Abstract Interpretation for Secrecy Correctness in E-Commerce Protocols

K. ADI†, M. DEBBABI‡ & L. PENE†
Computer Security Research Laboratory

†Computer Science and Engineering Department
Université du Québec en Outaouais
Gatineau, Qc, Canada
{adi, penl01}@uqo.ca
‡ Concordia Institute for Information Systems Engineering,
Concordia University,
Montreal, Qc, Canada
debbabi@ciise.concordia.ca

***Abstract*— Given its special nature, e-commerce yields concerns for providing secure transactions. Accordingly, a number of security properties, such as secrecy, authentication and fairness, have to be guaranteed. In this paper, we present a new method based on abstract interpretation for secrecy verification in e-commerce protocols. Hence, we define an abstract message domain and an abstract interpretation over finite and approximated models of e-commerce protocols. This allows us to build a semi-decidable procedure for e-commerce protocol correctness with respect to the secrecy property. Our approach is fully automatic from cryptographic protocol description to results and requires no user input except the protocol description and the level of the abstraction.**

***Index Terms*— E-commerce protocols, Abstract Interpretation, Secrecy, Protocol Correctness.**

I. INTRODUCTION AND RELATED WORK

With the dazzling expansion of computer networks and the emergence of new technologies such as World Wide Web and electronic commerce, security became a major concern for the computer research community. Accordingly, a surge of interest is devoted to the design, implementation and analysis of security protocols that are a basis of e-commerce protocols.

A considerable number of security protocols have been devised during the past two decades. Many among them have been shown flawed years after their first use. Consequently, a great deal of interest has been expressed in the development of formal techniques for the specification, design and verification of security protocols. Furthermore, we anticipate that the rapid expansion of distributed systems, communication networks, Internet, web applications, etc., will certainly bring a major need in security protocols. It is then imperative to have appropriate environments for the correct development of these protocols.

This research is supported by a research grant from the Natural Science and Engineering Council of Canada, NSERC, and the "Fonds québécois de la recherche sur la nature et les technologies", FQRNT.

Security protocols verification has known a significant progress. Burrows, Abadi and Needham [1] introduced the notion of modal logic of belief (the BAN logic) on security protocols more then a decade ago. A new linear and modal logic for specifying security properties has been advanced by Adi *et al.* [2]. The logic is compact, expressive and formal and has been used to specify classical security protocols and electronic commerce protocols. Lowe and others introduced the concept of model-checking and used it to successfully finding flaws in security protocols [3], [4] with the aid of systems such as FDR [5] and NRL [6]. Several other important results have to be mentioned, such as the use of theorem proving by Bolignano [7] and the use of inductive rules by Paulson [8].

Abstract interpretation has been applied successfully in the verification of security protocols and a number of valuable abstract models for security protocols have been obtained, such as the tree automata of Monniaux [9], the $\vee$-parameterized tree abstractions of Goubault-Larrecq [10], the abstract game semantics of Adi *et al.* [11] and the pattern-based abstraction of Lakhnech *et al.* [12]. Abstract interpretation allows semantics manipulations by performing simulations on data description in order to obtain correct, implementable and most accurate analysis. Other related approaches are the secrecy conditions of Houmani and Mejri's [13] and the trace model for authentication of Cremers *et al.* [14].

In general, models describing protocol-behaviors are infinite. In most cases, for verification purposes, only a finite and approximated model is required. For this reason, we consider the problem of computing such an approximation and we propose to simulate the required partial protocol execution at an abstract level. More precisely, we define a method that computes abstract finite models for security protocols with the aid of an abstraction function that bounds the size of the intruder-messages. The abstract model is then used to define a

semi-decidable procedure for secrecy correctness. Unlike the other approaches which require that the user designs himself the abstraction or manually helps a program to compute invariants (see for instance [7]), our approach is fully automatic from cryptographic protocol description to results and requires no user input except the protocol description and the level of the abstraction. The latter parameter allows the user to tune the precision of the abstraction to get the best results while minimizing the cost of the analysis. Our method finds immediate applicability in secure communications over private and public networks for e-commerce, teleconferencing, mobile computing etc.

The remainder of this paper is organized as follows. The next section is devoted to the definition of a trace-based model for security protocols. Section III is devoted to the definition of an abstract domain of messages and an abstraction function on messages. In section IV we present a method for computing abstract cryptographic protocol and abstract trace-based models. Section V states a sufficient condition for verifying the secrecy property of a cryptographic protocols. Finally, section VII contains concluding remarks.

II. Trace-Based Models for Security Protocols

The message syntax used in security protocol descriptions is captured by the following BNF (Backus-Naur Form) grammar:

m	::=	cst	Constant Message
	|	$\{m\}_{m'}$	Encrypted Message
	|	$m.m'$	Message Concatenation
	|	$f(m)$	Function Application

All messages that keep constant their values along a given protocol run are considered as constants and will be abstracted to terms of the form cte. A message m encrypted with key m' is written $\{m\}_{m'}$ and forms a message by itself. The operator "." is used to separate concatenated messages. All non-constant messages will be represented as terms through functional application. We use A_x, B_y, $x, y : nat$, to denote principals and S to denote the server. A shared key between A_x and B_y is represented by $K_{a_x b_y}$. A fresh nonce N, created by a principal A_x in a protocol run σ, is represented by the term $N(A_x, \sigma)$. For convenience, messages may be annotated. We use superscript annotations to indicate run identifiers and subscript annotations to indicate the principal association. Accordingly, the term $N(A_x, \sigma)$ will be simply represented as $N^{\sigma}_{A_x}$. If in some context more than one nonce are needed within the same session, they will be represented by $N1(A_x, \sigma)$, $N2(A_x, \sigma)$, etc. Let m, m_1 and m_2 be messages. We say that m is atomic if $m \neq m_1.m_2$ and $m \neq \{m_1\}_{m_2}$. We suppose that encryption keys are always atomic messages.

In the presence of a malicious intruder, a protocol can be described as a finite sequence of statements of the form:

$$\begin{array}{lcccl} \sigma.i, & A_x & \longrightarrow & I(B_y) & : \ m \\ \sigma.i, & I(A_x) & \longrightarrow & B_y & : \ m \end{array}$$

which state that during a protocol step i, A_x sends the message m intended for B_y over the network, then B_y gets the message m from the network, as intended by A_x. $I(A_x)$ stands for the intruder playing the A_x's role. The motivation underlying such a notation is that we assume that the network is under the control of a malicious smart intruder. All messages sent or received by honest principals transit by the intruder. This is used to capture the fact that the intruder is aware of any message circulating over the network. For the sake of convenience, we can also represent the two above actions by a unique action $\sigma.i,\ A_x \longrightarrow B_y : m$.

The execution of a security protocol generates a trace consisting of a sequence of events. Each event results from the execution of a protocol step corresponding to a send action or a receive action. A run of a protocol is a particular execution of the protocol. We refer to a protocol run as a session. The execution of a protocol is based on an interleaving model in which all events, including concurrent events, occurring during a run are interleaved to form a single trace of that execution. A protocol trace is said to be valid if all the messages sent by the intruder could be derived from the intruder's cumulated knowledge (initial knowledge and received messages), and all the involved principals respect the protocol. We assume that the intruder is able to perform the following actions: overhear messages, intercept messages, replay messages and generate new messages using his initial knowledge and the overheard messages. In the following, we introduce formally the notion of protocol traces. A protocol trace is a sequence of protocol events resulting from any interleaving of (possibly partial) protocol runs. We have no restrictions on traces in the sense that we support multi-session (an agent could participate in many sessions) and multi-role (an agent can be an initiator in some sessions and responder in others).

The set $\mathcal{T}$ of traces is defined inductively as follows:

- $\epsilon \in \mathcal{T}$
- if $t \in \mathcal{T}$ and a is a protocol event then $t.a \in \mathcal{T}$

where ϵ stands for the empty trace and "." is the concatenation operator for sequences.

A protocol can then be modelled as a subset of traces $P \subseteq \mathcal{T}$. More formally, following Paulson's model [8], we describe the set of protocol traces P as the closure of a set of inductive rules. In this approach, a security protocol is modelled by a set of rules representing protocol steps and intruder capabilities. To give an example of such a modelling, consider a version of the *Woo and Lam* authentication protocol [15]. The goal of this protocol is to authenticate the identity of the principal A_x with respect to the principal B_y. To achieve this objective, the protocol uses an authentication server S. The specification of this protocol is given in Table I. Here, $K_{a_x s}$ is a key shared between A_x and S and $K_{b_y s}$ is a key shared between B_y and S. The message $N^{\sigma}_{b_y}$ is a nonce generated by B_y during the session σ. It is used by B_y to preclude the replay of messages coming from preceding sessions. The *Woo and Lam* protocol traces P are built up inductively by a set of rules shown in Table II.

The inductive definition starts with the empty rule. The empty

TABLE I

THE *Woo and Lam* AUTHENTICATION PROTOCOL

Message σ.1.	A_x	$\longrightarrow$	B_y	:	A_x
Message σ.2.	B_y	$\longrightarrow$	A_x	:	$N_{b_y}^{\sigma}$
Message σ.3.	A_x	$\longrightarrow$	B_y	:	$\{N_{b_y}^{\sigma}\}_{K_{a_x s}}$
Message σ.4.	B_y	$\longrightarrow$	S	:	$\{A_x.\{N_{b_y}^{\sigma}\}_{K_{a_x s}}\}_{K_{b_y s}}$
Message σ.5.	S	$\longrightarrow$	B_y	:	$\{N_{b_y}^{\sigma}\}_{K_{bs}}$

TABLE II

Woo and Lam INDUCTIVE RULES

empty
$$\frac{}{\epsilon \in P}$$

Receive
$$\frac{t \in P \qquad (\sigma.j,\ A_x \longrightarrow I(B_y) : m) \in \overline{t}}{t.\ (\sigma.j,\ I(A_x) \longrightarrow B_y : m)\ \in P}$$

Intruder
$$\frac{t \in P \qquad m \in \text{Message}(t)_{\Downarrow}}{t.\ (\sigma.j,\ I(B_y) \longrightarrow A_x : m)\ \in P}$$

Message 1
$$\frac{t \in P}{t.\ (\sigma.j,\ A_x \longrightarrow I(B_y) : A_x)\ \in P}$$

Message 2
$$\frac{t \in P \qquad N_b^{\sigma} \notin \text{Message}(t)_{\Downarrow}}{t.\ (\sigma.j,\ B_y \longrightarrow I(A_x) : N_b^{\sigma})\ \in P}$$

Message 3
$$\frac{t \in P \qquad (\sigma.j,\ I(B_y) \longrightarrow A_x : N_b^{\sigma}) \in \overline{t}}{t.\ (\sigma.j,\ A_x \longrightarrow I(B_y) : \{N_b^{\sigma}\}_{K_{as}})\ \in P}$$

Message 4
$$\frac{t \in P \qquad (\sigma.j,\ I(A_x) \longrightarrow B_y : A_x) \in \overline{t} \qquad (\sigma.j,\ I(A_x) \longrightarrow B_y : \{N_b^{\sigma}\}_{K_{as}}) \in \overline{t}}{t.\ (\sigma.j,\ B_y \longrightarrow I(S) : \{A_x.\{N_b^{\sigma}\}_{K_{as}}\}_{K_{bs}})\ \in P}$$

Message 5
$$\frac{t \in P \qquad (I(B_y) \longrightarrow S : \{A_x.\{N_b^{\sigma}\}_{K_{as}}\}_{K_{bs}}) \in \overline{t}}{t.\ (\sigma.j,\ S \longrightarrow I(B_y) : \{N_b^{\sigma}\}_{K_{bs}})\ \in P}$$

where

Message(t) is the set of messages in the trace t
$\overline{t}$ is the set of components in the sequence t

trace belongs always to P. For each protocol step, we have a corresponding rule. For example, in the rule Message 2, a trace $t \in P$ can be extended with the event $(\sigma.j,\ B_y \longrightarrow I(A_x) : N_b^{\sigma})$ whenever N_b^{σ} is a fresh nonce, *i.e.*, it has not been used in t. The rule Receive states that a principal can get a message only if it has been previously sent to her. The rule Intruder models the capacity of the intruder to send any message built up from the past traffic.

The intruder closure operation $\_\Downarrow$ allows us to capture the usual intruder capabilities: encryption, decryption, messages concatenation and message decomposition.

Definition 1: **(Closure)** Let M be a set of messages. We denote by $M_{\Downarrow}$ the closure of the set M under the conventional intruder computations (encryption decryption, message catenation, etc.). The closure $M_{\Downarrow}$ is defined as the smallest set which satisfies the following conditions:

1) $M \subseteq M_{\Downarrow}$
2) If $K \in M_{\Downarrow}$ and $\{m\}_K \in M_{\Downarrow}$ then $m \in M_{\Downarrow}$
3) If $K \in M_{\Downarrow}$ and $m \in M_{\Downarrow}$ then $\{m\}_K \in M_{\Downarrow}$
4) If $m \in M_{\Downarrow}$ and $m' \in M_{\Downarrow}$ then $(m, m') \in M_{\Downarrow}$
5) If $(m, m') \in M_{\Downarrow}$ then $m \in M_{\Downarrow}$
6) If $(m, m') \in M_{\Downarrow}$ then $m' \in M_{\Downarrow}$

III. ABSTRACT DOMAIN OF MESSAGES

The abstract domain of messages is built as an extension of the concrete domain by the abstract messages $\top$ and $\bot$. Intuitively, the value $\top$ is used to abstract a (possibly infinite) set of messages. For instance, the message $\{\top\}_K$ stands for any message encrypted with the key K. It is an abstraction of a (possibly infinite) set of concrete messages encrypted with the key K. We use also the abstract message value $\bot$ to represent the empty set of messages. From now on, we note D the

concrete domain of messages and $D^{\#}$ the abstract domain of messages.

We define an ordering relation on abstract messages $\leq_a$ that captures the notion of approximation on messages. $\forall m, m', m_1, m'_1 \in D^{\#}$:

$$\begin{cases} m \leq_a m & \\ \bot \leq_a m & \\ m \leq_a \top & \\ m \leq_a m' \text{ and } m' \text{ is atomic} & \Rightarrow m = m' \\ m \leq_a m' \text{ and } m' \leq_a m'' & \Rightarrow m \leq_a m'' \\ m \leq_a m' & \Leftrightarrow \{m\}_K \leq_a \{m'\}_K \\ m \leq_a m' \text{ and } m_1 \leq_a m'_1 & \Leftrightarrow m.m_1 \leq_a m'.m'_1 \end{cases}$$

We define a function $a^{\rightarrow}(\_)$ to approximate (abstract) messages. This function takes as parameters a message m and limits l_1 and l_2 (which are natural numbers). It replaces some sub-messages of m by the abstract value $\top$. Intuitively, the function $a^{\rightarrow}(\_)$ is used to prune the algebraic structure of a message while keeping the external form of that message. The value l_1 acts on the depth of encryptions in a message while the value l_2 acts on the depth of concatenations. Note that similar approaches have been used for other purposes. See for instance, k-limiting in may-alias analysis [16] and abstract rewriting [17], [18].

Definition 2 (Abstraction Function): The message abstraction function $a^{\rightarrow}(\_)$ is defined as follows:

$$a \ : \ \mathbb{N} \times \mathbb{N} \times D \longrightarrow D^{\#}$$

$$\begin{cases} a^{l_1,l_2}(m) & = m & \text{if } m \text{ is atomic} \\ a^{l_1,l_2}(m.m') & = a^{l_1,l_2-1}(m).a^{l_1,l_2-1}(m') & \text{if } l_2 \geq 1 \\ a^{l_1,l_2}(\{m\}_{m'}) & = \{a^{l_1-1,l_2}(m)\}_{m'} & \text{if } l_1 \geq 1 \\ a^{l_1,0}(m.m') & = \top & \\ a^{0,l_2}(\{m\}_K) & = \top & \\ a^{l_1,l_2}(\top) & = \top & \\ a^{l_1,l_2}(\bot) & = \bot & \end{cases}$$

From now on, when l_1 and l_2 are understood, we simply write m^a instead of $a^{l_1,l_2}(m)$.

In the following, we present some interesting properties of the abstraction function $a^{\rightarrow}(\_)$.

Proposition 1: The message abstraction function $a^{\rightarrow}(\_)$ has the following properties:

$$\forall m \in D^{\#}, \ \forall l_1, l_2, l'_1, l'_2 \in \mathbb{N}, \ l_1 \geq l'_1, \ l_2 \geq l'_2 \ : \ a^{l_1,l_2}(m) \leq_a a^{l'_1,l'_2}(m) \quad (1)$$

$$\forall m \in D^{\#}, \ \forall l_1, l_2 \in \mathbb{N} \ : \ m \leq_a a^{l_1,l_2}(m) \quad (2)$$

$$a \circ a = a \quad (3)$$

$$\forall m, m' \in D^{\#}, \ \forall l_1, l_2 \in \mathbb{N} \ : \ m \leq_a m' \Rightarrow a^{l_1,l_2}(m) \leq_a a^{l_1,l_2}(m') \quad (4)$$

Proofs are given in the appendix of the paper.

- The proposition 1(1) establishes a relation between the values of l_1 and l_2 and the precision of the abstraction. As we may expect, the precision of the abstraction increases when the values of l_1 and l_2 increase.
- The proposition 1(2) establishes that the abstraction function $a^{\rightarrow}(\_)$ is extensive, i.e. the abstracted message can not be more precise than the original message.
- The proposition 1(3) states that the abstraction function $a^{\rightarrow}(\_)$ is idempotent. This property can be interpreted as the fact that all the abstracted information is lost at once.
- The proposition 1(4) states that the abstraction function $a^{\rightarrow}(\_)$ is monotone. This property can be interpreted as the fact that the abstraction process preserves the soundness of the approximation.

A function f is an upper closure with respect to the ordering relation if and only if f is monotone, extensive and idempotent. The abstraction function has all these properties. Intuitively, we can acknowledge that the initial ordering relation is preserved on set of abstracted messages.

Corollary 1: The abstraction function $a^{\rightarrow}(\_)$ is an upper closure on the abstract domain of messages with respect to the ordering relation $\leq_a$.

Proof: The proof is immediate from the propositions 1(2), 1(3) and 1(4). ■

For instance, let $l_1 = 2$, $l_2 = 4$ and let x be the following message: $x = \{m_1.\{m_2.m_3\}_{K_1}.\{\{m_4\}_{K_2}.m_5\}_{K_3}.m_6\}_{K_4}$. The messages $m_1, m_2, m_3, m_4, m_5, m_6$ are atomic messages and K_1, K_2, K_3, K_4 are cryptographic keys. Then, message x is abstracted by the abstraction function $a^{\rightarrow}(\_)$ as shown in Table III.

The following proposition states that the abstraction function $a^{\rightarrow}(\_)$ bounds the size of messages.

Proposition 2: Let m be a message, l_1 and l_2 be two natural numbers then:

$$|a^{l_1,l_2}(m)| \leq 2^{l_1+l_2} - 1$$

The intruder closure operation captures the usual intruder capabilities: encryption, decryption, message concatenation and message decomposition. Given a finite set of messages M, two natural numbers l_1, l_2, we define an abstract closure of M noted $M^{\#}_{\Downarrow}$, as the smallest set satisfying:

1) If $m \in M$ then $a^{l_1,l_2}(m) \in M^{\#}_{\Downarrow}$
2) If $K \in M^{\#}_{\Downarrow}$ and $\{m\}_K \in M^{\#}_{\Downarrow}$ then $a^{l_1,l_2}(m) \in M^{\#}_{\Downarrow}$
3) If $m.m' \in M^{\#}_{\Downarrow}$ then $a^{l_1,l_2}(m) \in M^{\#}_{\Downarrow}$
4) If $m.m' \in M^{\#}_{\Downarrow}$ then $a^{l_1,l_2}(m') \in M^{\#}_{\Downarrow}$
5) If $m \in M^{\#}_{\Downarrow}$ and $K \in M^{\#}_{\Downarrow}$ then $a^{l_1,l_2}(\{m\}_K) \in M^{\#}_{\Downarrow}$
6) If $m \in M^{\#}_{\Downarrow}$ and $m' \in M^{\#}_{\Downarrow}$ then $a^{l_1,l_2}(m.m') \in M^{\#}_{\Downarrow}$

The next proposition establishes the finiteness of a set of messages under particular conditions.

TABLE III
MESSAGE ABSTRACTION EXAMPLE

$$
\begin{array}{rcl}
a^{2,4}(x) & = & \{a^{1,4}(m_1.\{m_2.m_3\}_{K_1}.\{\{m_4\}_{K_2}.m_5\}_{K_3}.m_6)\}_{K_4} \\
& = & \{a^{1,3}(m_1).a^{1,3}(\{m_2.m_3\}_{K_1}.\{\{m_4\}_{K_2}.m_5\}_{K_3}.m_6)\}_{K_4} \\
& = & \{m_1.a^{1,2}(\{m_2.m_3\}_{K_1}).a^{1,2}(\{\{m_4\}_{K_2}.m_5\}_{K_3}.m_6)\}_{K_4} \\
& = & \{m_1.\{a^{0,2}(m_2.m_3)\}_{K_1}.a^{1,1}(\{\{m_4\}_{K_2}.m_5\}_{K_3}).a^{1,1}(m_6)\}_{K_4} \\
& = & \{m_1.\{a^{0,1}(m_2).a^{0,1}(m_3)\}_{K_1}.\{a^{0,1}(\{m_4\}_{K_2}.m_5)\}_{K_3}.m_6\}_{K_4} \\
& = & \{m_1.\{m_2.m_3\}_{K_1}.\{a^{0,0}(\{m_4\}_{K_2}).a^{0,0}(m_5)\}_{K_3}.m_6\}_{K_4} \\
& = & \{m_1.\{m_2.m_3\}_{K_1}.\{\top.m_5\}_{K_3}.m_6\}_{K_4}
\end{array}
$$

Proposition 3: Let M be a set of messages such that the set of atomic messages in M is finite. If the depth of each message in M is bounded then the set M is finite.

The intruder closure operation in the concrete model leads to an infinite set of messages. The abstraction process gives us a way to obtain an abstract finite representation. The following proposition states that $M^{\#}_{\Downarrow}$ is a finite set.

Proposition 4: Let M be a finite set of messages. Then $M^{\#}_{\Downarrow}$ is finite.

IV. SIMULATING CRYPTOGRAPHIC PROTOCOL RUNS

In this section, we define an abstract semantics for security protocols. This new semantics is used to approximate execution models of protocols. The obtained abstract models are then used to decide if security protocols satisfy the secrecy property with respect to a sensitive message.

The intruder can use the protocol in two different ways. The first involves a passive attack on the protocol. It allows the intruder to intercept messages that will add new information to his knowledge. However, not all the new data is necessarily useful, as we will prove in this section. The second way of benefiting from protocol runs is called protocol instrumentation. The intruder sends certain data expecting a valuable message in response. We analyze in the following what would be helpful for an intruder that tries to build an attack. The intruder can use his concatenation and decomposition abilities at any time, as they don't depend on the value of the information. However, he might try to encrypt or decrypt messages with keys that he does not possess. For instance, we consider the following protocol:

$$
\begin{array}{lrcl}
\textsf{Message } \sigma.1. & A_x & \longrightarrow & B_y \ : \ N^{\sigma}_{a_x} \\
\textsf{Message } \sigma.2. & B_y & \longrightarrow & A_x \ : \ \{N^{\sigma}_{a_x}\}_{K_{a_x b_y}}
\end{array}
$$

The intruder instruments the protocol as follows. He knows that the message transmitted during the second step is an encryption of the first message with a session key that he does not know. By playing the role of the initiator agent, he introduces the piece of data, possibly from a different session, that he wants encrypted. In return, he receives information that he could not derive from his knowledge set using the standard closure capabilities. Of course, it is sometimes sufficient to run a protocol sequence, rather than a complete session. This observation allows the presence of partial runs in the trace modeling the intruder's behavior. The same approach can be used for having messages decrypted by unsuspecting agents as shown in the following protocol:

$$
\begin{array}{lrcl}
\textsf{Message } \sigma.1. & A_x & \longrightarrow & B_y \ : \ \{N^{\sigma}_{a_x}\}_{K_{a_x b_y}} \\
\textsf{Message } \sigma.2. & B_y & \longrightarrow & A_x \ : \ N^{\sigma}_{a_x}
\end{array}
$$

A. Abstracting Atomic Messages

Our aim is to compute the intruder's knowledge in a parallel multi-session protocol run. Unfortunately, as shown in previous sections, the model capturing such behavior is infinite and uses an unbounded number of principals. As it is impossible to simulate such models, we introduce in the following a safe computable upper approximation of the intruder's knowledge.

1) Abstracting Agent Names:

We use a finite set of constants $\{A, B, \cdots\}$ to denote agent names. This set of agent identities is sufficient for modelling any protocol execution, including multi-sessions and masquerading or impersonation attacks. Each constant abstracts a (possibly infinite) set of agent names. For instance, for a protocol step $\sigma.i.A_x \longrightarrow B_y : A_x$, the agent name A will be used to abstract the (possibly infinite) set of initiator agents A_x. This abstraction is correct since it overestimates the intruder's knowledge. We conclude that if the intruder can get A, then he can get any agent name $A_i,\ i \in \mathbb{N}$ running the concrete communication protocol as initiator. We make the same assumption about the responder. Usually, only three constants are employed: A, B, and S, where S abstracts a server name for protocols that involve one.

2) Abstracting Keys:

Each agent has one pair of public-private keys. Public keys are considered known to all potential agents participating in a protocol run, including the intruder. Therefore, we assert that all public keys associated with agent names are part of the intruder's initial knowledge. For instance, the constant K_b abstracts the concrete public key K_{b_y} of an agent named B_y. The private keys are the inverses of the public keys. They are not part of the intruder's initial knowledge and are never transmitted during the protocol run. As a consequence of the number of agents being bounded, the number of their public and private keys is a finite set.

Session keys are variables that are shared by agents for the

duration of one session only. We will abstract the session keys by a constant for any pair of agents. The rationale behind this is that if the intruder is able to get an exchanged session key in a particular session σ of a protocol run, then it could get the session keys used in any other session of the protocol run.

3) Abstracting Nonces:

Nonces are fresh messages unique to a particular session. Since we are interested in secrecy, we will abstract away from the notion of message-freshness and consider these messages as constants for all sessions of the protocol. The intuition is that if the intruder is able to get an exchanged nonce in a particular session σ of a protocol run, then it could get the same nonce in any other session of the protocol run. Hence, the nonce $N^{\sigma}_{a_x}$ is abstracted by the atomic constant message N_a. This abstraction is an overestimation of the intruder capabilities and therefore it is correct for our analysis.

We consider that the intruder owns a nonce N_i that is part of his initial knowledge. N_i is a constant atomic message. Since there is no way for a regular agent to check the value of a nonce created by any other agent, including the intruder, we empower the intruder to re-use the same nonce N_i in any protocol session.

All the atomic messages other than those already introduced are abstracted by constant values cte, as the security properties depend only on the keys and nonces.

B. Abstracting Protocols

Let P be a concrete protocol and let l_1 and l_2 be two natural numbers. To abstract protocol specifications, we define an abstraction function, noted Ψ as in Table IV-B.

In the remainder of the paper, we note $\mathsf{P}^{\#} = \Psi(\mathsf{P})$ the abstract protocol. For instance, let P be the following authentication protocol:

$$\begin{array}{llllll} \mathsf{Message}\ \sigma.1. & A_x & \longrightarrow & B_y & : & \{A_x.N^{\sigma}_{a_x}\}_{K_{b_y}} \\ \mathsf{Message}\ \sigma.2. & B_y & \longrightarrow & A_x & : & \{N^{\sigma}_{a_x}\}_{K_{a_x}} \end{array}$$

If we fix the values of $l_1 = 1$ and $l_2 = 1$, the above protocol is then abstracted as:

$$\begin{array}{llllll} \mathsf{Message}\ \sigma.1. & A & \longrightarrow & B & : & \{A.N_a\}_{K_b} \\ \mathsf{Message}\ \sigma.2. & B & \longrightarrow & A & : & \{N_a\}_{K_a} \end{array}$$

A simulated run of $\mathsf{P}^{\#}$ reveals that the abstract intruder's knowledge with limits $l_1 = 1$ and $l_2 = 1$ is $\{N_i, \{A.N_a\}_{K_{ab}}, \{N_a\}_{K_{ab}}\}_{\Downarrow}$, which means that the nonce N_a is never revealed by the abstract protocol. This means that $N^{\sigma}_{a_x}$ is secret in P.

Let $t = a_1.a_2.\ldots$ be a (possibly infinite) trace where a_i is a communication action. In order to determine the abstracted knowledge of the intruder, we fix the values of l_1 and l_2. The abstraction of each individual action produces an abstracted protocol step. The sequence of all abstracted actions defines the abstracted trace $t^{\#}$. To generate an abstract trace $t^{\#}$ from an abstract protocol $\mathsf{P}^{\#}$, we can use a modified version of the Paulson's inductive rules presented above [8] by applying the abstraction function $a\text{-}\cdot\text{-}(\_)$ to each exchanged message and replace the intruder closure function $\_ \Downarrow$ by the abstract version $\_^{\#}_{\Downarrow}$.

Let t be the following trace:

$$\begin{array}{rl} t = & (\sigma.1.\ A_1 \longrightarrow I(B_1) : A_1). \\ & (\sigma.1.\ I(A_1) \longrightarrow B_1 : A_1). \\ & (\sigma.2.\ B_1 \longrightarrow I(A_1) : N^{\sigma}_{b_1}.A_1.B_1). \\ & (\sigma.2.\ I(B_1) \longrightarrow A_1 : m). \\ & (\sigma.3.\ A_1 \longrightarrow I(B_1) : \{m\}_{K_{a_1 s}}) \end{array}$$

where m is part of the intruder knowledge. For $l_1 = 1$ and $l_2 = 1$, the abstract trace $t^{\#}$ is:

$$\begin{array}{rl} t^{\#} = & (\sigma.1.\ A \longrightarrow I(B) : A). \\ & (\sigma.1.\ I(A) \longrightarrow B : A). \\ & (\sigma.2.\ B \longrightarrow I(A) : N_b.\top). \\ & (\sigma.2.\ I(B) \longrightarrow A : m^a). \\ & (\sigma.3.\ A \longrightarrow I(B) : \{m^a\}_{K_{as}}) \end{array}$$

Given a trace t and a finite set of intruder's initial knowledge $\mathcal{K}_I$, then the intruder knowledge associated with the abstract trace $t^{\#}$, noted $\mathcal{IK}(t^{\#})$, is generated by the relation $\vdash\!\sim$ defined in Table V.

The following proposition states that the abstract trace model $t^{\#}$ for an abstract protocol $\mathsf{P}^{\#}$ is finite, i.e. has a finite number of actions.

Proposition 5 (Finiteness): Let P be a cryptographic protocol and let $\mathsf{P}^{\#}$ be the

corresponding abstracted protocol. Then, the multi-session, multi-role trace execution model $t^{\#}$ of $\mathsf{P}^{\#}$ is finite.

V. Secrecy Correctness

Secrecy is the fact of keeping secret a given piece of information. This aspect of security is certainly the oldest and the best known. We say that a protocol preserves the secrecy of one of its parameters if it does not leak any information about these parameters during its execution. The parameters of the protocol that have to be kept secret can be cryptographic keys, nonces, or any other sensitive data. For instance, the following protocol does not guarantee the secrecy of the message m since the key used to encrypt m has been made public.

$$\begin{array}{llllll} \sigma.1\ A_x & \longrightarrow & B_y & : & K^{\sigma}_{a_x b_y} \\ \sigma.2\ B_y & \longrightarrow & A_x & : & \{m\}_{K^{\sigma}_{a_x b_y}} \end{array}$$

In the following we give a sufficient condition to check if a protocol preserves the secrecy of a message.

Definition 3 (Secrecy): Given a security protocol P and a sensitive message m, we say that P guarantees the secrecy of m if:

$$\forall x \in \mathcal{IK}(t^{\#})\ :\ x \sqcap m^a = \bot$$

TABLE IV

THE PROTOCOL ABSTRACTION FUNCTION Ψ

$$\begin{cases} \Psi(A_x) = A \\ \Psi(B_y) = B \\ \Psi(S) = S \\ \Psi(N^{\sigma}_{a_x}) = N_a \\ \Psi(K_{a_x}) = K_a \\ \Psi(K^{-1}_{a_x}) = K_a^{-1} \\ \Psi(K_{a_x b_y}) = K_{ab} \\ \Psi(K^{\sigma}_{a_x b_y}) = K_{ab} \\ \Psi(\{m_1\}_{m_2}) = a^{l_1,l_2}(\{\Psi(m_1)\}_{\Psi(m_2)}) \\ \Psi(m_1.m_2) = a^{l_1,l_2}(\Psi(m_1).\Psi(m_2)) \\ \Psi((\sigma.i.\ X \longrightarrow Y : m).\ \mathsf{P}) = (\sigma.i.\ \Psi(X) \longrightarrow \Psi(Y) : \Psi(m)).\ \Psi(\mathsf{P}) \\ \Psi(x) = cte \quad \text{otherwise} \end{cases}$$

TABLE V

ABSTRACT INTRUDER KNOWLEDGE GENERATION FROM ABSTRACT TRACES

$$\textsf{Init} \qquad \frac{}{\epsilon \vdash\!\sim \mathcal{K}_I{}^{\#}_{\Downarrow}}$$

$$\textsf{Get knowledge} \qquad \frac{t^{\#} \vdash\!\sim M}{t^{\#}.\ (\sigma.i.\ A \longrightarrow B : m^a) \vdash\!\sim M \cup m^a}$$

where $t^{\#}$ is the finite abstract trace generated from the abstract protocol $\mathsf{P}^{\#}$ and $\sqcap$ is the greatest lower bound (glb) function over the poset $D^{\#}(\leq_a)$.

It is clear that with our definition, we can build only a semi-decidable procedure since if there exists $x \in \mathcal{IK}(t^{\#})$ such that $x \sqcap m^a \neq \bot$, we cannot decide that the protocol contains a secrecy flaw.

VI. CASE STUDY

The protocol presented in Table VI has been proved to be correct with respect to secrecy in [13]. The protocol aims to distribute the key $K_{a_x b_y}$ to the agents A_x and B_y with the aid of the trusted server S. A_x initiates the session by sending to S his identity, the identity of the agent he wants to communicate with, B_y, and a nonce N_{a_x}, all encrypted with the long-term key $K_{a_x s}$ that he shares with S. The server replies by sending to A_x a message that contains a short-term key $K_{a_x b_y}$ that will be shared by A_x and B_y, encrypted with $K_{a_x s}$. Then S sends a message to B_y containing the identities of A_x and B_y and the key $K_{a_x b_y}$, all encrypted with the long-term key $K_{b_y s}$ that he shares with the agent B_y. Through this last message, B_y learns that A_x intends to communicate with him using the secret key $K_{a_x b_y}$.

To prove the correctness of the $Houmani - Mejri$ protocol with respect to secrecy, we demonstrate that the protocol complies with definition 3. In order to do this, we need to compute the intruder's knowledge associated with the abstract trace $t^{\#}$ generated by the execution of the protocol. As stated in section IV, the intruder knows the identities of the agents and server. The intruder also possesses a nonce N_i and a key shared with the server K_{is}. The initial intruder knowledge is illustrated by:

$$\mathcal{K}_I = \{A, B, S, N_i, K_{is}\}$$

For convenience, we denote by m_1, m_2 and m_3 the messages transmitted during the first, second and third protocol steps, respectively. For $l_1 = 0$ or $l_2 = 0$, $m_1^a = m_2^a = m_3^a = \top$. The abstraction has gone too deep and there is no more useful information for the intruder. The problem of protocol correctness is undecidable in this case. For $l_1 \geq 2$ and $l_2 \geq 2$, $m_1^a = m_1$, $m_2^a = m_2$ and $m_3^a = m_3$. There is no benefit in abstracting the messages, as the intruder does not gain any information. Therefore, the values of the pair (l_1, l_2) that could be useful for the intruder are the tuples (1,1), (2,1) and (1,2). We fix the values of the parameters to $l_1 = 1$ and $l_2 = 1$ and compute the abstracted messages m_1^a, m_2^a and m_3^a:

$$\begin{aligned} a^{1,1}(m_1) &= a^{1,1}(\{A.B.N_a\}_{K_{as}}) \\ &= \{a^{0,1}(A.B.N_a)\}_{K_{as}} \\ &= \{a^{0,0}(A).a^{0,0}(B.N_a)\}_{K_{as}} \\ &= \{A.\top\}_{K_{as}} \end{aligned}$$

TABLE VI
THE KEY DISTRIBUTION PROTOCOL

Message σ.1.	A_x	$\longrightarrow$	S	$: \{A_x.B_y.N^{\sigma}_{a_x}\}_{K_{a_x s}}$
Message σ.2.	S	$\longrightarrow$	A_x	$: \{\{A_x\}_{N^{\sigma}_{a_x}}.B_y.K^{\sigma}_{a_x b_y}\}_{K_{a_x s}}$
Message σ.3.	S	$\longrightarrow$	B_y	$: \{A_x.B_y.K^{\sigma}_{a_x b_y}\}_{K_{b_y s}}$

$$\begin{aligned} a^{1,1}(m_2) &= a^{1,1}(\{\{A\}_{N_a}.B.K_{ab}\}_{K_{as}}) \\ &= \{a^{0,1}(\{A\}_{N_a}.B.K_{ab})\}_{K_{as}} \\ &= \{a^{0,0}(\{A\}_{N_a}).a^{0,0}(B.K_{ab})\}_{K_{as}} \\ &= \{\top.\top\}_{K_{as}} \end{aligned}$$

$$\begin{aligned} a^{1,1}(m_3) &= a^{1,1}(\{A.B.K_{ab}\}_{K_{bs}}) \\ &= \{a^{0,1}(A.B.K_{ab})\}_{K_{bs}} \\ &= \{a^{0,0}(A).a^{0,0}(B.K_{ab})\}_{K_{bs}} \\ &= \{A.\top\}_{K_{bs}} \end{aligned}$$

The abstract trace $t^{\#}$ for $l_1 = 1$ and $l_2 = 1$ is:

$$\begin{aligned} t^{\#} = \; & (\sigma.1.\ A \longrightarrow I(S) : \{A.\top\}_{K_{as}}). \\ & (\sigma.1.\ I(A) \longrightarrow S : \{A.\top\}_{K_{as}}). \\ & (\sigma.2.\ S \longrightarrow I(A) : \{\top.\top\}_{K_{as}}). \\ & (\sigma.2.\ I(S) \longrightarrow A : \{\top.\top\}_{K_{as}}). \\ & (\sigma.3.\ S \longrightarrow I(B) : \{A.\top\}_{K_{bs}}) \end{aligned}$$

The intruder is not able to decrypt any of the abstracted messages. Consequently, he can not replace any message in subsequent protocol executions. Since no fresh messages can be transmitted, the trace can not be extended with new actions. The complete abstract trace of the $Houmani-Mejri$ protocol is then $t^{\#}$. The intruder knowledge associated with the abstract trace $t^{\#}$ is:

$$\mathcal{IK}(t^{\#}) = (\mathcal{K}_I \cup \{\{A.\top\}_{K_{as}}, \{\top.\top\}_{K_{as}}, \{A.\top\}_{K_{bs}}\})_{\Downarrow}$$

We compare the elements of the set $\mathcal{IK}(t^{\#})$ with the value K_{ab} that is supposed to be kept secret by P. The process is trivial for the elements of the initial intruder knowledge set $\mathcal{K}_I$:

$$\begin{aligned} K_{ab} \sqcap A &= K_{ab} \sqcap B = K_{ab} \sqcap S \\ &= K_{ab} \sqcap N_i = K_{ab} \sqcap K_{is} \\ &= \bot \end{aligned}$$

The three elements of $\mathcal{IK}(t^{\#})$ obtained by abstraction are encrypted messages that don't contain K_{ab}. In this case, we have the following relation:

$$\begin{aligned} K_{ab} \sqcap \{A.\top\}_{K_{as}} &= K_{ab} \sqcap \{\top.\top\}_{K_{as}} \\ &= K_{ab} \sqcap \{A.\top\}_{K_{bs}} \\ &= \bot \end{aligned}$$

The rest of the elements of the set $\mathcal{IK}(t^{\#})$ can be obtained by applying the closure operations from definition 1. Since none of the basic components contain K_{ab}, the resulting elements will also not contain K_{ab}. Therefore, the condition for secrecy correctness is fulfilled by the protocol:

$$\forall x \in \mathcal{IK}(t^{\#}) \; : \; x \sqcap K_{ab} = glb(x, K_{ab}) = \bot$$

VII. CONCLUSION

The main intent of this work is to characterize the verification of the secrecy property as an abstract interpretation of security protocols. The abstract model was defined by the means of a finite abstract trace that models all of the potential intruder behaviors. The model was applied to approximate the unbounded space of intruder knowledge to a finite one. This information was then used to decide if the security protocol satisfies the secrecy property for sensitive data. Since the abstract trace is finite, it becomes possible to compute the finite abstract set $M^{\#}$ of messages that the intruder can get by running a security protocol P. Given a message m, if for all x in $M^{\#}$, $glb(x, m^a) = x \sqcap m^a = \bot$ then m is distinct from all messages in M and consequently P does not leak the message m. The approximation is correct with respect to the original protocol execution, as it overestimates the capabilities of the intruder.

REFERENCES

[1] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication, from Proceedings of the Royal Society, volume 426, number 1871, 1989," in *William Stallings, Practical Cryptography for Data Internetworks*. IEEE Computer Society Press, 1996, 1996. [Online]. Available: citeseer.nj.nec.com/burrows90logic.html

[2] K. Adi, M. Debbabi, and M. Mejri, "A New Logic for Electronic Commerce Protocols," *Theoretical Computer Science (TCS)*, vol. 291, no. 2, 2002.

[3] G. Lowe, "Breaking and Fixing the Needham Schroeder Public-Key Protocol using FDR," in *Proceedings of TACAS*, vol. 1055. Springer Verlag, 1996, pp. 147–166.

[4] ——, "SPLICE-AS: A Case Study in Using CSP to Detect Errors in Security Protocols," Programming Research Group, Oxford, Tech. Rep., 1996.

[5] B. Donovan, P. Norris, and G. Lowe, "Analyzing a library of security protocols using Casper and FDR," in *Workshop on Formal Methods and SecurityProtocols*, 1999.

[6] C. Meadows, "The NRL protocol analyzer: An overview," *Journal of Logic Programming*, vol. 26, no. 2, pp. 113–131, 1996. [Online]. Available: citeseer.ist.psu.edu/meadows96nrl.html

[7] D. Bolignano, "Towards a mechanization of cryptographic protocol verification," in *9th International Computer-Aided Verification Conference, CAV'97*, Jun 1997.

[8] L. C. Paulson, "Proving properties of security protocols by induction," in *10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997, pp. 70–83.

[9] D. Monniaux, "Abstracting cryptographic protocols with tree automata," in *Static Analysis Symposium*, 1999, pp. 149–163. [Online]. Available: citeseer.nj.nec.com/monniaux99abstracting.html

[10] J. Goubault-Larrecq, "A method for automatic cryptographic protocol verification (extended abstract)," in *FMPPTA'2000, Cancun, Mexico*, May 2000. [Online]. Available: citeseer.nj.nec.com/303679.html
[11] K. Adi and M. Debbabi, "A game semantics approach for security protocols," in *6th International Symposium on Programming and Systems, ISPS'2003.*, 2002, pp. 209–227.
[12] L. Bozga, Y. Lakhnech, and M. Périn, "Pattern-based abstraction for verifying secrecy in protocols," in *TACAS*, ser. Lecture Notes in Computer Science, vol. 2619. Springer, 2003, pp. 299–314.
[13] H. Houmani and M. Mejri, "Secure protocols for secrecy," in *Foundations of Computer Security*, I. Cervesato, Ed., Ottawa, Canada, 26–27 June 2003, pp. 59–68.
[14] C. Cremers, S. Mauw, and E. de Vink, "Defining authentication in a trace model," in *FAST 2003*, ser. Proceedings of the first international Workshop on Formal Aspects in Security and Trust. Pisa: IITT-CNR technical report, 2003, pp. 131–145.
[15] T. Y. C. Woo and S. S. Lam, "A lesson on authentication protocol design," *Operating Systems Review*, vol. 28, no. 3, pp. 24–37, 1994.
[16] A. Deutsch, "Interprocedural may-alias analysis for pointers: beyond k-limiting," *ACM SIGPLAN Notices*, vol. 29, no. 6, pp. 230–241, 1994. [Online]. Available: citeseer.ist.psu.edu/deutsch94interprocedural.html
[17] D. Bert, R. Echahed, and B. Ostvold, "Abstract rewriting," in *Third International Workshop on Static Analysis*. Lecture Notes in Computer Science 724, Springer-Verlag, 1993, pp. 178–192.
[18] D. Bert, R. Echahed, and K. Adi, "Resolution of goals with the functional and logic programming language LPG: Impact of abstract interpretation," pp. 629–632, 1996.

APPENDIX

Proposition 1 (1): Let $m \in D^{\#}$, l_1, l_2, l'_1 and l'_2 four natural numbers such that $l_1 \geq l'_1$ and $l_2 \geq l'_2$, then:

$$a^{l_1,l_2}(m) \leq_a a^{l'_1,l'_2}(m)$$

Proof: The proof is by structural induction on m.

- m is atomic:

$$a^{l_1,l_2}(m) = m \leq_a m = a^{l'_1,l'_2}(m) \tag{5}$$

- $m = m_1.m_2$: We distinguish the following cases:
  - $l'_2 = 0$:

$$a^{l_1,l_2}(m) \leq_a \top = a^{l'_1,0}(m) = a^{l'_1,l'_2}(m) \tag{6}$$

  - $l'_2 \neq 0$:
    By definition of the function $a^{-,-}(\_)$ we have:

$$a^{l_1,l_2}(m) = a^{l_1,l_2-1}(m_1).a^{l_1,l_2-1}(m_2) \tag{7}$$

    By induction hypothesis, we have:

$$\left\{ \begin{array}{lcl} a^{l_1,l_2-1}(m_1) & \leq_a & a^{l'_1,l'_2-1}(m_1) \\ a^{l_1,l_2-1}(m_2) & \leq_a & a^{l'_1,l'_2-1}(m_2) \end{array} \right. \tag{8}$$

    By definition of the ordering and 8, we have:

$$\begin{array}{c} a^{l_1,l_2-1}(m_1).a^{l_1,l_2-1}(m_2) \leq_a \\ a^{l'_1,l'_2-1}(m_1).a^{l'_1,l'_2-1}(m_2) \end{array} \tag{9}$$

    By definition of the function $a^{-,-}(\_)$ and 9, we have:

$$\begin{array}{c} a^{l_1,l_2}(m_1.m_2) = a^{l_1,l_2}(m) \leq_a \\ a^{l'_1,l'_2}(m) = a^{l'_1,l'_2}(m_1.m_2) \end{array} \tag{10}$$

- $m = \{m_1\}_K$: we distinguish the following cases
  - $l'_1 = 0$:

$$a^{l_1,l_2}(m) \leq_a \top = a^{0,l'_2}(m) = a^{l'_1,l'_2}(m) \tag{11}$$

  - $l'_1 \neq 0$:
    By definition of the function $a^{-,-}(\_)$ we have:

$$a^{l_1,l_2}(m) = \{a^{l_1-1,l_2}(m_1)\}_K \tag{12}$$

    By induction hypothesis, we have:

$$a^{l_1-1,l_2}(m_1) \leq_a a^{l'_1-1,l'_2}(m_1) \tag{13}$$

    By definition of the ordering and 13, we have:

$$\{a^{l_1-1,l_2}(m_1)\}_K \leq_a \{a^{l'_1-1,l'_2}(m_1)\}_K \tag{14}$$

    By definition of the function $a^{-,-}(\_)$ we deduce:

$$a^{l_1,l_2}(m) \leq_a a^{l'_1,l'_2}(m) \tag{15}$$

■

Proposition 1 (2): The abstraction function $a^{-,-}(\_)$ is extensive, i.e.:

$$\forall m \in D^{\#},\ \forall l_1, l_2 \in \mathbb{N} \ :\ m \leq_a a^{l_1,l_2}(m) \tag{16}$$

Proof: Let $m \in D^{\#}$ and let l_1 and l_2 be two natural numbers. The proof is by structural induction on m.

- m is atomic:

$$m \leq_a m = a^{l_1,l_2}(m) \tag{17}$$

- $m = m_1.m_2$: By induction hypothesis, we have:

$$\left\{ \begin{array}{lcl} m_1 & \leq_a & a^{l_1,l_2}(m_1) \\ m_2 & \leq_a & a^{l_1,l_2}(m_2) \end{array} \right. \tag{18}$$

  By definition of the ordering and 18, we have:

$$m_1.m_2 \leq_a a^{l_1,l_2}(m_1).a^{l_1,l_2}(m_2) \tag{19}$$

  By definition of $a^{-,-}(\_)$, we have:

$$a^{l_1,l_2}(m_1).a^{l_1,l_2}(m_2) = a^{l_1,l_2+1}(m_1.m_2) \tag{20}$$

  By 19 and 20, we have:

$$m_1.m_2 \leq_a a^{l_1,l_2+1}(m_1.m_2) \tag{21}$$

  By the proposition 1, we have

$$a^{l_1,l_2+1}(m_1.m_2) \leq_a a^{l_1,l_2}(m_1.m_2) \tag{22}$$

  By 21 and 22, we have:

$$m_1.m_2 \leq_a a^{l_1,l_2}(m_1.m_2) \tag{23}$$

- $m = \{m_1\}_K$: By induction hypothesis, we have:

$$m_1 \leq_a a^{l_1,l_2}(m_1) \tag{24}$$

  By definition of the ordering, we have:

$$\{m_1\}_K \leq_a \{a^{l_1,l_2}(m_1)\}_K \tag{25}$$

By definition of a, we have:

$$\{m_1\}_K \leq_a a^{l_1+1,l_2}(\{m_1\}_K) \tag{26}$$

By the proposition 1, we have:

$$a^{l_1+1,l_2}(\{m_1\}_K) \leq_a a^{l_1,l_2}(\{m_1\}_K) \tag{27}$$

By 26 and 27, we deduce:

$$\{m_1\}_K \leq_a a^{l_1,l_2}(\{m_1\}_K) \tag{28}$$

■

Proposition 1 (3): The abstraction function $a^{-,-}(\_)$ is idempotent, i.e.:

$$a \circ a = a \tag{29}$$

Proof: Let $m \in D^{\#}$ and let l_1 and l_2 be two natural numbers. The proof is by structural induction on m.

- m is atomic:

$$a^{l_1,l_2}(a^{l_1,l_2}(m)) = m = a^{l_1,l_2}(m) \tag{30}$$

- $m = m_1.m_2$. By definition of $a^{-,-}(\_)$, we have:

$$\begin{array}{c} a^{l_1,l_2}(a^{l_1,l_2}(m_1.m_2)) = \\ a^{l_1,l_2-1}(a^{l_1,l_2-1}(m_1)).a^{l_1,l_2-1}(a^{l_1,l_2-1}(m_2)) \end{array} \tag{31}$$

By induction hypothesis, we have:

$$\left\{ \begin{array}{rcl} a^{l_1,l_2-1}(a^{l_1,l_2-1}(m_1)) & = & a^{l_1,l_2-1}(m_1) \\ a^{l_1,l_2-1}(a^{l_1,l_2-1}(m_2)) & = & a^{l_1,l_2-1}(m_2) \end{array} \right. \tag{32}$$

By 31 and 32, we have:

$$a^{l_1,l_2}(a^{l_1,l_2}(m_1.m_2)) = a^{l_1,l_2-1}(m_1).a^{l_1,l_2-1}(m_2) \tag{33}$$

By definition of $a^{-,-}(\_)$ and 33, we deduce:

$$a^{l_1,l_2}(a^{l_1,l_2}(m_1.m_2)) = a^{l_1,l_2}(m_1.m_2) \tag{34}$$

- $m = \{m_1\}_K$. By definition of $a^{-,-}(\_)$, we have:

$$a^{l_1,l_2}(a^{l_1,l_2}(\{m_1\}_K)) = \{a^{l_1-1,l_2}(a^{l_1-1,l_2}(m_1))\}_K \tag{35}$$

By induction hypothesis, we have:

$$a^{l_1-1,l_2}(a^{l_1-1,l_2}(m_1)) = a^{l_1-1,l_2}(m_1) \tag{36}$$

By 35 and 36, we have:

$$a^{l_1,l_2}(a^{l_1,l_2}(\{m_1\}_K)) = \{a^{l_1-1,l_2}(m_1)\}_K \tag{37}$$

By definition of $a^{-,-}(\_)$ and 37, we deduce:

$$a^{l_1,l_2}(a^{l_1,l_2}(\{m_1\}_K)) = a^{l_1,l_2}(\{m_1\}_K) \tag{38}$$

■

In the sequel we prove that the abstraction function $a^{-,-}(\_)$ is monotone. First, we have to prove technical lemmas which state that the size of an abstracted message can be smaller than or equal to the size of the original message. The intuition is that the abstraction leads to loss of detail, which is reflected in the contracted depth and width of the resulting message. The equality corresponds to the special case of the original and the abstracted message being the same.

We define the size of a message, written $|\_|$ as follows:

$$\left\{ \begin{array}{l} m \text{ is atomic } \Rightarrow |m| = 0 \\ |m.m'| = |m| + |m'| + 1 \\ |\{m\}_{m'}| = |m| + 1 \end{array} \right.$$

Lemma 1: $\forall m \in D^{\#},\ \forall l_1, l_2 \in \mathbb{N}\ :\ |a^{l_1,l_2}(m)| \leq |m|$

Proof: Let $m \in D^{\#}$ and let l_1 and l_2 be two natural numbers. The proof is by structural induction on m.

- m is atomic. By definition $a^{l_1,l_2}(m) = m$, then we have:

$$|a^{l_1,l_2}(m)| = |m| \leq |m| \tag{39}$$

- $m = m_1.m_2$. By definition of $a^{-,-}(\_)$, we have:

$$a^{l_1,l_2}(m_1.m_2) = a^{l_1,l_2-1}(m_1).a^{l_1,l_2-1}(m_2) \tag{40}$$

By 40 and the definition of $|\_|$, we have:

$$\begin{array}{rcl} |a^{l_1,l_2}(m_1.m_2)| & = & |a^{l_1,l_2-1}(m_1)| \\ & + & |a^{l_1,l_2-1}(m_2)| \\ & + & 1 \end{array} \tag{41}$$

By induction hypothesis, we have:

$$\left\{ \begin{array}{l} |a^{l_1,l_2-1}(m_1)| \leq |m_1| \\ |a^{l_1,l_2-1}(m_2)| \leq |m_2| \end{array} \right. \tag{42}$$

By 42, we have:

$$|a^{l_1,l_2-1}(m_1)| + |a^{l_1,l_2-1}(m_2)| + 1 \leq |m_1| + |m_2| + 1 \tag{43}$$

By 43 and the definition of $|\_|$ we have:

$$|a^{l_1,l_2-1}(m_1).a^{l_1,l_2-1}(m_2)| \leq |m_1.m_2| \tag{44}$$

By 44 and the definition of $a^{-,-}(\_)$ we have:

$$|a^{l_1,l_2}(m_1.m_2)| \leq |m_1.m_2| \tag{45}$$

- $m = \{m_1\}_K$. By definition of $a^{-,-}(\_)$, we have:

$$a^{l_1,l_2}(\{m_1\}_K) = \{a^{l_1-1,l_2}(m_1)\}_K \tag{46}$$

By 46 and the definition of $|\_|$, we have:

$$|a^{l_1,l_2}(\{m_1\}_K)| = |a^{l_1-1,l_2}(m_1)| + 1 \tag{47}$$

By induction hypothesis, we have:

$$|a^{l_1-1,l_2}(m_1)| \leq |m_1| \tag{48}$$

By $\leq$ property and 48 we have:

$$|a^{l_1-1,l_2}(m_1)| + 1 \leq |m_1| + 1 \quad (49)$$

By 49 and the definition of $|\_|$ we have:

$$|\{a^{l_1-1,l_2}(m_1)\}_K| \leq |\{m_1\}_K| \quad (50)$$

By 50 and the definition of $a^{-,-}(\_)$ we have:

$$|a^{l_1,l_2}(\{m_1\}_K)| \leq |\{m_1\}_K| \quad (51)$$

■

Lemma 2: $\forall m \in D^{\#},\ \forall l_1, l_2 \in \mathbb{N}\ :\ |a^{l_1,l_2}(m)| = |m| \Rightarrow a^{l_1,l_2}(m) = m$

Proof: Let $m \in D^{\#}$ and let l_1 and l_2 be two natural numbers. The proof is by structural induction on m.

- m is atomic. By definition of $a^{-,-}(\_)$, we have:

$$a^{l_1,l_2}(m) = m$$

- $m = m_1.m_2$. By definition of $a^{-,-}(\_)$ and $|\_|$ operator, we have:

$$\left\{ \begin{array}{lcl} |m_1.m_2| & = & |m_1| + |m_2| + 1 \\ |a^{l_1,l_2}(m_1.m_2)| & = & |a^{l_1,l_2-1}(m_1)| \\ & + & |a^{l_1,l_2-1}(m_2)| + 1 \end{array} \right. \quad (52)$$

By hypothesis and 52, we deduce:

$$|a^{l_1,l_2-1}(m_1)| + |a^{l_1,l_2-1}(m_2)| = |m_1| + |m_2| \quad (53)$$

By lemma 1 and 53, we deduce:

$$\left\{ \begin{array}{lcl} |a^{l_1,l_2-1}(m_1)| & = & |m_1| \\ |a^{l_1,l_2-1}(m_2)| & = & |m_2| \end{array} \right. \quad (54)$$

By induction hypothesis and 54, we deduce:

$$\left\{ \begin{array}{lcl} a^{l_1,l_2-1}(m_1) & = & m_1 \\ a^{l_1,l_2-1}(m_2) & = & m_2 \end{array} \right. \quad (55)$$

By definition of $a^{-,-}(\_)$ and 55, we have:

$$\begin{array}{rcl} a^{l_1,l_2-1}(m_1).a^{l_1,l_2-1}(m_2) & = & a^{l_1,l_2}(m_2.m_2) \\ & = & m_1.m_2 \end{array} \quad (56)$$

- $m = \{m_1\}_K$. By definition of $a^{-,-}(\_)$ and $|\_|$ operator, we have:

$$\left\{ \begin{array}{lcl} |a^{l_1,l_2}(\{m_1\}_K)| & = & |\{a^{l_1-1,l_2}(m_1)\}_K| \\ & = & |a^{l_1-1,l_2}(m_1)| + 1 \\ |\{m_1\}_K| & = & |m_1| + 1 \end{array} \right. \quad (57)$$

By hypothesis and 57, we deduce:

$$|a^{l_1-1,l_2}(m_1)| = |m_1| \quad (58)$$

By induction hypothesis and 58, we deduce:

$$a^{l_1-1,l_2}(m_1) = m_1 \quad (59)$$

By definition of $a^{-,-}(\_)$ and 59, we deduce:

$$\{a^{l_1,l_2-1}(m_1)\}_K = a^{l_1,l_2}(\{m_1\}_K) = \{m_1\}_K \quad (60)$$

■

Proposition 1 (4): The abstraction function $a^{-,-}(\_)$ is monotone, i.e.:

$$\forall m, m' \in D^{\#},\ \forall l_1, l_2 \in \mathbb{N}\ :\ m \leq_a m' \Rightarrow a^{l_1,l_2}(m) \leq_a a^{l_1,l_2}(m') \quad (61)$$

Proof:

Let $m, m' \in D^{\#}$ such that $m \leq_a m'$. Let l_1 and l_2 be two natural numbers. The proof is by induction on $H = |m|$:

- $H = 0$. This means that m is atomic. We have two cases
  - $m' = m$.

$$a^{l_1,l_2}(m) = m \leq_a m' = a^{l_1,l_2}(m') \quad (62)$$

  - $m' = \top$. By definition of the relation $\leq_a$ and a, we have:

$$\begin{array}{rcl} a^{l_1,l_2}(m) & = & m \leq_a \top = a^{l_1,l_2}(\top) \\ & = & a^{l_1,l_2}(m') \end{array} \quad (63)$$

- $H = n$. By the proposition 1, we have:

$$m' \leq_a a^{l_1,l_2}(m') \quad (64)$$

Since $m \leq_a m'$ and by 64, we deduce:

$$m \leq_a a^{l_1,l_2}(m') \quad (65)$$

By the lemma 1, we have two cases:

  - $|a^{l_1,l_2}(m)| < |m|$, then by induction hypothesis, we have:

$$a^{l_1,l_2}(a^{l_1,l_2}(m)) \leq_a a^{l_1,l_2}(m') \quad (66)$$

By proposition 1 and 66, we have:

$$a^{l_1,l_2}(m) \leq_a a^{l_1,l_2}(m') \quad (67)$$

  - $|a^{l_1,l_2}(m)| = |m|$. By lemma 2, we deduce that:

$$a^{l_1,l_2}(m) = m \quad (68)$$

By 68 and 65, we deduce:

$$a^{l_1,l_2}(m) \leq_a a^{l_1,l_2}(m') \quad (69)$$

■

Proposition 2: Let m be a message, l_1 and l_2 be two natural numbers then:

$$|a^{l_1,l_2}(m)| \leq 2^{l_1+l_2} - 1$$

Proof: The proof is by structural induction on m.

- m is atomic. By definition $a^{l_1,l_2}(m) = m$, then we have:

$$|a^{l_1,l_2}(m)| = |m| = 0 \leq 2^{l_1+l_2} - 1 \quad (70)$$

- $m = \{m'\}_K$. By definition of the abstraction function a, we have:

$$a^{l_1,l_2}(\{m'\}_K) = \{a^{l_1-1,l_2}(m')\}_K \quad (71)$$

By definition of the operator $|\_|$ and 71, we have:

$$|a^{l_1,l_2}(\{m'\}_K)| = |a^{l_1-1,l_2}(m')| + 1 \quad (72)$$

By induction hypothesis, we have:

$$\begin{aligned} |a^{l_1-1,l_2}(m')| &\leq 2^{l_1-1+l_2} - 1 = \frac{2^{l_1+l_2}-2}{2} \\ &\leq 2^{l_1+l_2} - 2 \end{aligned} \quad (73)$$

By 71, 72 and 73, we deduce:

$$|a^{l_1,l_2}(\{m'\}_K)| \leq 2^{l_1+l_2} - 1 \quad (74)$$

- $m = m_1.m_2$. By definition of the abstraction function a, we have:

$$a^{l_1,l_2}(m_1.m_2) = a^{l_1,l_2-1}(m_1).a^{l_1,l_2-1}(m_2) \quad (75)$$

By definition of the operator $|\_|$ and 75, we have:

$$\begin{aligned} |a^{l_1,l_2}(m_1.m_2)| &= |a^{l_1,l_2-1}(m_1)| \\ &+ |a^{l_1,l_2-1}(m_2)| \\ &+ 1 \end{aligned} \quad (76)$$

By induction hypothesis, we have:

$$\begin{cases} |a^{l_1,l_2-1}(m_1)| \leq 2^{l_1+l_2-1} - 1 \\ |a^{l_1,l_2-1}(m_2)| \leq 2^{l_1+l_2-1} - 1 \end{cases} \quad (77)$$

By 76, 77, we deduce:

$$\begin{aligned} |a^{l_1,l_2}(m_1.m_2)| &\leq (2^{l_1+l_2-1} - 1) \\ &+ (2^{l_1+l_2-1} - 1) + 1 \\ &= 2^{l_1+l_2} - 1 \end{aligned} \quad (78)$$

■

Proposition 3: Let M be a set of messages such that the set of atomic messages in M is finite. If the depth of each message in M is bounded then the set M is finite.

Proof: Let $|M| = Max\{|m| \mid m \in M\}$ be the maximum length of the messages in M (a bound). The proof is by induction on $|M|$.

- $|M| = 0$. We deduce that each message in M is atomic. By hypothesis, we conclude that M is finite.
- $|M| = n > 0$. Let $M = M_1 \cup M_2$ such that $|M_1| = n - 1$ and $M_2 = \{m \in M \mid |m| = n\}$. By induction hypothesis, we have:

$$M_1 \text{ is a finite set} \quad (79)$$

Let $m \in M_2$. Since $|m| = n > 0$, we have the two following cases:

$$\begin{cases} m = \{m_1\}_K \ \wedge \ |m_1| = n - 1 \\ m = m_1.m_2 \ \wedge \ |m_1| < n \ \wedge \ |m_2| < n \end{cases} \quad (80)$$

We build the set M' as follows:

$$\begin{cases} \{m_1\}_K \in M_2 \ \Rightarrow \ \{m_1, K\} \subseteq M' \\ m_1.m_2 \in M_2 \ \Rightarrow \ \{m_1, m_2\} \subseteq M' \end{cases} \quad (81)$$

Since the encryption keys are atomic messages, then, by 80, we deduce that:

$$|M'| < n \quad (82)$$

By induction hypothesis, we have:

$$M' \text{ is a finite set} \quad (83)$$

We build the set M'' as follows:

$$\begin{cases} \{m_1, K\} \subseteq M' \ \Rightarrow \ \{m_1\}_K \in M'' \\ \{m_1, m_2\} \subseteq M' \ \Rightarrow \ m_1.m_2 \in M'' \end{cases} \quad (84)$$

By 81 and 84, we deduce:

$$M_2 \subseteq M'' \quad (85)$$

By 83 and 84, we have:

$$M'' \text{ is finite} \quad (86)$$

By 85 and 86, we have:

$$M_2 \text{ is finite} \quad (87)$$

By 79 and 87, we deduce that:

$$M \text{ is finite} \quad (88)$$

■

Proposition 4: Let M be a finite set of messages. Then $M^{\#}_{\Downarrow}$ is finite.

Proof: Since M is finite, then:

$$\{m \in M \mid |m| = 0\} \text{ is finite} \quad (89)$$

Let $m \in M^{\#}_{\Downarrow}$, by the definition of $M^{\#}_{\Downarrow}$ and Proposition 2, we have:

$$|m| \leq 2^{l_1+l_2} - 1 \quad (90)$$

By 89, 90 and Proposition 3, we deduce that $M^{\#}_{\Downarrow}$ is finite.

■

Proposition 5 (Finiteness): Let P be a cryptographic protocol and let $\mathsf{P}^{\#}$ be the corresponding abstracted protocol. Then, the multi-session, multi-role trace execution model $t^{\#}$ of $\mathsf{P}^{\#}$ is finite.

Proof: As stated in section II, a trace can be extended with an event a whenever the message transmitted during that action is fresh. We will prove that, because of the protocol abstraction, the number of distinct messages that can be issued is bounded.

The abstraction of atomic messages bounds the number of agents, keys (public and session) and nonces. Henceforth, we denote by Nb_{atoms} their total number. We remind the reader that the inductive rules that can lead to a trace extension are Receive and Intruder (Table II). They define how the intruder can increase his knowledge set either by intercepting messages and decomposing them, or by building new messages from pieces of information that he already possesses. We fix l_1 and l_2 and adapt the two rules to our abstract trace model $t^{\#}$. In the following, we analyze their effect on the abstracted protocol runs.

The Receive rule is modified to reflect that the intruder can only obtain abstracted messages m^a. Likewise, the Intruder rule is changed to indicate that the intruder can only send m^a. The size of those messages, $|m^a|$ is limited, as shown in Proposition 2:

$$|m^a| \leq 2^{l_1+l_2} - 1$$

The intuition is quite simple: if, for any message size between 0 and $2^{l_1+l_2} - 1$, the number of messages is bounded, then there is a limited number of possible messages m^a.

Let m, m' and m'' be atomic messages. The total number of messages of size 0 (atomic messages) is Nb_{atoms}. For the messages of size 1, we have to count both encryptions and concatenations of atoms. For each of them, the number is $(Nb_{atoms})^2$. So, the total number of possible messages of size 1 is $2 \times (Nb_{atoms})^2$. For size 2, the computations become more complex. We have several different combinations of operations on atoms that can produce messages of size 2: $m.m'.m''$, $\{m.m'\}''_m$, $\{\{m\}'_m\}''_m$, $m.\{m'\}''_m$ and $\{m\}'_m.m''$. For each of them, there can be $(Nb_{atoms})^3$ possible messages. Consequently, the total number of messages of size 2 is $5 \times (Nb_{atoms})^3$. The reasoning can be continued until size $2^{l_1+l_2} - 1$ is reached. For each size $s \in [0, 2^{l_1+l_2} - 1]$, there will be an increasing number of combinations of messages, each of them with $(Nb_{atoms})^s$ possible messages. Although the total number increases dramatically, it is still finite.

New actions can be added to the trace only if the messages are fresh. Since the number of messages that can be sent or received by the intruder is bounded, the number of inductive rules that describe the corresponding actions is also limited. Therefore, the abstract trace $t^{\#}$ is rendered finite.

■