m_2\}_{k_2} \hookrightarrow \left(k_1 \sim k_2 \wedge m_1 \sim m_2 \wedge k_1\right) \vee \left(\neg k_1 \wedge \neg k_2\right)$$

The messages $\{m_1\}_{k_1}$ and $\{m_2\}_{k_2}$ are similar in two cases : if none of the keys are compromised or if the keys are equals and the encoded messages are similar.

However, using only unification-like rules will not work. The main difference is that a predicate like $x \sim \{x\}_k$ has some solutions if the key k is not deducible by the intruder. For example, $x = \{a\}_k$ satisfies the former predicate. The usual "occur check" rule could not apply directly, so we will have to use a method called *key quantification*.

The idea of key quantification lies upon the fact that the set $Keys$ of keys occuring in the protocol is finite. That is why we will make tries for every possible partition $Keys^+ \cup Keys^-$ of $Keys$ with the following hypothesis : if σ is a solution, then for every k^+ in $Keys^+$ and k^- in $Keys^-$, we have $env\sigma \vdash k^+\sigma$ and $\neg env\sigma \vdash k^-\sigma$. So we quantify over the set $Keys^+$ of compromised keys. Furthermore, if the intruder knows initially some of the keys $Keys_0^+$, we will only quantify for $Keys^+ \supseteq Keys_0^+$. After choosing $Keys^+$, the first step is to substitute the conjunction by :

$$\bigwedge_{i=1}^{n} fresh(m_i) \sim fresh(n_i)$$

Let us call $freshVar(m_i)$ the set of fresh variables used to compute $fresh(m_i)$. By extension, let us define $freshVar(P)$ by :

$$freshVar(P) = \bigcup_{i=1}^{n} (freshVar(m_i) \cup freshVar(n_i))$$

The rewriting system R over $Conj$ is defined by the following rules.

- Variable Resolution (Res) : if the variable x occurs in C and not in m,

$$x \sim m \wedge C \hookrightarrow x \sim m \wedge C[x \backslash m]$$

If m is the variable x,

$$x \sim x \wedge C \hookrightarrow C$$

Else, if the variable x occurs in m,

$$x \sim m \wedge C \hookrightarrow \bot$$

- Pair Decomposition (Pair) :

$$\langle m_1, m_2 \rangle \sim \langle n_1, n_2 \rangle \wedge C \hookrightarrow m_1 \sim n_1 \wedge m_2 \sim n_2 \wedge C$$

- Axiom (Ax) : for a and b two distinct atoms

$$a \sim a \wedge C \hookrightarrow C$$

$$a \sim b \wedge C \hookrightarrow \bot$$

- Type Mismatch (Type) :

$$\langle m_1, m_2 \rangle \sim \{m\}_k \wedge C \hookrightarrow \bot$$
$$a \sim \langle m_1, m_2 \rangle \wedge C \hookrightarrow \bot$$
$$a \sim \{m\}_k \wedge C \hookrightarrow \bot$$

- Code Decomposition (Code) : if k_1 and k_2 are in $Keys^+$,

$$\{m_1\}_{k_1} \sim \{m_2\}_{k_2} \wedge C \hookrightarrow C \wedge k_1 \sim k_2 \wedge m_1 \sim m_2$$

If k_1 and k_2 are in $Keys^-$,

$$\{m_1\}_{k_1} \sim \{m_2\}_{k_2} \wedge C \hookrightarrow C$$

Else,

$$\{m_1\}_{k_1} \sim \{m_2\}_{k_2} \wedge C \hookrightarrow \bot$$

Definition 6 (Solved Variable/Form) *A variable x from X is* solved *in a predicate P iff x appears exactly in one similarity of P and this similarity has the form $x \sim m$ where m is a message that does not contain x.*

A solved form *is an element P of the set $Conj$ of the form*

$$P = \bigwedge_{i=1}^{n} x_i \sim m_i$$

Where for every i, the variable x_i is solved.

Note that, in a solved form, some of the free variables could be unsolved. This will be the case, in particular, for our fresh variables.

Theorem 1 *The rewriting system R terminates and the normal forms are solved forms and $\bot$. Moreover, R is correct and complete, i.e. the solutions of a predicate are exactly the solutions of its normal forms.*

Proof 1 *To prove the termination of the rewriting system, we will use the lexicographic order $(sf, sp, sc, np)_{lex}$ where sf is the number of non-solved variables, sp is the number of pair used in the predicate, sc is the number of encryptions and np is the number of atomic formulas. Then the values decreases strictly during rewriting as shown in the following array. This proves the termination of the rewriting system.*

Rule	*sf*	*sp*	*sc*	*np*
Res 1	$<$			
Res 2	$\leq$	$\leq$	$\leq$	$<$
Res 3	$<$			
Pair	$\leq$	$<$		
Ax 1 2	$\leq$	$\leq$	$\leq$	$<$
Type 1 2 3	$\leq$	$\leq$	$<$	
Code 1 2 3	$\leq$	$\leq$	$<$	

For the completeness and correction of rewriting, we have to prove for each rule $P_1 \hookrightarrow P_2$ that P_1 and P_2 have exactly the same sets of solutions.

∎

The predicate is equivalent to a solved form :

$$\bigwedge_{i=1}^{n} x_i \sim m_i$$

We could now describe the set Σ of possible substitutions σ satisfying our predicate P by : for any variable x that is not solved, $x\sigma$ ranges over all the possible messages. These variables includes in particular some of the fresh variables included in the sets $freshVar(m_i)$. For the solved variables x_i,

$$x_i\sigma = m_i\sigma$$

As m_i could only contain unsolved variables, the former definition is not recursive.

At last, we have to check that the hypothesis we made over $Keys^+$ and $Keys^-$ is correct, i.e. there exists a σ among Σ such that $\neg env\sigma \vdash Keys^-\sigma$. We will not consider the hypothesis over $Keys^+$ as soon as the keys of $Keys^+\sigma$ could be considered as part of the initial knowledge of the intruder. To check that $Keys^-$ is not deducible, it is possible to try with the worst solution (for Dolev-Yao theory), i.e. use the same fresh atom a for all the $x\sigma$ where x is not a key and use different fresh atoms for keys.

Property 5 *The σ defined above is the worst according to Dolev-Yao theory, formally for every couple of messages m and n,*

$$\big(\exists \eta \in \Sigma, env\eta \vdash m\eta \sim n\eta\big) \Rightarrow env\sigma \vdash m\sigma \sim n\sigma$$

$$\big(\exists \eta \in \Sigma, env\eta \vdash m\eta\big) \Rightarrow env\sigma \vdash m\sigma$$

Proof 2 *We suppose that there exists $\eta \in \Sigma$ such that $env\eta \vdash m\eta \sim n\eta$. By induction on the proof's structure of $env\eta \vdash m\eta \sim n\eta$, the property is easy to prove using the following lemma : for every key variable k :*

$$env\eta \vdash k\eta \Leftarrow env\sigma \vdash k\sigma$$

As $k\sigma$ is atomic and keys are atomic, $k\sigma$ could be obtained from $env\sigma$ using only decomposition rules. Then, given the nature of σ, we have that $env \vdash k$ which proves that $env\eta \vdash k\eta$. ∎

To finish our check, we just have to prove $\neg env\sigma \vdash Keys^-$ for our "worst" σ. This last check is of course decidable. To conclude this section, let us recall the main steps of our decision algorithm :

- Write the opacity property as two constraints. Process these constraints one after the other.
- Choose a set $Keys^+$ included in K (finite number of possibilities).
- Rewrite the constraints using the given rules.
- Check that the set $Keys \setminus Keys^+$ could not be inferred by the intruder using the worst solution.

If for the two constraints, there exists a set $Keys^+$ such that the worst solution is valid, then the studied property is opaque. Otherwise, the property is not opaque.

5 Example : The Limited Cryptographs Dinner

To give a simple application of opacity, the example of the cryptographs dinner will be taken. In this example, only two cryptographs will be present : Alice (A) and Bob (B). They have dinner together in a restaurant. When comes the time to pay, the waiter tells them that someone already paid the bill. A and B want to know if the person who paid is one of them or not, but if this is the case, they also want that name to remain anonymous. They suppose that an intruder Charlie (C) could listen to whatever they say. They decide to flip two coins (C can't see the result), if both are head or both are tail, A have to tell 1 if he didn't pay, 0 else, same thing for B. If the coins gives two different results, then A and B act in the opposite way. Obviously A and B could know with that protocol who paid the dinner. C could also know if A or B paid. Now we want to check that C cannot deduce who paid.

Let us formalize this protocol. They will toss two coins p_1 and p_2 with result the booleans x_1 and x_2. The predicate x_A is true iff A paid, the predicate x_B is true iff B paid. The first step of the protocol is the distribution of x_1 and x_2 by a third actor S using a key k not deducible by C.

$$S \to A : \{\langle x_1, x_2\rangle\}_k$$

$$S \to B : \{\langle x_1, x_2\rangle\}_k$$

The following of the protocol is detailed below with respect to the possible values for the different variables.

x_A	0	0	0	0	**0**	0	0	0	1	1	**1**	1	1	1	1	1
x_B	0	0	0	0	**1**	1	1	1	0	0	**0**	0	1	1	1	1
x_1	0	0	1	1	**0**	0	1	1	0	0	**1**	1	0	0	1	1
x_2	0	1	0	1	**0**	1	0	1	0	1	**0**	1	0	1	0	1
$A \to B$	1	0	0	1	**1**	0	0	1	0	1	**1**	0	impossible			
$B \to A$	1	0	0	1	**0**	1	1	0	1	0	**0**	1	impossible			

Let us suppose the trace of the protocol is :

$$S \to A, B : \{\langle 0, 1\rangle\}_k$$

$$A \to B : 1$$

$$B \to A : 0$$

So one of A and B paid the dinner. Intuitively, we could conclude immediately. Let us consider the two bold columns, they propose the right execution trace but in one case A paid, in the other one, it is B. The identity of the payer remains anonymous. The opacity property is the following with the environment $\{0, 1\}$.

$$\exists\sigma_1, \big(\{\langle x_1, x_2\rangle\}_k \sim \{\langle 0, 1\rangle\}_k \wedge A \to B = 1 \wedge B \to A = 0 \wedge x_A\big)\sigma_1$$

$$\exists\sigma_2, \big(\langle x_1, x_2\rangle\}_k \sim \{\langle 0, 1\rangle\}_k \wedge A \to B = 1 \wedge B \to A = 0 \wedge \neg x_A\big)\sigma_2$$

Let suppose that k is in $Keys^-$ (otherwise, the property is not opaque). By developing, we obtain that the possible sessions for that trace are : $[x_A\backslash 0, x_B\backslash 1, x_1\backslash 0, x_2\backslash 0]$, $[x_A\backslash 0, x_B\backslash 1, x_1\backslash 1, x_2\backslash 1]$, $[x_A\backslash 1, x_B\backslash 0, x_1\backslash 0, x_2\backslash 1]$ or $[x_A\backslash 1, x_B\backslash 0, x_1\backslash 1, x_2\backslash 0]$. And we could take for example :

$$\sigma_1 = [x_A\backslash 0, x_B\backslash 1, x_1\backslash 0, x_2\backslash 0]$$

$$\sigma_2 = [x_A\backslash 1, x_B\backslash 0, x_1\backslash 1, x_2\backslash 0]$$

This proves the anonymity of the payer for this trace.

6 Conclusion

In this paper, we presented a new simple and intuitive definition for opacity. With that definition, the opacity of a given property is decidable. The decision algorithm has been implemented and tested in some example cases. As far as we know, other versions of opacity ([1], [8]) have been given in the literature but none of these criterion were implemented. This work has some limitation, in particular, the hypothesis made over the session between A and B : only atomic keys are used and public key cryptography is not allowed. This gives some natural extension to this paper that will be explored later. For example, using tree automata techniques should allow the use of non-atomic keys. Another interesting extension would be to make the intruder active. If C could intercept and modify the messages, could he find the right messages to alter such that the property is not opaque any more ? Another interesting extension would be to add syntactic equality to constraints : this equality means that the intruder has receive two exactly identical messages. With that knowledge, the intruder could make new deductions.

References

[1] A. Boisseau. *Abstractions pour la vérification de propriétés de sécurité de protocoles cryptographiques*. PhD thesis, Laboratoire Spécification et Vérification (LSV), ENS de Cachan, 2003. 6

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

[3] 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

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

[5] 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

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

[7] 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

[8] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2004. 6

[9] 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. 4.2

[10] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *ESORICS*, pages 198–218, 1996. 4.1