A Theory of Dictionary Attacks and its Complexity *

Stéphanie Delaune
France Télécom R&D
LSV, CNRS UMR 8643, ENS de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex, France
stephanie.delaune@lsv.ens-cachan.fr

Florent Jacquemard
INRIA, Research Unit Futurs, project SECSI
LSV, CNRS UMR 8643, ENS de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex, France
florent.jacquemard@lsv.ens-cachan.fr

Abstract

We consider the problem of automating proofs of cryptographic protocols when some data, like poorly chosen passwords, can be guessed by dictionary attacks. First, we define a theory of these attacks: we introduce an inference system modeling the guessing capabilities of an intruder. This system extends the classical Dolev–Yao rules. Using proof rewriting techniques, we show a locality lemma for our inference system which yields the PTIME–completeness of the deduction problem.

This result is lifted to the simultaneous solving of intruder deduction constraints with variables. Constraint solving is the basis of a NP algorithm for the protocol insecurity problem in the presence of dictionary attacks, assuming a bounded number of sessions. This extends the classical NP–completeness result for the Dolev–Yao model.

We illustrate the procedure with examples of published protocols. The model and decision algorithm have been validated on some examples in a prototype implementation.

1. Introduction

While the automatic verification of cryptographic protocols is undecidable, even with several restrictions, it has aroused a lot of interest during the last years. The undecidability of this problem results from several factors: the ability of agents to generate fresh random data (nonces), the unlimited size of terms, the unboundedness of the number of sessions. Removing the last condition is sufficient for decidability (while removing the others is not, see [12, 7, 2]) and several procedures have been proposed to decide the protocol insecurity problem with a bounded number of sessions [3, 20] and [21] where the problem is shown NP–complete.

In the works cited above, as well as in most approaches concerning automated verification of security protocols, the underlying cryptographic primitives are based on the so called Dolev–Yao model [13]. In this model, a malicious agent called *intruder* is assumed to have a complete control over the communication network: he is able to eavesdrop and replay messages, impersonate honest agents, generate nonces... These abilities still comply to the *perfect cryptography assumption*, which states that there is no way to obtain knowledge about an encrypted plaintext without knowing the decryption key, and no way to obtain the key from ciphertexts. This abstraction happened to be accurate enough to reveal many logical attacks on known cryptographic protocols in an automated way. However, it may be too strong to capture some specific attacks that may occur in real word situations. For instance, in the Dolev–Yao model it is not possible to take into account attacks based on algebraic properties of cryptographic operators, like exclusive or, in presence of which the protocol insecurity problem with a bounded number of sessions is still decidable [8, 6].

In this paper, we formalize another interesting attack technique which appears to be out of the scope of the Dolev–Yao model: the so-called *dictionary attacks* [15, 18]. In some situations, an intruder is able to guess poorly chosen passwords (or other data belonging to a reasonably small domain) by an offline brute force iteration through a *dictionary*, using messages previously collected on the network to verify his guess at each step. The reason for this interest is simple: password-guessing attacks are a common avenue for breaking into systems, and the application of formal method to analyze password protocols can help.

There are number of facets to dictionary attacks. At the beginning of the nineties, several examples of dictionary attacks have been analyzed, and some countermeasures have been proposed to design protocols resistant to this kind of attacks [15, 23, 5]. Lot of effort seems to have been put into

* This work has been partly supported by the RNTL project PROUVÉ 03V360 and the ACI-SI Rossignol.

using *provable security* [14] for analysis such protocols that use poorly–chosen data [24, 4, 17]. However, the application of automatic verification methods, which has been used for cryptographic protocol analysis, was not used for password protocols. Perhaps, this is due to the complex nature of dictionary attacks, whose analysis involves complications similar to combining cryptanalytical and abstract protocol analysis. Only recently, some procedures have been implemented to automatically find dictionary attacks [18, 9]. However, neither the complexity or the completeness of the procedures, nor the decidability of the problem have been studied in these works.

In this paper, we propose a formal definition of an inference system modeling an intruder which extends the Dolev–Yao model with guessing abilities. We show, by a locality result *à la* [19] for this system, that the corresponding *intruder deduction problem* (whether the intruder is able to deduce a given message from a given set of messages received) is decidable in polynomial time. With a lifting of this decision result to the simultaneous solving of intruder deduction constraints with variables, we propose a non–deterministic polynomial procedure to solve the problem of protocol insecurity in presence of dictionary attacks for a bounded number of sessions. Though this complexity is the same as in the Dolev–Yao model, see [21], the proofs of our decision procedure are made dramatically harder by the introduction of guessing abilities in the inference system. Indeed, some basic results easy to prove in the standard Dolev-Yao model are not anymore in our extended model. The accuracy of our model has been confronted to many known examples of protocols using a prototype based on an efficient approximation of our procedure. A comparison with other models such as CSP [18] and the spi–calculus [1] can be found at the end of the paper.

After some motivating examples of dictionary attacks (Section 2) and preliminary definitions of protocols syntax and semantics (Section 3) we define in Section 4 our extended intruder model, and formalize in particular the brute force procedure mentioned above. We then prove a locality theorem from which it follows that the ground intruder deduction problem can be decided in polynomial time. After the formal definition of dictionary attacks (Section 5), in term of solutions of symbolic constraint systems (where each individual constraint is a lifting of the ground intruder deduction problem) we give in Section 6 a non–deterministic polynomial time procedure to decide their existence.

2. Examples

A simple subcase of dictionary attacks is the *known–plaintext* attacks, where an intruder intercepts a message $\{M\}_K$ encrypted with a weak password K and whose encrypted content M is known (for instance an instruction like `hello`). The intruder can then try to decrypt this ciphertext with each word in a dictionary one by one, and verify for each guess d whether the value obtained is the known plaintext M, which means with a high probability that $d = K$. This method also works against a challenge–response scheme where a server sends to a user A a nonce N as a challenge and A responds with $\{N\}_K$, where K is its a weak password.

The examples below show that similar attacks are also possible in some cases where the plaintext is not known, with more subtle techniques to verify the guesses.

2.1. Naive Vote Protocol

Consider the following naive vote protocol:

$$0. \quad A \quad \rightarrow \quad S: \quad \{V\}_{pub(S)}$$

The voter A encrypts his vote V with the public key $pub(S)$ of the vote server S. The server decrypts the message with his private key $priv(S)$ and registers the vote. The security requirement is that, only A and S know V. This protocol is secure in the standard Dolev–Yao model, because an intruder who intercepts the message $\{V\}_{pub(S)}$ will not be able to learn the value of the vote V as long as he does not know $priv(S)$. However, if we assume that the intruder knows a finite set $\mathcal{D}$ (reasonably small) of values that V can take, then he can deduce V without knowing $priv(S)$: for each value $d \in \mathcal{D}$, he encrypts d with $pub(S)$ and verifies whether the ciphertext $\{d\}_{pub(S)}$ obtained is equal to $\{V\}_{pub(S)}$, which means that the guess $d = V$.

2.2. Handshake Protocol

Consider this challenge–response transaction which is commonly used in authentication protocols (see [15]):

$$\begin{array}{llcll} 0. & A & \rightarrow & B: & \{N\}_{pw(A,B)} \\ 1. & B & \rightarrow & A: & \{N+1\}_{pw(A,B)} \end{array}$$

A generates a random number (*nonce*) N and sends it to B encrypted with $pw(A, B)$, B decrypts the message, computes $N + 1$, and returns to A the encrypted result. The cryptosystem is symmetric. In the standard Dolev–Yao model, an intruder who intercepts the messages cannot deduce $pw(A, B)$, and the incrementation of N in the second message prevents replay attacks. However, if $pw(A, B)$ is a poorly chosen password, and belongs to a finite dictionary $\mathcal{D}$, then the challenge–response transaction can be attacked in other ways: the intruder guesses $d \in \mathcal{D}$ and tries to decrypt both messages 0 and 1 with d. He obtains two values, v_0 and v_1 respectively. If $v_1 = v_0 + 1$, then the attacker has guessed the correct value $d = K$.

This attack is called a dictionary attack and has been used to exploit systems in the past, often quite successfully.

2.3. Enhanced Kerberos Protocol

As outlined in [15], the Kerberos protocol [22] contains some messages which make it vulnerable to known–plaintext attacks. To avoid this problem, Gong et al. [15] propose the following modification, which will be used as a running example.

$$
\begin{array}{llll}
0. & A \rightarrow S: & \{A, B, N_1, N_2, Ca, \{Ta\}_{pw(A,S)}\}_{pub(S)} \\
1. & S \rightarrow A: & \{N_1, K \oplus N_2\}_{pw(A,S)}, \{A, K, Ts\}_{pw(B,S)} \\
2. & A \rightarrow B: & \{A, K, Ts\}_{pw(B,S)}
\end{array}
$$

In this protocol, the user A obtains from S a secret key K to be shared between himself and the ticket–granting service B. Afterward, A can obtain tickets for other services from B using this key K. The symbol $\oplus$ denotes the bit–wise exclusive–or operation. We do not consider any algebraic properties of this operation here and rather see it as an encryption: $K \oplus N$ is equivalent to $\{K\}_N$. $pub(S)$ is the public–key of the server S, and $pw(A, S)$, $pw(B, S)$ are symmetric keys (passwords) that A and B respectively share with S.

The password $pw(B, S)$ can be assumed to be well–chosen since B is a server, but the password $pw(A, S)$ of the user A is likely to come from a dictionary. This protocol implements some protections against dictionary attacks on $pw(A, S)$, using the nonces N_1, N_2, the confounder Ca (which is a long nonce whose role is to confound attacks like in Section 2.2), and the timestamps Ta and Ts, added in order to prevent the replay of messages 0 and 1. We refer the reader to [15] for the details about this protocol.

As described in [23, 15, 18] for similar protocols, if the server S does not record the timestamps Ta, and if moreover the clocks of S and A are not well synchronized, an intruder can replay a copy of an eavesdropped message 0 within the clock skew, making possible the attack described below.

$$
\begin{array}{lll}
\alpha.0. & A \rightarrow S: & \{A, B, N_1, N_2, Ca, \{Ta\}_{pw(A,S)}\}_{pub(S)} \\
\alpha.1. & S \rightarrow A: & \{N_1, K \oplus N_2\}_{pw(A,S)}, \\
 & & \{A, K, Ts\}_{pw(B,S)} \\
\\
\beta.0. & I(A) \rightarrow S: & \{A, B, N_1, N_2, Ca, \{Ta\}_{pw(A,S)}\}_{pub(S)} \\
\beta.1. & S \rightarrow I(A): & \{N_1, K' \oplus N_2\}_{pw(A,S)}, \\
 & & \{A, K', Ts'\}_{pw(B,S)}
\end{array}
$$

I guesses $pw(A, S)$ offline

$$
\begin{array}{lll}
\gamma.0. & I(A) \rightarrow S: & \{A, B, M_1, M_2, Ci, \{Ti\}_{pw(A,S)}\}_{pub(S)} \\
\gamma.1. & S \rightarrow I(A): & \{M_1, K'' \oplus M_2\}_{pw(A,S)}, \\
 & & \{A, K'', Ts''\}_{pw(B,S)} \\
\gamma.2. & I(A) \rightarrow B: & \{A, K'', Ts''\}_{pw(B,S)}
\end{array}
$$

The notation $I(A)$ represents: on the left of an arrow, the intruder I masquerading A to send a message, and on the right of an arrow, the intruder intercepting a message intended for A.

In the session β, the intruder replays A's message $\alpha.0$, so as to get the server to issue (in $\beta.1$) another message using the same nonces N_1 and N_2. Hence, N_1 can be used as a verifier to guess $pw(A, S)$: the intruder can decrypt $\{N_1, K \oplus N_2\}_{pw(A,S)}$ and $\{N_1, K' \oplus N_2\}_{pw(A,S)}$ with a value d in a dictionary, and if the first field of the two values obtained is the same, then it means that the value guessed $d = pw(A, S)$. After that, the intruder can impersonate A in session γ, with chosen nonces M_1 and M_2, and obtains in $\gamma.1$ the session key K'' which is assumed to be a secret shared between A, S and B.

3. Protocols

We assume given a signature $\mathcal{F}$ containing the symbols $\langle \_, \_ \rangle$ (pairing), $\{\_\}_\_$ (encryption), some unary function symbols representing invertible functions, others representing one way functions, and 3 other special constructors $pw(\_)$ (for symmetric keys, or passwords shared between agents[1]), $pub(\_)$ (for asymmetric public keys of agents) and $priv(\_)$ (for the corresponding private keys). The signature $\mathcal{F}$ contains also an arbitrary subset $\mathcal{F}_0$ of constant symbols representing objects like keys, agent names, nonces... We also assume given an infinite set of variables $\mathcal{X}$.

The set of terms built with $\mathcal{F}$ and $\mathcal{X}$ is denoted $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and the subset of ground terms (terms without variables) $\mathcal{T}(\mathcal{F})$. We denote $vars(t)$ the set of variables occurring in a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $st(t)$ the set of subterms of t. These two notations are extended as expected to a structure T containing some terms: $vars(T)$ (resp. $st(T)$) is the union of the sets $vars(t)$ (resp. $st(t)$) for every term t contained in T.

Among the terms of $\mathcal{T}(\mathcal{F})$, we shall distinguish a restricted subset $\mathcal{G} \subseteq \mathcal{F}_0 \cup \{pw(t), pub(t), priv(t) | t \in \mathcal{T}(\mathcal{F})\}$ of *guessable* values, *i.e.* values which are known to belong to a finite dictionary. Moreover, we assume a bijective mapping denoted $\_^{-1}$ from $\mathcal{T}(\mathcal{F}, \mathcal{X})$ into $\mathcal{T}(\mathcal{F}, \mathcal{X})$ which associates to a public key the corresponding private key and reciprocally. More precisely, if $k \in \mathcal{F}_0$ represents an asymmetric key, public or private, then $k^{-1} \in \mathcal{F}_0$ represents its private (resp. public) counterpart. For every $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we have $pub(t)^{-1} = priv(t)$ and $priv(t)^{-1} = pub(t)$, and for every other $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ (which does not represent a public or private key), we have $s^{-1} = s$.

In the paper, $|S|$ denotes the cardinal of the set S. The *size* $\|t\|$ of a term t is the number of nodes in t. This notation is extended as expected to a set of terms $\|T\|$. The *dag-size* $\|T\|_d$ of a term container T is the number of distinct subterms of T (*i.e.* the number of nodes in a representation of T as a dag with maximal sharing).

---

1 We sometimes write $pw(t_1, t_2)$ for $pw(\langle t_1, t_2 \rangle)$.

A *replacement* is the term morphism extension of a finite mapping $\{s_1 \mapsto t_1, \ldots, s_n \mapsto t_n\}$ where $s_1, \ldots, s_n, t_1, \ldots t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If $t_1, \ldots t_n \in \mathcal{T}(\mathcal{F})$, the replacement is called *ground*. A *substitution* is a replacement which domain is a subset of $\mathcal{X}$. As usual, the application of a replacement σ to a term t and the composition of replacements σ_1 by σ_2 are written in postfix notation, respectively $t\sigma$ and $\sigma_1\sigma_2$. A substitution σ is *grounding for t* is $t\sigma \in \mathcal{T}(\mathcal{F})$.

Definition 1 *A* protocol *is a finite set of programs, each program being a finite sequence of pairs of* instructions *of the form* $\mathsf{recv}(r); \mathsf{send}(s)$ *with* $r, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

It is equivalent to consider programs which are sequences of send and recv in an arbitrary order, since we may add some instructions $\mathsf{send}(0)$ and $\mathsf{recv}(0)$ where $0 \in \mathcal{F}_0 \setminus \mathcal{G}$ is a special constant known to everyone.

Example 1 *The Kerberos protocol variant described in Section 2.3 is made of three programs:*

role A

0. $\mathsf{recv}(0);$
$\mathsf{send}\big(\{x_A^0, x_B^0, x_{N_1}^0, x_{N_2}^0, x_{Ca}^0, \{x_{Ta}^0\}_{x^0_{pw(A,S)}}\}_{x^0_{pub(S)}}\big)$
1. $\mathsf{recv}\big(\{x_{N_1}^0, x_K^0 \oplus x_{N_2}^0\}_{x^0_{pw(A,S)}}, x^0\big);\ \mathsf{send}(x^0)$

role S

0. $\mathsf{recv}\big(\{x_A^1, x_B^1, x_{N_1}^1, x_{N_2}^1, x_{Ca}^1, \{x_{Ta}^1\}_{pw(x_A^1, x_S^1)}\}_{x^1_{pub(S)}}\big);$
$\mathsf{send}\big(\{x_{N_1}^1, x_K^1 \oplus x_{N_2}^1\}_{pw(x_A^1, x_S^1)}, \{x_A^1, x_K^1, x_{Ts}^1\}_{pw(x_B^1, x_S^1)}\big)$

role B

0. $\mathsf{recv}\big(\{x_A^2, x_K^2, x_{Ts}^2\}_{x^2_{pw(B,S)}}\big);\ \mathsf{send}(0)$

The symbols $x_A^i \ldots$ *(*$i = 0, 1, 2$*) are all distinct variables of* $\mathcal{X}$*. Note that* A *receives (in step* 1*) the cipher* $\{A, K, Ts\}_{pw(B,S)}$ *as a value* x^0*, and forwards it blindly (to B), since he does not know B's password* $pw(B,S)$*. The program of role B implements only the reception of the last message by B.*

Definition 2 *A* process (p, σ) *is made of a program p and a ground substitution* σ *whose domain is a subset of* $vars(p)$.

Every program of a protocol defines a *role*, and a process (p, σ) is an honest *agent* playing the role p.

Let N be a set of ground terms called the network. We define small step semantics for the execution of processes. A process (p, σ) changes to (p', σ') by an execution step if the first element of the program p is $\mathsf{recv}(r); \mathsf{send}(s)$ and p' is the rest of the sequence p, the instruction $\mathsf{recv}(r)$ is executed properly, *i.e.* there exists a ground substitution θ such that $r\theta \in N$ and $\sigma' = \sigma\theta$, and $\mathsf{send}(s)$ is executed, by adding the term $s\sigma'$ to the network N. We assume that the protocol and the initial configuration are such that $s\sigma'$ is ground. It means that every participant is able to construct a term to be sent with the substitution in its initial process (its initial knowledge) or with the values received from other participants, as defined below.

Definition 3 *An initial configuration* $(p_0, \sigma_0), \ldots, (p_m, \sigma_m)$ *of a protocol* $\mathcal{P}$ *is called* runnable *iff* $p_0, \ldots p_m$ *are copies of programs of* $\mathcal{P}$ *whose variables sets are pairwise disjoint and for each* $i \leq m$ *such that the program* p_i *is the sequence* $(\mathsf{recv}(r_{i,j}); \mathsf{send}(s_{i,j}))_{j \leq n}$*, for each* $j \leq n$*, for each* $x \in vars(s_{i,j})$*,* x *is in the domain of* σ_i *or there exists* $k \leq j$ *such that* $x \in vars(r_{i,k})$.

Example 2 *The sequence of processes* $\big((p_0, \sigma_0), (p_1, \sigma_1), (p_2, \sigma_2)\big)$ *described below is a runnable initial configuration of the protocol of Example 1:*

$$\sigma_0 = \left\{ \begin{array}{l} x_A^0 \mapsto A, x_B^0 \mapsto B, x_S^0 \mapsto S, x_{Ca}^0 \mapsto Ca, \\ x_{N_1}^0 \mapsto N_1, x_{N_2}^0 \mapsto N_2, x_{Ta}^0 \mapsto Ta, \\ x_{pub(S)}^0 \mapsto pub(S), x_{pw(A,S)}^0 \mapsto pw(A,S), \end{array} \right\}$$

$$\sigma_1 = \left\{ \begin{array}{l} x_K^1 \mapsto K, x_{Ts}^1 \mapsto Ts, \\ x_S^1 \mapsto S, x_{pub(S)}^1 \mapsto pub(S) \end{array} \right\}$$

$\sigma_2 = \big\{x^2_{pw(B,S)} \mapsto pw(B,S)\big\}$*, where* A*,* B*,* S*,* N_1*,* N_2*,* Ca*,* Ta*,* K*,* Ts *are constants of* $\mathcal{F}_0$*.*

In this paper, we are interested in proving the confidentiality of some data sent by processes of a given runnable initial configuration, assuming the presence of an intruder who has strong control over the network N, as defined in the next section.

4. Intruder Model

We assume that the intruder has a complete control over the network: he systematically diverts messages, possibly modifies them and forwards them to the receiver under the identity of the official sender. In other words all the communications are mediated by a hostile environment represented by the intruder. The intruder actions for analyzing and modifying the messages are modeled below by inference rules.

4.1. Dolev–Yao Model

The most widely used deduction relation representing the control of the intruder on the network is known as the Dolev–Yao model [13] and is defined in Figure 1. The sequent $T \vdash u$ means that if the intruder knows the messages in $T \subseteq \mathcal{T}(\mathcal{F})$, then he can deduce the message $u \in \mathcal{T}(\mathcal{F})$. In this model, the intruder can form pairs and ciphertexts from known terms (rules P, E), decompose pairs, and decrypt ciphertexts only when he can deduce the decryption key (rules $\mathsf{UL}, \mathsf{UR}, \mathsf{D}$). The latter condition is known as the *perfect cryptography assumption*.

4.2. Extended Dolev–Yao Model

We shall now describe how dictionary attacks can be modeled by extending the intruder Dolev–Yao model. For

$$
\begin{array}{ll}
\textsf{Axiom} & \dfrac{u \in T}{T \vdash u}\ (\mathsf{A}) \\[2ex]
\textsf{Pairing} & \dfrac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v\rangle}\ (\mathsf{P}) \\[2ex]
\textsf{Encryption} & \dfrac{T \vdash u \quad T \vdash v}{T \vdash \{u\}_v}\ (\mathsf{E}) \\[2ex]
\textsf{Unpairing} & \dfrac{T \vdash \langle u, v\rangle}{T \vdash u}\ (\mathsf{UL}) \qquad \dfrac{T \vdash \langle u, v\rangle}{T \vdash v}\ (\mathsf{UR}) \\[2ex]
\textsf{Decryption} & \dfrac{T \vdash \{u\}_v \quad T \vdash v^{-1}}{T \vdash u}\ (\mathsf{D})
\end{array}
$$

Figure 1. The Dolev–Yao model.

$$
\begin{array}{ll}
\textsf{Axiom} & \dfrac{u \in T}{T/\emptyset \vdash' u}\ (\mathsf{A}') \qquad \textsf{Guess} \quad \dfrac{u \in \mathcal{G}}{T/u \vdash' u}\ (\mathsf{G}) \\[2ex]
\textsf{Pairing} & \dfrac{T/T'_1 \vdash' u \quad T/T'_2 \vdash' v}{T/T'_1 \cup T'_2 \vdash' \langle u, v\rangle}\ (\mathsf{P}') \\[2ex]
\textsf{Encryption} & \dfrac{T/T'_1 \vdash' u \quad T/T'_2 \vdash' v}{T/T'_1 \cup T'_2 \vdash' \{u\}_v}\ (\mathsf{E}') \\[2ex]
\textsf{Unpairing} & \dfrac{T/T' \vdash' \langle u, v\rangle}{T/T' \vdash' u}\ (\mathsf{UL}') \qquad \dfrac{T/T' \vdash' \langle u, v\rangle}{T/T' \vdash' v}\ (\mathsf{UR}') \\[2ex]
\textsf{Decryption} & \dfrac{T/T'_1 \vdash' \{u\}_v \quad T/T'_2 \vdash' v^{-1}}{T/T'_1 \cup T'_2 \vdash' u}\ (\mathsf{D}')
\end{array}
$$

Figure 2. The Dolev–Yao' model.

$$
\dfrac{
P_1 \left\{ \dfrac{\dfrac{\vdots}{\vdash' u_1} \ \ldots \ \dfrac{\vdots}{\vdash' u_{n_1}}}{T/T'_1 \vdash' u}\ (R_1) \right.
\qquad
P_2 \left\{ \dfrac{\dfrac{\vdots}{\vdash' v_1} \ \ldots \ \dfrac{\vdots}{\vdash' v_{n_2}}}{T/T'_2 \vdash' v}\ (R_2) \right.
}{T \vdash g}\ (\mathsf{C})
$$

where:

(i).	$g \in T'_1 \cup T'_2$
(ii).	P_1 and P_2 are normal DY'-proofs
(iii).	$R_1 \neq R_2$ or $\{u, u_1, .., u_{n_1}\} \neq \{v, v_1, ..., v_{n_2}\}$
(iv).	$R(u, v)$ where $R = Id \cup \{(k, k^{-1}) \mid$ k is a key$\}$

Figure 3. The rule Compare.

this purpose, we use the following definition of dictionary attacks from [18] which generalizes the definition of [15]:

A dictionary attack consists of the intruder guessing a value d, and then verifying it. The verification will be by the intruder using d to produce a value v, which we call the verifier and can take a number of different forms:

1. *the intruder knew v initially, (cf. Section 2.1)*
2. *the intruder produced v in two distinct ways from d, (cf. Section 2.2)*
3. *v is an asymmetric key, and the intruder knows v's inverse from somewhere.*

Intuitively, the intruder knows that $g \in \mathcal{G}$ belongs to a dictionary, in which he picks d. If the verifier v, built with d and the intruder's knowledge ensures one of the three conditions above, then the probability is high that $d = g$. We shall use a variant of the rules of Figure 1 in order to model the guessing of a d and the production of a verifier $v \in \mathcal{T}(F)$ by the intruder. In the rules of this variant, presented in Figure 2, we introduce a new form of sequent $T/T' \vdash' v$, which means that if the intruder knows the messages in $T \subseteq \mathcal{T}(\mathcal{F})$ and guessed values for the symbols of $T' \subseteq \mathcal{G}$, then he can build the verifier $v \in \mathcal{T}(\mathcal{F})$. In other words, he can deduce that v belongs to a finite set that he can compute. The members of T and T' are respectively called the *strong* and *weak hypotheses* of $T/T' \vdash' v$, and v is called its *target*.

Figure 3 introduces a deduction rule Compare which models the verification of a guess d following one of the 3 cases described above. The conditions (i) and (ii) ensure that one of the proofs P_1 or P_2 really uses the guessable value g and it is necessary to prevent certain false attacks (see Section 4.4). The normality condition (ii) will prohibit deduction steps that simply undo previous steps. It refers to the notion of *normal* DY'-proof which is formally defined in Section 4.3. Lastly, (iii) ensures that the two proofs P_1 and P_2 do not end with the same instance of the same rule. Hence, the two ways to obtain the verifier v are really distinct.

The inference system made of the rules displayed in Figures 1 and 2 and the rule Compare of Figure 3 is called the *Dolev–Yao extended model*.

4.3. Proofs

Definition 4 *A* DY-proof *(resp.* DY'-proof*) P of $T \vdash u$ (resp. $T/T' \vdash' u$) is a tree such that:*

- *every leaf of P is labeled with some $v \in T$ (resp. $v \in T \cup \mathcal{G}$),*
- *for every node n labeled with s with k sons labeled with $s_1, \ldots, s_k$, $(s_1, \ldots, s_k, s)$ is an instance of an inference rule R of Figure 1 (resp. Figure 2) which $s_1, \ldots, s_k$ are the premises and s the conclusion. We say that P contains the instance $(s_1, \ldots, s_k, s)$, or*

ends with this instance (or simply with rule R) if n is the root of P,

- *the root is labeled with some $T \vdash u$ (resp. $T/T' \vdash' u$).*

A guessing-proof *is a tree ending with an instance of the rule* Compare *and whose two sons are DY'-proofs which satisfy the conditions (i)–(iv) of Figure 3.*

Let P be a DY-proof (resp. DY'-proof). We say that the pair $\langle \gamma_1, \gamma_2 \rangle$ is *decomposed* in P if P contains an instance of the rule UL or UR (resp. UL' or UR') whose target of the premise is $\langle \gamma_1, \gamma_2 \rangle$. Similarly, the ciphertext $\{\gamma_1\}_{\gamma_2}$ is said decomposed in P if P contains an instance of the rule D (resp. D') whose targets of premises are $\{\gamma_1\}_{\gamma_2}$ and γ_2^{-1}.

Definition 5 *A DY-proof or DY'-proof P is called* minimal *if it does not contain two nodes on the same path labeled by sequents with the same target.*

Definition 6 *A DY'-proof P is called* normal *if the rewrite rules defined in Figure 4 can not be applied to P.*

Note that every minimal DY'-proof is normal. Indeed, if a DY'-proof P in not normal, then a rewrite rule can be applied to P which means that P contains two nodes on the same path labeled by sequents with the same target (but not necessarily with the same set of weak hypotheses). The normality of DY-proofs is defined with a similar set of rewrite rules, where D', E', P', UL',UR' are replaced respectively by D, E, P, UL,UR.

Lemma 1 *If there exists a DY-proof of $T \vdash u$, then there exists a minimal DY-proof of $T \vdash u$.*

Proof. We can show this result by induction on the proof P of $T \vdash u$. If P is reduced to an instance of the rule A, then it is obvious. Otherwise, by induction hypothesis the direct subproofs of P are minimal. So, if P is not minimal, there exists two nodes on the same path labeled by the sequent $T \vdash u$ and one of these is the root of P. So, we can consider the other which is the root of a minimal proof of $T \vdash u$. $\square$

However this result is not valid for DY'-proofs, unless the set of weak hypotheses is empty for every node. The problem with DY'-proofs is that we can not assume that two nodes on the same path and with the same target have the same set of weak hypotheses.

4.4. Rule Compare

In this section, we shall explain in more details the application conditions of the rule Compare.

Condition (ii) By the condition (i), one son P_1 or P_2 contains the guess g among the weak hypotheses, but it is not sufficient to ensure that P_1 or P_2 really depends on the guess g: only the condition (ii) ensures this property. Indeed, without this condition, there would be a guessing-proof with the two (non normal) sons below, which would mean that the intruder is able to guess any $g \in \mathcal{G}$ from any message m known by the intruder.

$$\dfrac{m \in T}{T/\emptyset \vdash' m}\,(\mathsf{A}') \qquad \dfrac{\dfrac{\dfrac{m \in T}{T/\emptyset \vdash' m}\,(\mathsf{A}') \quad \dfrac{g \in \mathcal{G}}{T/\{g\} \vdash' g}\,(\mathsf{G})}{T/\{g\} \vdash' \langle m, g \rangle}\,(\mathsf{P}')}{T/\{g\} \vdash' m}\,(\mathsf{UL}')$$

Condition (iii) The condition (iii) also prevents certain false attacks. Consider the program which is made up of one pair of instructions $\mathsf{recv}(x); \mathsf{send}(\langle x, x \rangle)$. An agent executing this program will answer to any message m with the pair $\langle m, m \rangle$. Thus, if the intruder knows the message $m = \{n\}_g$, he obtains from the agent the answer $\langle \{n\}_g, \{n\}_g \rangle$. The two DY'-proofs below verify the conditions (i), (ii) and (iv) of the rule Compare. However, the condition (iii) is not verified and these two proofs indeed represent a fake dictionary attack. Intuitively, in this attack, the intruder guesses a value d for g, computes the left and right projections ((UL') and (UR')) of m, tries to decrypt each projection with d (rule (D')) and compares the values obtained. But, since both projections are $\{n\}_g$, the values will always be equal, even when $d \neq g$. We see that the two proofs differ (the first step is left or right projection) but that their last instance is the same (decryption applied to the same premises).

$$\dfrac{\dfrac{\dfrac{\langle \{n\}_g, \{n\}_g \rangle \in T}{T/\emptyset \vdash' \langle \{n\}_g, \{n\}_g \rangle}\,(\mathsf{A}')}{T/\emptyset \vdash' \{n\}_g}\,(\mathsf{UL}') \qquad \dfrac{g \in \mathcal{G}}{T/\{g\} \vdash' g}\,(\mathsf{G})}{T/\{g\} \vdash' n}\,(\mathsf{D}')$$

$$\dfrac{\dfrac{\dfrac{\langle \{n\}_g, \{n\}_g \rangle \in T}{T/\emptyset \vdash' \langle \{n\}_g, \{n\}_g \rangle}\,(\mathsf{A}')}{T/\emptyset \vdash' \{n\}_g}\,(\mathsf{UR}') \qquad \dfrac{g \in \mathcal{G}}{T/\{g\} \vdash' g}\,(\mathsf{G})}{T/\{g\} \vdash' n}\,(\mathsf{D}')$$

4.5. Locality

In order to prove the intruder deduction problem is polynomial in time (Section 4.6), we shall prove below 3 *locality* [19] results for the respective theories DY, DY' and DY'+Compare.

The proposition 1 is a locality result for DY-proofs, its proof (folklore knowledge, see *e.g.* [8]) can be straightforwardly extended to show the similar locality result (Proposition 2) for the DY'-proofs.

Proposition 1 *A normal DY-proof P of $T \vdash u$ contains only terms in $st(T \cup \{u\})$. If moreover P ends with a decomposition rule (*A, UL, UR, D*), then P contains only terms in $st(T)$.*

$$\cfrac{\cfrac{\cfrac{Q_1}{T/T'_1 \vdash' u_1} \qquad \cfrac{Q_2}{T/T'_2 \vdash' u_2}}{T/T'_1 \cup T'_2 \vdash' \{u_1\}_{u_2}}\,(\mathsf{E}') \qquad \cfrac{Q_3}{T/T'_3 \vdash' u_2^{-1}}}{T/T'_1 \cup T'_2 \cup T'_3 \vdash' u_1}\,(\mathsf{D}') \quad \rightsquigarrow \quad \cfrac{Q_1}{T/T'_1 \vdash' u_1}$$

$$\cfrac{\cfrac{\cfrac{Q_1}{T/T'_1 \vdash' \{u_1\}_{u_2}} \qquad \cfrac{Q_2}{T/T'_2 \vdash' u_2^{-1}}}{T/T'_1 \cup T'_2 \vdash' u_1}\,(\mathsf{D}') \qquad \cfrac{Q_3}{T/T'_3 \vdash' u_2}}{T/T'_1 \cup T'_2 \cup T'_3 \vdash' \{u_1\}_{u_2}}\,(\mathsf{E}') \quad \rightsquigarrow \quad \cfrac{Q_1}{T/T'_1 \vdash' \{u_1\}_{u_2}}$$

Similar rules exist for (P'/UR') and (P'/UL')

Figure 4. The proof rewriting rules.

Proposition 2 *A normal DY'-proof P of $T/T' \vdash' u$ contains only terms in $st(T \cup T' \cup \{u\})$, and if P ends with a decomposition rule* (G, A', UL', UR', D')*, then P contains only terms in $st(T \cup T')$.*

Proposition 3 *A guessing-proof of $T \vdash g$ contains only terms in $st(T \cup \mathcal{G})$.*

Proof. (sketch) A guessing-proof is made up of two normal DY'-proofs P_1 of $T/T'_1 \vdash' u_1$ and P_2 of $T/T'_2 \vdash' u_2$. We distinguish two cases, whether $u_1 = u_2$ or not. In both cases, we show that P_1 or P_2 ends with a decomposition rule. Hence, we conclude thanks to Proposition 2. ☐

This result allows us to consider only subterms of the attacker's knowledge as potential verifiers to do a dictionary attack.

4.6. Intruder Deduction Problem

The intruder deduction problem, which corresponds to the security decision problem in the presence of a passive attacker, is a significant question to the verification problem as well as to the search for attacks. We can formulate this problem in the following way: given a finite set T of messages (ground terms) and a message s (the secret), can the intruder deduce s? In other words, does there exist a proof of $T \vdash s$ in the intruder deduction model? This deduction problem clearly depends on the deduction capabilities of the intruder. The proofs of the following complexity results use the locality results of Propositions 1 and 3.

Proposition 4 *Given a set of messages $T \subseteq \mathcal{T}(\mathcal{F})$, and a message $u \in \mathcal{T}(\mathcal{F})$, the existence of a DY-proof of $T \vdash u$ can be decided in polynomial time in $\|T \cup \{u\}\|_d$.*

Proof. This result follows from Lemma 1, which guarantees the existence of a normal DY-proof P of $T \vdash u$, and Proposition 1, which says that P only involves terms in $st(T \cup \{u\})$. Indeed, we can code, following [19], by a set $\mathcal{S}$ of ground Horn clauses the instances of the inference rules that can be used in a normal DY-proof of $T \vdash u$. This way, we reduce the problem of deciding whether there exists a (normal) DY-proof of $T \vdash u$ to the HORN–SAT problem for $\mathcal{S}$. Hence, the existence of a DY-proof of $T \vdash u$ can be decided in polynomial time in $\|T \cup \{u\}\|_d$, since $|\mathcal{S}|$ is linear in $\|T \cup \{u\}\|_d$. ☐

Proposition 5 *Given a set of messages $T \subseteq \mathcal{T}(\mathcal{F})$, and a guessable symbol $g \in \mathcal{G}$, the existence of a guessing-proof of $T \vdash g$ can be decided in polynomial time in $\|T \cup \mathcal{G}\|_d$.*

Proof. Like in the above proof of Proposition 4, we reduce the problem with the construction of a set of ground Horn clauses $\mathcal{S}$. In order to code the instances of the rule Compare with its conditions of application, we need though to add some additional information into the atoms of the clauses. The construction of $\mathcal{S}$ is detailed in Appendix A. ☐

Corollary 1 *Given a set of messages $T \subseteq \mathcal{T}(\mathcal{F})$, and a message $s \in \mathcal{T}(\mathcal{F})$, the intruder deduction problem $T \vdash s$ can be decided in polynomial time in $\|T \cup \mathcal{G} \cup \{s\}\|_d$ for the extended Dolev–Yao intruder model.*

Proof. Let us compute first the set $\mathcal{G}_1$ of guessable symbols $g \in \mathcal{G}$ such that there exists a guessing-proof of $T \vdash g$ (Proposition 5) and then solve the problem of the existence of a DY-proof of $T \cup \mathcal{G}_1 \vdash s$ (Proposition 4), which is equivalent to the intruder deduction problem $T \vdash s$ in the extended Dolev–Yao intruder model. Both above steps can be performed in polynomial time in $\|T \cup \mathcal{G} \cup \{s\}\|_d$. ☐

5. Attacks

We assume given a protocol $\mathcal{P}$, a runnable initial configuration $\mathcal{S} = (p_0, \sigma_0), \ldots, (p_m, \sigma_m)$ of $\mathcal{P}$, and a finite set $S_0 \subseteq \mathcal{T}(\mathcal{F})$ of initial knowledge of the intruder, such that $0 \in S_0$.

Now that we have semantics for the protocol execution (Section 3) and for the intruder (Section 4), we shall define the security attacks (with guessing) that the intruder can mount, starting with S_0, against the processes of $\mathcal{S}$. Our definition involves simultaneous solving of intruder deduction problems with variables, presented as symbolic constraints of the following form.

Definition 7 *A constraint is a sequent of the form $T \vdash_{\mathsf{dy}} u$ (*DY–constraint*) or $T \vdash_{\mathsf{g}} u$ (*guess–constraint*) where T is a finite subset of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and $u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. A solution of a finite set (or* system*) $\mathcal{C}$ of constraints is a grounding substitution σ such that for every $T \vdash_{\mathsf{dy}} u \in \mathcal{C}$ (resp. every $T \vdash_{\mathsf{g}} u \in \mathcal{C}$) there exists a DY-proof (resp. guessing-proof) of $T\sigma \vdash u\sigma$.*

Note that we do not assume that the constraints of the set $\mathcal{C}$ are variable disjoint.

We assume a linear well–founded ordering $\prec$ on the ground terms of $\mathcal{T}(\mathcal{F})$ such that the constant 0 is minimal w.r.t. $\prec$. We shall use below the (well–founded) extension $\ll$ of $\prec$ to multisets of ground terms. For sake of notation, given two solutions σ_1 and σ_2 of sets of constraints, we write $\sigma_1 \ll \sigma_2$ iff $img(\sigma_1) \ll img(\sigma_2)$ ($img(\sigma_i)$ is the multiset of all $x\sigma$ such that $x \in dom(\sigma_i)$).

An *interleaving* of $\mathcal{S}$ is a finite sequence I, without repetition, of values which can be either constants of $\mathcal{G}$ or pairs of integers (i, j) where $0 \leq i \leq m$ (i is the index of a process of $\mathcal{S}$) and $0 \leq j < |p_i|$ (j is the index of a pair of instruction of p_i), which satisfies the following *ordering condition*: for each i with $0 \leq i \leq m$, the subsequence of I of pairs with first component i has the form $(i, 0), (i, 1), \ldots, (i, n)$ with $n < |p_i|$. This condition expresses that I describes a partial linear execution of the respective programs of the processes, up to some point.

Let $s \in \mathcal{T}(\mathcal{F})$ be a term whose confidentiality must be ensured by the protocol. We associate to s, $\mathcal{S}$, S_0 and to an interleaving I of length ℓ a set $\mathcal{C} = \mathcal{C}(s, \mathcal{S}, S_0, I) = \{C_0, \ldots, C_\ell\}$ of DY– and guess–constraints, which solvability defines the problem of protocol insecurity in presence of dictionary attacks. We construct in parallel the constraints of $\mathcal{C}$ and the sequences $T_0, \ldots, T_\ell$ of their hypotheses sets. Let $T_0 = S_0$. For each $k < \ell$,

- if $I_k \in \mathcal{G}$, then $C_k := T_k \vdash_{\mathsf{g}} g$ (the intruder can deduce g by guessing) and $T_{k+1} := T_k \cup \{g\}$ (he adds this value to his knowledge),
- if $I_k = (i, j)$, then let $\mathsf{recv}(r_{i,j}); \mathsf{send}(s_{i,j})$ be the jth instruction pair of the program p_i, then $C_k := T_k \vdash_{\mathsf{dy}} r_{i,j}\sigma_i$ ($r_{i,j}\sigma_i$ can be received from the network) and $T_{k+1} := T_k \cup \{s_{i,j}\sigma_i\}$ ($s_{i,j}\sigma_i$ is sent to the network).

And finally, if $s \in \mathcal{G}$ then $C_\ell := T_\ell \vdash_{\mathsf{g}} s$ and otherwise, $C_\ell := T_\ell \vdash_{\mathsf{dy}} s$ (the secret s is revealed).

Example 3 *The attack described in Section 2.3 can be executed starting with a (runnable) initial configuration $\mathcal{S}$ made of 5 processes: the processes* 0, 1 *for the respective roles A and S of session α which were described in Example 2, a process* 4 *for the role S of session β and other processes* 7 *and* 8 *for the role S and B of session γ, with:*

$$\sigma_4 = \left\{ \begin{array}{ccc} x^4_K & \mapsto & K', \\ x^4_{Ts} & \mapsto & Ts', \\ x^4_S & \mapsto & S, \\ x^4_{pub(S)} & \mapsto & pub(S) \end{array} \right\}$$

$$\sigma_7 = \left\{ \begin{array}{ccc} x^7_K & \mapsto & K'', \\ x^7_{Ts} & \mapsto & Ts'', \\ x^7_S & \mapsto & S, \\ x^7_{pub(S)} & \mapsto & pub(S) \end{array} \right\}$$

$$\sigma_8 = \{x^8_{pw(B,S)} \mapsto pw(B, S)\}$$

where K', Ts', K'', Ts'' are constants of $\mathcal{F}_0$.

The interleaving I describing the trace of the attack is:

$$\begin{array}{cccccc} (0,0), & (1,0), & (4,0), & pw(A,S), & (7,0), & (8,0) \\ \alpha.0 & \alpha.1 & \beta.1 & pw(A,S) & \gamma.1 & \end{array}$$

The steps $\beta.0$, $\gamma.0$ and $\gamma.2$ do not occur in I since they are performed by the intruder (on behalf of A) in the attack. The step $(8, 0)$ (reception of last message by B) has no corresponding label.

We define the intruder's initial knowledge by $S_0 = \{0, S, pub(S), A, B, M_1, M_2, Ci, Ti\}$. The constraint system $\mathcal{C}(K'', \mathcal{S}, S_0, I)$ is described in Figure 5.

Lemma 2 *For all $i \leq \ell$ and all variable x of a term in the hypotheses of C_i, there exists $j < i$ such that x occurs in the target of C_j.*

Proof. By the construction of $\mathcal{C}(s, \mathcal{S}, S_0, I)$, the ordering condition for the interleaving I and the hypothesis that the initial configuration $\mathcal{S}$ is runnable (Definition 3). □

Definition 8 *An attack on the security of $s \in \mathcal{T}(\mathcal{F})$ with the protocol $\mathcal{P}$, the runnable initial configuration $\mathcal{S}$ and the initial intruder's knowledge $S_0 \subseteq \mathcal{T}(\mathcal{F})$ is a pair (I, σ) where I is an interleaving of $\mathcal{S}$ and σ is a solution of $\mathcal{C}(s, \mathcal{S}, S_0, I)$.*

Note that this definition conforms to the semantics of Section 3, in the sense that a solution σ of $\mathcal{C}(s, \mathcal{S}, S_0, I)$ is the (disjoint) union of the substitutions of the processes, after the execution of the trace summarized in I. In the above construction of $\mathcal{C}(s, \mathcal{S}, S_0, I)$, we instantiate the terms by the substitutions σ_i of the processes of the initial configuration $\mathcal{S}$, to ensure that every solution of $\mathcal{C}(s, \mathcal{S}, S_0, I)$ really subsumes these substitutions.

(0,0)	$T_0 := S_0, \quad C_0 := T_0 \vdash_{\mathsf{dy}} 0$	A receives 0
	$T_1 := T_0, \{A, B, N_1, N_2, Ca, \{Ta\}_{pw(A,S)}\}_{pub(S)}$	A sends his request to S
(1,0)	$C_1 := T_1 \vdash_{\mathsf{dy}} \{x^1_A, x^1_B, x^1_{N_1}, x^1_{N_2}, x^1_{Ca}, \{x^1_{Ta}\}_{pw(x^1_A,S)}\}_{pub(S)}$	S receives the request
	$T_2 := T_1, \{x^1_{N_1}, K \oplus x^1_{N_2}\}_{pw(x^1_A,S)}, \{x^1_A, K, Ts\}_{pw(x^1_B,S)}$	S answers the request
(4,0)	$C_2 := T_2 \vdash_{\mathsf{dy}} \{x^4_A, x^4_B, x^4_{N_1}, x^4_{N_2}, x^4_{Ca}, \{x^4_{Ta}\}_{pw(x^4_A,S)}\}_{pub(S)}$	S receives a second request
	$T_3 := T_2, \{x^4_{N_1}, K' \oplus x^4_{N_2}\}_{pw(x^4_A,S)}, \{x^4_A, K', Ts'\}_{pw(x^4_B,S)}$	S answers, I diverts S answer
$pw(A,S)$	$C_3 := T_3 \vdash_{\mathsf{g}} pw(A,S)$	I guesses $pw(A,S)$ using S answer
	$T_4 := T_3, pw(A,S)$	$pw(A,S)$ is added to I's knowledge
(7,0)	$C_4 := T_4 \vdash_{\mathsf{dy}} \{x^7_A, x^7_B, x^7_{N_1}, x^7_{N_2}, x^7_{Ca}, \{x^7_{Ta}\}_{pw(x^7_A,S)}\}_{pub(S)}$	S receives a third request
	$T_5 := T_4, \{x^7_{N_1}, K'' \oplus x^7_{N_2}\}_{pw(x^7_A,S)}, \{x^7_A, K'', Ts''\}_{pw(x^7_B,S)}$	S answers, I diverts S answer
(8,0)	$C_5 := T_5 \vdash_{\mathsf{dy}} \{x^8_A, x^8_K, x^8_{Ts}\}_{pw(B,S)}$	B accepts I's message
	$T_6 := T_5, 0$	B answers 0
	$C_6 := T_6 \vdash_{\mathsf{dy}} K''$	and the secret K'' is revealed

Figure 5. The constraint system $\mathcal{C}(K'', \mathcal{S}, S_0, I)$.

6. Decision Procedure

We present in this section a non deterministic polynomial time algorithm to find security attacks, given a protocol, a secret, an initial configuration and an initial intruder's knowledge. The idea is that if there exists an attack, then there exists a minimal attack whose dag-size is polynomial in the size of the problem. This fact has been shown in [21] for the Dolev-Yao model and we can use it to treat DY-constraints. However the case of guess-constraints is much more difficult because some results which are obvious for DY-proofs are not true for DY'-proofs (see Section 4.3). Indeed, we have shown that we can always assume that a DY-proof is in normal form, and we often use this result, but the transformation rules to normalize a DY'-proof of $T/T' \vdash u$ must be used very carefully since we can lose weak hypotheses in T' when we apply them. Moreover, a guessing-proof ends with an instance of the rule Compare which is inherently difficult.

6.1. Algorithm

The following decision algorithm takes as input a secret $s \in \mathcal{T}(\mathcal{F})$, a protocol $\mathcal{P}$, a set of initial processes $\mathcal{S}$ and an initial intruder's knowledge $S_0 \subseteq \mathcal{T}(\mathcal{F})$ like in Section 5, and checks the existence of an attack.

1. choose an interleaving I of $\mathcal{S}$; let $\mathcal{C} = \mathcal{C}(s, \mathcal{S}, S_0, I)$ and let $\{x_1, \ldots, x_m\} = vars(\mathcal{C})$,
2. choose for each $i \leq m$ a term $t_i \in st(\mathcal{C})$, and let σ be a most general unifier (if any) of the equational problem $\mathcal{E} = \{x_1 \approx t_1, \ldots, x_m \approx t_m\}$,
3. if σ is ground, check whether σ is a solution of $\mathcal{C}$, if so return Yes.

6.2. Complexity

By construction, the number m of variables in $\mathcal{C}$ is smaller than $|vars(\mathcal{S})|$, hence, $\|\mathcal{E}\| \leq |vars(\mathcal{S})|.M$, where M is the maximal size of a term in $\mathcal{P} \cup S_0 \cup \{s\}$. At step 2, the mgu σ (represented as a dag) can be computed in polynomial time in $\|\mathcal{E}\|$, using syntactic transformation rules for solving unification problems, see *e.g.* [16]. The dag-size of the terms in the codomain of σ is polynomial in the size of $\mathcal{E}$, hence in the size of $\mathcal{S}$, $\mathcal{P}$, S_0 and s.

The test of step 3 consists in checking that each constraint of $\mathcal{C}$ is satisfied by σ. The number $|\mathcal{C}|$ of constraints, which is also the length $|I|$ of the chosen interleaving I is at most $|\mathcal{G}|.|\mathcal{S}|. \sum_{p \in \mathcal{P}} |p|$. Moreover, by construction of $\mathcal{C}$, and according to the above bound on the size of σ, the instance of any constraint of $\mathcal{C}$ by σ has a dag-size polynomial in the size of $\mathcal{S}$, $\mathcal{P}$, S_0 and s. The results of Propositions 4 and 5 yield polynomial procedures for checking the satisfaction of each constraint by σ. Altogether, the complexity of step 3 is polynomial in the sizes of $\mathcal{G}$, $\mathcal{S}$, $\mathcal{P}$, S_0, and s.

6.3. Completeness

The completeness of our algorithm is ensured by the corollary 2 of the key proposition 6. The following technical lemmas 3,4, 5 will be used in the proof of this proposition. Given a proof P of a sequent $T \vdash u$, the aim of Lemmas 3 and 5 is to ensure the existence of a particular proof of $T \vdash u$ which respects some extra conditions in order to guarantee some results when we are going to apply transformations, as replacement, on proof trees.

Lemma 3 *[21] Let P be a DY-proof of $T \vdash t$ and P' be a minimal DY-proof of $T \vdash \gamma$ ending with a composition rule (E or P). There exists a proof of $T \vdash t$ in which γ is never decomposed.*

One may observe that in a guessing-proof of $T \vdash g$, the only relevant information in the weak hypotheses T' of a node $T/T' \vdash t$ is whether T' contains g or not. Below, in order to simplify the notations, the weak hypotheses in DY'– proofs will be noted g^+, g^- or $\emptyset$: g^+ and g^- represent arbitrary subsets of $\mathcal{G}$ respectively containing g and not containing g. Although $\emptyset$ is a subcase of g^-, we shall still use this notation to emphasis that every DY'-proof of $T/\emptyset \vdash u$ is isomorphic a DY-proof of $T \vdash u$. Note that a set of guessing-proof defined with the g^+ and g^- notation can be represented by a unique guessing-proof, which real contents of weak hypotheses are deduced from the leaves.

Lemma 5 below is an analogous of Lemma 3 for guessing proofs. Its proof requires the following auxiliary Lemma 4 which gives us sufficient conditions to have a guessing-proof of a given sequent $T \vdash g$.

Lemma 4 *Let P_1 and P_2 be two normal DY'-proofs of respectively $T/\emptyset \vdash' t$ and $T/g^+ \vdash' t$. There exist two DY'-proofs P_1' and P_2', subtrees of P_1 and P_2 respectively and a guessing-proof of $T \vdash g$ whose two sons are P_1' and P_2'.*

Proof. We prove this result by induction on the proof P_2. If the proof P_2 is an instance of G, then we can apply the rule Compare since the conditions (i), (ii) and (iv) are clearly verified, and (iii) also because the proof P_1 of $T/\emptyset \vdash' t$ can not end with an instance of G.

If the last rule of P_2 is UL' then the conditions (i), (ii) and (iv) of Compare are clearly met. Either the proof P_1 and P_2 verify the condition (iii), and we can apply the rule Compare, or these proofs end respectively by the instances $(T/\emptyset \vdash' \langle t_1, t_2\rangle, T/\emptyset \vdash' t_1)$ and $(T/g^+ \vdash' \langle t_1, t_2\rangle, T/g^+ \vdash' t_1)$ of UL'. In such a case we apply the induction hypotheses on the subtrees of P_1 and P_2 which root are labeled with $T/\emptyset \vdash' \langle t_1, t_2\rangle$ and $T/g^+ \vdash' \langle t_1, t_2\rangle$ respectively. The other cases are very similar. ☐

In the following Lemma 5, equivalent of Lemma 3 for the guessing-proofs, we impose an extra condition (condition b) to ensure that the replacement we are going to perform on a such guessing-proof doesn't lose the only relevant information in the weak hypothesis set.

Lemma 5 *Let P be a guessing-proof of $T \vdash g$ and P' a minimal DY'-proof of $T/\emptyset \vdash' \gamma$ ending with a composition rule (E' or P'). There exists a guessing-proof of $T \vdash g$ in which:*
(a) γ is never decomposed,
(b) every instance $(s_1, s_2, T/T' \vdash' \gamma)$ of a composition rule (P' or E') is such that $g \notin T'$.

Proof. We make an induction on the number of instances of rules in P which do not satisfy (a) or (b).

First, we consider the case of an instance which does not satisfy the condition (a), and we distinguish two subcases, depending on whether a premise of the instance is labeled with $T/g^+ \vdash' \gamma$ or $T/g^- \vdash' \gamma$.

Case (a^+): Let $\gamma = \langle\gamma_1, \gamma_2\rangle$ (the case $\gamma = \{\gamma_1\}_{\gamma_2}$ is similar), let $(T/g^+ \vdash' \gamma, s)$ be an instance of UL' (the case UR' is similar) in P which does not satisfy the condition (a), and let P_1 the subproof of P whose root is the above $T/g^+ \vdash' \gamma$. We can apply the induction hypothesis to the guessing-proof P'' of $T \vdash g$ whose direct subproofs are P_1 and P'

Case (a^-): Let P_1 and P_2 be the two direct (DY'-) subproofs of P. Let $\gamma = \langle\gamma_1, \gamma_2\rangle$ (the case $\gamma = \{\gamma_1\}_{\gamma_2}$ is similar), let $(T/g^- \vdash' \gamma, s)$ be an instance of UL' (the case UR' is similar) in P_i (say $i = 1$ for simplicity) which does not satisfy (a) and let P_1' be the tree obtained from P_1 by replacing the subproof of P_1 whose root is the above $T/g^- \vdash' \gamma$ by P' (whose root is $T/\emptyset \vdash' \gamma$). This tree P_1' is a DY'-proof since $\emptyset$ is a subcase of g^- in our notation. It is not normal, and hence the condition (ii) of Compare is not satisfied by P_1'. We normalize the proof P_1' using simplification rules which are not described here for sake of place, and show by induction on the number of simplification steps (using Lemma 4) that we obtain a guessing-proof P'' to which we can apply the induction hypothesis. The simplification rules and the detailed induction proof can be found in the Appendix B of the long version of this paper [10].

Case (b): Let $(s_1, s_2, T/g^+ \vdash' \gamma)$ be an instance of a composition rule in P which does not satisfy (b), with $\gamma = \langle\gamma_1, \gamma_2\rangle$ (the case $\{\gamma_1\}_{\gamma_2}$ is similar). Hence, we have a normal DY'-proof P_1' of $T/g^+ \vdash' \gamma$, and we can apply Lemma 4 to P' and P_1' in order to obtain a guessing-proof P'' of $T \vdash g$ to which we apply the induction hypothesis. ☐

Proposition 6 *Let σ be a minimal (w.r.t. $\ll$) solution of $\mathcal{C}$. For all $x \in vars(\mathcal{C})$, there exists $t \in st(\mathcal{C}) \setminus vars(\mathcal{C})$ such that $t\sigma = x\sigma$.*

Proof. We reason by contradiction. Assume that there exists $x \in vars(\mathcal{C})$ such that for all $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \setminus \mathcal{X}$ with $t\sigma = x\sigma$, we have $t \notin st(\mathcal{C})$. We will show that under this condition there exists a smaller solution σ' of $\mathcal{C}$.

Let $\mathcal{C} = \{C_1, \ldots C_\ell\}$ and for each $i \leq \ell$, let r_i be the target of C_i and $C_i\sigma$ be the (ground) constraint obtained from C_i by instantiating all the terms in its hypotheses and target with σ.

Fact 1 *If $x\sigma \in st(s\sigma)$ for some hypothesis s of C_i ($i \leq \ell$), then there exists $j < i$ such that $x\sigma \in st(r_j\sigma)$.*

Proof. This is a consequence of Lemma 2 since, by hypothesis, if $x\sigma \in st(C_i\sigma)$ then $x\sigma \in st(y\sigma)$ for some

$y \in vars(C_i)$ (otherwise there exists a $t \in st(C_i) \setminus \mathcal{X}$ such that $t\sigma = x\sigma$). $\square$

Fact 1 allows us to define: $m = \min\{j \mid x\sigma \in st(r_j\sigma)\}$. Note that the constraint C_m is a DY–constraint of the form $S_m \vdash_{\mathsf{dy}} r_m$. Otherwise, r_m would be a ground term, hence $x\sigma \in st(r_m\sigma) \subseteq st(\mathcal{C})$, a contradiction. The proof of the following Fact can be found in Appendix B.

Fact 2 *There exists a minimal DY-proof of $S_m\sigma \vdash x\sigma$ ending with a composition rule E or P.*

Now, we let δ be the replacement $\{x\sigma \mapsto 0\}$ (0 is a special constant introduced in Section 3). We will show that $\sigma' := \sigma\delta$ is also a solution of $\mathcal{C}$, which is a contradiction since $\sigma' \ll \sigma$. For this purpose, we have to build a proof of each $C_i\sigma'$, $i \leq l$. For each $i < m$, $x\sigma \notin st(C_i\sigma)$, by definition of m and Fact 1. Hence, $(C_i\sigma)\delta = C_i\sigma = C_i\sigma'$, *i.e.* σ' is a solution of C_i.

Let us show that σ' is also a solution of C_i for each $i \geq m$. We may note first that $C_i(\sigma\delta) = (C_i\sigma)\delta$, because of the hypothesis that there does not exists $t \in st(C_i) \setminus \mathcal{X}$ such that $t\sigma = x\sigma$. So, we are going to show that there exists a (DY- or guessing-) proof of $(C_i\sigma)\delta$ for each $i \geq m$.

Case (1): C_i is a guess–constraint $S_i \vdash_{\mathsf{g}} g$. There exists a guessing-proof of $S_i\sigma \vdash g$ and moreover, thanks to Fact 2, there exists a normal DY'-proof of $S_i\sigma/\emptyset \vdash' x\sigma$ ending with a composition rule. Thanks to Lemma 5, there exists a guessing-proof GP of $S_i\sigma \vdash g$ which verify the condition (a) and (b). We shall build from GP a guessing-proof of $(S_i\sigma)\delta \vdash g$. We replace first in GP every subtree ended by an instance $(s_1, s_2, S_i\sigma/g^- \vdash' x\sigma)$ of composition rule (E' or P') by the following "instance" of A': $(x\sigma \in S_i\sigma, S_i\sigma/g^- \vdash' x\sigma)$. Then we apply δ to every term of the tree obtained, getting GP'.

Fact 3 *GP' is a guessing-proof of $(S_i\sigma)\delta \vdash g$.*

This fact is proved in Appendix B. It follows that σ' is a solution of C_i.

Case (2): C_i is a DY–constraint $S_i \vdash_{\mathsf{dy}} r_i$. By hypothesis, there exists a DY-proof of $S_i\sigma \vdash r_i\sigma$, and thanks to Fact 2 and Lemma 3, there exists a DY-proof of $S_i\sigma \vdash r_i\sigma$ in which $x\sigma$ is never decomposed. We can build as in the previous case a DY-proof of $(S_i\sigma)\delta \vdash (r_i\sigma)\delta$. $\square$

Corollary 2 *If $\mathcal{C}$ admits a solution then there exists an equational problem of the form $\mathcal{E} = \{x_1 \approx t_1, \ldots, x_n \approx t_n\}$, where $\{x_1, \ldots, x_n\} = vars(\mathcal{C})$ and $t_1, \ldots, t_n \in st(\mathcal{C})$ such that σ is the unique most general unifier of $\mathcal{E}$.*

Proof. Let σ be a minimal solution of $\mathcal{C}$. By Proposition 6, for each x_i, $i \leq n$, there exists $t_i \in st(\mathcal{C}) \setminus vars(\mathcal{C})$ such that $t_i\sigma = x_i\sigma$, *i.e.* such that σ is a solution of $\mathcal{E} = \{x_1 \approx t_1, \ldots, x_n \approx t_n\}$. We can permute the indexes of the variables $x_1, \ldots, x_n$ in order to have $\mathcal{E}$ in dag solved form, *i.e.* such that:

$$\text{for every } 1 \leq i < j \leq n,\ x_i \notin vars(t_j) \tag{1}$$

(the x_i are pairwise distinct and every $t_j \notin \mathcal{X}$ by construction). The converse would mean that $\mathcal{E}$ has no solution, using the completeness results for the dag based syntactic unification procedure presented in [16]. Indeed, the only transformation rule of this procedure applicable to a system of the form of $\mathcal{E}$ is the "occur–check", and its application would mean that $\mathcal{E}$ has no solution.

Hence, see [16], $\mathcal{E}$ has a unique most general unifier $\theta = \theta_1 \ldots \theta_n$ where each θ_i is $\{x_i \mapsto t_i\}$. Since for each $i \leq n$, $t_i \in st(\mathcal{C})$, and hence $vars(t_i) \in \{x_1, \ldots, x_n\}$, and by condition (1), θ is ground. It implies that $\theta = \sigma$. $\square$

6.4. NP–hardness

If we choose $\mathcal{G} = \emptyset$, then we fall into the problem of [21] which has been shown NP–hard.

7. Related Work

In the spi–calculus [1], the protocol security is defined by behavioral equivalence between processes: the secrecy of V is ensured by the protocol $P(V)$ if for every V', and every process O, we have $P(V)|O \cong P(V')|O$ (the relation $\cong$ is *barbed equivalence*). In particular, the "intruder" O can perform some comparisons, by matching, and we may wonder how Definition 8 can be compared to security in spi–calculus.

For instance, assume that (the role A of) the protocol of Section 2.1 is represented by the process $P(V) = \overline{p}\big\langle \{V\}_{pub(S)} \big\rangle$ (the encrypted vote is sent on the channel p). Consider the observer process $O := p(x).[x\ is\ \{V\}_{pub(S)}]\,\overline{q}\langle x\rangle$ (x is received on channel p, and if it is equal to $\{V\}_{pub(S)}$ then x is resent on channel q). For $V' \neq V$, $P(V)|O$ and $P(V')|O$ do not have the same behavior w.r.t. the channel q, hence $P(V)|O \not\cong P(V')|O$. Therefore, the vote V, which is secret in the Dolev–Yao model, is vulnerable in spi–calculus like in our extended intruder model. Nevertheless, some protocols that are considered secure in our model are not in spi–calculus. Consider a process $P(K) = (\nu n)\overline{p}\big\langle \{n\}_K \big\rangle$ which sends a nonce n encrypted with key K on channel p. The one message protocol implemented by $P(K)$ is considered secured for the key K in our settings, and not in spi–calculus: consider for instance the observer $O := p(x).case\ x\ of\ \{y\}_K\ in\ \overline{q}\langle x\rangle$.

Lowe presents in [18] a formal CSP model of an intruder able of dictionary attacks and a procedure to detect such attacks, which is implemented as an extension of the framework based on the protocol compiler Casper and the model checker FDR. We believe that our model is compatible with the one of [18], though the formalisms differ, and hence that our decidability and complexity results are also valid for this system.

Another implementation of dictionary attacks detection is presented in [9]. The authors, like us, define the existence of dictionary attacks by the solvability of a system of constraints, but unlike us, they use only Dolev–Yao constraints (of the form $\vdash_{\mathsf{dy}}$). Indeed, the guessing-constraints $T \vdash_{\mathsf{g}} g$ are encoded in [9] into some DY-constraints and negation of DY–constraints. However, this gives a definition of attacks strictly coarser than our Definition 8. For instance, the tree of the second example of Section 4.4, which is not a guessing-proof, and which, in our opinion does not represent a real dictionary attack, is considered as a dictionary attack in [9].

8. Conclusion

We have defined a formal model of an intruder with Dolev–Yao and guessing abilities, and of the security attacks that he can mount against cryptographic protocols. A non–deterministic polynomial time procedure is described which decides whether there exists such attacks on a given protocol and a given finite set of sessions. It is based on a polynomial time algorithm for the decision of the intruder deduction problem. A conservative approximation of our procedure has been implemented in a prototype which has been executed successfully on some of the examples presented in this paper and others.

One may note that our procedure is not restricted to one intruder's guess per attack. We may also wonder whether it is easier to detect attacks of a weaker intruder, for instance a passive intruder who can only listen to communications on the network and guess offline weakly chosen passwords. The existence of such attacks, for a finite number of processes, can actually be reduced to the intruder deduction problem in the extended Dolev–Yao model, and hence can be decided in polynomial time by Corollary 1.

In this paper, we focused on offline dictionary attacks. The problem of online dictionary attacks, where the intruder uses an online exchange of messages to verify each of his guesses, can also be realistic in some situations [11], but its formalization needs quite a different model than the one presented here. As future work we consider to study the design of a general framework, in the style of the one of Section 4, covering several kinds of intruder techniques such as online and offline dictionary attacks or e.g. chosen plaintext attacks.

Acknowledgments

The authors wish to thank Hubert Comon–Lundh for his precious assistance to this work, and Ralf Treinen for his reading of a preliminary version of this paper.

References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997.

[2] R. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR)*, volume 2421 of *LNCS*, 2002.

[3] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proceedings of the 12th International Conference on Concurrency Theory (CONCUR)*, volume 1877 of *LNCS*, 2000.

[4] M. Bellare, D. Pointcheval and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proceedings of Advances in Cryptology (EUROCRYPT)*, volume 1807 of *LNCS*, 2000.

[5] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1992.

[6] Y. Chevalier, R. Kuester, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with xor. In *Proceedings of Logic in Computer Science (LICS)*, 2003.

[7] H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. In *Theoretical Computer Science*, 2003. To appear.

[8] H. Comon-Lundh and V. Shmatikov. Constraint solving, exclusive or and the decision of confidentiality for security protocols assuming a bounded number of sessions. In *Proceedings of Logic in Computer Science (LICS)*, 2003.

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

[10] S. Delaune and F. Jacquemard. A theory of guessing attacks and its complexity. Research Report LSV-04-1, Lab. Specification and Verification, 2004.

[11] Y. Ding and P. Horster. Undetectable on-line password guessing attacks. *ACM Operating Systems Review*, 29(4), 1995.

[12] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proceedings of Workshop on Formal Methods and Security Protocols*, 1999.

[13] D. Dolev and A. Yao. On the security of public key protocols. In *Proceedings of the IEEE Transactions on Information Theory, 29(2)*, 1983.

[14] S. Goldwasser ans S. Miscali. Probabilistic encryption. In *Journal of Computer and System Sciences*, volume 28, 1984.

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

[16] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: a rule-based survey of unification. In *Computational Logic. Essays in honor of Alan Robinson*. The MIT press, 1991.

[17] J. Katz, R. Ostrovsky, M. Yung Efficient password-authenticated key exchange using human-memorable passwords. In *Proceedings of Advances in Cryptology (EUROCRYPT)*, 2001.

[18] G. Lowe. Analysing protocols subject to guessing attacks. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS)*, 2002.

[19] D. McAllester. Automatic recognition of tractability in inference relations. In *Journal of the ACM, 40(2)*, 1993.

[20] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, 2001.

[21] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th IEEE Computer Security Foundations Workshop*, 2001.

[22] J.G. Steiner, C. Neuman, and J.I Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Winter 1988 Usenix Conference*, 1988.

[23] G. Tsudik and E. Van Herreweghen. Some remarks on protecting weak secrets and poorly-chosen keys from guessing attacks. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 1993.

[24] T. Wu. The secure remote password protocol. In *Proceedings of the Internet Society Network and Distributed System Security Symposium*, 1998.

Appendix A. Proof of Proposition 5

Given a set of messages $T \subseteq \mathcal{T}(\mathcal{F})$, and a guessable symbol $g \in \mathcal{G}$, we describe below a set $\mathcal{S}$ of ground Horn clauses from which one can derive the empty clause if and only if there exists a (normal) guessing-proof of $T \vdash g$. Since the dag size of $\mathcal{S}$ is polynomial is $\|T \cup \mathcal{G}\|_d$, this provides a decision procedure with the complexity wanted. The ground terms in the clauses of $\mathcal{S}$ are either elements of $st(T, \mathcal{G})$ or one of the nullary symbols A, P, E, G corresponding to rules of the system of Figure 2, or also $UL(u)$, $UR(u)$, $D(u)$ with $u \in st(T, \mathcal{G})$. The meaning of $I_0(u, l)$ is that there exists a normal DY'-proof which root is labeled with $T/T' \vdash' u$, with $T' \subseteq \mathcal{G}$, and moreover, if l is A, P, E or G then the proof ends by the corresponding inference rule, and if $l = UL(v)$, then the proof has the following form (and similarly if $l = UR(v)$ or if $l = D(v)$):

$$\frac{\dfrac{\cdots}{T/T' \vdash' \langle u, v\rangle}}{T/T' \vdash' u}\ (\mathsf{UL}')$$

This second argument l is used to ensure the conditions (ii) and (iii) of the rule Compare. The meaning of $I_1(u, l)$ is the same as $I_0(u, l)$ except that moreover $g \in T'$. The set $\mathcal{S}$ contains ($\varepsilon + \varepsilon'$ denotes the Boolean or, for $\varepsilon, \varepsilon' \in \{0, 1\}$):

$\Rightarrow I_1(g, G)$,

$\Rightarrow I_0(u, A)$ for all $u \in T \cup \mathcal{G}$,

$I_\varepsilon(u, l), I_{\varepsilon'}(v, l') \Rightarrow I_{\varepsilon+\varepsilon'}(\langle u, v\rangle, P)$ with $l \neq UL(v)$ and $l' \neq UR(u)$,

$I_\varepsilon(u, l), I_{\varepsilon'}(v, l') \Rightarrow I_{\varepsilon+\varepsilon'}(\{u\}_v, E)$ with $l \neq D(v^{-1})$,

$I_\varepsilon(\langle u, v\rangle, l) \Rightarrow I_\varepsilon(u, UL(v))$ with $l \neq P$,

$I_\varepsilon(\langle u, v\rangle, l) \Rightarrow I_\varepsilon(v, UR(u))$ with $l \neq P$,

$I_\varepsilon(\{u\}_v, l), I_{\varepsilon'}(v^{-1}, l') \Rightarrow I_{\varepsilon+\varepsilon'}(u, D(v^{-1}))$ with $l \neq E$,

$I_\varepsilon(u, l), I_{\varepsilon'}(u, l') \Rightarrow Goal$ with $l \neq l'$ and $\varepsilon + \varepsilon' = 1$,

$I_\varepsilon(u, l), I_{\varepsilon'}(u^{-1}, l') \Rightarrow Goal$ with $x \neq x^{-1}$, $\varepsilon + \varepsilon' = 1$,

$Goal \Rightarrow$.

Appendix B. Proof of Proposition 6

Fact 2 *There exists a minimal DY-proof of $S_m\sigma \vdash x\sigma$ ending with a composition rule* E *or* P.

Proof. By hypothesis, there exists a DY–proof P of $S_m\sigma \vdash r_m\sigma$ and by Lemma 1, we can assume that P is a minimal DY–proof of $S_m\sigma \vdash r_m\sigma$. If P contains a node labeled by $S_m\sigma \vdash x\sigma$, then it is the root of a minimal subproof as expected. This proof indeed ends with a composition rule: otherwise, by minimality of P, we would have an occurrence of $x\sigma$ as a subterm of $S_m\sigma$, which contradicts the definition of m by Fact 1.

We show now that P necessarily contains one node labeled by $S_m\sigma \vdash x\sigma$. Assume that P contains no such node. We will construct recursively a path in P, from the root up to one leaf, every node of which is labeled by $S_m\sigma \vdash u$ such that $x\sigma \in st(u)$, and we shall show in parallel that the existence of such a path conducts to a contradiction.

By definition of m, the condition $x\sigma \in st(r_m\sigma)$ is true for the root of P, which is labeled by $S_m\sigma \vdash r_m\sigma$. Assume that this condition is also true for each node of (a prefix of) a path labeled by $S_m\sigma \vdash u_0, \ldots, S_m\sigma \vdash u_k$, with $u_0 = r_m\sigma$ and let us consider the sons of $s = S_m\sigma \vdash u_k$ is P:

- if s has 1 son s_1 and (s_1, s) is an instance of A, then $u_k \in S_m\sigma$ and $x\sigma \in st(u_k)$ contradicts the definition of m, because of Fact 1.
- if s has 1 son s_1 and (s_1, s) is an instance of UL or UR, then u_k is a subterm of the target of s_1, hence also $x\sigma$, and we let s_1 be the next node of the path.
- if s has 2 sons s_1, s_2 and (s_1, s_2, s) is an instance of D, then u_k is a subterm of the target of s_1 or s_2 (say s_1), hence also $x\sigma$, and we let s_1 be the next node of the path.
- if s has 2 sons s_1, s_2 and (s_1, s_2, s) is an instance of P or E. By hypothesis, we have $x\sigma \neq u_k$ since we have assume that P contains no such node. So, $x\sigma$ is a strict subterm of u_k and it is also a subterm of the target of one of s_1 and s_2 (say s_1). Hence, we can let s_1 be the next node of the path. $\square$

Fact 3 *GP' is a guessing-proof of $(S_i\sigma)\delta \vdash g$.*

Proof. We show first that in the two sons GP'_1 and GP'_2 of GP' are DY'–proofs, *i.e.* for every node labeled by s' and with n sons labeled respectively by $s'_1, \ldots, s'_n$, $(s'_1, \ldots, s'_n, s')$ is an instance of a rule of Figure 2.

- if $n = 1$ and (s'_1, s') is an instance of A' added by replacement in GP of an instance (s_1, s_2, s) of a composition rule in the construction of GP'. By construction, we have $s' = s\delta = (S_i\sigma)\delta/g^- \vdash' 0$ and $0 \in S_0 \subseteq (S_i\sigma)\delta$. Hence, (s'_1, s') is an instance of A'.
- if $n = 1$ and we are not in the above case, we have $(s'_1, s') = (s_1\delta, s\delta)$ where (s_1, s) is an instance of G, A',UL' or UR' contained in GP.

  Case G: let (s_1, s) be $(u \in \mathcal{G}, S_i\sigma/u \vdash' u)$. As shown above, $x\sigma \notin \mathcal{G}$, therefore $(s'_1, s') = (s_1\delta, s\delta)$ is also an instance of G.

  Case A': this case is immediate.

  Case UL', UR': by construction, we have $(s'_1, s) = (s_1\delta, s\delta)$ where (s_1, s) is an instance of UL' (the case of UR' is similar). Let (s_1, s) be $(S_i\sigma/g^\varepsilon \vdash' \langle u_1, u_2\rangle, S_i\sigma/g^\varepsilon \vdash' u_1)$. By condition (a) on GP, $\langle u_1, u_2\rangle \neq x\sigma$. Hence $\langle u_1, u_2\rangle\delta = \langle u_1\delta, u_2\delta\rangle$. Therefore (s'_1, s') is an instance of UL$'$.
- if $n = 2$, by construction we have $(s'_1, s'_2, s') = (s_1\delta, s_2\delta, s\delta)$ where (s_1, s_2, s) is an instance of a composition rule, say P' (the case of E' is similar), contained in GP. Let (s_1, s_2, s) be $(S_i\sigma/g^{\varepsilon_1} \vdash' u_1, S_i\sigma/g^{\varepsilon_2} \vdash' u_2, S_i\sigma/g^{\varepsilon_1\cup\varepsilon_2} \vdash' \langle u_1, u_2\rangle)$ (where $\varepsilon_1 \cup \varepsilon_2 = -$ iff $\varepsilon_1 = \varepsilon_2 = -$). By construction of GP', $\langle u_1, u_2\rangle \neq x\sigma$. Indeed, according to condition (b), all the instances of composition rules with $x\sigma$ as target of the root has been replaced in the construction of GP''. Therefore, $\langle u_1, u_2\rangle\delta = \langle u_1\delta, u_2\delta\rangle$. Thus, (s'_1, s'_2, s') is an instance of P$'$.
- if $n = 2$ and we have $(s'_1, s'_2, s') = (s_1\delta, s_2\delta, s\delta)$ where (s_1, s_2, s) is an instance of D' contained in GP. Let (s_1, s_2, s) be $(S_i\sigma/g^{\varepsilon_1} \vdash' \{u_1\}_{u_2}, S_i\sigma/g^{\varepsilon_2} \vdash' u_2^{-1}, S_i\sigma/g^{\varepsilon_1\cup\varepsilon_2} \vdash' u_1)$. By condition (a) on GP, $\{u_1\}_{u_2} \neq x\sigma$. Hence, $\{u_1\}_{u_2}\delta = \{u_1\delta\}_{u_2\delta}$. Therefore (s'_1, s'_2, s') is an instance of D'.

We have shown that GP'_1 and GP'_2 are both DY'–proofs. The condition (i) of the rule (Compare) of Figure 3 is ensured, and the other conditions (ii)–(iv), for GP, are preserved in the construction of GP'. Hence GP' if a guessing–proof of $(S_i\sigma)\delta \vdash g$. □