Theoretical Computer Science 538 (2014) 2–15

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Protocol insecurity with a finite number of sessions and a cost-sensitive guessing intruder is NP-complete

Pedro Adão [a,b,1], Paulo Mateus [a,c,1], Luca Viganò [d,*,2]

[a] SQIG, Instituto de Telecomunicações, Portugal
[b] Departamento de Engenharia Informática, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[c] Departamento de Matemática, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[d] Department of Informatics, King's College London, UK

ARTICLE INFO

Keywords:
Security
Protocols
Guessing
Verification
Complexity

ABSTRACT

Guessing attacks in security protocols arise when honest agents make use of data easily guessable by an intruder, such as passwords generated from a small dictionary. A way to model such attacks is to formalize a Dolev–Yao style model with inference rules that capture the additional capabilities of the intruder concerning guessable data. In this paper, we formalize a cost-sensitive intruder deduction system where information is available at a cost. The intruder may apply standard operations to deduce new messages from his current knowledge, or invoke an oracle rule that allows him to get hold of data that was previously unknown to him. Our system manipulates data items by means of inference rules and uses labels to keep track of the costs associated to the application of each rule. This allows us to answer the question of what is the cost of deducing a particular data that was meant to remain a secret between honest protocol participants. We also investigate the complexity of this quantitative insecurity problem and show that it is NP-complete in the case of a finite number of protocol sessions.

1. Introduction

A *Dolev–Yao* (*DY*) *intruder* [13] is an agent who completely controls the network over which security protocols are executed. He can impersonate other agents, prevent messages from reaching their destination, or reroute them to other agents. He can also generate messages from the knowledge he has acquired and send them to any agent, or decrypt messages using his knowledge. However, he cannot break cryptography, which is assumed to be *perfect*, i.e., the intruder cannot break an encryption unless he knows the corresponding key and cannot compute the content of a hashed message. One of the core problems of security protocol analysis is thus the so-called *intruder deduction problem*: given a state of the protocol execution, can the intruder deduce a given message, e.g., one that is intended to be secret such as a key shared between two honest agents? Deduction here is relative to the terms currently in the intruder knowledge, i.e., relative to the closure under a set of deduction rules of his initial knowledge augmented with the messages that he has observed so far during the protocol execution.

* Corresponding author.
E-mail addresses: pedro.adao@ist.utl.pt (P. Adão), pmat@math.ist.utl.pt (P. Mateus), luca.vigano@kcl.ac.uk (L. Viganò).

[1] Partially supported by FCT projects ComFormCrypt PTDC/EIA-CCO/113033/2009 and PEst-OE/EEI/LA0008/2013.

[2] Partially supported by the FP7-ICT-2009-5 Project no. 257876, "SPaCIoS: Secure Provision and Consumption in the Internet of Services", and by the PRIN 2010–2011 project "Security Horizons". Much of this work was carried out while Luca Viganò was at the Dipartimento di Informatica, Università di Verona, Italy.

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.09.015

Since the DY model abstracts away from cryptography, complexity, and probability, it reduces attacks to unexpected protocol interleavings that "leak" information to the intruder. Hence, following the terminology of [3,13–15,21], *protocol security* is reduced to checking a safety condition; roughly speaking, is it true that the intruder never obtains some private data? In general, for the case of an arbitrary number of protocol sessions that can be executed in an interleaved way, the problem is undecidable since the halting problem can be reduced to it (see, e.g., [15]). However, restricting it to a finite number of sessions, the problem of searching for such an attack in the DY model is an NP-complete problem [21] and checking if a protocol is secure is co-NP complete.

Guessing (or *dictionary*) *attacks* arise when an intruder exploits the fact that honest agents executing a security protocol make use of certain data like passwords that may have low entropy, e.g., stem from a small set of values, and thus may be easily guessable by the intruder. One way to model such attacks is to formalize a DY-style model with inference rules that capture the additional capabilities of the intruder concerning guessable data, e.g., decrypting a message that is encrypted with a guessable password. Several approaches (e.g., [2,7,8,10,11,17,18,23]) have been given for formal protocol analysis in the presence of an intruder who can perform *off-line guessing*, i.e., an intruder that employs guesses when analyzing observed messages.

In this paper, we follow an approach that differs from the previous ones in that we extend the classical DY intruder model by formalizing a cost-sensitive intruder deduction system where information is available at a cost: to get hold of data he does not know, the intruder may invoke an *Oracle* rule, which associates a cost to each data the intruder deduces in this way. As a consequence, we effectively drop the perfect cryptography assumption as the intruder is able to break cryptography, by guessing any message with the help of the oracle, but in a traceable and controlled way. This is achieved by our system manipulating data items (message terms) by means of inference rules that keep track of the associated costs by using labels, so that we can answer the question of what is the cost of deducing a particular data that was meant to remain a secret between honest protocol participants. In other words, in our approach the intruder deduction problem as formulated above is not relevant anymore since any term can now be deduced via the oracle rule. However, we can now focus on a different question: is it possible for the intruder to deduce a message given a fixed budget? We also investigate the complexity of this quantitative insecurity problem and show that it is NP-complete in the case of a finite number of protocol sessions (i.e., for a fixed number of interleaved protocol runs).

We proceed as follows. In Section 2, we define the language of our cost-sensitive guessing intruder, introduce the rules of the proposed deduction system based on rewriting, and discuss in detail the *Oracle* rule. In Section 3, we prove that all deductions in our system can be reduced to a normal form that satisfies a subterm principle, which characterizes the messages that may occur in a deduction. In Section 4, we define protocol executions and attacks to protocols, and in Section 5, we show that the resulting quantitative protocol insecurity problem is NP-complete in the case of a finite number of protocol sessions. In Section 6, we draw conclusions, discuss related work (Section 6.1) and compare the approach that we chose to follow with some possible alternatives (Section 6.2), and close the paper with some ideas for future work. Further examples and all proofs can be found in [1].

2. Modeling security protocols and the Dolev–Yao intruder

We define a deduction system, based on rewriting, that formalizes how the intruder can manipulate the messages he knows in order to generate new ones. We extend the classical DY intruder model by introducing an *Oracle* rule that allows the intruder to guess any message in a deduction, but at a cost. To keep track of the costs the intruder is incurring into in a deduction, we let the rules manipulate not only the messages but also the cost of generating new messages. The main aim of this deduction system is not to increase the number of messages the intruder can deduce, as he can obtain any message by an oracle call, but rather we are interested in how to manipulate and minimize the costs of deducing these messages. Costs thus represent a way to keep the use of the "omniscient" *Oracle* rule under control (and thus to interpret the behavior of this new intruder).

2.1. Language

Let V be a set of variables and Σ a set of function symbols, which is the union of disjoint sets $\bigcup_{n\in\mathbb{N}} \Sigma_n$, where Σ_0 is the set of *atomic messages* (or *Atoms*, for short) and Σ_n for $n > 0$ is the set of n-ary *function symbols*.

Definition 2.1 *(Message terms).* The set $\mathcal{M}_\Sigma(V)$ of *message terms* (or *messages*, for short) is defined inductively as follows: $Atoms \subseteq \mathcal{M}_\Sigma(V)$; $V \subseteq \mathcal{M}_\Sigma(V)$; $op(m_1, \dots, m_n) \in \mathcal{M}_\Sigma(V)$ if $op \in \Sigma_n$ and $\{m_1, \dots, m_n\} \subseteq \mathcal{M}_\Sigma(V)$.

We assume given a set of atomic keys $Keys \subset Atoms$ such that if $k \in Keys$ is a public key (respectively, private key), then $k^{-1} \in Keys$ and k^{-1} denotes the corresponding private key (respectively, public key).

Our results are general and independent of the particular language but, for brevity, in the following we will focus on the operators usually considered in protocol analysis and thus consider messages that are atomic or are composed (i) using *pairing*, denoted by $\langle m_1, m_2\rangle$ or, when there is no risk of confusion simply "m_1, m_2", or (ii) using the cryptographic operators $\{m_1\}_{m_2}$ for the *asymmetric* encryption of m_1 with m_2, $\{|m_1|\}_{m_2}$ for the *symmetric* encryption of m_1 with m_2, or $m_1(m_2)$ for the *application* of the (possibly composed) function m_1 to the message m_2, representing a hash-function or keytable.

$$\begin{array}{llll}
G_{pair}(\langle m_1, m_2\rangle): & m_1,\, m_2 & \rightarrow & m_1,\, m_2,\, \langle m_1, m_2\rangle \\
G_{crypt}(\{m_1\}_{m_2}): & m_1,\, m_2 & \rightarrow & m_1,\, m_2,\, \{m_1\}_{m_2} \\
G_{scrypt}(\{|m_1|\}_{m_2}): & m_1,\, m_2 & \rightarrow & m_1,\, m_2,\, \{|m_1|\}_{m_2} \\
G_{apply}(m_1(m_2)): & m_1,\, m_2 & \rightarrow & m_1,\, m_2,\, m_1(m_2) \\
A_{pair,i\in\{1,2\}}(\langle m_1, m_2\rangle): & \langle m_1, m_2\rangle & \rightarrow & \langle m_1, m_2\rangle,\, m_i \\
A_{crypt}(\{m_1\}_{m_2}): & \{m_1\}_{m_2},\, m_2^{-1} & \rightarrow & \{m_1\}_{m_2},\, m_2^{-1},\, m_1 \\
A^{-1}_{crypt}(\{m_1\}_{m_2^{-1}}): & \{m_1\}_{m_2^{-1}},\, m_2 & \rightarrow & \{m_1\}_{m_2^{-1}},\, m_2,\, m_1 \\
A_{scrypt}(\{|m_1|\}_{m_2}): & \{|m_1|\}_{m_2},\, m_2 & \rightarrow & \{|m_1|\}_{m_2},\, m_2,\, m_1
\end{array}$$

where in A_{crypt} and A^{-1}_{crypt} we have that $m_2, m_2^{-1} \in \mathit{Keys}$.

Fig. 1. Deduction rules for the classical DY intruder.

Any message m can be used as a key for symmetric encryption but only messages from *Keys* can be used for public-key encryption.

Other operators can be added in a straightforward way. Moreover, we consider the distinguished finite subset $\mathit{Names} \subset \mathit{Atoms}$ to represent agent names, including the name of the intruder *Charlie*; we could also consider, as is often done, a distinguished subset *Fresh*, used for the generation of fresh data, but this is not required for our results.

We can adapt the standard notions of *ground term*, *subterm*, and *degree*. Let $\mathit{vars}(m)$ be the set of all the variables in a message m. A message m is *ground* if it has no variables, i.e., $\mathit{vars}(m) = \{\}$, and the *set of ground messages* is represented by $\mathcal{M}_\Sigma$. The set $st(m)$ of *subterms* of a message is the set of message terms that are subterms of m, including m itself. A *toplevel subterm* is an *immediate* subterm of a term, e.g., $m_1(m_2)$ and m_3 are the toplevel subterms of $\{m_1(m_2)\}_{m_3}$. The *degree* of a message m is defined recursively as $deg(\mathit{Charlie}) = 0$, $deg(m) = 1$ for all $m \in \mathit{Atoms} \setminus \{\mathit{Charlie}\}$, and for an $op \in \Sigma_n$, $deg(op(m_1, \dots, m_n)) = 1 + deg(m_1) + \dots + deg(m_n)$. A (ground) *substitution* σ assigns to each variable a (ground) message, where the application of a substitution σ to a message m is denoted by $m\sigma$. We extend these notions to sets of messages in the expected way, e.g., $M\sigma = \{m\sigma \mid m \in M\}$.

Definition 2.2 *(Interpretation of terms).* Let us consider a function v that provides the *interpretation* of the terms of our language. Messages are then simply interpreted in the free algebra of messages: $v(m) = m$ and $v(M) = \{v(m_1), \dots, v(m_n)\} = M$ for $M = \{m_1, \dots, m_n\}$.

2.2. Deduction rules

In Fig. 1, we show the standard (e.g., [4,21]) rewriting rules for a classical DY intruder, which are divided into two groups: (i) *Generation rules* (or *G-rules*), which express that the intruder can compose messages from known messages using pairing, asymmetric and symmetric encryption, and function application, and (ii) *Analysis rules* (or *A-rules*), which describe how the intruder can decompose messages.

We name rules accordingly, e.g., G_{scrypt} and A_{scrypt} are the G-rule and A-rule associated with the symmetric encryption operation, respectively. Note that no rules are given that allow the intruder to analyze function applications, e.g., to recover m_2 from $m_1(m_2)$. Note also that each of our A-rules is such that the conclusion is always a subterm of the premise that is decomposed, that the other premise or its inverse is also a subterm of the premise that is decomposed, and that each of our G-rules is such that its premises are subterms of the conclusion and all toplevel subterms of the conclusion are premises of the rule; this will be exploited in several of the results below (e.g., Theorems 3.3 and 3.6).

Although we have not yet defined deductions (which we will do in Definition 2.5), the following example should be understandable enough to illustrate the main ideas underlying our approach.

Example 2.3. Assume that the intruder knows the messages $S_0 = \{N_b,\ \{|\langle m_1, N_a\rangle|\}_k\}$ and wishes to build the message $\langle N_a, N_b\rangle$. Clearly, a classical DY intruder cannot do that, as can be formally shown by considering that the only possible deduction from S_0 aiming to obtain $\langle N_a, N_b\rangle$ in this deduction system is the following one[3]:

$$\begin{array}{ll}
 & N_b,\ \left\{\left|\langle m_1, N_a\rangle\right|\right\}_k \\
\rightarrow_{?} & N_b,\ \left\{\left|\langle m_1, N_a\rangle\right|\right\}_k,\ ?k \\
\rightarrow_{A_{scrypt}(\{|\langle m_1, N_a\rangle|\}_k)} & N_b,\ \left\{\left|\langle m_1, N_a\rangle\right|\right\}_k,\ ?k,\ \langle m_1, N_a\rangle \\
\rightarrow_{A_{pair,2}(\langle m_1, N_a\rangle)} & N_b,\ \left\{\left|\langle m_1, N_a\rangle\right|\right\}_k,\ ?k,\ \langle m_1, N_a\rangle,\ N_a \\
\rightarrow_{G_{pair}(\langle N_a, N_b\rangle)} & N_b,\ \left\{\left|\langle m_1, N_a\rangle\right|\right\}_k,\ ?k,\ \langle m_1, N_a\rangle,\ N_a,\ \langle N_a, N_b\rangle
\end{array}$$

[3] In fact, to show that no other deduction is possible, one should exploit formal results on the normalization of the deductions in the classical DY setting, which are widely available, e.g., [21]; we return to this below.

$$\begin{array}{llcl}
G_{pair}(f_{G_{pair}}(m_1, m_2) : \langle m_1, m_2\rangle): & m_1, m_2 & \rightarrow & m_1, m_2, \langle m_1, m_2\rangle \\
G_{crypt}(f_{G_{crypt}}(m_1, m_2) : \{m_1\}_{m_2}): & m_1, m_2 & \rightarrow & m_1, m_2, \{m_1\}_{m_2} \\
G_{scrypt}(f_{G_{scrypt}}(m_1, m_2) : \{|m_1|\}_{m_2}): & m_1, m_2 & \rightarrow & m_1, m_2, \{|m_1|\}_{m_2} \\
G_{apply}(f_{G_{apply}}(m_1, m_2) : m_1(m_2)): & m_1, m_2 & \rightarrow & m_1, m_2, m_1(m_2) \\
A_{pair,i}(f_{A_{pair,i}}(\langle m_1, m_2\rangle) : \langle m_1, m_2\rangle): & \langle m_1, m_2\rangle & \rightarrow & \langle m_1, m_2\rangle, m_i \\
A_{crypt}(f_{A_{crypt}}(\{m_1\}_{m_2}, m_2^{-1}) : \{m_1\}_{m_2}): & \{m_1\}_{m_2}, m_2^{-1} & \rightarrow & \{m_1\}_{m_2}, m_2^{-1}, m_1 \\
A_{crypt}^{-1}(f_{A_{crypt}^{-1}}(\{m_1\}_{m_2^{-1}}, m_2) : \{m_1\}_{m_2^{-1}}): & \{m_1\}_{m_2^{-1}}, m_2 & \rightarrow & \{m_1\}_{m_2^{-1}}, m_2, m_1 \\
A_{scrypt}(f_{A_{scrypt}}(\{|m_1|\}_{m_2}, m_2) : \{|m_1|\}_{m_2}): & \{|m_1|\}_{m_2}, m_2 & \rightarrow & \{|m_1|\}_{m_2}, m_2, m_1 \\
Oracle(f_{Oracle}(m, M) : m): & & \rightarrow & m
\end{array}$$

where in A_{crypt} and A_{crypt}^{-1} we have that $m_2, m_2^{-1} \in Keys$ and the *Oracle* rule has the side condition that m is ground, i.e., $m \in \mathcal{M}_\Sigma$.

Fig. 2. Deduction rules for the cost-sensitive intruder.

This deduction is, however, incomplete, in the sense that the key k is not known and thus remains as an open assumption (as highlighted by the "?" in front of the term and in the inference rule). But our cost-sensitive intruder has the possibility of asking for the help of an oracle and we can thus complete the deduction as follows:

$$\begin{array}{ll}
 & N_b, \{|\langle m_1, N_a\rangle|\}_k \\
\rightarrow_{Oracle(\ell_1 : k)} & N_b, \{|\langle m_1, N_a\rangle|\}_k, k \\
\rightarrow_{A_{scrypt}(\ell_2 : \{|\langle m_1, N_a\rangle|\}_k)} & N_b, \{|\langle m_1, N_a\rangle|\}_k, k, \langle m_1, N_a\rangle \\
\rightarrow_{A_{pair,2}(\ell_3 : \langle m_1, N_a\rangle)} & N_b, \{|\langle m_1, N_a\rangle|\}_k, k, \langle m_1, N_a\rangle, N_a \\
\rightarrow_{G_{pair}(\ell_4 : \langle N_a, N_b\rangle)} & N_b, \{|\langle m_1, N_a\rangle|\}_k, k, \langle m_1, N_a\rangle, N_a, \langle N_a, N_b\rangle
\end{array}$$

where we justify the intruder's use of the key k for message decryption by means of the *Oracle* rule and keep track of the costs of the inference by adding labeled messages as parameters of the rules. So, in order to capture situations like this, we just need to explain how to deal with the *Oracle* rule and with the labeled messages.

The deductive power of our cost-sensitive intruder is in all similar to that of the classical DY intruder, since the way the message terms are manipulated remains the same. The crucial difference is the possibility that the intruder has to increase his knowledge at any point in a deduction, if he is willing to pay a price for the added information, which we formalize by exploiting the labels in the parameters of the rules. The rules of our cost-sensitive intruder, deducing messages and keeping track of the costs via labels, are given in Fig. 2, where for simplicity we have used the same names as in Fig. 1. In this case, rules have the form

$$X(\ell : m): \quad m_1, \dots, m_n \rightarrow m_1, \dots, m_n, m'$$

and tell us that with rule X we can deduce the conclusion m', with cost ℓ, from the premises $m_1, \dots, m_n$, where m is either the conclusion of the rule (G-rules and *Oracle*) or the decomposed premise of the rule (A-rules). To each rule X, we associate a function f_X that assigns a cost ℓ to the conclusion of X.

We will use $X(\ell : m)$, $G(\ell : m)$, and $A(\ell : m)$ to refer respectively to an arbitrary rule, an arbitrary G-rule, and an arbitrary A-rule. We may also omit the parameter $\ell : m$ when not necessary for the discussion.

At each step during the execution of a protocol, the intruder increases his initial knowledge (consisting of his own public, private, and shared keys, his own name and the names and public keys of agents, hash-functions, etc.) with the messages that he observes during the execution, as well as with the messages that he deduces from his current knowledge, and with the messages obtained via oracle calls. Formally, every time the intruder deduces a new message, he has to pay a price for the new information. Hence, our inference rules manipulate messages (as premises and conclusions) and *labeled messages* (ℓ, m) that are composed of the message term m, that is either the conclusion of the rule (G-rules and *Oracle*) or the decomposed premise of the rule (A-rules), and of a *label term* ℓ, which represents the cost of deducing that message. For brevity, we will often refer to label terms and labeled message terms just as *labels* and *labeled messages*.

To generate the label terms, we consider a set of function symbols, one function f_X for each rule X of the system, where the arity of f_X is the number of premises of X, except for the function $f_{Oracle}(m, M)$ of the oracle rule, which takes as argument the guessed message and the current intruder knowledge (we will return to this in Section 2.3 in more detail). The intuition is that $f_X = \ell$ represents the cost of deducing m when applying the rule X.[4]

Definition 2.4 *(Rules: principal terms, minor and major premises).* Each rewriting rule $X(\ell : m) : lhs \rightarrow rhs$ has an argument m, also called the *principal term* of the rule, and a parameter $\ell : m$ that associates, via the function f_X, a cost ℓ to the application of this rule, and has on the left-hand side (lhs) its premises and on the right-hand side (rhs) its conclusions. We call the premises of an A-rule *minor premises* except for its principal term, which we call its *major premise*.

[4] Alternatively, but at the cost of additional technical complexity, we could also consider a set of *introduction functions*, which tell us the initial cost of a message, and *propagation functions*, which tell us how costs are propagated from hypothesis to conclusions of rules.

For simplicity, we may also use $X(\ell : m) : m_1, \dots, m_n \to lhs, m'$ or even $X(\ell : m) : lhs \to lhs, m'$ to avoid (re-)writing the *lhs* of the rule. We may also call m' the conclusion of the rule ignoring the conclusions that are also premises.

We adapt some standard definitions and results. Having labeled messages instead of simply messages, albeit only as parameters of the rules, requires us to define how we interpret the labels (message terms are interpreted in the free algebra of messages, as we defined above). We thus assume that the interpretation v also assigns values to labels.

Definition 2.5 *(Deductions).* Given a set M of messages, we write $M \to_X M'$ if there exist sets *lhs* and *rhs* of messages and a rule $X : lhs \to rhs$ such that $lhs \subseteq M$ and $M' = M \cup rhs$.

A *deduction* Π of a message m from the set of messages M, denoted by $M \vdash m$, is a sequence $M_0 \to_{X_1} M_1 \to_{X_2} \cdots \to_{X_n} M_n$ such that $m \in M_n$ and where $M_0 = M$. We say in this case that Π has *length* n, *goal* m, and (abusively) that each $X_i \in \Pi$.

If, for some i, $X_i = A(\ell : m)$, then we say that m is *decomposed* in Π; if $X_i = Oracle(f_{Oracle}(m_i, M_{i-1}) : m_i)$, then we call M_{i-1} the *support-set* of X_i. We also define $msg(\Pi) = \bigcup_{i=0}^{n} M_i$ and the *total cost of deduction* $v(\Pi) = \sum_{i=1}^{n}\{v(\ell_i) \mid X_i(\ell_i : m_i) \in \Pi\}$ to be the sum of all cost labels.

Note that in our rules, $lhs(X) \subseteq rhs(X)$ so, given a deduction $\Pi = M_0 \to_{X_1} M_1 \to_{X_2} \cdots \to_{X_n} M_n$, we have, for all i, that $M_i \subseteq M_{i+1}$ and $msg(\Pi) = M_n$.

In this paper, we give an abstract presentation and do not commit ourselves to particular cost functions nor to a fixed interpretation function v, whose choice largely depends on the case study under consideration. We just generically require that the oracle function f_{Oracle} is interpreted as a cost function that satisfies $\overline{f_{Oracle}} : (\mathcal{M}_\Sigma) \times \mathcal{P}(\mathcal{M}_\Sigma) \mapsto \mathbb{R}_0^+$, and that every other function f_X is interpreted as a cost function that satisfies $\overline{f_X} : (\mathcal{M}_\Sigma)^n \mapsto \mathbb{R}_0^+$, where n is the number of premises of X. We also require that all cost functions are computable in polynomial-time.

Example 2.6. For example, we could consider the cost of pairing, $f_{G_{pair}}$, simply to be the sum of the sizes of messages m_1 and m_2. That is, we could refine the G_{pair} rule of Fig. 2 to

$$G'_{pair}\big(sum\big(size(m_1), size(m_2)\big) : \langle m_1, m_2\rangle\big): \quad m_1, m_2 \to m_1, m_2, \langle m_1, m_2\rangle$$

assuming that $v(sum) = +$ and $v(size) = |\dots|$, where $|m|$ computes the size of a message m. Similarly, we could have the cost of symmetric encryption be a multiplication

$$G'_{scrypt}\big(times\big(size(m_1), size(m_2)\big) : \{|m_1|\}_{m_2}\big): \quad m_1, m_2 \to m_1, m_2, \{|m_1|\}_{m_2}$$

with $v(times) = \times$.[5]

For concreteness, using the rules of the cost-sensitive intruder, we can rewrite the deduction of Example 2.3 as

$$\begin{array}{ll}
 & N_b,\ \{|\langle m_1, N_a\rangle|\}_k \\
\to Oracle(f_{Oracle}(k, \{N_b, \{|\langle m_1,N_a\rangle|\}_k\}): k) & N_b,\ \{|\langle m_1, N_a\rangle|\}_k,\ k \\
\to A_{scrypt}(f_{A_{scrypt}}(\{|\langle m_1,N_a\rangle|\}_k,k):\{|\langle m_1,N_a\rangle|\}_k) & N_b,\ \{|\langle m_1, N_a\rangle|\}_k,\ k,\ \langle m_1, N_a\rangle \\
\to A_{pair,2}(f_{A_{pair,i}}(\langle m_1,N_a\rangle): \langle m_1,N_a\rangle) & N_b,\ \{|\langle m_1, N_a\rangle|\}_k,\ k,\ \langle m_1, N_a\rangle,\ N_a \\
\to G_{pair}(f_{G_{pair}}(N_a,N_b):\langle N_a,N_b\rangle) & N_b,\ \{|\langle m_1, N_a\rangle|\}_k,\ k,\ \langle m_1, N_a\rangle,\ N_a,\ \langle N_a, N_b\rangle
\end{array}$$

and if the cost of pairing and of symmetric decryption is given by addition and multiplication, respectively, and the cost of projection is 0, then the cost of deducing $\langle N_a, N_b\rangle$ is

$$f_{Oracle}\big(k, \{N_b, \{|\langle m_1, N_a\rangle|\}_k\}\big) + \big(|\langle m_1, N_a\rangle_k| \times |k|\big) + 0 + \big(|N_a| + |N_b|\big)$$

Given that the *Oracle* rule is at the core of our approach, we discuss in detail the requirements for its cost function in the following subsection.

[5] We could even add special rules for the case when $m_2 = m_1$ so that the result of pairing and encryption is just the size of m_1, i.e.,

$$\begin{array}{lll}
G''_{pair}\big(size(m_1) : \langle m_1, m_1\rangle\big): & m_1 \to & m_1,\ \langle m_1, m_1\rangle \\
G''_{scrypt}\big(size(m_1) : \{|m_1|\}_{m_2}\big): & m_1 \to & m_1,\ \{|m_1|\}_{m_1}
\end{array}$$

We believe that, in this context, it will be particularly interesting to consider possible connections to, and synergies with, [5], in which Baudet extends existing Dolev–Yao models with systems with computation times and probabilities to account for random polynomial-time computability. We leave this investigation for future work.

2.3. The Oracle rule

The idea behind the *Oracle* rule is to allow the intruder to make use of any ground message $m \in \mathcal{M}_\Sigma$ (cf. the rule's side condition) at any point in a deduction. We do not restrict the use of this rule and so the intruder is free to make as many *Oracle* calls as he needs to proceed with a given deduction. It is clear that this is a very powerful rule and provides an unrealistic solution to the intruder deduction problem: *can a given message be deduced from the current intruder knowledge*[6]? For this rule to make sense, we thus need a way to control its application and consider the costs the intruder incurs into when applying it, and at the same time to reformulate the problem itself. The interesting question is not *what message can be deduced* with the *Oracle* rule—as all ground messages are—but rather *what is the cost of deducing it*. Each application of the rule incurs into a cost and it is our goal to see what can be deduced within a certain budget. This amounts to the following optimization problem:

Problem 2.7 *(Quantitative Intruder Deduction Problem).* Given a set of messages M, a goal $m \in \mathcal{M}_\Sigma$, and a budget $\gamma \in \mathbb{R}$, is there a deduction $M \vdash m$ such that $v(M \vdash m) \leqslant \gamma$?

In Section 5, we study the complexity of this problem and show that it is NP-complete. To that end, let us formalize the properties we require of the *Oracle* rule. For any set M of messages and goal $m \in \mathcal{M}_\Sigma$, we require the cost function $\overline{f_{Oracle}}$ to satisfy the following condition:

$$\overline{f_{Oracle}}(m, M) \geqslant card\big(st(M, m)\big) \tag{1}$$

where *card* returns the cardinality of a set. The *rationale* behind this condition is that to apply the *Oracle* rule the intruder needs at least to "read" the messages in the support-set M and to "write" the result. In Section 6.2, we present alternatives to this condition discussing the advantages and disadvantages of the different solutions.

Having justified and explained the *Oracle* rule, we can take stock and describe our approach in more detail. In a nutshell, the deduction system we introduced amounts to extending the classical DY intruder model by allowing attacks on cryptography. This is so since in the classical model the intruder can only decrypt messages if he knows the corresponding key, but with the *Oracle* rule we have given the intruder access to any message, in particular any key, at a cost to be determined by the oracle (e.g., it is not unreasonable to assume that passwords or weak keys can be broken in feasible time, if enough resources are allocated to the task). Our work is thus motivated by questions such as: Can focused brute-force attack expose a vulnerability in a security protocol, if it is performed against the right key in the right moment? And, assuming that the intruder has access to an unusual amount of computational resources for a limited time, which is the best way for him to use these resources to attack a protocol?

By assuming that cryptography can be subjected to an attack and allowing the intruder, through the *Oracle* rule, to make use of any message and quantifying the effort necessary to obtain it, we thus provide a formal system to reason about protocol attacks triggered by the use of poorly protected data, i.e., data that is retrievable by investing an acceptable amount of resources.

3. Normal forms and subterm principle

Results on the normalization of deductions in the classical DY setting are widely available, e.g., [21]. Deductions are transformed and shaped in a canonical way, its normal form, by removing unnecessary rule applications and eliminating redundancies. In our case, we prove a normalization result for our labeled deductions by extending the results applied in the classical case. This extension is, however, not trivial as we must take into account the labels and thus, ultimately, the cost of the original deduction and the normalized one.

3.1. Normalization

In general, normalization is based on the observation that the A-rules are, in a sense, the inverse of the corresponding G-rules: applying an A-rule to the conclusion of a G-rule essentially restores what had already been established. This relationship between A-rules and G-rules is called the *inversion principle*. If this principle is not taken into account, then it is possible to have arbitrarily long deductions simply by combining G-rules and A-rules. However, by taking the principle into account, we can reduce a deduction to some normal form, provided that we take care of the labels and of the possible detours introduced by the *Oracle* rule.

[6] For instance, returning to Example 2.3, another possible deduction of $\langle N_a, N_b\rangle$ is just

$$Oracle\big(f_{Oracle}\big(\langle N_a, N_b\rangle, \{N_b, \{|\langle m_1, N_a\rangle|\}_k\}\big) : \langle N_a, N_b\rangle\big): \quad \rightarrow \langle N_a, N_b\rangle$$

since it is always possible to deduce any ground message in one step using the *Oracle* rule.

Definition 3.1 *(Non-optimal application of rules, normal deductions).* Given a deduction Π of m, of the form $M_0 \rightarrow_{X_1} M_1 \rightarrow_{X_2} \cdots \rightarrow_{X_n} M_n$, we say that $X_i \in \Pi$ is *applied non-optimally* if one of the following holds:

(i) $m \in M_{i-1}$; or
(ii) $M_i = M_{i-1}$; or
(iii) $i < n$, $m_i \in M_i \setminus M_{i-1}$ and for all $i < j, j' \leqslant n$, X_j is not an *Oracle* rule and $m_i \notin lhs(X_{j'})$.

A deduction $M \vdash m$ is *normal* if it does not contain non-optimal application of rules.

A deduction is "compressed" if one can remove all its non-optimal applications of rules. Condition (i) ensures that we do not perform any further step as soon as our goal is reached, whereas (ii) ensures that each rule application increases the intruder knowledge by adding at least one new message. Condition (iii) ensures that we do not deduce unnecessary messages unless an *Oracle* rule is applied in the future, in which case the deduced message is needed as it may reduce the cost of the *Oracle* deduction.

As a consequence of Definition 3.1, we can show in Proposition 3.2 that we never apply an A-rule to a message that resulted from a previous application of a G-rule, and vice-versa, as this would restore what had already been established.

Proposition 3.2. *Let Π be a normal deduction $M_0 \rightarrow_{X_1} M_1 \rightarrow_{X_2} \cdots \rightarrow_{X_n} M_n$. Then*

(i) *if $X_i = A(\ell_i : m_i)$, then there is no $j < i$ such that $X_j = G(\ell_j : m_i)$;*
(ii) *if $X_i = G(\ell_i : m_i)$, then there is no $j < i$ such that $X_j = A(\ell_j : m_i)$.*

Contrary to the classical case where finding a normalization procedure is enough, in our case we also need the normalized deduction to not increase the cost of the deduction. This is because we want to study minimal attacks and so normalized deductions should be at most as costly as the original ones. We can now extend standard results of proof theory and give a normalization procedure that yields a normalized reduction by removing redundancies and unnecessary rule applications from the deductions.

Theorem 3.3 *(Normalization). If Π is a deduction $M \vdash m$, then there exists a normal deduction $M \vdash m$, denoted by Π', such that $v(\Pi') \leqslant v(\Pi)$.*

3.2. Subterm principle

In this subsection, we show that the messages in any normal deduction that does not use the oracle rule are subterms of its premises or of its conclusion. This is especially important since it bounds the number of terms that need to be deduced and the number of rules that need to be applied in a deduction, a fact that is fundamental to establish decidability and complexity bounds for our cost-sensitive intruder. The number of applications of the *Oracle* rule is then bounded via a complexity argument.

Proposition 3.4. *Let Π be normal deduction $M \vdash m$. If m_i is decomposed in Π, i.e., is a major premise of an A-rule, and there are no previous applications of the Oracle rule, then m_i is a subterm of a labeled message in $M_0 = M$.*

We can exploit Proposition 3.4 to show a property about subterms and then the subterm principle theorem.

Proposition 3.5. *Let Π be a normal deduction $M \vdash m$ with no applications of the Oracle rule. Then, for any $m_i \in msg(\Pi)$, either*

(i) *m_i is a subterm of a message in M_0, or*
(ii) *m_i is not decomposed in Π and is the conclusion of a G-rule in Π.*

Theorem 3.6 *(Subterm principle). Let Π be normal deduction $M \vdash m$ with no applications of the Oracle rule. Then, every message in $msg(\Pi)$ is either*

(i) *a subterm of the conclusion m, or*
(ii) *a subterm of a message in M.*

Remark 3.7. Throughout the rest of this paper we will assume, without loss of generality, that all deductions are normal.

4. Protocol specifications and protocol attacks

To focus our attention on the problem of protocol (in)security, we first need to formalize what protocol specification and execution are. We will be brief, as these definitions are quite standard.

A *security protocol* can be defined as an ordered sequence of steps for each principal, i.e., as a finite partially-ordered set of execution steps $(W_A, \leqslant_A)$ for each protocol agent A. An execution step is a pair of terms $R \Rightarrow S$ that represents that whenever message R is received by the agent, it replies with message S. A *protocol specification* is then given by

$$\{(i, R_i \Rightarrow S_i) \mid i \in \mathcal{I}\}$$

where the set $\mathcal{I} = \{(A, w) \mid A \in \mathit{Names} \text{ and } w \in W_A\}$ is the *execution order*. We denote by $|\mathcal{I}|$ the *size of the protocol*. We use dummy messages *Init* and *End* to initiate and close a protocol session. We also require that the variables occurring in S_i were already instantiated in a previous step (of the principal), i.e., that, for all $A \in \mathit{Names}$, we have $\mathit{vars}(S_{(A,w)}) \subseteq \bigcup_{w' \leqslant_A w} \mathit{vars}(R_{(A,w')})$.

A *correct execution order* is a one-to-one mapping $\omega : \mathcal{I} \to \{1, \ldots, |\mathcal{I}|\}$ that respects the partial order of each agent, i.e., for all $A \in \mathit{Names}$ and $j \leqslant_A j'$ we have $\omega(A, j) \leqslant \omega(A, j')$. We can also consider the partial order $(W_A^k, \leqslant_A^k)$ to specify k protocol sessions of an agent A where $(w_1, \ldots, w_k) \leqslant_A^k (w'_1, \ldots, w'_k)$ if $w_j \leqslant_A w'_j$ for all $j \in 1, \ldots, k$, i.e., each of the k sessions respects the agent's partial order.

An *execution environment* for a protocol is a set of messages E. A ground substitution σ, a correct execution order ω, and a sequence of execution environments $E_0, \ldots, E_{|\mathcal{I}|}$ define a *protocol execution* if (i) $\mathit{Init} \in E_0$, (ii) $\mathit{End} \in E_{|\mathcal{I}|}$, and (iii) for all $1 < k \leqslant |\mathcal{I}|$, $R_{\omega^{-1}(k)}\sigma \in E_{k-1}$ and $S_{\omega^{-1}(k)}\sigma \in E_k$.

Note that, as we are considering a finite number of sessions, we may assume that all nonces generated by the agents are already in their initial knowledge.

Let us now consider attacks. An attack to a protocol is no more than a correct execution of the protocol where the attacker interacts with the protocol, collects information, and from that information is able to retrieve some secret term. In our case, in addition to the standard monotonicity property of the deduction, we also need to consider that, if a term was deduced (and consequently payed for) in an earlier step of the attack, the cost of deducing it again should be 0. To account for this, we explicitly add those terms to the intruder knowledge after each step of the attack. We show, however, that with these extra hypothesis the deductive power of the intruder is the same and only the cost of deduction decreases.

Proposition 4.1. *We have a sequence of deductions $\Pi_j = \{M_0, \ldots, M_{j-1}\} \cup \mathit{msg}(\Pi_1) \cup \cdots \cup \mathit{msg}(\Pi_{j-1}) \vdash m_j$, for $j \geqslant 1$, if and only if we have a sequence of deductions $\Pi_j^- = \{M_0, \ldots, M_{j-1}\} \vdash m_j$. Moreover $\mathit{msg}(\Pi_j) = \mathit{msg}(\Pi_j^-)$.*

Definition 4.2 *(Protocol attack).* Let $P = \{(i, R'_i \Rightarrow S'_i) \mid i \in \mathcal{I}\}$ be a protocol specification, $m \in \mathcal{M}_\Sigma$ a secret term, and $S_0 \supseteq \{\mathit{Charlie}\}$ a set of messages that constitutes the initial intruder knowledge. Moreover, assume that there exists a ground substitution σ and an execution order $\omega : \mathcal{I} \to 1, \ldots, k$, such that for all $j \in 1, \ldots, k$, Π_j is the deduction $\{S_0\sigma, \ldots, S_{j-1}\sigma\} \cup \mathit{msg}(\Pi_1) \cup \cdots \cup \mathit{msg}(\Pi_{j-1}) \vdash R_j\sigma$ and Π is the deduction $\{S_0\sigma, \ldots, S_k\sigma\} \cup \mathit{msg}(\Pi_1) \cup \cdots \cup \mathit{msg}(\Pi_k) \vdash m$, where $R_j = R'_{\omega^{-1}(j)}$ and $S_j = S'_{\omega^{-1}(j)}$.

We say that σ and ω constitute an *attack on protocol P that exposes the secret m when the initial intruder knowledge is S_0*. Further, the *cost of this attack* is computed by adding the intermediate costs of each deduction plus the cost of deducing the secret, i.e., $\mathit{attackCost} = \sum_{j=1}^{k} v(\Pi_j) + v(\Pi)$.

To simplify our notation, when in the presence of a protocol attack (σ, ω), we will use Hyp_j to denote $\{S_0\sigma, \ldots, S_{j-1}\sigma\} \cup \mathit{msg}(\Pi_1) \cup \cdots \cup \mathit{msg}(\Pi_{j-1})$ and Hyp_j^- to denote $\{S_0\sigma, \ldots, S_{j-1}\sigma\}$.

Note that in fact a protocol attack also depends on the Π_j as *attackCost* depends on that. However, and in order to simplify our discussion, we will consider an attack to be composed of just (σ, ω) rather than $(\sigma, \omega, \{\Pi_j\}_j, \Pi)$.

We can now define the Quantitative Protocol Insecurity Problem.

Problem 4.3 *(Quantitative Protocol Insecurity (QPI) Problem).* Given a protocol specification P, initial intruder knowledge S_0, a goal $m \in \mathcal{M}_\Sigma$, and a budget γ, is there a ground substitution σ and an execution order ω such that (σ, ω) is an attack to P and $\mathit{attackCost} \leqslant \gamma$?

5. The Quantitative Protocol Insecurity Problem is in NP

In this section, we give our main result by proceeding as follows. We start with the definition of normal attack and with the introduction of the relevant complexity notions and restrictions that our cost functions need to satisfy. Then, we show that for deductions that do not use the *Oracle* rule, $\sigma(x)$, for all $x \in V$, is unifiable with some term already in the protocol description. This allows us to bound the size of the substitutions and we do that reusing a result from [21]. We can then bound the size of attacks (for deductions that do not use *Oracle*) in terms of the number of messages and length and, using a complexity argument, we extend these to arbitrary deductions. We conclude this section proving that the QPI Problem is NP-complete. Let us start with some preliminary definitions.

A *finite multiset* over natural numbers is a function $ms : \mathbb{N} \nrightarrow \mathbb{N}$ with a finite domain. We will denote by $\{1, 1, 2, 2, 2, 4\}$ the function ms such that $ms(1) = 2, ms(2) = 3, ms(4) = 1$ and $ms(x)$ is undefined for all other $x \in \mathbb{N}$. We extend the ordering in $\mathbb{N}$ to multisets as $ms_1 <_{ms} ms_2$ if (i) $ms_1 \neq ms_2$, and (ii) if for some x, $ms_1(x) > ms_2(x)$ then there exists $y > x$ such that $ms_1(y) < ms_2(y)$. As an example $\{2, 2, 2, 2, 4\} <_{ms} \{1, 5\}$.

We can then define normal attacks as the "simplest" and "cheapest".

Definition 5.1 *(Normal attack).* Given a protocol specification $P = \{(i, R'_i \Rightarrow S'_i) \mid i \in \mathcal{I}\}$, an attack (σ, ω) is *normal* if there is no other attack (σ^*, ω^*) such that $\{deg(R^*_1\sigma^*), \ldots, deg(R^*_k\sigma^*)\} <_{ms} \{deg(R_1\sigma), \ldots, deg(R_{k'}\sigma)\}$ and $attackCost^* \leqslant attackCost$, where $R_j = R'_{\omega^{-1}(j)}$ and $S_j = S'_{\omega^{-1}(j)}$, and $R^*_j = R'_{\omega^{*-1}(j)}$ and $S^*_j = S'_{\omega^{*-1}(j)}$.

By well-foundedness of $<_{ms}$ over non-negative integers, if there is an attack, there is also a normal attack. Note, however, that the existence of a normal attack is not unique and that two attacks may be incomparable.

5.1. Cost after substitution by a simpler message

In order to prove our results, the cost functions need to satisfy one extra property that requires that if one replaces a message by another one with a smaller degree, then the cost of the new deduction should be at most the same as the cost of the original one. So, let Π be a deduction $M \vdash m$ that does not use the *Oracle* rule, messages $m_1, m_2 \in \mathcal{M}_\Sigma$ be such that $deg(m_2) < deg(m_1)$, and $t\delta$ be the message obtained from t after replacing m_1 by m_2. We require that if Π^δ is the deduction $M\delta \vdash m\delta$ then

$$v\left(\Pi^\delta\right) \leqslant v(\Pi) \tag{2}$$

We show in Lemma 5.3 that a sufficient condition for the existence of $M\delta \vdash m\delta$ is that m_1 is never decomposed in Π.

We can easily see that if the rules used in Π^δ are the same as the ones used in Π (up to δ), a sufficient condition that enforces (2) is to require each of the resulting rules to cost less than the original one, that is,

$$v\left(f_{A_{pair,i}}\left(m'\delta\right)\right) \leqslant v\left(f_{A_{pair,i}}\left(m'\right)\right)$$
$$v\left(f_{op}\left(m'\delta, m''\delta\right)\right) \leqslant v\left(f_{op}\left(m', m''\right)\right) \quad \text{for all the other operators}$$

While these latter restrictions are point-wise and not over a deduction, hence easier to check, they are more restrictive as there are cost functions that could satisfy (2) and not satisfy them. For that reason, we leave (2) as the required property.

5.2. Bounding the size of messages used in attacks

We can now bound the size of normal attacks. Suppose that there exists an attack (σ, ω) against P and let as usual $R_i = R'_{\omega^{-1}(i)}$ and $S_i = S'_{\omega^{-1}(i)}$ for all $i \in \{1, \ldots, k\}$. Define $st(P)$ as the set of subterms of the terms in $S_0 \cup \{R_j, S_j \mid j = 1, \ldots, k\}$, and $st_{\leqslant i}(P)$ as the set of subterms in $S_0\sigma \cup \{R_j\sigma, S_j\sigma \mid j = 1, \ldots, i\}$. Recall that we also assumed $Charlie \in S_0$ and $deg(Charlie) = 0$. By definition of (σ, ω) being an attack, we have that (i) for all $1 \leqslant j \leqslant k$, $Hyp_j \vdash R_j\sigma$, and (ii) $Hyp_{k+1} \vdash m$. Let us call Π_j each such deduction. By Remark 3.7, we may assume without loss of generality that each Π_j is normal.

Definition 5.2. Let t and t' be two terms and θ a ground substitution. Then t is a θ-*match* of t', denoted by $t \sqsubseteq_\theta t'$, if t is not a variable and $t\theta = t'$.

The next lemma is one of the fundamental results as it bounds the size of the substitutions in normal attacks. In particular, we show that substitutions in normal attacks, also called *normal substitutions*, only use terms already in the protocol specification, hence bounded by the size of the specification of the protocol. The proof is adapted from the one in [21] considering both the costs of deductions and the size of its conclusions. The proof strategy is the following. We assume, for the sake of contradiction, that given a normal attack (σ, ω), there is a variable x such that $\sigma(x)$ is not unifiable with any subterm (different from a variable) of the protocol specification. The reasoning proceeds by first showing that $x \in st(R_i)$ for some i and that $\sigma(x) \in msg(\Pi_i)$; then that $\sigma(x)$ is never decomposed in the deductions Π_j; and finally that replacing $\sigma(x)$ by a "simpler" message yields a "simpler" attack contradicting the normality of the attack (σ, ω). The details can be found in [1].

Lemma 5.3. *If (σ, ω) is a normal attack that does not use the Oracle rule, then, for all variables $x \in V$, there exists $t \in st(P)$ such that $t \sqsubseteq_\sigma \sigma(x)$.*

Consider now the *Directed Acyclic Graph* (DAG) representation of messages as in [21], where the nodes of the graph are the subterms of the message, and the edges are defined from terms to their subterms, i.e., $op(m_1, m_2) \rightarrow_i m_i$.

We can remark immediately that the DAG-representation of a message is unique, and that if $n = card(st(m))$, then the DAG-representation has at most n nodes and $2n$ edges. We write $|m|_{\mathrm{DAG}}$ to denote the *DAG-size* of m and extend it to sets in the trivial way. We have that the DAG-size of a set is the number of (distinct) subterms of E, i.e., $|E|_{\mathrm{DAG}} = card(st(E))$.

Theorem 5.4. *If (σ, ω) is a normal attack that does not use the Oracle rule, then we have for all $x \in Var$, $|\sigma(x)|_{\mathrm{DAG}} \leqslant |P|_{\mathrm{DAG}}$.*

5.3. Bounding the size of attacks

We can now compute the maximum number of messages, and their respective size, that an intruder needs to deduce when creating an attack.

Throughout the rest of the paper, we will assume that $P = \{(i, R'_i \Rightarrow S'_i) \mid i \in \mathcal{I}\}$ is a protocol specification with a finite set of variables V, and that (σ, ω) and deductions $\Pi_1, \ldots, \Pi_k, \Pi$ form a normal attack with initial intruder knowledge S_0 that exposes $m \in \mathcal{M}_\Sigma$. Recall also that each Π_j is a deduction $Hyp_j \vdash R_j\sigma$ and Π is a deduction $Hyp_{k+1} \vdash m$.

The next proposition follows straightforwardly from Theorem 5.4 knowing that $|R'_i, S'_i|_{\mathrm{DAG}} \leqslant |P|_{\mathrm{DAG}}$ for all $i = 1, \ldots, k$. For this proposition, it is essential to use DAGs in order to compress the representation of sets of terms.

Proposition 5.5. *There exists a polynomial $p_{msg}(\cdot)$ such that if (σ, ω) is a normal attack against P that does not use the Oracle rule, then the maximum number of messages that the intruder needs to deduce is bounded by $p_{msg}(n)$, where $n = |P, S_0, m|_{\mathrm{DAG}}$, that is, $|msg(\Pi_1), \ldots, msg(\Pi_k), msg(\Pi)|_{\mathrm{DAG}} \in O(p_{msg}(n))$.*

We can now limit the number of rewrite rules used in a deduction.

Proposition 5.6. *There exists a polynomial $p_{rules}(\cdot)$ such that if (σ, ω) is a normal attack against P that does not use the Oracle rule, then the number of rewrite rules applied is bounded by $p_{rules}(n)$, where $n = |P, S_0, m|_{\mathrm{DAG}}$.*

The motivation for the next theorem is that if an attack has some cost γ, then the number of usages of the *Oracle* rule cannot be exponential in γ. This results from the fact that the cost of using an *Oracle* rule is at least the cost of "reading" all terms in the hypothesis plus the cost of "writing" the conclusion and so, if the *Oracle* rule is used very often, then the cost will grow too much and so cannot be limited by γ.

Theorem 5.7. *There exist polynomials $p_m(\cdot)$, $p_r(\cdot)$ and $p_\sigma(\cdot)$ such that if (σ, ω) is a normal attack against P and attackCost $\leqslant \gamma$, then the number of messages in the intruder knowledge, the number of rules used, and $|\sigma(x)|_{\mathrm{DAG}}$, for all $x \in V$, are bounded respectively by $p_m(n)$, $p_r(n)$ and $p_\sigma(n)$, where $n = |P, S_0, m|_{\mathrm{DAG}} + \gamma$.*

Proof. Recall that, since (σ, ω) is an attack, we have for all $1 \leqslant j \leqslant k$, $Hyp_j \vdash R_j\sigma$, and $Hyp_{k+1} \vdash m$. If there are no applications of the *Oracle* rule, then, by Proposition 5.5, we know that the maximum number of terms that the intruder needs to deduce is bounded by $p_{msg}(n)$; by Proposition 5.6, the number of rewrite rules needed is bounded by $p_{rules}(n)$; and by Theorem 5.4, we can choose a DAG-representation of $\sigma(x)$ in $O(n)$ for each $x \in Var$.

Now suppose that there are applications of *Oracle* rules and let us divide our attack into two parts $\Pi_1, \ldots, \Pi_i^-$ (with final set M_i) and $\Pi_i^+, \Pi_{i+1}, \ldots, \Pi_k, \Pi$ where: (i) there are no occurrences of the *Oracle* rule in $\Pi_i^+, \Pi_{i+1}, \ldots, \Pi_k, \Pi$; (ii) Π_i^- corresponds to the initial part of Π_i up to the point where the last *Oracle* rule is applied; and (iii) Π_i^+ is the remaining deduction of Π_i.

Note that if we consider the protocol $P_{<i}$ corresponding to the first $i-1$ steps of P and $P_{\geqslant i}$ the protocol corresponding to the other steps, then we have that $(\sigma, \omega_{<i})$ is an attack against protocol $P_{<i}$ with initial knowledge S_0 that exposes all messages in set M_i with cost $\gamma_{<i}$ (note that we can extend without loss of generality attacks to sets of messages), and $(\sigma, \omega_{\geqslant i})$ is an attack against protocol $P_{\geqslant i}$ with initial knowledge M_i that exposes m with cost $\gamma - \gamma_{<i}$ ($\omega_{<i}$ and $\omega_{\geqslant i}$ are the relevant execution orders).

Let $n_{\geqslant i} = |P_{\geqslant i}, M_i, m|_{\mathrm{DAG}}$. Since, by construction, $P_{\geqslant i}$ does not use *Oracle* rules, we can apply the same reasoning as above and show that the maximum number of terms that the intruder needs to deduce is bounded by $p_{msg}(n_{\geqslant i})$, the number of rewrite rules needed is bounded by $p_{rules}(n_{\geqslant i})$, and that we can choose a DAG-representation of $\sigma(x)$ in $O(n_{\geqslant i})$ for each $x \in vars(P_{\geqslant i})$. We will show next that $|M_i|_{\mathrm{DAG}}$ is polynomial in n, which imply that $n_{\geqslant i}$ is also polynomial in n, and so there exist $p'_m(\cdot)$, $p'_r(\cdot)$ and $p'_\sigma(\cdot)$ such that the previous polynomials are bounded by $p'_m(n)$, $p'_r(n)$ and $p'_\sigma(n)$ and consequently everything in $P_{\geqslant i}$ is polynomial in n. Let us now look at $P_{<i}$.

We first observe that since the cost of the application of an *Oracle* with support-set M and conclusion m is at least $|M, m|_{\mathrm{DAG}}$ (1) and each step of a deduction adds one element to the set, if the attack cost is less than γ, then the number of applications of the *Oracle* rule has to be polynomial in n (otherwise its cost would be greater than γ). In fact, the number of all rules has to be polynomial in n, otherwise the support-set of the last *Oracle* call in Π_i^- would be too "big" and this call would be too expensive.

For a similar reason (and $msg(\Pi_1) \cup \cdots \cup msg(\Pi_{i-1}) \subseteq msg(\Pi_i^-) = M_i$), we also have that $|msg(\Pi_1), \ldots, msg(\Pi_i^-)|_{\mathrm{DAG}} = |M_i|_{\mathrm{DAG}}$ has to be polynomial in n; otherwise producing such a "big" set would cost more than γ. It is then left to show that $|\sigma(x)|_{\mathrm{DAG}}$ is polynomial for each $v \in vars(P_{<i})$ but this follows easily from the fact that $|M_i|_{\mathrm{DAG}}$ is polynomial in n and $R_1\sigma, \ldots, R_{i-1}\sigma \in M_i$.

So, there exist $p_m(n)$, $p_r(n)$ and $p_\sigma(n)$ that bound the number of messages, rule applications and $|\sigma(x)|_{\mathrm{DAG}}$ for the attack (σ, ω) against P with budget γ, where $n = |P, S_0, m|_{\mathrm{DAG}} + \gamma$. $\square$

5.4. The QPI Problem is in NP

NP problems can be characterized as those for which, given a witness y for the problem, it is possible to check in polynomial-time that y is indeed a witness. Rigorously, a language L is in NP if and only if there exist polynomials p and q, and a deterministic Turing machine M, such that (i) for all x and y, machine M runs in time $p(x)$ on input (x, y); (ii) for all $x \in L$ there exists a witness y of length at most $q(x)$ such that $M(x, y) = 1$; and (iii) for all $x \notin L$ and all strings y of length at most $q(x)$, $M(x, y) = 0$.

In our case, L is the language of tuples (P, S_0, m, γ) such that there exists an attack to P, with initial knowledge S_0, that reveals m with a cost less than γ, and y are instances of attacks.

We define M as follows: given an attack (σ, ω) with $\Pi_1, \ldots, \Pi_k, \Pi$ (a witness), we apply the following procedure, where we let $R_i = R'_{\omega^{-1}(i)}$ and $S_i = S'_{\omega^{-1}(i)}$ for all $i \in \{1, \ldots, k\}$:

1. Check that $\sigma(x)$ is a ground substitution for all $x \in V$.
2. For each $i \in \{1, \ldots, k\}$, check that Π_i is a deduction $Hyp_i \vdash R_i\sigma$ and compute cost c_i.
3. Check that Π is a deduction $Hyp_{k+1} \vdash m$ and compute cost c_{k+1}.
4. Compute the sum $c = \sum_{i=1}^{k+1} c_i$.
5. If all checks are successful and $c \leqslant \gamma$, then answer 1, else answer 0.
6. M returns 0 once executed $p(n) + 1$ steps (the polynomial p is made precise in the proof below to avoid repeating the same arguments).

Theorem 5.8. *The Quantitative Protocol Insecurity Problem is in NP for a fixed number of sessions.*

Proof. Consider a protocol specification $P = \{(i, R'_i \Rightarrow S'_i) \mid i \in \mathcal{I}\}$ with a finite set of variables V, a secret m, an initial intruder knowledge set S_0 and a budget γ, and let $n = |P, S_0, m|_{\mathrm{DAG}} + \gamma$.

It is clear that if there is no attack ($x \notin L$) then the procedure M above always returns 0 as there is no witness to pass the test (condition (iii) of NP definition).

If there is an attack ($x \in L$) we recall that, by Theorem 3.3, we can restrict ourselves to normal attacks as we can always normalize any attack to obtain one with smaller cost. Now, by Theorem 5.7, we know that there exist polynomials $p_m(\cdot)$, $p_r(\cdot)$ and $p_\sigma(\cdot)$ that bound the number of messages, the number of rules used, and $|\sigma(x)|_{\mathrm{DAG}}$ for all $x \in V$, for any normal attack (σ, ω) with cost less than γ. So, there exists a polynomial-time witness (in n) for which M will return 1 (condition (ii) of NP definition). The only "problem" is that, in order to enforce M to run in polynomial-time for every (x, y), we required M to stop after $p(n) + 1$ steps. What we are going to show next is that these $p(n) + 1$ steps are enough to verify our witness.

We first convert P, S_0 to DAG-representation which can be done in polynomial-time $p_1(n)$.

Step 1. By Theorem 5.7, checking that $\sigma(x)$ is ground can be done in $p_\sigma(n)$ and for σ in $(p_\sigma(n))^2$.

Steps 2 and 3. Checking if a rewrite rule $lhs \to lhs, rhs$ is applicable to a set E and computing the DAG-representation of $E' = E, rhs$ can be done in $p'_2(|E|_{\mathrm{DAG}})$ provided we have a DAG-representation of E, lhs, rhs (we just need to check that all terms in lhs are in E). Since the maximum number of messages that the intruder needs to learn is bounded by $p_m(n)$, we know that checking the applicability of each rule is bounded by $p_2(n) = p'_2(p_m(n))$. Finally, the number of rewrite rules needed in each Π_j is bounded by $p_r(n)$ as well as the degree of principal terms and so, checking each Π_j of Step 2 takes at most $p_r(n) \times p_2(n)$, and for all j takes at most $n \times p_r(n) \times p_2(n)$. Idem for Step 3.

Step 4. Finally, the cost of this attack reduces to summing a polynomial number of terms (Definition 4.2), where each term can be computed in polynomial-time $p_4(n)$, as required in the definition of cost functions in Section 2.

So, there is a polynomial-time witness verification algorithm that, given the trace of the attack (a witness), can check in polynomial-time $p(n) = p_1(n) + (p_\sigma(n))^2 + (n + 1) \times p_r(n) \times p_2(n) + p_4(n)$ whether this attack is possible with cost less than γ, and so return 1 whenever $x \in L$.[7]

[7] Note that the polynomial p defined here does not depend on the input (P, S_0, m, γ) nor on the attack witness y. We defined it within this proof just to avoid repeating the same argument. We could instead have presented this discussion before defining M and show (repeatedly) in this proof that the given p is sufficient to perform the verification.

Since p is a polynomial, the last step of definition of M immediately enforces condition (i) of the NP definition. We can thus conclude that the QPI Problem is in NP for a fixed number of sessions. □

Theorem 5.9. *The Quantitative Protocol Insecurity Problem is NP-complete for a fixed number of sessions.*

Proof. To show NP-hardness, we just need to see that, considering all cost functions to be 0 and $\gamma = 0$, we can reduce the standard Protocol Insecurity Problem that is known to be NP-complete [21] to the QPI Problem. □

6. Conclusions

We have given an abstract presentation of a cost-sensitive guessing DY intruder and established the complexity of the corresponding insecurity problem for a finite number of sessions. We extended the standard DY intruder with an *Oracle* rule that allows the adversary to deduce any message, and proposed a deduction system where rules are decorated with a label that represents the cost of applying such rule. Since in our framework any message can be deduced using the *Oracle* rule, we extended the standard Protocol Insecurity Problem proposing a quantitative approach that aims at deciding whether there is an attack against some protocol P with a cost at most γ. In order to show that this problem is also NP, we have given a series of conditions that these cost functions need to satisfy.

6.1. Related work

We have based our results on the work of Rusinowitch and Turuani [21], who have shown that the protocol insecurity problem with a finite number of sessions and composed keys is NP-complete. We have extended this by allowing the intruder to guess information at a cost and shown that the complexity of the resulting quantitative protocol insecurity problem is also NP-complete. The extension is in itself quite intuitive—we have introduced an inference rule called *Oracle* and decorated inference rules with labeled parameters that keep track of the associated costs—but the proofs of the corresponding results, in particular those concerning complexity associated with the *Oracle* rule, are actually quite intricate as we have to deal with this additional information and, ultimately, show that the costs can be kept under control.

Delaune and Jacquemard [11,12] have also considered a guessing DY intruder by introducing a probabilistic encryption operator and guessing abilities for the intruder, and provided NP-complete decision procedures for the resulting protocol insecurity problem. (Several other approaches have been given for formal protocol analysis in the presence of an intruder who can perform off-line guessing, e.g., [2,7,8,10,11,17,18,23].) The main difference with respect to this work lies in our use of labels so as to consider explicitly the cost of guessing. A similar, albeit for now unrelated, use of costs has been proposed by Meadows for the analysis of denial of service (DoS) in computer networks: in [19], she formalized a framework in which the capabilities of the intruder are extended to include the cost of intruder actions, which allows her to model and compare the costs of the possible DoS attacks and defenses. Several researchers have extended this approach, e.g., Groza and Minea, who in [16] give a framework for the formal modeling and automatic detection of resource exhaustion attacks. Such a use of costs is different from our approach, and for a different purpose, but we actually believe that there might be interesting synergies to consider, e.g., to include guessing among the capabilities of the DoS intruder of [19]. As we remarked in Section 2, we plan to carry out a similar investigation also for the work of Baudet [5], in which Dolev–Yao models are extended with systems with computation times and probabilities to account for random polynomial-time computability.

Time plays a crucial role also in [6,9]. Corin et al. [9] propose an intruder model in which, in a nutshell, temporal constraints allow one to model the time spent deducing messages, whereas Benerecetti and Peron [6] extend the DY intruder model by decorating protocol steps with time constraints to model time-dependent protocol executions and attacks. Like we did, they also extend [21] and show that the resulting protocol insecurity problem, under the assumption of a finite number of sessions, remains NP-complete. Both of these works thus aim at modeling time as a quantitative measure that explicitly influences protocol attacks, as opposed to the abstract cost measure that we have considered. One could indeed take time as a possible concrete cost but other kinds of cost could be considered as well (e.g., the size of the dictionary or even the complexity of the guessed passwords or keys). Although not simple, as the formalisms are quite different, it would be interesting to combine these approaches with ours to reason explicitly about timing costs, both when deducing messages and when protocols explicitly depend on timing issues, e.g., in the case of timestamps.

Another work that is close to ours is that of Zunino and Degano [23], who consider secrecy and authentication in a process calculus with cryptographic primitives, where the Dolev–Yao intruder is extended with a rule to guess a secret key that decrypts an intercepted message. They assume that guessing succeeds with a given negligible probability and that the resources available to the intruder are polynomially bounded. Although the complexity analysis they perform is different from ours, they reach a similar result, namely that their extended Dolev–Yao intruder is as powerful as the standard one. Similar approaches and results are presented in [20,22]. Finally, the present paper extends and actually supersedes our previous exploratory publication [2] as the formalization that we present here is at the same time conceptually simpler and more expressive.

6.2. Discussion and possible extensions

The solution proposed in this paper relies on the intuitive fact that the usage of the *Oracle* rule is a costly operation and should be minimized. We added this explicitly in the properties of the cost functions, cf. (1), and this was extensively used in Theorem 5.7 to limit both the number of *Oracle* calls and the size of the generated terms. As mentioned, the *rationale* behind this hypothesis is that the attacker needs at least to "read" the messages in his knowledge and "write" the result. We could however have considered different alternatives and we discuss some of them in this subsection.

A similar assumption would have been to require the cost of *Oracle* to depend on the conclusion of the rule and on a subset of the intruder's knowledge (that we could call *oracle-relevant*) rather than on the whole set. This can be accomplished changing the definition of deduction in order to consider a support-set $M' \subseteq M_{i-1}$ for the *Oracle* rule rather than the whole set M_{i-1}. To follow this approach, one has to keep track of these *oracle-relevant* messages and (i) change the definition of normalization removing the messages that were not used as premises of any rule nor are relevant for any *Oracle* call; and (ii) show in Theorem 5.7 that the number of *oracle-relevant* messages, the number of calls to the *Oracle* rule, as well as the size of generated messages, has to be bounded. To prove case (ii), we can use an argument similar to the one used in Theorem 5.7 to show that after each *Oracle* call the resulting set is polynomial in the original n. Since the attack has a given cost γ, since calling the *Oracle* rule costs at least the size of the *relevant* messages and its conclusion, and since the messages on the deduction are either used as hypothesis of a rule or are relevant to some oracle, we can see that the number of *oracle-relevant* messages has to be small as expected. Although quite intuitive, the proof above would have been more complex as we would have to "break" the deduction into several smaller deductions and apply to each of them the argument used in Theorem 5.7. And so, since our assumption over the *Oracle* function is a realistic one, we decided not follow this alternative as it would add unnecessary complexity to the paper without a significant improvement to our main result.

Another alternative approach would be to limit the size and cost of *Oracle* calls through normalization discarding condition (1). We can do this in two different ways: the first is to require *Oracle* to generate only atomic messages; whereas the second is to require normalization to substitute *Oracle* calls with conclusion $op(m_1, m_2)$ by *Oracle* calls that generate both m_1 and m_2, and then compose them using the respective G_{op}. Both these solutions are, however, quite restrictive: the first limits the applicability of guessing, disallowing, for instance, to guess composed passwords, whereas the second only preserves the cost of deductions if the cost functions are such that deducing components m_1 and m_2 and composing them is "cheaper" than guessing directly $op(m_1, m_2)$ (which is not the case for instance with malleable cryptographic schemes). Anyway, even if one is comfortable with the above restrictions, the number of *oracle-relevant* messages would still be unbounded. The only way to limit these messages is to disallow them and restrict all deduced messages to be used as premises of some rule. This would, however, be a serious limitation as it would amount to assuming that it does not make sense to deduce messages that simplify the usage of *Oracle*, which is something that cannot be done beforehand. And although one could argue that it is possible to incorporate the cost of deducing an *oracle-relevant* message m in the cost of the *Oracle* call that uses such message, this would become problematic if m is relevant for two *Oracle* calls as then one would have to pay twice the cost of m while in our approach it is just paid once.

Weighing all the factors, we opted for a set of restrictions that were reasonable to accept and natural to justify and, most importantly, were sufficient to enforce our result.

In addition to what we already mentioned above, as future work, we are considering a number of significant case studies in order to come up with a characterization of appropriate cost functions for the different cryptographic operators, with the ultimate goal of devising a library of costs that mimic the realistic complexity of guessing.

Acknowledgements

The authors would like to thank Tiago Reis, Michael Rusinowitch and Mathieu Turuani.

References

[1] P. Adão, P. Mateus, L. Viganò, Protocol insecurity with a finite number of sessions and a cost-sensitive guessing intruder is NP-complete (extended version), Technical Report, SQIG-IT and IST, ULisboa, Portugal, available at: http://sqig.math.ist.utl.pt/pub/AdaoPM/13-AMV-quantana.pdf, 2013.
[2] P. Adão, P. Mateus, T. Reis, L. Viganò, Towards a quantitative analysis of security protocols, in: QAPL 2006, Electron. Notes Theor. Comput. Sci. 164 (3) (2006) 3–25.
[3] R. Amadio, D. Lugiez, On the reachability problem in cryptographic protocols, in: Concur, in: Lect. Notes Comput. Sci., vol. 1877, Springer, 2000.
[4] D. Basin, S. Mödersheim, L. Viganò, OFMC: A symbolic model checker for security protocols, Int. J. Inf. Secur. 4 (3) (June 2005) 181–208.
[5] M. Baudet, Random polynomial-time attacks and Dolev–Yao models, J. Autom. Lang. Comb. 11 (1) (2006) 7–21.
[6] M. Benerecetti, A. Peron, Timed protocol insecurity problem is NP-complete, Future Gener. Comput. Syst. 29 (3) (2011) 843–862.
[7] E. Cohen, Proving cryptographic protocols safe from guessing attacks, in: Foundations of Computer Security'02, 2002, pp. 85–92.
[8] R. Corin, J. Doumen, S. Etalle, Analysing password protocol security against off-line dictionary attacks, in: WISP'04, 2004.
[9] R. Corin, S. Etalle, P.H. Hartel, A. Mader, Timed analysis of security protocols, J. Comput. Secur. 15 (6) (2007) 619–645.
[10] R. Corin, S. Malladi, J. Alves-Foss, S. Etalle, Guess what? Here is a new tool that finds some new guessing attacks (extended abstract), in: WITS'03, 2003.
[11] S. Delaune, F. Jacquemard, A theory of guessing attacks and its complexity, Technical report, Research Report LSV-04-1, Lab. Specification and Verification, ENS de Cachan, France, 2004.

[12] S. Delaune, F. Jacquemard, Decision procedures for the security of protocols with probabilistic encryption against offline dictionary attacks, J. Autom. Reason. 36 (1–2) (Jan. 2006) 85–124.
[13] D. Dolev, A. Yao, On the security of public key protocols, IEEE Trans. Inf. Theory 29 (2) (1983) 198–208.
[14] N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov, Undecidability of bounded security protocols, in: Workshop on Formal Methods and Security Protocols, 1999.
[15] S. Even, O. Goldreich, On the security of multi-party ping pong protocols, Technical Report 285, Israel Institute of Technology, 1983.
[16] B. Groza, M. Minea, Formal modelling and automatic detection of resource exhaustion attacks, in: Proceedings of the 6th ASIACCS, ACM Press, 2011, pp. 326–333.
[17] P. Hankes Drielsma, S. Mödersheim, L. Viganò, A formalization of off-line guessing for security protocol analysis, in: LPAR'04, in: LNAI, vol. 3452, Springer, 2005, pp. 363–379.
[18] G. Lowe, Analysing protocols subject to guessing attacks, J. Comput. Secur. 12 (1) (2004).
[19] C. Meadows, A cost-based framework for analysis of denial of service in networks, J. Comput. Secur. 9 (1/2) (2001) 143–164.
[20] J.C. Mitchell, A. Ramanathan, A. Scedrov, V. Teague, A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols, Theor. Comput. Sci. 353 (1–3) (2006) 118–164.
[21] M. Rusinowitch, M. Turuani, Protocol insecurity with a finite number of sessions and composed keys is NP-complete, Theor. Comput. Sci. 299 (1–3) (2003) 451–475.
[22] A. Troina, A. Aldini, R. Gorrieri, Towards a formal treatment of secrecy against computational adversaries, in: Post-Proceedings of GC'04, in: Lect. Notes Comput. Sci., vol. 3267, Springer, 2005, pp. 77–92.
[23] R. Zunino, P. Degano, Weakening the perfect encryption assumption in Dolev–Yao adversaries, Theor. Comput. Sci. 340 (1) (2005) 154–178.