Towards the Correctness of Security Protocols

Mourad Debbabi[1]

Panasonic Information and Networking Technologies Laboratory
Princeton, New Jersey, USA

Mohamed Mejri[2]

Computer Science Department, Laval University
Sainte Foy, Quebec, Canada

Abstract

In [19], the authors presented a type-theoretic approach to the verification of security protocols. In this approach, a universal type system is proposed to capture in a finite way all the possible computations (internal actions or protocol instrumentations) that could be performed by a smart malicious intruder. This reduces the verification of cryptographic protocols to a typing problem where types are attack scenarios. In this paper, we recall this type system and we prove its completeness i.e. if the intruder can learn a message from a given protocol instrumentation, then this message could be infered from the type system. A significant result of this paper is the presentation of a new transformation that allows us to abstract a non-terminating type inference system into a terminating deductive proof system. We demonstrate how these results could be used to establish the security of cryptographic protocols from the secrecy standpoint. Finally, the usefulness and the efficiency of the whole approach is illustrated by proving the correctness of a new version of the Needham-Shoreder protocol with respect to the secrecy property.

1 Motivations and Background

Information technology is becoming, more and more, a vitally important underpinning of our economy and society. It is embedded in our everyday applications and animates a wide class of systems that range from small to large, and from simple to extremely complex. Actually, information systems increasingly govern nearly every aspect of our lives. This omnipresence is largely increased by the dazzling expansion of Internet, World Wide Web, parallel and distributed systems and mobile computation. In such contexts, information must be protected against mystification, destruction and disclosure. Accordingly, a great deal of interest has been expressed in the development and use of cryptographic protocols.

[1] Email: debbabim@research.panasonic.com
[2] Email: mejri@ift.ulaval.ca

A protocol is an orderly defined sequence of communication and computation steps. A communication step transfers messages from one principal (sender) to another (receiver), while a computation step updates a principal's internal state. A protocol with a security objective is called a cryptographic protocol. Cryptographic functions are used to achieve such an objective. In the literature, two major classes of cryptographic protocols have been advanced: authentication protocols and key distribution protocols. The primary objective of authentication protocols is to allow principals to identify themselves to each other. Key distribution protocols aim to distribute cryptographic keys between principals.

Today, it is well known that the design of cryptographic protocols is error prone. Several protocols have been shown flawed in computer security literature [14] many years after their publication and use. Moreover, the correctness of these protocols is paramount, especially when we consider the size of the networks involved and the desire of principals to put confidential information and to allow for financial transactions to take place across the network.

The primary objective of this work is to build on top of the work proposed in [19] to contribute to the correctness of security protocols. Actually, in [19], the authors presented a type-theoretic approach to the verification of security protocols. In this approach, a universal type system is proposed to capture in a finite way all the possible computations (internal actions or protocol instrumentations) that could be performed by a smart malicious intruder. This reduces the verification of cryptographic protocols to a typing problem where types are attack scenarios. In this paper, we recall this type system and we prove its completeness i.e. if the intruder can learn a message from a given protocol instrumentation, then this message could be infered from the type system. A significant result of this paper is the presentation of a new transformation that allows us to abstract a non-terminating type inference system into a terminating deductive proof system. We demonstrate how these results could be used to establish the security of cryptographic protocols from the secrecy standpoint. Finally, the usefulness and the efficiency of the whole approach is illustrated by proving the correctness of a new version of the Needham-Shoreder protocol with respect to the secrecy property.

The rest of this paper is organized as follows: In Section 2, we present an overview of related work in the area of modeling and verification of cryptographic protocols. In Section 3, we review the basic notation and terminology used for cryptographic protocols. In Section 4, we present the type system and the results related to its correctness and completeness. In Section 5, we present a new schema allowing to handle the termination problem within the type system. The efficiency on this schema is illustrated on a concret example. Finally, in Section 6, some concluding remarks are ultimately sketched as a conclusion.

2 Related Work

Formal modeling and verification of cryptographic protocols has received much attention in recent years. Several frameworks for the description and analysis of cryp-

tographic protocols have been proposed. A complete bibliography and a comparative study of these methods can be found in [12,13,14,31,34,40,50,49] [55,56]. These methods could be classified as follows: logical methods, general purpose formal methods and process algebra methods.

Typically, logical methods rest on the use of modal (epistemic, temporal and/or doxatic) logics. The logic is used to specify the protocol (idealization) as well as the security properties. In 1989, Burrows, Abadi and Needham devised BAN, a modal logic of belief for the specification and verification of cryptographic protocols [10,11]. BAN is the most known and famous logic dedicated to cryptographic protocols. Since then, plenty of derived logics have been advanced [3,23,27,28,39]. In 1990, Bieber [4] developed CKT5, a modal logic of knowledge that has been revised and extended by Carlsen in [13] and Snekkenes in [54]. Concurrently, many other logics attempted to combine several aspects of modal logic such as belief, knowledge and trust [26,46,47,58]. These methods have been successfully used to detect many flaws in cryptographic protocols and they are very expressive while specifying security properties. Nevertheless, they are not very suitable for specifying the protocols themselves. In fact, the protocols are often translated into a set of logical formulas. The translation process, often referred to as idealization, is error-prone since it aims to translate an operational description into a logical one. Furthermore, the idealization is not systematic. Moreover, most of the proposed logics, while proved sound with respect to some semantics are generally incomplete. In addition, the verification of the protocol is always manual and semi-formal.

Another trend in formal cryptographic development is to make use of some well-known general purpose formal methods. Representative specification languages that have been used include LOTOS [9,60,61,62], B, VDM [5,6,59], HOL [54], Ina Jo [32,33], Z [8,53] and Coq [7]. Although these formal methods are now firmly established and known to be of great use in specification and verification, it remains that these methods are not dedicated to cryptographic protocols. In addition, these methods need much expert assistance during the verification process. In fact, they rely on manual or interactive theorem proving techniques.

Recently, the use of process algebra for cryptographic protocol specification and verification has been explored. In 1995, Gavin Lowe [35,36,37,38] was the first to use CSP [29] and model-checking techniques for cryptographic protocol analysis. The protocol is specified as a set of communicating sequential processes that are running in parallel and interacting with their surroundings. The verification is performed by extracting a model (usually a finite state transition system) from the specification and checking that model against a logical specification (a formula over a modal temporal logic) or a behavioral specification (a process term). A similar approach was developed by Bill Roscoe, Paul Gardiner, Dave Jackson and Janson Hulance in [24,25,48] and Steve Schneider in [51]. In [44,45,52] Mitchell et al. analyzed cryptographic protocols using the general-purpose state enumeration tool Murϕ [15]. Abadi and Gordon [1,2] advanced Spi, a calculus for cryptographic protocols. Spi is built on top of the π-calculus [41,42,43], a mobile process algebra. It has been devised for the description and analysis of security protocols. Many other approachs

[30,50,57] are based on the Dolev-Yao model [22] that consider protocols from an attacker's standpoint. The process algebra-based methods have been successfully used to detect several flaws in well-known cryptographic protocols. The approach seems to be very promising and useful. However, it is well known that the underlying verification techniques, mainly those based on model-checking, are generally problematic in the presence of processes that exhibit infinite behaviors. Accordingly, the infinite aspects of cryptographic protocols are usually not supported in the verification process. Notice also that the specification of security properties in terms of process agents or modal formulae is neither straightforward nor systematic (except in the case of Spi).

3 Basics

In this section, we introduce the basic notations that will be used throughout this paper. This protocol notation, which we refer to as the *standard notation*, is based on a fairly standard informal notation used by the security protocol community. The statement $A \rightarrow B : m$ denotes the transmission of a message m from the principal A to the principal B. A message is composed of one or more primitive words. A message m encrypted with key k is written $\{m\}_k$ and forms a word by itself. Concatenated messages are separated by commas. Message contents (words) have the following naming conventions: Encryption keys and nonces are respectively written k and N. Principals are written A, B, S and I, where A and B stand for principals who wish to communicate, S for a trusted server and I for a potential intruder. Subscripts will be used to denote an association to a principal; thus, for example N_a is a nonce that belongs to A and k_{as} is a shared key between A and S. Here is the BNF syntax of messages:

m	$::=$	A	Principal Identifier
	$\mid$	N_a	Nonce
	$\mid$	k	Key
	$\mid$	n	Numeral
	$\mid$	X	Message Variable
	$\mid$	$\{m\}_k$	Encrypted Message
	$\mid$	m, m'	Message Concatenation
	$\mid$	$m\ op\ m'$	Arithmetic Operation

where $op \in \{+, -, *, /\}$.

A message m is said to be atomic if it is a principal identity, a nonce, a key, or a numeral. As an example, we show in Table 1 the Needham Schroeder protocol with symmetric keys. This protocol achieves both authentication and key distribution between two principals A and B by the means of a trusted server S.

1.	A	$\rightarrow$	S	:	A, B, N_a
2.	S	$\rightarrow$	A	:	$\{N_a, B, k_{ab}, \{k_{ab}, A\}_{k_{bs}}\}_{k_{as}}$
3.	A	$\rightarrow$	B	:	$\{k_{ab}, A\}_{k_{bs}}$
4.	B	$\rightarrow$	A	:	$\{N_b\}_{k_{ab}}$
5.	A	$\rightarrow$	B	:	$\{N_b - 1\}_{k_{ab}}$

Table 1
The Needham Schroeder Protocol with Conventional Keys

Here N_a and N_b are nonces, *i.e.*random numbers generated respectively by A and B especially for this protocol run. S is a server and k_{as} and k_{bs} are keys that A and B initially share with S. The key k_{ab} is a fresh key dynamically generated by the server S, for use by the principals A and B. The description of the protocol can be read as follows: First, A initiates the protocol and claims to the server S its desire to authenticate and communicate with B by sending the message A, B, N_a. Second, the server S generates a fresh key k_{ab} and sends it back to A wrapped in the message $\{N_a, B, k_{ab}, \{k_{ab}, A\}_{k_{bs}}\}_{k_{as}}$. Third, the principal A decrypts the previous message, extracts the component $\{k_{ab}, A\}_{k_{bs}}$ and sends it back to B. Fourth, receiving the previous message, the principal B decrypts it and extracts the key k_{ab} that it uses to encrypt a fresh nonce N_b to get $\{N_b\}_{k_{ab}}$, which it sends to A. Finally, A extracts the nonce N_b, decrements it by 1, and sends the result back to B encrypted by the key k_{ab}, thus proving A's identity to B.

4 Type System

The main idea underlying the type system is to come up with a universal model that captures in a finite way all the intruder abilities. In addition, the intruder is made powerful, smart and lazy. By powerful, we mean that the intruder has a complete control over the network. By smart, we mean that the intruder knows which messages he needs to break a security property. By lazy, we mean that the intruder will act in a goal-directed manner. He will neither generate nor send any message that is not useful to achieve an attack. Furthermore, the intruder is able to:

- overhear every message and decrypt cyphertext when he has the appropriate key,
- intercept and store messages,
- generate new messages using his initial knowledge and the intercepted messages.

The intruder abilities are formally captured as a type system in which protocols are represented as static environments, messages as programs and attack scenarios as types. The type system captures an insecurity property. When a message is provable under the typing rules this means that the intruder could get that message by combining its traditional abilities together with protocol instrumentations.

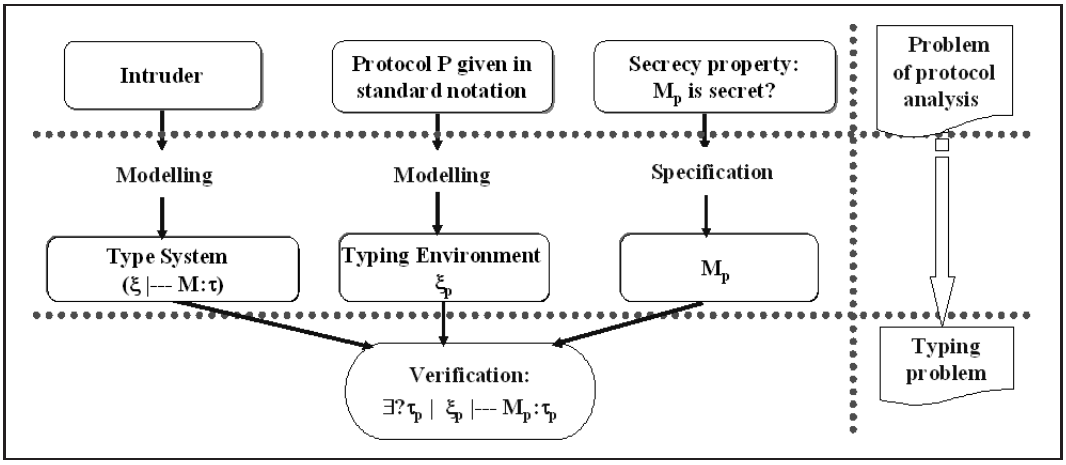


Fig. 1. Approach Outline

As shown by Fig. 1, the type system aims to transform the problem of the cryptographic protocol analysis to a typing problem.

4.1 Protocols, Roles and Environments

In this sequel, we explain how to generate a role-based specification from protocol descriptions written in the standard notation. These specifications will be used to provide static environments for the type system. These specifications were first introduced in [16,17,18] for the purpose of verification of authentication protocols. First, we review the definition of a role and a generalized role, in order to introduce the notion of a static environment. We will illustrate this concept on the Needham Schroeder Protocol.

4.1.1 Role

A role is a protocol abstraction where the emphasis is put on a particular principal. More precisely, roles are extracted from the protocol according to the following steps:

- For each principal, we extract all the steps in the protocol in which it participates. Furthermore, we add the same session identifier i to all those steps. Finally, to each fresh message we add i as exponent to indicate the fact that those messages change their values from one execution of the protocol to another.
- We introduce explicitly an intruder I to indicate that all the messages sent or received by the principal of the role have to be sent or received to/from the intruder.

For instance, in the case of the Needham Schroeder protocol of Table 1, three roles can be extracted, $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$. They respectively correspond to principals playing the roles A, B and S. Note that throughout the rest of this paper the terms "role" and "principal" will be used interchangeably, whenever the meaning is unambiguous. For example, in the following, "S" should be understood as "a principal playing the role of S", and similarly for A and B.

The first step of the role generation gives the following:

$$\begin{aligned}
\mathcal{A} = \; & \langle i.1, A \to S : A, B, N_a^i \rangle \\
& \langle i.2, S \to A : \{N_a^i, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}} \rangle \\
& \langle i.3, A \to B : \{k_{ab}^i, A\}_{k_{bs}} \rangle \\
& \langle i.4, B \to A : \{N_b^i\}_{k_{ab}^i} \rangle \\
& \langle i.5, A \to B : \{N_b^i - 1\}_{k_{ab}^i} \rangle
\end{aligned}$$

$$\begin{aligned}
\mathcal{S} = \; & \langle i.1, A \to S : A, B, N_a^i \rangle \\
& \langle i.2, S \to A : \{N_a^i, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}} \rangle
\end{aligned}$$

$$\begin{aligned}
\mathcal{B} = \; & \langle i.3, A \to B : \{k_{ab}^i, A\}_{k_{bs}} \rangle \\
& \langle i.4, B \to A : \{N_b^i\}_{k_{ab}^i} \rangle \\
& \langle i.5, A \to B : \{N_b^i - 1\}_{k_{ab}^i} \rangle
\end{aligned}$$

After the second step, the following results are obtained. Here $I(S)$ indicates the intruder acting in the role of S, etc.

$$\begin{aligned}
\mathcal{A} = \; & \langle i.1, A \to I(S) : A, B, N_a^i \rangle \\
& \langle i.2, I(S) \to A : \{N_a^i, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}} \rangle \\
& \langle i.3, A \to I(B) : \{k_{ab}^i, A\}_{k_{bs}} \rangle \\
& \langle i.4, I(B) \to A : \{N_b^i\}_{k_{ab}^i} \rangle \\
& \langle i.5, A \to I(B) : \{N_b^i - 1\}_{k_{ab}^i} \rangle
\end{aligned}$$

$$\begin{aligned}
\mathcal{S} = \; & \langle i.1, I(A) \to S : A, B, N_a^i \rangle \\
& \langle i.2, S \to I(A) : \{N_a^i, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}} \rangle
\end{aligned}$$

$$\begin{aligned}
\mathcal{B} = \; & \langle i.3, I(A) \to B : \{k_{ab}^i, A\}_{k_{bs}} \rangle \\
& \langle i.4, B \to I(A) : \{N_b^i\}_{k_{ab}^i} \rangle \\
& \langle i.5, I(A) \to B : \{N_b^i - 1\}_{k_{ab}^i} \rangle
\end{aligned}$$

4.1.2 Generalized Roles

From the roles, we extract what we call generalized roles. A generalized role is an abstraction of a role where some messages are replaced by variables. Intuitively, we replace a message or a component of message by a variable, if the receiver of this

message could not do any verification on it. Generalized roles give a precise idea about the principals' behaviours during the protocol execution.

For instance, the principal, playing the role A, participates in the protocol through these 5 steps:

(i) The principal A initiates the protocol by sending the message A, B, N_a^i to the principal S.

(ii) The principal A receives the message

$$\{N_a^i, B, k_{ab}, \{k_{ab}, A\}_{k_{bs}}\}_{k_{as}}$$

from S. Actually, A receives a message of the form $\{N_a^i, B, X, Y\}_{k_{as}}$ where X and Y are message variables. Here, we replace the message k_{ab} by X and $\{k_{ab}, A\}_{k_{bs}}$ by Y. The rationale underlying such substitutions is that the principal A is waiting for a message (k_{ab}, A) encrypted under a key (k_{bs}) that is not known to A. The principal A has no means to check either that the received message at the second step is actually encrypted under k_{bs}, or that its content is "k_{ab}, A". Hence, A can accept any message, say Y, at this second step. A similar argument applies to the substitution of k_{ab} by X.

(iii) The principal A reacts according to the protocol by sending the message Y to B.

(iv) The principal A receives the message $\{N_b^i\}_{k_{ab}}$ and is supposed to react by sending $\{N_b^i - 1\}_{k_{ab}}$. However, notice that: (i) The principal A has no prior knowledge about the value of the expected nonce, so A has no means to verify the value of the received message at that step; (ii) The freshness of N_b^i is a local property. In other words, the freshness of the nonce cannot be attested by any principal other than B. So, A cannot require the freshness of the received message; (iii) For the sake of generality, we assume that the exchanged messages are not typed. Consequently, A is unable to verify the type of the message received at the fourth step of the protocol. Owing to these three facts, A can verify neither the value, the freshness nor the type of the message received at the fourth step of the protocol. Accordingly, N_b^i will be replaced by Z.

(v) Finally, A sends the message $\{Z - 1\}_X$ to B.

In summary, the role associated with A, for a session i, can be written as the following sequence of actions:

$$\begin{aligned}
\mathcal{A} = \; & \langle i.1, A \to I(S) : A, B, N_a^i \rangle \\
& \langle i.2, I(S) \to A : \{N_a^i, B, X, Y\}_{k_{as}} \rangle \\
& \langle i.3, A \to I(B) : Y \rangle \\
& \langle i.4, I(B) \to A : \{Z\}_X \rangle \\
& \langle i.5, A \to I(B) : \{Z - 1\}_X \rangle
\end{aligned}$$

More formally, a generalized role could be introduced as follows. Let $\mathcal{A}$, be the

role of A, $\mathcal{K}_A$ the initial knowledge of the principal A (for the sake of simplicity, we suppose that $\mathcal{K}_A$ contains only atomic messages) and $M^{\mathcal{A}}$ the set of messages exchanged in the role $\mathcal{A}$. We define the component of $M^{\mathcal{A}}$ that is unknown to A, denoted by $M^{\mathcal{A}}_{\rightarrow_{\mathcal{K}_A}}$, using the following rewriting rules:

$$\begin{array}{lll} M \cup \{m\} & \rightarrow_{\mathcal{K} \cup \{m\}} & M \\ M \cup \{\{m\}_k\} & \rightarrow_{\mathcal{K} \cup \{k\}} & M \cup \{m\} \\ M \cup \{\{m\}_k\} \cup \{k\} & \rightarrow_{\mathcal{K}} & M \cup \{m\} \cup \{k\} \\ M \cup \{m.m'\} & \rightarrow_{\mathcal{K}} & M \cup \{m\} \cup \{m'\} \\ M \cup \{m \ op \ m'\} & \rightarrow_{\mathcal{K}} & M \cup \{m\} \cup \{m'\} \end{array}$$

Now let f be any injective function from $M^{\mathcal{A}}_{\rightarrow_{\mathcal{K}_A}}$ to the set of message variables. The generalized role of $\mathcal{A}$, is obtained from $\mathcal{A}$ by replacing each exchanged message m by $G_f(m)$, where $G_f(m)$ is defined as follows:

$$\begin{array}{ll} G_f(m) & = f(m) \text{ If } f(m) \text{ is defined} \\ G_f(m) & = m \text{ If } m \text{ is atomic and} \\ & \quad f(m) \text{ is not defined} \\ G_f(m.m') & = G_f(m).G_f(m') \\ G_f(m \ op \ m') & = G_f(m) \ op \ G_f(m') \\ G_f(\{m\}_k) & = \{G_f(m)\}_{G_f(k)} \text{If} f(\{m\}_k) \\ & \quad \text{is not defined} \end{array}$$

Notice that for the sake of simplicity, we deal only with symmetric key protocols and the results could be easily extended to others kind of protocols.

For the role $\mathcal{S}$, we have:

$$M^{\mathcal{S}} = \{A.B.N_a^i, \{N_a^i, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}}\}$$

Let $\mathcal{K}_S = \{A, B, S, k_{as}, k_{bs}, k_{ab}^i\}$ be the initial knowledge of S, therefore:

$$M^{\mathcal{S}}_{\rightarrow_{\mathcal{K}_S}} = \{N_a^i\}$$

Now, let f be the injective function from $M^{\mathcal{S}}_{\rightarrow_{\mathcal{K}_S}}$ to the set of message variables defined as follows:

$$f(N_a^i) = X$$

Since:

$$G_f(A.B.N_a^i) = A.B.X$$

and

$$G_f(\{N_a^i, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}}) = \{X, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}}$$

We conclude that the generalized role of $\mathcal{S}$ is:

$$\begin{aligned}\mathcal{S} = \langle i.1, I(A) \rightarrow S : A, B, X\rangle \\ \langle i.2, S \rightarrow I(A) : \{X, B, k^i_{ab}, \\ \{k^i_{ab}, A\}_{k_{bs}}\}_{k_{as}}\rangle\end{aligned}$$

Similarly, the generalized roles associated with $\mathcal{B}$ could be written as follows:

$$\begin{aligned}\mathcal{B} = \langle i.3, I(A) \rightarrow B : \{X, A\}_{k_{bs}}\rangle \\ \langle i.4, B \rightarrow I(A) : \{N^i_b\}_X\rangle \\ \langle i.5, I(A) \rightarrow B : \{N^i_b - 1\}_X\rangle\end{aligned}$$

4.1.3 Environment

We are interested in role instrumentation by an active, malicious intruder. The idea is to supply a given role with the messages it expects, in order to get the role to produce a particular message that will be used in an attack. Hence, we are interested in the output communication steps of the roles. We split each role into many sub-roles, each of which ends with an output communication step. In the case of the principal A, we have three outputs. Accordingly, we will split $\mathcal{A}$ into three sub-roles $\mathcal{A}^1$, $\mathcal{A}^2$ and $\mathcal{A}^3$. Thus, we get the following:

$$\mathcal{A}^1 = \langle i.1, A \rightarrow I(S) : A, B, N^i_a\rangle$$

$$\begin{aligned}\mathcal{A}^2 = \langle i.1, A \rightarrow I(S) : A, B, N^i_a\rangle \\ \langle i.2, I(S) \rightarrow A : \{N^i_a, B, X, Y\}_{k_{as}}\rangle \\ \langle i.3, A \rightarrow I(B) : Y\rangle\end{aligned}$$

$$\begin{aligned}\mathcal{A}^3 = \langle i.1, A \rightarrow I(S) : A, B, N^i_a\rangle \\ \langle i.2, I(S) \rightarrow A : \{N^i_a, B, X, Y\}_{k_{as}}\rangle \\ \langle i.3, A \rightarrow I(B) : Y\rangle \\ \langle i.4, I(B) \rightarrow A : \{Z\}_X\rangle \\ \langle i.5, A \rightarrow I(B) : \{Z - 1\}_X\rangle\end{aligned}$$

In the case of the principals S and B, we have only one output communication step. Consequently, we will have $\mathcal{S}^1$ and $\mathcal{B}^1$ defined as follows:

$$\mathcal{S}^1 = \langle i.1, I(A) \rightarrow S : A, B, X\rangle$$
$$\langle i.2, S \rightarrow I(A) : \{X, B, k^i_{ab},$$
$$\{k^i_{ab}, A\}_{k_{bs}}\}_{k_{as}}\rangle$$

$$\mathcal{B}^1 = \langle i.3, I(A) \rightarrow B : \{X, A\}_{k_{bs}}\rangle$$
$$\langle i.4, B \rightarrow I(A) : \{N^i_b\}_X\rangle$$

Each sub-role ends with an output communication step. The last step of the protocol, i.e:

$$\langle i.5, I(A) \rightarrow B : \{N^i_b - 1\}_X\rangle$$

is then removed from $\mathcal{B}^1$. This is motivated by the fact that this step is not needed to get the message $\{N^i_b\}_X$ from B.

Now we are ready to introduce the notion of static environment. This is simply a mapping that takes a multiset containing the last messages output by a given sub-role to the sequence of communication steps representing this sub-role. We will see later why the notion of a multiset is useful. An empty sequence of communication steps corresponds to the multiset containing the intruder's initial knowledge. A multiset will be written between the following curly brackets $\{\!|\ldots|\!\}$ (to avoid ambiguity, two concatenated messages m_1, m_2 used in a set or a multiset will be denoted by $m_1.m_2$). If M is a multiset, then we use $\overline{M}$ to denote the set containing all the messages in M. An empty sequence of communication steps is denoted by ϵ. As an illustration, the static environment associated with the Needham Schroeder protocol of Table 1 is shown in Table 2. The multiset K_I contains messages initially known by the intruder. Notice that these roles and the corresponding environments can be generated automatically from the protocol descriptions written in the standard notation. The underlying algorithms are detailed in [16,17,18].

In the remainder of this paper, we use the following notation: If $\mathcal{A}$ is a generalized role, then $\mathcal{A}^*$ will denote the following set:

$$\mathcal{A}^* = \bigcup_{\sigma\in\Gamma}\{\mathcal{A}\sigma\}$$

where Γ is the set of all possible substitutions. Also, if $\mathcal{A}_1 \ldots \mathcal{A}_{n_1}$, $\mathcal{B}_1 \ldots \mathcal{B}_{n_2}$, ... are the generalized roles associated with a given protocol P, then $\mathcal{P}^*$ is the following set:

$$\mathcal{P}^* = \mathcal{A}^*_1 \cup \ldots \cup \mathcal{A}^*_{n_1} \cup \mathcal{B}^*_1 \cup \ldots \cup \mathcal{B}^*_{n_2} \ldots$$

If ξ is a static environment, we will use the notation ξ^* to denote the following set:

$$\xi^* = \bigcup_{\sigma\in\Gamma}\bigcup_{\mathcal{A}\in Rng(\xi)}\{\mathcal{A}\sigma\}$$

$$
\begin{array}{rcl}
\xi = \{ & & \\
K_I & \mapsto & \epsilon \\
\{\!|A.B.N_a^i|\!\} & \mapsto & \langle i.1, A \to I(S) : A, B, N_a^i \rangle \\
\{\!|Y|\!\} & \mapsto & \begin{array}{l} \langle i.1, A \to I(S) : A, B, N_a^i \rangle \\ \langle i.2, I(S) \to A : \{N_a^i, B, X, Y\}_{k_{as}} \rangle \\ \langle i.3, A \to I(B) : Y \rangle \end{array} \\
\{\!|\{Z-1\}_X|\!\} & \mapsto & \begin{array}{l} \langle i.1, A \to I(S) : A, B, N_a^i \rangle \\ \langle i.2, I(S) \to A : \{N_a^i, B, X, Y\}_{k_{as}} \rangle \\ \langle i.3, A \to I(B) : Y \rangle \\ \langle i.4, I(B) \to A : \{Z\}_X \rangle \\ \langle i.5, A \to I(B) : \{Z-1\}_X \rangle \end{array} \\
\{\!|\{X, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}}|\!\} & \mapsto & \begin{array}{l} \langle i.1, I(A) \to S : A, B, X \rangle \\ \langle i.2, S \to I(A) : \{X, B, k_{ab}^i, \{k_{ab}^i, A\}_{k_{bs}}\}_{k_{as}} \rangle \end{array} \\
\{\!|\{N_b^i\}_X|\!\} & \mapsto & \begin{array}{l} \langle i.3, I(A) \to B : \{X, A\}_{k_{bs}} \rangle \\ \langle i.4, B \to I(A) : \{N_b^i\}_X \rangle \end{array} \\
\} & &
\end{array}
$$

Table 2
The Needham Schroeder Environment

4.2 Scenario

Our typing rules will make use of the notion of a scenario, which we formalize here. Given a static environment ξ, we associate a scenario τ (a type) to a message m (a program). This means that the message m could be inferred by the intruder by following the scenario τ in which the intruder uses some of its usual abilities and some instrumentations of the protocol ξ. Intuitively, a scenario is a valid trace denoting a protocol execution, *i.e.*, a sequence of communication steps in which principals act according to the protocol specification and all the messages sent by the intruder are derivable from its knowledge.

A message m is said to be derivable by the intruder from its knowledge, say M (a message set), whenever m can be obtained from M by using the intruder's conventional abilities (encryption, decryption, composition, and decomposition). More formally, we say that m is derivable from M if $m \in M_{\Downarrow}$, where $M_{\Downarrow}$ is the closure of

the message set M under the conventional operations. Here is the formal definition:

Definition 4.1 [Closure] Let M be a message set. We denote by $M_{\Downarrow}$ the closure of the set M under the conventional intruder computations. The closure $M_{\Downarrow}$ is defined as the least set satisfying the following conditions:

(i) $M \subseteq M_{\Downarrow}$,
(ii) $(k \in M_{\Downarrow}) \wedge (\{m\}_k \in M_{\Downarrow}) \Rightarrow m \in M_{\Downarrow}$,
(iii) $(m \in M_{\Downarrow}) \wedge (m' \in M_{\Downarrow}) \Rightarrow m \; op \; m' \in M_{\Downarrow}$
(iv) $(m \; op \; m' \in M_{\Downarrow}) \wedge (m' \in M_{\Downarrow}) \Rightarrow m \in M_{\Downarrow}$,
(v) $(m \; op \; m' \in M_{\Downarrow}) \wedge (m \in M_{\Downarrow}) \Rightarrow m' \in M_{\Downarrow}$,
(vi) $(k \in M_{\Downarrow}) \wedge (m \in M_{\Downarrow}) \Rightarrow \{m\}_k \in M_{\Downarrow}$,
(vii) $(m \in M_{\Downarrow}) \wedge (m' \in M_{\Downarrow}) \Rightarrow m.m' \in M_{\Downarrow}$,
$(viii)$ $(m.m' \in M_{\Downarrow}) \Rightarrow \{m, m'\} \subseteq M_{\Downarrow}$.
where $op \in \{+, -, *, /\}$,

Now, we need to define the algebra $\mathcal{T}$ of types. The structure of $\mathcal{T}$ is given by the following BNF grammar:

$$\begin{aligned} \tau ::= \; & \epsilon \\ | \; & \langle i.j, I(A) \rightarrow B : m \rangle.\tau \\ | \; & \langle i.j, A \rightarrow I(B) : m \rangle.\tau \end{aligned}$$

where the type $\langle i.j, I(A) \rightarrow B : m \rangle.\tau$ stands for the sequence in which the intruder starts by an usurpation of the principal A and then sends the message m to the principal B. The rest of the sequence is τ. The type $\langle i.j, A \rightarrow I(B) : m \rangle.\tau$ stands for the sequence whose first step is a combination of an usurpation of the principal B by the intruder together with an interception of the message m. The rest of the sequence is τ. The term $i.j$ denotes the identifier of the session i and the number of the communication step j according to the protocol description.

Throughout the rest of this paper, we denote by $\mathcal{C}_{i,j}$ the set of all possible communication steps labeled with $i.j$. We write $c_{i,j}$ to denote a communication step in $\mathcal{C}_{i,j}$ and $\mathsf{label}(c)$ to denote the label of the communication step c. Also, we use the notation $\mathcal{C}$, $\mathcal{C}_i$, $\mathcal{C}^A$ and $\mathcal{C}_i^A$ to denotes the following sets:

$$\mathcal{C} = \bigcup_{i,j} \mathcal{C}_{i,j}$$

$$\mathcal{C}_i = \bigcup_{j} \mathcal{C}_{i,j}$$

$$\mathcal{C}^A = \{c \in \mathcal{C} \mid c = \langle i.j, I(B) \to A : m\rangle \text{ or } c = \langle i.j, A \to I(B) : m\rangle\}$$

$$\mathcal{C}_i^A = \mathcal{C}^A \cap \mathcal{C}_i$$

$$\mathcal{C}_{ij}^A = \mathcal{C}^A \cap \mathcal{C}_{ij}$$

To lighten the notation, we will omit the sequence concatenation operator "." and the trivial type ϵ when the meaning is unambiguous.

Although the intuitive meaning of a valid trace seems to be clear at first glance, we define the notion more precisely here. In what follows, we will introduce a set of conditions on traces of communication steps that characterizes valid scenarios.

The most important condition that has to be satisfied by a valid trace is a dataflow constraint. Such a constraint is inspired by dataflow analysis which stipulates that a variable should be defined before its use. Analogously, in a cryptographic protocol setting, we require that a message should be defined (*i.e.*in the initial knowledge, previously received, or deducible by a closure) before its use (sending over the network). For that, we need to introduce the following Def/Use definitions:

Definition 4.2 [Def/Use] Let τ be a type and k_i a message set denoting the intruder initial knowledge. We define the operations Def and Use as follows:

- The set $\mathsf{Def}_{k_i}(\tau)$ stands for the messages received by the intruder in the communication steps reported in the type τ together with its initial knowledge. It is inductively defined as follows:

$$\begin{aligned}
\mathsf{Def}_{k_i}(\epsilon) &= k_i \\
\mathsf{Def}_{k_i}(\langle i.j, A \to I(B) : m\rangle.\tau) &= \{m\} \cup \mathsf{Def}_{k_i}(\tau) \\
\mathsf{Def}_{k_i}(\langle i.j, I(A) \to B : m\rangle.\tau) &= \mathsf{Def}_{k_i}(\tau)
\end{aligned}$$

- The set $\mathsf{Use}(\tau)$ stands for the set of messages sent by the intruder in the communication steps reported in the type τ. This set is inductively defined as follows:

$$\mathsf{Use}(\epsilon) = \emptyset$$
$$\mathsf{Use}(\langle i.j, A \to I(B) : m\rangle.\tau) = \mathsf{Use}(\tau)$$
$$\mathsf{Use}(\langle i.j, I(A) \to B : m\rangle.\tau) = \{m\} \cup \mathsf{Use}(\tau)$$

Throughout all this paper and when there is no ambiguity, $\mathsf{Def}_{k_i}(\tau)$ will be merely written $\mathsf{Def}(\tau)$. We need to introduce the multiset extension $\underline{\mathsf{Use}}$ of Use and $\underline{\mathsf{Def}}$ of Def as follows:

$$\underline{\mathsf{Use}}(\epsilon) = \emptyset$$
$$\underline{\mathsf{Use}}(\langle i.j, A \to I(B) : m\rangle.\tau) = \underline{\mathsf{Use}}(\tau)$$
$$\underline{\mathsf{Use}}(\langle i.j, I(A) \to B : m\rangle.\tau) = \{\!|m|\!\} \cup \underline{\mathsf{Use}}(\tau)$$

$$\underline{\mathsf{Def}}_{k_i}(\epsilon) = k_i$$
$$\underline{\mathsf{Def}}_{k_i}(\langle i.j, A \to I(B) : m\rangle.\tau) = \{\!|m|\!\} \cup \underline{\mathsf{Def}}_{k_i}(\tau)$$
$$\underline{\mathsf{Def}}_{k_i}(\langle i.j, I(A) \to B : m\rangle.\tau) = \underline{\mathsf{Def}}_{k_i}(\tau)$$

Now, we are able to formalize the condition that captures the dataflow constraint. This condition states that at each communication step c in a trace τ, the messages used by the intruder have to be previously defined, *i.e.*they could be derived form its initial knowledge and the messages received before this step.

Definition 4.3 [Well-Defined Type] A type τ is well-defined if, for each communication step c, with type τ_1 and type τ_2 such that $\tau = \tau_1.c.\tau_2$, we have:

$$\mathsf{Use}(c) \subseteq \mathsf{Def}(\tau_1)_{\Downarrow}.$$

The second important condition that has to be satisfied by a valid trace is an ordering relation between the communication steps used in a valid trace. This condition states that the communication steps used in a valid trace have to follow the order established by the protocol specification. For example, in the same session i, the communication step $c_{i,2}$ cannot occur before the communication step $c_{i,1}$. Furthermore, a valid trace should not contain duplicated communication steps. To formalize such an ordering condition, we need to introduce the definition of the so-called flattening operation. This operation takes a type to the set of communication steps that form it. The flattening is defined as follows:

Definition 4.4 [Flattening] Let τ be a type. We define the flattening of τ, written $\overline{\tau}$, as follows:

$$\overline{\epsilon} = \emptyset$$
$$\overline{c.\tau} = \{c\} \cup \overline{\tau}$$

where c is a communication step in $\mathcal{C}$.

Next we introduce a precedence order between communication steps in a given type.

Definition 4.5 [$\prec_\tau$] Let $\tau \in \mathcal{T}$, and c and c' be two communication steps such that $\{c, c'\} \subseteq \overline{\tau}$. We say that c precedes c' in τ, and we write $c \prec_\tau c'$, if there exist two types τ_1 and τ_2 in $\mathcal{T}$ such that $\tau = \tau_1.\tau_2$, $\{c'\} \cap \overline{\tau_1} = \emptyset$ and $\{c\} \cap \overline{\tau_2} = \emptyset$.

Types that respect the ordering relation will be called well-ordered types. They are formally defined as follows:

Definition 4.6 [Well-Ordered Type] We say that a type $\tau \in \mathcal{T}$ is a well-ordered type if it satisfies the following two conditions:

(i) For each session identifier i and for each natural number j, there exists at most one communication step in τ labeled with $i.j$. In other words, if there exist three types τ_1, τ_2 and τ_3 and two communication steps c_1 and c_2 such that $\tau = \tau_1.c_1.\tau_2.c_2.\tau_3$, then $\mathsf{label}(c_1) \neq \mathsf{label}(c_2)$,

(ii) If there exist two communication steps $c_{i,j}$ and $c_{i,j'}$, and a role identifier A such that $\{c_{i,j}, c_{i,j'}\} \subseteq \overline{\tau} \cap \mathcal{C}_i^A$ and $j < j'$, then $c_{i,j} \prec_\tau c_{i,j'}$.

It is straightforward to see that the two previous constraints (well-defined and well-ordered) do not guarantee that the honest principals will act according to the protocol specification. For that reason, we need another condition to complete the characterization of a valid trace. Given some trace, to verify whether principals have acted with respect to the protocol specification or not, we need to compare the behaviors of these principals with their generalized roles. Each session in the scenario has to correspond to an instantiation of some generalized role from a static environment ξ. This is captured by the notion of ξ-induced types introduced hereafter.

Let S be a set of communication steps. We say that S is ξ-induced if all its communication steps that belong to the same session are generated from a certain role in the protocol specification ξ. More formally:

Definition 4.7 [ξ-Induced Set] Let ξ be a static environment denoting a protocol specification and S a set of communication steps. We say that S is ξ-induced if for each session identifier i and role identifier A such that $S \cap \mathcal{C}_i^A \neq \emptyset$, there exists τ in ξ^* such that $\overline{\tau} = S \cap \mathcal{C}_i^A$.

Notice that we say that a type τ is ξ-induced if $\overline{\tau}$ is a ξ-induced set.

To sum up, we have now 3 constraints: First, the $\mathsf{Def/Use}$ condition ensures that all the messages sent by the intruder can be derived from its initial knowledge and the received messages. Second, the well-ordered type condition ensures that the communication steps are well-ordered with respect to a given session. Third, the ξ-induced type condition ensures that the honest principals act according to the protocol specification. Now, we define a valid trace with respect to a static environment ξ (also called a ξ-scenario), as follows:

Definition 4.8 [ξ-Scenario] Let τ be a type and ξ a static environment denoting a

protocol specification. We say that τ is a ξ-scenario if:

(i) τ is a well-ordered type.

(ii) τ is a well-defined type.

(iii) τ is a ξ-induced type.

4.3 Typing Rules

In the sequel, we present a type system that propagates sequences of communication steps in order to construct the sequence needed by the intruder to compute a given message. The type system gives a type to a multiset of messages with respect to an environment generated according to the protocol specification. The notion of multiset is introduced to allow the extraction of a wide variety of protocol flaws. In particular, this notation allows us to extract flaws that contain two different types for the same message. To illustrate this, let us take a concrete example. Suppose that we have a static environment that contains the following association:

$$\{\!|m|\!\} \mapsto \begin{array}{l} \langle i.1, I(B) \rightarrow A : m_1 \rangle \\ \langle i.2, A \rightarrow I(B) : m_2 \rangle \\ \langle i.3, I(B) \rightarrow A : m_1 \rangle \\ \langle i.4, A \rightarrow I(B) : m_3 \rangle \\ \langle i.5, I(B) \rightarrow A : m_4 \rangle \\ \langle i.6, A \rightarrow I(B) : m \rangle \end{array}$$

The association above means that the intruder has to derive the messages m_1 and m_4 to get the message m. A type of the message m reflects the scenario that leads the intruder to the message m. Such a type must also contain the types of the messages m_1 and m_4. In the example above, what makes the use of the multiset notation more beneficial than the set notation is the multiple presence of the message m_1. Suppose that we use a simple set and we say that the intruder must infer a type for the set of messages $\{m_1, m_4\}$ to obtain the message m. In this case, the intruder needs types for the messages m_1 and m_4. Furthermore, the type of the message m_1 will be used twice since this message is needed in two places. Then, the scenario will have the following form:

$$\text{Type of message } m_1$$
$$\langle i.1, I(B) \to A : m_1 \rangle$$
$$\langle i.2, A \to I(B) : m_2 \rangle$$
$$\langle i.3, I(B) \to A : m_1 \rangle$$
$$\langle i.4, A \to I(B) : m_3 \rangle$$
$$\text{Type of message } m_4$$
$$\langle i.5, I(B) \to A : m_4 \rangle$$
$$\langle i.6, A \to I(B) : m \rangle$$

However, if we use the multiset, we can say that the intruder has to prove the following multiset of messages

$$\{\!|m_1, m_1, m_4|\!\}$$

and in this case we have to prove the message m_1 twice, as well as the message m_4. Suppose that there are many types that could be associated to the message m_1. Then, we can obtain a scenario in which we associate two different types to the message m_1, as shown below.

$$\text{Type 1 of message } m_1$$
$$\langle i.1, I(B) \to A : m_1 \rangle$$
$$\langle i.2, A \to I(B) : m_2 \rangle$$
$$\text{Type 2 of message } m_1$$
$$\langle i.3, I(B) \to A : m_1 \rangle$$
$$\langle i.4, A \to I(B) : m_3 \rangle$$
$$\text{Type of message } m_4$$
$$\langle i.5, I(B) \to A : m_4 \rangle$$
$$\langle i.6, A \to I(B) : m \rangle$$

Thus, with a multiset notation, we are no longer confined to have the same types at the two occurrences where types for m_1 are needed.

Our static semantics manipulates judgments of the form:

$$\xi \vdash M : \tau$$

meaning that the message multiset M has the type τ in the environment ξ. In other words, the intruder could get all the messages in M by executing the sequence τ of communication steps which may instrument the protocol ξ.

Before we introduce the typing rules, we need a way to merge the types of two

(**Triv**) $$\frac{\square}{\xi \ \vdash \ \{\!|\}\!\} : \epsilon}$$

(**Op1**) $$\frac{\xi \ \vdash \ \{\!|m|\}\!\} : \tau_1 \quad \xi \ \vdash \ \{\!|m'|\}\!\} : \tau_2 \quad op \in \{+, -, *, /\} \quad \tau_1 \sharp_\xi \tau_2}{\xi \ \vdash \ \{\!|m \ op \ m'|\}\!\} : \tau_1 \dagger \tau_2}$$

(**Op2**) $$\frac{\xi \ \vdash \ \{\!|m|\}\!\} : \tau_1 \quad \xi \ \vdash \ \{\!|m \ op \ m'|\}\!\} : \tau_2 \quad op \in \{+, -, *, /\} \quad \tau_1 \sharp_\xi \tau_2}{\xi \ \vdash \ \{\!|m'|\}\!\} : \tau_1 \dagger \tau_2}$$

(**Op3**) $$\frac{\xi \ \vdash \ \{\!|m'|\}\!\} : \tau_1 \quad \xi \ \vdash \ \{\!|m \ op \ m'|\}\!\} : \tau_2 \quad op \in \{+, -, *, /\} \quad \tau_1 \sharp_\xi \tau_2}{\xi \ \vdash \ \{\!|m|\}\!\} : \tau_1 \dagger \tau_2}$$

(**Comp**) $$\frac{\xi \ \vdash \ \{\!|m|\}\!\} : \tau_1 \quad \xi \ \vdash \ \{\!|m'|\}\!\} : \tau_2 \quad \tau_1 \sharp_\xi \tau_2}{\xi \ \vdash \ \{\!|m.m'|\}\!\} : \tau_1 \dagger \tau_2}$$

(**Dec1**) $$\frac{\xi \ \vdash \ \{\!|m.m'|\}\!\} : \tau}{\xi \ \vdash \ \{\!|m|\}\!\} : \tau}$$

(**Dec2**) $$\frac{\xi \ \vdash \ \{\!|m.m'|\}\!\} : \tau}{\xi \ \vdash \ \{\!|m'|\}\!\} : \tau}$$

(**Encr**) $$\frac{\xi \ \vdash \ \{\!|m|\}\!\} : \tau_1 \quad \xi \ \vdash \ \{\!|k|\}\!\} : \tau_2 \quad \tau_1 \sharp_\xi \tau_2}{\xi \ \vdash \ \{\!|\{m\}_k|\}\!\} : \tau_1 \dagger \tau_2}$$

(**Decr**) $$\frac{\xi \ \vdash \ \{\!|\{m\}_k|\}\!\} : \tau_1 \quad \xi \ \vdash \ \{\!|k|\}\!\} : \tau_2 \quad \tau_1 \sharp_\xi \tau_2}{\xi \ \vdash \ \{\!|m|\}\!\} : \tau_1 \dagger \tau_2}$$

(**Inst**) $$\frac{\xi \dagger [M \mapsto \tau_1] \ \vdash \ \underline{\mathsf{Use}}(\tau_1\sigma) : \tau_2 \quad \overline{M_1} \subseteq \overline{M\sigma} \quad \tau_1\sigma \sharp_\xi \tau_2}{\xi \dagger [M \mapsto \tau_1] \ \vdash \ M_1 : \tau_2 \dagger \tau_1\sigma}$$

(**Cup**) $$\frac{\xi \ \vdash \ M_1 : \tau_1 \quad \xi \ \vdash \ M_2 : \tau_2 \quad \tau_1 \sharp_\xi \tau_2}{\xi \ \vdash \ M_1 \cup M_2 : \tau_1 \dagger \tau_2}$$

(**Equiv**) $$\frac{\xi \ \vdash \ M : \tau_1 \quad \tau_1 \approx \tau_2}{\xi \ \vdash \ M : \tau_2}$$

Table 3
The Typing Rules

multisets of messages into one multiset, and to check for compatible types. Suppose for example that we want to merge the following two types:

$$\text{type 1} = \begin{cases} \dots \\ \langle i.1, I(A) \rightarrow B : m_1 \rangle \\ \dots \end{cases}$$

$$\text{type 2} = \begin{cases} \dots \\ \langle i.1, I(A) \rightarrow B : m_2 \rangle \\ \dots \end{cases}$$

Suppose also that m_1 and m_2 are different messages. It is straightforward that these two types cannot be merged since we have two different communication steps having the same label.

The compatibility notion is more complicated than to simply guarantee that there is no more than one communication step having the same label. In fact, we must also ensure that each principal will not play more than one role in the same session. Let us take a concrete example to explain this fact clearly. Suppose that we have the following protocol:

$$\langle i.1, A \rightarrow S : \{A, B, k\}_{k_{as}} \rangle$$
$$\langle i.2, S \rightarrow B : \{B, A, k\}_{k_{bs}} \rangle$$

Suppose also that we have the following two scenarios:

$$\text{type 1} = \begin{cases} \dots \\ \langle i.1, C \rightarrow I(S) : \{C, D, k\}_{k_{cs}} \rangle \\ \dots \end{cases}$$

$$\text{type 2} = \begin{cases} \dots \\ \langle i.2, I(S) \rightarrow C : \{C, D, k\}_{k_{cs}} \rangle \\ \dots \end{cases}$$

Those two scenarios could not be combined. In fact, in the first scenario, the principal C plays the role of A in the session i. However, in the second scenario, the principal C plays the role of B in the session i. Then, if we put together these two scenarios, we will obtain a type that contains the same honest principal playing two different roles in the same session. To avoid this situation, we require that the set of communication steps used in the two merged scenarios form a ξ-induced set. Now, we give the following definition that stipulates whether two types are compatible or not.

Definition 4.9 [Compatible Set] Let S be a set of communication steps. We say that S is compatible with respect to an environment ξ, if the two following conditions hold:

(i) S is a ξ-induced set.

(ii) For each session label i and each step label j, we have $|\mathcal{C}_{ij} \cap S| \leq 1$.

Suppose that τ_1 and τ_2 are two types and ξ is an environment. We say that τ_1 and τ_2 are compatible in ξ, and we write $\tau_1 \sharp_\xi \tau_2$, if the set $\overline{\tau_1} \cup \overline{\tau_2}$ is compatible in ξ. Finally, if $\tau \sharp_\xi \tau$, we say that τ is a self-compatible type in ξ.

Now, if we have two compatible scenarios τ_1 and τ_2, then the type $\tau_1.\tau_2$ is well-defined and ξ-induced. Therefore, it remains only to check that the type $\tau_1.\tau_2$ forms a well-ordered type to guarantee that the two concatenated types form a scenario. For that reason, we need a definition of type merging that specifies how two types will be gathered in order to ensure that the merge of two compatible scenarios gives another scenario. Actually, the merge of two types τ_1 and τ_2 is a sort of type concatenation that eliminates the redundant steps in τ_2 with respect to the type τ_1.

Definition 4.10 [Type Merge] Let τ_1 and τ_2 be two types and c a communication step. The merge of τ_1 and τ_2, denoted by $\tau_1 \dagger \tau_2$, is defined as follows:

$$\tau \dagger \epsilon = \tau$$
$$\tau \dagger c.\tau' = \tau \dagger \tau' \text{ if } c \in \overline{\tau}$$
$$\tau \dagger c.\tau' = \tau.c \dagger \tau' \text{ if } c \notin \overline{\tau}$$

Actually, there are many ways to merge subproofs in order to construct the global attack scenario. Suppose that τ_1 is a type of m_1 and τ_2 a type of m_2. Suppose also that τ_1 and τ_2 are two compatible types. Therefore, we can easily prove that $\tau_1 \dagger \tau_2$ and $\tau_2 \dagger \tau_1$ are two scenarios for the message $m_1.m_2$. Obviously, $\tau_1 \dagger \tau_2$ could be different from $\tau_2 \dagger \tau_1$. Consequently, in order to allow the inference of many equivalent communication sequences, we need to introduce the following equivalence on types.

Definition 4.11 [Equivalence] Let τ and τ' be two well-ordered and well-formed types. We say that τ is equivalent to τ' and we write $\tau \approx \tau'$ if $\overline{\tau} = \overline{\tau'}$.

Let $\mathcal{T}_1$ be a subset of $\mathcal{T}$. In the sequel, we use the notation $\mathcal{T}_1^{\approx}$ to denote the following set:

$$\mathcal{T}_1^{\approx} = \{\tau \in \mathcal{T} \mid \exists\ \tau_1 \in \mathcal{T}_1 \text{ and } \tau \approx \tau_1\}$$

The typing rules are given in Table 3. The intuitive ideas underlying each rule are as follows:

Triv: According to this rule, the empty communication step sequence, ϵ, is the type of the empty multiset, $\{\!|\}$.

Op1: If τ_1 is a type for the multiset $\{\!|m|\!\}$, τ_2 is a type for the multiset $\{\!|m'|\!\}$ and τ_1 and τ_2 are two compatible types, then $\tau_1 \dagger \tau_2$ is a type for $\{\!|m\ op\ m'|\!\}$, where

$op \in \{+, -, *, /\}$. Since many cryptographic protocols use arithmetic operations, like

$$\langle i.1, A \to B : \{N_b + 1\}_{k_{ab}} \rangle,$$

we have introduced the set of operators $\{+, -, *, /\}$ to deal with a large class of protocols. The rules **Op1**, **Op2** and **Op3** show how to cope with messages containing an operator in $\{+, -, *, /\}$. There is no evaluation of messages that contain operators. For example, the message $m * 0$ is treated differently from the message 0.

Comp: This is a composition rule and it illustrates how to give a type to a message of the form $m.m'$ knowing the types of the messages m and m'. Accordingly, if τ_1 is a type for the multiset $\{\!|m|\!\}$, τ_2 is a type for the multiset $\{\!|m'|\!\}$ and τ_1, τ_2 are two compatible types, then the merged type $\tau_1 \dagger \tau_2$ is a type for $\{\!|m.m'|\!\}$. Similarly, the rules **Dec1** and **Dec2** are decomposition rules. They stipulate that a type τ for the message multiset $\{\!|m.m'|\!\}$ is also a type for the one-message multisets $\{\!|m|\!\}$ and $\{\!|m'|\!\}$.

Encr: This rule stipulates that if τ_1 is a type of the message m and τ_2 is the type of the key k such that τ_1 and τ_2 are two compatible types, then $\tau_1 \dagger \tau_2$ is a type of the message $\{m\}_k$. The decryption rule **Decr** shows that the type $\tau_1 \dagger \tau_2$ could be a type of the message m, if τ_1 is a type for a message $\{m\}_k$, τ_2 is a type for the key k and τ_1 and τ_2 are two compatible types.

Inst: The rules **Op1**, **Op2**, **Op3**, **Comp**, **Dec1**, **Dec2**, **Encr** and **Decr** represent the intruder's usual abilities. However, the rule instantiation, **Inst**, gives a supplementary ability to the intruder based on the instrumentation of the protocol itself. The rule **Inst** is the most important rule in the type system. In fact, all the usual rules can merge only the types that are basically generated by the rule **Inst**. The intuition underlying the instantiation rule can be explained by the following example. Suppose that we have a static environment ξ that contains the following association:

$$\{\!|\{X\}_{k_{as}}|\!\} \mapsto \begin{array}{l} \langle i.1, A \to I(B) : A \rangle \\ \langle i.2, I(B) \to A : X \rangle \\ \langle i.3, A \to I(B) : \{X\}_{k_{as}} \rangle \end{array}$$

This association means that the intruder is able to generate any instantiation of

$$\{\!|\{X\}_{k_{as}}|\!\}$$

i.e.:

$$\{X\}_{k_{as}}\sigma,$$

if it is able to generate all messages used in the type appearing on the right instantiated by the same substitution σ. Now, since $\{\!|X|\!\}$ is the multiset of messages used in the sequence of communication steps of this association, then the intruder is able to generate $\{X\}_{k_{as}}\sigma$, if it can generate $X\sigma$, where σ is a substitution.

Cup: This rule shows that if τ_1 is a type for a multiset M_1 and τ_2 is a type for a multiset M_2 such that τ_1 and τ_2 are two compatible types, then $\tau_1 \dagger \tau_2$ is a type for the union of the two multisets, *i.e.*$M_1 \cup M_2$.

Equiv: Since the merge operator gives us only one possibility to merge two scenarios, the other possibilities can be specified as an equivalence between scenarios. So, if τ_1 is a type for a multiset M and τ_2 is a type equivalent to τ_1, *i.e.*τ_2 is a scenario that contains the same communication steps used in τ_1 but ordered in a different way, then τ_2 is also a type for the multiset M.

4.4 Correctness Result

The correctness theorem states that if a multiset of messages M has the type τ under some static environment ξ (protocol specification), then the type τ is a legitimate sequence that could be used by the intruder to derive all the messages in M.

Theorem 4.12 (Correctness Theorem) *Let M be a multiset of messages and τ is a type. If $\xi \vdash M : \tau$, then τ is a ξ-scenario and for all $m \in M$, we have $m \in \mathsf{Def}(\tau)_{\Downarrow}$.*

4.5 Completeness Result

The intent hereafter is to prove the completeness of the type system.

4.5.1 Definition of Completeness

The soundness property has been introduced as follows: if $\xi \vdash M : \tau$, then τ is a ξ-scenario and $\overline{M} \subseteq \mathsf{Def}(\tau)_{\Downarrow}$. Informally, this definition means that τ is a ξ-scenario and all the messages in M are defined in τ modulo the usual compositions and decompositions introduced by the operator "$\Downarrow$". In other words, the intruder can generate all the messages in M and the proof is given by the scenario τ. Similarly, the completeness property could be defined as follows: If M is a multiset of messages and τ is a ξ-scenario such that $\overline{M} \subseteq \mathsf{Def}(\tau)_{\Downarrow}$, then $\xi \vdash M : \tau$.

Our intention hereafter is to refine the definition of completeness without affecting its global meaning. First, let us introduce the following notation:

Definition 4.13 [Minimal ξ-Scenario]

Let ξ be an environment associated with a given protocol, τ a ξ-scenario and M a multiset of messages. We say that τ is a minimal ξ-scenario for M if the following conditions hold:

- $\overline{M} \subseteq \mathsf{Def}(\tau)_{\Downarrow}$.
- For all ξ-scenarios τ' such that $\overline{M} \subseteq \mathsf{Def}(\tau')_{\Downarrow}$ and $\overline{\tau} \subseteq \overline{\tau'}$, we have: $\overline{\tau} = \overline{\tau'}$.

Intuitively a ξ-scenario τ is not minimal for a multiset of messages M if we can eliminate some communication steps from τ and the remaining type is also a ξ-scenario for M. For instance, let M be a multiset of messages and τ a ξ-scenario such that $\overline{M} \subseteq \mathsf{Def}(\tau)_{\Downarrow}$. By the definition of a ξ-scenario, if τ' is another ξ-scenario such that τ and τ' are compatible, then $\tau \dagger \tau'$ is also a ξ-scenario and $\overline{M} \subseteq \mathsf{Def}(\tau \dagger \tau')_{\Downarrow}$.

However, the type $\tau \dagger \tau'$ is generally a non minimal ξ-scenario. Let us consider a concrete example. Let P be the following protocol:

$$\langle \alpha.1\ A{\rightarrow}B\ : A\rangle$$

We can easily show that $\langle \alpha.1\ A{\rightarrow}I(B)\ : A\rangle$ is a ξ-scenario for the multiset $\{\!|A|\!\}$. We can also prove that:

$$\langle \alpha.1\ A{\rightarrow}I(B)\ : A\rangle$$
$$\langle \beta.1\ A{\rightarrow}I(B)\ : A\rangle$$

is a ξ-scenario for the multiset $\{\!|A|\!\}$. However, the second ξ-scenario is not minimal.

The definition of completeness is given with respect to a minimal ξ-scenario as follows:

Definition 4.14 [Definition of Completeness]

Let τ be a ξ-scenario and M a multiset of messages. The type system is complete if:

$$\tau \text{ is a minimal } \xi\text{-scenario associated to } M \Longrightarrow \xi \vdash M : \tau.$$

4.5.2 Completeness Theorem

The following theorem shows that the type system is complete with respect to the completeness definition.

Theorem 4.15 (Completeness) *Let ξ be an environment, M a multiset of messages and τ a ξ-scenario. If τ is a minimal ξ-scenario for M, then $\xi \vdash M : \tau$.*

5 Correctness of Cryptoprotocols

It is a known fact that proving the correctness of security protocols with respect to a given set of security properties is at least a difficult and a subtle task, if not impossible. For that reason, almost all security researchers try to simplify the problem before its resolution. These simplifications consist generally in studying only one security property on a restricted class of cryptographic protocols. However, defining a sub-class of protocols for which their correctness with respect to some security properties can be verified leads in almost all cases to the definition of a big number of restrictions reducing considerably their applications in the security field.

Though the type system 3 has been shown sound and complete, verifying whether a message m can be known by an intruder or not may lead to an infinite computation. As a result, the secrecy of the message m may not be assured. The termination problem is generally related to the deduction strategy (forward chaining, backward chaining, etc.) used to prove the secrecy of a message m. Unfortunately, backward chaining and forward chaining may lead to an infinite computation when they are used to prove the sequent $\xi \vdash m : \tau$ within the type system. For backward chaining some non-oriented rules such as:

$$\frac{\xi \vdash m.m' : \tau}{\xi \vdash m : \tau}$$

can lead to a termination problem. Similarly, the presence of some rules such as:

$$\frac{\xi \vdash m : \tau \quad \xi \vdash m' : \tau' \quad \tau \sharp \tau'}{\xi \vdash m.m' : \tau \dagger \tau'}$$

can also lead to an infinite computation for forward chaining.

This section introduces some techniques to resolve the termination problem within inference systems. In particular, these techniques will be combined to the abstract interpretation ones to resolve the termination problem within the type system. Besides, we introduce a modified version of the typing system making easier the resolution of the termination problem. This modification consists in restricting the type system so that it will be bound to deal with only a sub-class of security protocols.

5.1 Outline of the Methodology

The approach used in this paper to ensure the correctness of security protocols with respect to the secrecy property is essentially based on the resolution of the termination problem within the type system. Indeed, if there is no termination problem within the type system, then proving whether a protocol $P(M)$ keeps a message M secret or not will be always decidable thanks to completeness and correction theorems of the type system. The completeness theorem ensures that a message M is secret if it is not provable using the typing system. The correction theorem, on the other hand, ensures that a message M is not secret if it is provable using the type system.

The basic idea outlining the approach can be summarized as follows:

(i) Resolving the termination problem within a given inference system consists in generating another equivalent inference system (the two inference systems have the same induced theory) in which all the rules are ordered by a well-founded measure [3]. Given a non-terminating inference system, the idea consists in adding new rules that make redundant (can be removed from the inference system without affecting the underlying theory) the inference rules that are the origin of the termination problem. The new generated rules are extracted from the original inference system and they are by themselves redundant in this inference system.

(ii) The type system, in its actual form, is indeed not appropriate to be handled by the termination techniques. Therefore, an abstract function, to simplify the type system, is defined. This simplification consists essentially in eliminating from the type system all the information related to the construction of attack scenarios. Actually, the type system provides the ability to deduce whether a message M can be hold by a smart intruder. Furthermore, an attack scenario,

[3] A partial ordering $\succ$ over a set T is well-founded if any descending chain $t_1 \succ t_2 \succ \ldots$ of elements in T is finite.

exhibiting the sequence of actions executed by an intruder to get possession of the message M, is constructed. For instance, the rule

$$\frac{\Box}{\xi \vdash m : \tau}$$

states that an intruder can know the message m and the attack scenario is provided by τ. Since we deal with the correction of a protocol with respect to the secrecy property, then the information related to the construction of an attack scenario are no more important. In fact, proving that a message m can be hold by an intruder, without providing how an attack scenario can be constructed, is enough to conclude that the protocol does not keep the message m secret. In the other hand, if the analyzed protocol is secure, then there is, of course, no attack scenario to generate. In both cases, the information related to the construction of an attack scenario can be removed without affecting the conclusion. As a result, the rule

$$\frac{\Box}{\xi \vdash m : \tau}$$

can merely be replaced by

$$\frac{\Box}{\xi \vdash m}.$$

Another important abstraction consists in replacing the type system by another more general inference system (the theory of the original inference system is contained in the theory induced by the abstract inference system). For instance, if S_1 is an inference system containing the rule R_1, then the inference system $S_2 = (S_1 \backslash \{R_1\}) \cup \{R_2\}$, where R_2 is a rule obtained from R_1 by eliminating some premises, is an abstraction. As a result, if a message m cannot be proved using S_2, then it is also the case when S_1 is used. Besides, the inference system S_2 may resolve the termination problem of S_1. For instance, let S_1 and S_2 be the following inference systems:

$$S_1 = \{\frac{\Box}{a}, \frac{x, f(f(x))}{f(x)}\}$$

$$\begin{aligned} S_2 &= (S_1 \backslash \{\frac{x, f(f(x))}{f(x)}\}) \cup \{\frac{x}{f(x)}\} \\ &= \{\frac{\Box}{a}, \frac{x}{f(x)}\} \end{aligned}$$

Proving $f(b)$ within S_1, using backward chaining, leads to an infinite computation. However, within S_2, there is no termination problem and it is clear that $f(b)$ cannot be proved.

Obviously, the soundness of the abstraction process is momentous. In fact, the soundness property ensures that if the abstract type system proves the secrecy of a message M within a protocol $P(M)$, then this is indeed correct, i.e. the original type system gives the same result.

5.2 Abstract Interpretation and Proof System Termination

Understanding the concept of backward and forward chaining, one generally chooses backward strategy to prove that a term t is a theorem of a given inference system S. In fact, unlike forward chaining, backward strategy involves in its proofs only the sub-goals contributing to the construction of the required proof. Nevertheless, similarly to forward chaining, backward strategy may lead to an infinite computation without even solving the problem.

Many efforts have been investigated, last years, in the resolution of the termination problem within inference systems. Though the termination problem is known to be in general not decidable, this problem can be resolved for many classes of inference systems. Proving that there is no termination problem within an inference system, when using the backward chaining strategy, consists generally in finding a well-founded ordering relation making the conclusion of each rule less than its smallest associated premise. Therefore, given a non-terminating inference system S_1, finding another inference system S_2 equivalent to S_1 so that all the rules of S_2 are ordered by a well-founded relation implies the resolution of the termination problem of S_1.

The intent of the sequel is twofold: First, a new approach to solve the termination problem within inference systems is introduced. Second, a synergy between abstract interpretation and termination techniques for the verification of cryptographic protocols with respect to the secrecy property is presented.

5.2.1 Inference System

The purpose hereafter is to address the termination problem at the level of the proof system.

Preliminaries

Given a set $\mathcal{F} = \bigcup_{n\geq 0} \mathcal{F}_n$ of function symbols called a signature and a set $\mathcal{X}$ of variable symbols, the set of terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$ over $\mathcal{F}$ and $\mathcal{X}$ is the smallest set containing $\mathcal{X}$ such that $f(t_1, \ldots, t_n)$ is in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ whenever f is in $\mathcal{F}_n$ and $\{t_1, \ldots, t_n\} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$.

An inference rule R over a set of terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is defined by a set of premises $\{p_1, \ldots, p_n\} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$ and a conclusion c in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and written as follows:

$$\frac{p_1, \ldots, p_n}{c}$$

Besides, an inference rule with an empty set of premises is called an axiom and written as follows:

$$\frac{\square}{c}$$

An inference system S over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is a set of inference rules over $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Let t be a term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We say that t is a theorem in S and we write $S \vdash t$, if the sequent $S \vdash t$ can be proved using the rules given in Table 4. Intuitively, $S \vdash t$ if there exist an inference rule

$$R = \frac{p_1, \ldots, p_n}{c}$$

in S and a substitution σ such that $c\sigma = t$ and $S \vdash p_i$ for all $i \in \{1, \ldots, n\}$. The pair (R, σ) is used as a label for the rule

$$\frac{S \vdash p_1\sigma, \ \ldots, \ S \vdash p_n\sigma}{S \vdash c\sigma}$$

to keep a trace of how the proof of $S \vdash t$ is constructed.

(R, σ)	$\frac{\Box}{S \vdash c\sigma}$	$(R : \frac{\Box}{c} \in S)$
(R, σ)	$\frac{S \vdash p_1\sigma, \ \ldots, \ S \vdash p_n\sigma}{S \vdash c\sigma}$	$(R : \frac{p_1, \ \ldots, \ p_n}{c} \in S)$

Table 4
The Inference System Rules

For example, let S be the following inference system:

$$S = \{R_1 : \frac{x, y}{f(x, y)}, R_2 : \frac{\Box}{a}, R_3 : \frac{\Box}{b}\}$$

Let $\sigma = [x \mapsto a, \ y \mapsto b]$ and $\sigma' = \emptyset$. The proof associated with $f(a, b)$ is:

$$(R_1, \sigma)\frac{(R_2, \sigma')\frac{\Box}{S \vdash a} \ (R_3, \sigma')\frac{\Box}{S \vdash b}}{S \vdash f(a, b)}$$

Notices that many proof trees can be associated with the same sequent $S \vdash t$. In the sequel, the set of proof trees associated with the sequent $S \vdash t$ will be denoted by $[\![S \vdash t]\!]$. If P is a proof tree, then we denote by $|P|$, the hight of this tree. Similarly, if R is a rule, then we denote by $|R|$, the number of the premisses used in this rule.

The following definition introduces an ordering relation between inference systems.

Definition 5.1 (Inference System Comparison) Let S_1 and S_2 be two inference systems. The ordering relations $\sqsubseteq$ and $\equiv$ are defined as follows:

(i) $S_1 \sqsubseteq S_2$ if $\forall\, t : \ S_1 \vdash t \Rightarrow S_2 \vdash t$.

(ii) $S_1 \equiv S_2$ if $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_1$.

An inference rule is said to be redundant in a given inference system, if it can be eliminated from this inference system without affecting its involved theory.

Definition 5.2 [Redundant rule] Let S be an inference system. A rule $R \in S$ is said to be redundant in S if:

$$S \equiv S \backslash \{R\}.$$

Inference System Equivalence

The following proposition gives some examples of trivially redundant rules.

Proposition 5.3 *Let S be an inference system and R a rule in S. If one of the following condition holds:*

(i) $R = \frac{p_1, \ldots, p_n}{c}$ *and* $c \in \{p_1, \ldots, p_n\}$.

(ii) $R = \frac{p_1, \ldots, p_n}{c}$ *and there exists a rule* $R' = \frac{p'_1, \ldots, p'_{n'}}{c'}$ *in S and a substitution σ such that:*

$$\begin{cases} R \neq R' \\ c'\sigma = c \\ \{p'_1\sigma, \ldots, p'_{n'}\sigma\} \subseteq \{p_1, \ldots, p_n\} \end{cases}$$

then rule R is redundant in S.

In what follows, we denote by $S_{\downarrow}$ the normal form[4] of the inference system S using the rewriting rules given in Table 5.

Delete:	$S \cup \{\frac{p_1, \ldots, p_n}{c}\}$	$\rightsquigarrow$	S	if $c \in \{p_1 \ldots, p_n\}$
Eliminate:	$S \cup \{\frac{p_1, \ldots, p_n}{c}, \frac{p'_1, \ldots, p'_n}{c'}\}$	$\rightsquigarrow$	$S \cup \{\frac{p'_1, \ldots, p'_n}{c'}\}$	if (*)

(*): $\exists\, \sigma |\ c = c'\sigma$ and $\{p'_1\sigma, \ldots, p'_n\sigma\} \subseteq \{p_1, \ldots, p_n\}$

Table 5
Redundant Rules Elimination

It is straightforward that $S_{\downarrow}$ exists for any finite system S, since the rewriting rules decrease the size of S. Furthermore, by the previous proposition, it is immediate that for any inference system S, we have $S \equiv S_{\downarrow}$.

In what follows, we show how we can introduce new rules in a given inference system without modifying its induced theory. The idea of adding new inference rules has been largely used by the rewriting system community to generate a well-ordered rewriting rule for a finite set of equations. The most famous application of this idea is the Knuth-Bendix completion algorithm [21]. For more details about rewriting systems, the reader can refer to [20].

The new rules to be introduced are generated by composing the existing rules according to the following definition of composition.

[4] A term t is said to be in a normal form in a given rewriting system, if it cannot be rewritten by any rule of this system.

Definition 5.4 [Rules Composition] Let $R = \frac{p_1, \ldots, p_n}{c}$ and $R_1, \ldots, R_n$ a sequence of n rules such that $R_i = \frac{p_{1i}, \ldots, p_{n_i i}}{c_i}$,$1 \leq i \leq n$. We define the composition of $R_1, \ldots, R_n$ with R, denoted by $\frac{R_1, \ldots, R_n}{R}$, as follows:

$$\frac{R_1, \ldots, R_n}{R} = \begin{cases} \{\frac{p_{11}\sigma, \ldots, p_{1n_1}\sigma, \ldots, p_{1n}\sigma, \ldots, p_{n_n n}\sigma}{c\sigma}\} & \text{if } \sigma \text{ exists.} \\ \emptyset & \text{otherwise} \end{cases}$$

where $\sigma = mgu(\{p_1 = c_1, \ldots, p_n = c_n\})$.

The $mgu(\{p_1 = c_1, \ldots, p_n = c_n\})$ is the most general unifier, i.e. the most general substitution that unify all the pairs $(p_1, c_1), \ldots, (p_n, c_n)$.

Example 5.5 Let S be the following inference system:

$$S = \{R_1 : \frac{ff(x_1), x_1}{f(x_1)}, R_2 : \frac{x_2}{fff(x_2)}, R_3 : \frac{\Box}{f(b)}.\}$$

Since $mgu(\{ff(x_1) = fff(x_2), x_1 = f(b)\}) = [x_1 \mapsto f(b)\ x_2 \mapsto b]$, therefore:

$$\frac{R_2, R_3}{R_1} = \frac{b}{ff(b)}.$$

The composition of rules will be extended to inference systems as follows: Let R be a rule such that $|R| = n$ and S an inference system. The composition of S with R, denoted by $\frac{S}{R}$, is defined as follows:

$$\frac{S}{R} = \bigcup_{(R_1, \ldots, R_n) \in S^n} \frac{R_1, \ldots, R_n}{R}.$$

Let S_1 and S_2 be two inference systems. The composition of S_1 with S_2, denoted by $\frac{S_1}{S_2}$, is defined as follows:

$$\frac{S_1}{S_2} = \bigcup_{R \in S_2} \frac{S_1}{R}.$$

The following proposition states that the addition to a given inference system of new rules induced by internal compositions does not change its underlying theory.

Proposition 5.6 *Let S_1, S_2 be two sets of rules. We have the following result:*

$$(S_1 \cup S_2) \equiv (S_1 \cup S_2 \cup \frac{S_1}{S_2}).$$

Obviously, introducing new rules in an inference system does not solve the termination problem. We need, indeed, to eliminate from this inference system the rules causing the termination problem. In this respect, the aim of introducing new rules is to make redundant the ones causing the termination problem.

The basic result of this section is given by the following theorem which aims to eliminate some non-trivial redundant rules from an inference system. As shown

later, the result of this theorem can be used to give a transformation schema allowing the resolution of the termination problem within some inference systems.

Theorem 5.7 *Let $S = S_1 \cup S_2$ be an inference system such that all the axioms of S are in S_2. If $(S_1 \cup S_2 \cup \frac{S_2}{S_1})_{\downarrow} = S_1 \cup S_2$, then $S \equiv S_2$.*

Handling Termination

In the sequel we introduce an efficient transformation schema which can lead to avoiding the termination problem within an inference system. The transformation schema is based on the previously established results related to an inference system.

Let S be an inference system and $\prec$ a partial order over the terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We define the inference system $S_{\prec}$ as follows:

$$\begin{aligned} S_{\prec} = & \{\frac{\Box}{c} \in S\} \\ & \cup \\ & \{\frac{p_1, \ldots, p_n}{c} \in S \mid \forall\, i,\ 1 \leq i \leq n :\ p_i \prec c\}. \end{aligned}$$

Also, we introduce the function $\phi_{\prec}(S)$ as follows:

$$\phi_{\prec}(S) = (S \cup \frac{S_{\prec}}{S \backslash S_{\prec}})_{\downarrow}.$$

The following corollary gives some important properties of the function ϕ.

Corollary 5.8 *Let S be an inference system and $\prec$ a partial ordering over the terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We have:*

$$\begin{cases} (i) \quad \phi_{\prec}(S) \equiv S. \\ (ii) \ \textit{if } \phi_{\prec}(S) = S \ \textit{ then } S \equiv S_{\prec}. \end{cases}$$

Let $\prec$ be a well-founded ordering. Thanks to the previous corollary, we can give a transformation schema that can solve the termination problem within an inference system. This transformation schema states that if S has a normal form using the rewriting rules given in Table 6, then this normal form is an inference system with no termination problem.

Simplify:	S	$\leadsto$ $S_{\prec}$	if $S = \phi_{\prec}(S)$
Compose:	S	$\leadsto$ $\phi_{\prec}(S)$	

Table 6
The Termination Schema

5.2.2 Abstract Type System

The intent hereafter is to abstract the type system so that it can be easily analyzed using the termination techniques. This abstraction consists in eliminating all the information used to construct a scenario whenever an attack is found. More formally, the function that allows to abstract a type system is denoted by Υ and defined as follows:

$$\Upsilon(\emptyset) = \emptyset$$

$$\Upsilon(\{\frac{\Box}{c}\} \cup S') = \{\frac{\Box}{\Upsilon_p(c)}\} \cup \Upsilon(S')$$

$$\Upsilon(\{\frac{p_1, \ldots, p_n}{c}\} \cup S') = \{\frac{\Upsilon_p(p_1), \ldots, \Upsilon_p(p_n)}{\Upsilon_p(c)}\} \cup \Upsilon(S')$$

where Υ_p is a function abstracting a premise or a conclusion of a type system rule as follows:

$$\Upsilon_p(\tau_1 \sharp \tau_2) = \Box$$
$$\Upsilon_p(\tau_1 \approx \tau_2) = \Box$$
$$\Upsilon_p(M_1 \subseteq M_2) = M_1 \subseteq M_2$$
$$\Upsilon_p(\xi \vdash M : \tau) = \Upsilon_e(\xi) \vdash M$$

The abstraction of a premise into "$\Box$" means its elimination from the abstracted rule. Finally, the function Υ_e is used to abstract an environment ξ into $\xi^\flat$ as follows:

$$\Upsilon_e(\emptyset) = \emptyset$$
$$\Upsilon_e(\{[M \mapsto \tau]\} \cup \xi) = \{[M \mapsto \underline{\mathsf{Use}}(\tau)]\} \cup \Upsilon_e(\xi)$$

The abstract type system is given in Table 7. The following theorem states that this abstraction is correct.

Theorem 5.9 (Abstraction Soundness) *Let ξ and $\xi^\flat$ be two environments such that $\xi^\flat = \Upsilon_e(\xi)$. Also, let M be a multiset of messages and τ a ξ-scenario:*

$$\textit{If } \xi \vdash M : \tau \textit{ then } \xi^\flat \vdash^\flat M$$

Though the abstract type system is simpler than the concrete one, it is not yet suitable to be handled by the termination. For that reason, a new version of the abstract type system is introduced as follows: Let $T_{\xi^\flat}$ be the abstract type system associated with $\xi^\flat$. We denote by $S_{\xi^\flat}$, the smallest inference system satisfying the conditions given in Table 8.

Notice that the rules of the simplified abstract type system can be modified when necessary to fit the requirement of the analyzed protocol. For instance, the symmet-

$(\mathbf{Triv})^\flat$	$\dfrac{\square}{\xi^\flat \vdash^\flat \{\!\|\}}$
$(\mathbf{Op1})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|m\|\!\}\quad \xi^\flat \vdash^\flat \{\!\|m'\|\!\}}{\xi^\flat \vdash^\flat \{\!\|m\ op\ m'\|\!\}}$
$(\mathbf{Op2})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|m\|\!\}\quad \xi^\flat \vdash^\flat \{\!\|m\ op\ m'\|\!\}}{\xi^\flat \vdash^\flat \{\!\|m'\|\!\}}$
$(\mathbf{Op3})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|m'\|\!\}\quad \xi^\flat \vdash^\flat \{\!\|m\ op\ m'\|\!\}}{\xi^\flat \vdash^\flat \{\!\|m\|\!\}}$
$(\mathbf{Comp})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|m\|\!\}\quad \xi^\flat \vdash^\flat \{\!\|m'\|\!\}}{\xi^\flat \vdash^\flat \{\!\|m.m'\|\!\}}$
$(\mathbf{Dec1})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|m.m'\|\!\}}{\xi^\flat \vdash^\flat \{\!\|m\|\!\}}$
$(\mathbf{Dec2})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|m.m'\|\!\}}{\xi^\flat \vdash^\flat \{\!\|m'\|\!\}}$
$(\mathbf{Encr})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|m\|\!\}\quad \xi^\flat \vdash^\flat \{\!\|k\|\!\}}{\xi^\flat \vdash^\flat \{\!\|\{m\}_k\|\!\}}$
$(\mathbf{Decr})^\flat$	$\dfrac{\xi^\flat \vdash^\flat \{\!\|\{m\}_k\|\!\}\quad \xi^\flat \vdash^\flat \{\!\|k\|\!\}}{\xi^\flat \vdash^\flat \{\!\|m\|\!\}}$
$(\mathbf{Inst})^\flat$	$\dfrac{\xi^\flat \dagger [M \mapsto M'] \vdash^\flat M'\sigma \quad \overline{M_1} \subseteq \overline{M'\sigma}}{\xi^\flat \dagger [M \mapsto M'] \vdash^\flat M_1}$
$(\mathbf{Cup})^\flat$	$\dfrac{\xi^\flat \vdash^\flat M_1\quad \xi^\flat \vdash^\flat M_2}{\xi^\flat \vdash^\flat M_1 \cup M_2}$

Table 7
The Abstract Type System

ric key encryption and decryption rules given in the type system can be replaced by public key encryption and decryption rules if the analyzed protocol uses public keys. Besides, the rules dealing with operators can be eliminated if the protocol does not use arithmetic operations ($m_1\ op\ m_2$, where $op \in \{+, -, *, /\}$).

The following proposition proves that the system $S_{\xi^\flat}$ is equivalent to $T_{\xi^\flat}$.

Proposition 5.10 *Let ξ be an environment associated with a given protocol and $\xi^\flat$*

$$\frac{m_1 \quad m_2}{\{m_1\}_{m_2}} \in S_{\xi^\flat} \quad \frac{\{m_1\}_{m_2} \quad m_2}{m_1} \in S_{\xi^\flat}$$

$$\begin{cases} M \mapsto \{\!|m'_1, \ldots, m'_{n'}|\!\} \in \xi^\flat \\ \text{and} \\ m \in M \end{cases} \Rightarrow \frac{m'_1 \ \ldots \ m'_{n'}}{m} \in S_{\xi^\flat}$$

$$\frac{m_1 \quad m_2}{m_1, m_2} \in S_{\xi^\flat} \quad \frac{m_1, m_2}{m_1} \in S_{\xi^\flat} \quad \frac{m_1, m_2}{m_2} \in S_{\xi^\flat}$$

$$\frac{m_1 \quad m_2}{m_1 \ op \ m_2} \in S_{\xi^\flat} \quad \frac{m_1 \ op \ m_2 \quad m_1}{m_2} \in S_{\xi^\flat} \quad \frac{m_1 \ op \ m_2 \quad m_2}{m_1} \in S_{\xi^\flat}$$

Table 8
The Simplified Abstract Type System

its abstracted version. We have:

$$\xi^\flat \vdash^\flat \{\!|m|\!\} \textit{ if and only if } S_{\xi^\flat} \vdash m.$$

5.2.3 Case Study

The aims hereafter is to use the abstract interpretation and the termination results to prove the correction of some security protocols with respect to the secrecy property.

Protocol

The analyzed protocol is inspired by the symmetric key Needham-Schroeder protocol (the first three steps of a corrected version of the Needham-Schroeder symmetric key protocol):

$$\langle \alpha.1 \ A \rightarrow S : \ A, B, N_a^\alpha \rangle$$
$$\langle \alpha.2 \ S \rightarrow A : \ \{N_a^\alpha, B, k_{ab}^\alpha, \{k_{ab}^\alpha, N_a^\alpha, A\}_{k_{bs}}\}_{k_{as}} \rangle$$
$$\langle \alpha.3 \ A \rightarrow B : \ \{k_{ab}^\alpha, N_a^\alpha, A\}_{k_{bs}} \rangle$$

This protocol aims to establish a new key between two principals A and B. This key will be used by A and B for their future communication and should be kept secret.

The generalized roles associated with this protocol are as follows:

$$\mathcal{A}^1 = \langle \alpha.1\ A \rightarrow I(S) : A, B, N_a^\alpha \rangle$$

$$\begin{aligned}\mathcal{A}^2 = \ & \langle \alpha.1\ A \rightarrow I(S) : A, B, N_a^\alpha \rangle \\ & \langle \alpha.2\ I(S) \rightarrow A :\ \{N_a^\alpha, B, X, Y\}_{k_{as}} \rangle \\ & \langle \alpha.3\ A \rightarrow I(B) :\ Y \rangle\end{aligned}$$

$$\begin{aligned}\mathcal{S} \ = \ & \langle \alpha.1\ I(A) \rightarrow S : A, B, X \rangle \\ & \langle \alpha.2\ S \rightarrow I(A) :\ \{X, B, k_{ab}^\alpha, \{k_{ab}^\alpha, X, A\}_{k_{bs}}\}_{k_{as}} \rangle\end{aligned}$$

The environment associated with this protocol is given in Table 9.

$$\begin{array}{l}\xi = \{ \\ \qquad \begin{array}{rcl} K_I & \mapsto & \epsilon \\ \{\!|A.B.N_a^\alpha|\!\} & \mapsto & \mathcal{A}^1 \\ \{\!|Y|\!\} & \mapsto & \mathcal{A}^2 \\ \{\!|\{X, B, k_{ab}^\alpha, \{k_{ab}^\alpha, X, A\}_{k_{bs}}\}_{k_{as}}|\!\} & \mapsto & \mathcal{S} \end{array} \\ \}\end{array}$$

Table 9
The Needham-Schroeder Protocol Environment

Protocol Abstract Type System

Since the protocol uses symmetric keys and since it does not use arithmetic operations, the simplified representation of the abstract type system will be as shown in Table 10. This type system is generated according to the rules presented in Table 8 and supposes that the intruder's initial knowledge contains A and k_{is}. This means that the intruder i knows all the other principal identities (A is a variable that can be replaced by any principal identifier) and it shares a key k_{is} with the server S.

The rules R_1 and R_2 reflect the intruder initial knowledge. They are indeed extracted from the following mapping:

$R_1 : \frac{\square}{A}$ $R_2 : \frac{\square}{k_{is}}$ $R_3 : \frac{\square}{A, B, N_a^\alpha}$	$R_7 : \frac{x, y}{x}$ $R_8 : \frac{x, y}{y}$
$R_4 : \frac{x \quad y}{x, y}$ $R_5 : \frac{x \quad y}{\{x\}_y}$	$R_9 : \frac{\{x\}_y \quad y}{x}$
$R_6 : \frac{A, B, x}{\{x, B, k_{ab}^\alpha, \{k_{ab}^\alpha, x, A\}_{k_{bs}}\}_{k_{as}}}$	$R_{10} : \frac{\{N_a^\alpha, B, x, y\}_{k_{as}}}{y}$

Table 10
The Needham-Schroeder Abstract Type System

$$K_I \mapsto \epsilon.$$

The rule R_3 is extracted from the following mapping:

$$\{\!|A.B.N_a^\alpha|\!\} \mapsto \mathcal{A}^1.$$

The rule R_6 is extracted from the following mapping:

$$\{\!|\{X, B, k_{ab}^\alpha, \{k_{ab}^\alpha, X, A\}_{k_{bs}}\}_{k_{as}}|\!\} \mapsto \mathcal{S}.$$

The rule R_{10} is extracted from the following mapping:

$$\{\!|Y|\!\} \mapsto \mathcal{A}^2.$$

Finally, the rules R_4, R_5, R_7, R_8 and R_9 reflect the usual intruder abilities.

Notice that all the rules on the left hand side of Table 10 belong to $S_\prec$ (S is the inference system given in Table 10 and $\prec$ is defined such that $t_1 \prec t_2$ if $|t_1| < |t_2|$). However, all the rules in the right hand side belong to $S \backslash S_\prec$.

Handling Termination

To solve the termination problem within the inference system given in Table 10, we use the transformation schema given in Table 6 and the ordering relation $\prec$ ($t_1 \prec t_2$ if $|t_1| < |t_2|$) as a well-founded measure. Notice also that for the sake of simplicity, we keep the standard notation used within cryptographic protocols in the inference rules when using the transformation schema. For instance, the messages A and B denote principal identity variables. In other words, these messages have to be treated as variables but their substitutions are bound to principal identities. More precisely, a message A can be replaced by the term $Agt(x)$ which means that A is a variable denoting a principal identity. Also, we use N_a^α to denote the term $N(\alpha, A)$ i.e. $N(\alpha, Agt(x))$, k_{ab} to denote the term $k(A, B)$ and k_{ab}^α to denote $k(\alpha, A, B)$. Finally, i and s are two constants, where i denotes the intruder identity and s is

the server identity. To simplify the result of the transformation schema, we suppose that the message $m, m', m'' = (m, m'), m''$.

i	$S^i_\prec$	$S^i \backslash S^i_\prec$	Inference
0	$R_1 : \frac{\square}{A}$ $R_2 : \frac{\square}{k_{is}}$ $R_3 : \frac{\square}{A, B, N_a^\alpha}$ $R_4 : \frac{x \quad y}{x, y}$ $R_5 : \frac{x \quad y}{\{x\}_y}$ $R_6 : \frac{A, B, x}{\{x, B, k_{ab}^\alpha, \{k_{ab}^\alpha, x, A\}_{k_{bs}}\}_{k_{as}}}$	$R_7 : \frac{x, y}{x}$ $R_8 : \frac{x, y}{y}$ $R_9 : \frac{\{x\}_y \quad y}{x}$ $R_{10} : \frac{\{N_a^\alpha, B, x, y\}_{k_{as}}}{y}$	
1	The rule of $S^0_\prec$ and $\frac{R_3}{R_8} : \frac{\square}{B, N_a^\alpha}$ $\frac{R_6}{R_{10}} : \frac{A, B, N_a^\alpha}{\{k_{ab}^\alpha, N_a^\alpha, A\}_{k_{bs}}}$ $\frac{R_6 \quad R_2}{R_9} : \frac{i, B, x}{x, B, k_{ib}^\alpha \{k_{ib}^\alpha, x, i\}_{k_{bs}}}$	The rule of $S^0 \backslash S^0_\prec$ and $\frac{R_5}{R_{10}} : \frac{N_a^\alpha, B, x, y \quad k_{as}}{y}$	Compose
2	$S^1_\prec$ $\frac{\frac{R_3}{R_8}}{R_8} : \frac{\square}{N_a^\alpha}$ $\frac{\frac{R_6 \quad R_2}{R_9}}{R_8} : \frac{i, B, x}{B, k_{ib}^\alpha, \{k_{ib}^\alpha, x, i\}_{k_{bs}}}$	$(S^1 \backslash S^1_\prec)$ $\frac{\frac{R_6 \quad R_2}{R_9}}{R_7} : \frac{i, B, x}{x}$	Compose

Table 11
Handling Termination: Part I

The application of the transformation schema is shown in Tables 11 and 12, Table 13 and Table 14. Notice that the rules $\frac{i \quad B, x}{k_{ib}^\alpha}$ and $\frac{i, B, x}{k_{ib}^\alpha}$ of Table 13 are made redundant by the rule $\frac{\square}{k_{ib}^\alpha}$.

3	$S^2_{\prec}$ $\dfrac{\dfrac{\dfrac{R_6 \quad R_2}{R_9}}{R_8}}{R_8} : \dfrac{i, B, x}{k^{\alpha}_{ib}, \{k^{\alpha}_{ib}, x, i\}_{k_{bs}}}$	$(S^2 \backslash S^2_{\prec})$ $\dfrac{R_4}{\dfrac{\dfrac{R_6 \quad R_2}{R_9}}{R_7}} : \dfrac{i \quad B, x}{x}$	Compose

Table 12
Handling Termination: Part I - cont'd

Protocol Correction

The result of the transformation schema shows that the abstract type system associated with the Needham-Schroeder protocol is equivalent to the terminating inference system given in Table 15.

Notice that all the rules in this inference system are ordered by $\prec$, i.e. the premises of each rule are smaller than the conclusion of the same rule with respect to the ordering relation $\prec$.

Since the termination problem is resolved, proving whether a message m can be inferred using the terminating inference system given in Table 15 is decidable. For instance, let a_1 and a_2 be two principal identities. It is clear, that the message $k^{\alpha}_{a_1 a_2}$ can not be inferred from the terminating inference system. Therefore, thanks to the soundness of the abstraction and the transformation schema, we deduce that the three communication steps inspired by the Needham-Schroeder symmetric key protocol keeps the new generated key secret. As a result, we conclude that the secrecy property is satisfied.

6 Conclusion

In this paper, we have combined different formal and elegant techniques to ensure the correctness of security protocols with respect to the secrecy property. Type system, abstract interpretation and proof system termination are the basic ones used to reach this goal. In fact, the approach is based on a sound and complete type system in which types are communication steps and typing constraints characterize all the messages available to the intruder. This reduces verification of authentication and secrecy properties to a typing problem in our type system. Furthermore, a transformation schema that can resolve the termination problem within inference systems. Also, we have introduced a sound abstraction for the type system. This abstraction aims to simplify the typing rules so that they can be easily handled by the transformation schema. Once the termination problem of the type system is resolved, the correctness of a security protocol with respect to the secrecy property can be easily analyzed. The efficiency of this approach has been illustrated on a concrete example.

i	$S^i_\prec$	$S^i \backslash S^i_\prec$	Inference
4	$S^3_\prec$ $\dfrac{\dfrac{\dfrac{\dfrac{R_6 \quad R_2}{R_9}}{R_8}}{R_8}}{R_8} : \dfrac{i, B, x}{\{k^\alpha_{ib}, x, i\}_{k_{bs}}}$	$(S^3 \backslash S^3_\prec)$ $\dfrac{\dfrac{\dfrac{\dfrac{R_6 \quad R_2}{R_9}}{R_8}}{R_8}}{R_7} : \dfrac{i, B, x}{k^\alpha_{ib}}$	Compose
5	$S^4_\prec$ $\dfrac{\dfrac{\dfrac{\dfrac{R_6 \quad R_2}{R_9}}{R_8}}{R_8} \quad R_2}{R_9} : \dfrac{i, B, x}{k^\alpha_{ii}, x, i}$	$(S^4 \backslash S^4_\prec)$ $\dfrac{\dfrac{\dfrac{\dfrac{R_4}{R_6 \quad R_2}}{R_9}}{R_8}}{R_8}}{R_7} : \dfrac{i \quad B, x}{k^\alpha_{ib}}$	Compose
6	$S^5_\prec$ $\dfrac{R_1 \quad \dfrac{R_3}{R_8}}{R_4} : \dfrac{\square}{k^\alpha_{ib}}$ $\dfrac{\dfrac{\dfrac{\dfrac{R_6 \quad R_2}{R_9}}{R_8}}{R_8}}{R_7}$	$(S^5 \backslash S^5_\prec) \backslash \{\dfrac{i \quad B, x}{k^\alpha_{ib}}, \ \dfrac{i, B, x}{k^\alpha_{ib}}\}$ $\dfrac{\dfrac{\dfrac{\dfrac{R_6 \quad R_2}{R_9}}{R_8}}{R_8} \quad R_2}{R_9} : \dfrac{i, B, x}{x, i}$	Compose
7	$S^6_\prec$	$(S^6 \backslash S^6_\prec)$ $\dfrac{\dfrac{\dfrac{\dfrac{R_4}{R_6 \quad R_2}}{R_9}}{R_8}}{R_8} \quad R_2}{R_9} : \dfrac{i \quad B, x}{x, i}$	Compose

Table 13
Handling Termination: Part II

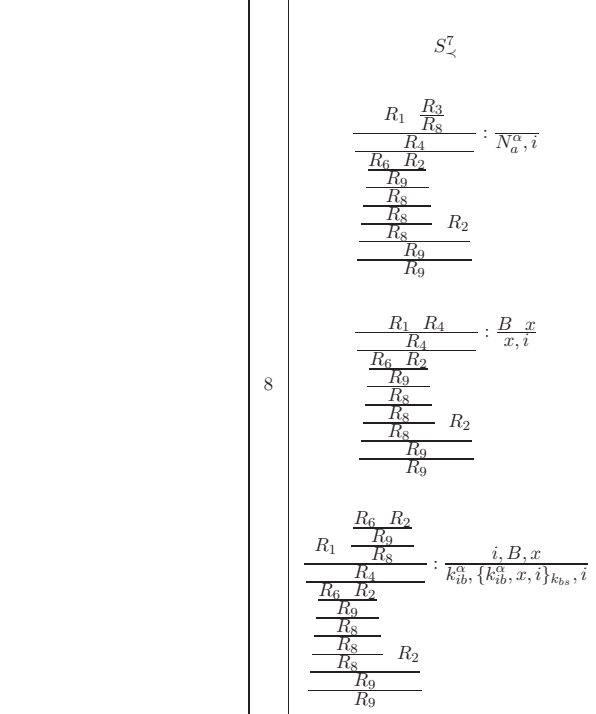
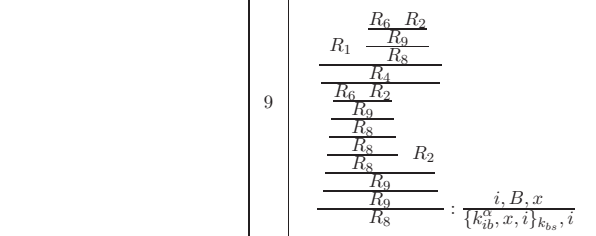
i	$S^i_\prec$	$S^i \backslash S^i_\prec$	Inference
8	$S^7_\prec$ 	$(S^7 \backslash S^7_\prec)$	Compose
9	$S^8_\prec$ 	$(S^8 \backslash S^8_\prec)$	Compose
10	$S^9_\prec$	$(S^9 \backslash S^9_\prec)$	Compose
11	$S^{10}_\prec$	$\emptyset$	Simplify

Table 14
Handling Termination: Part III

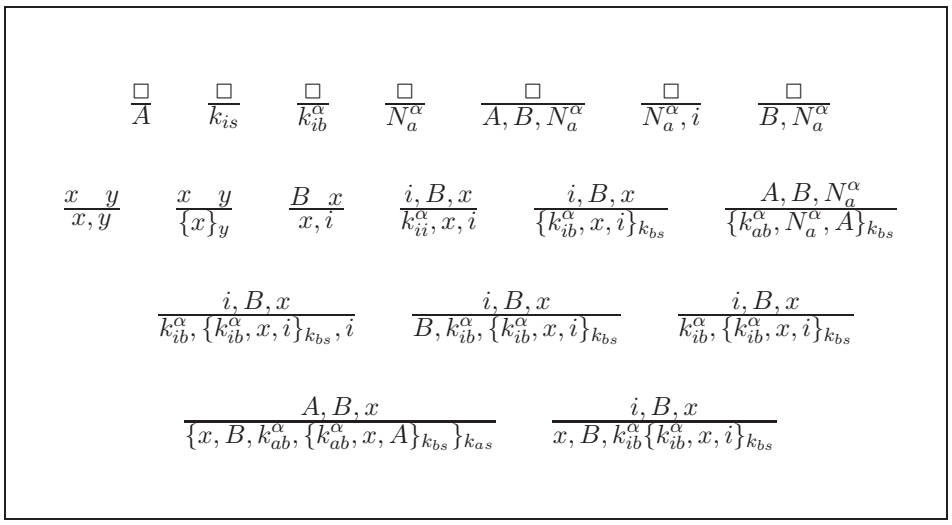


Table 15
Terminating Abstract System

References

[1] M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. Technical report, DEC/SRC, December 1996.

[2] M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*. ACM Press, April 1997.

[3] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication. In *Proceedings of the 10th Annual ACM Symposium On Principles of Distributed Computing*, pages 201–216, August 1991.

[4] P. Bieber. A Logic of Communication in a Hostile Environment. In *Proceedings of the Computer Security Foundations Workshop III*, pages 14–22. IEEE Computer Society Press, 1990.

[5] P. Bieber and N. B. Cuppens. Formal development of authentication protocols. In *In Proceedings of the BCS-FACS 6th Refinemment Workshop on software Engineering and its Applications*, January 1994.

[6] P. Bieber, N. B. Cuppens, T. Lehman, and E. V. Wickeren. Abstract machines for communication security. In *In Proceedings of the IEEE Computer Security Foundation Workshop VI*, Franconia,New-Hampshire, June 1993.

[7] D. Bolignano. An Approach to the Formal Verification of Cryptographic Protocols. In *Proceedings of the Third ACM Conference on Computer and Communications Security, CCS'96, New Delhi, India*, pages 106–118. ACM Press, 1996.

[8] C. Boyd. Security Architectures Using Formal Methods. *Journal on Selected Areas in Communications*, 11(5):694–701, June 1990.

[9] C. Boyd. A Framework for Authentication. In *Proceedings of the European Symposium on Research in Computer Security, ESORICS 92*, volume 648 of *Lecture Notes in Computer Science*, pages 273–292. Springer Verlag, November 1992.

[10] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proceedings of the Royal Society of London A Vol.426*, pages 233–271, 198.

[11] M. Burrows, M. Abadi, and R. Needham. Rejoinder to Nessett. *ACM Operating Systems Review*, 24(2):39–40, April 1990.

[12] L. Buttyan. Formal methods in the design of cryptographic protocols (state of the art). Technical Report No. SSC/1999/38, Swiss Federal Institute of Technology (EPFL), Lausanne, November 1999.

[13] U. Carlsen. *Formal Specification and Analysis of Cryptographic Protocols*. PhD thesis, Thèse d'Informatique soutenue à l'Université PARIS XI, October 1994.

[14] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature. Unpublished Article Available at `http://dcpu1.cs.york.ac.uk/~jeremy`, August 1996.

[15] D. L. Dill. The murphi verification system. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 390–393, New Brunswick, NJ, USA, July/August 1996. Springer Verlag.

[16] M. Debbabi, M. Mejri, N. Tawbi, and I. Yahmadi. A New Algorithm for Automatic Verification of Authentication Cryptographic Protocols. In *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols, DIMACS Center, Core Building, Rutgers University, New Jersy, USA*, Sep 1997.

[17] M. Debbabi, M. Mejri, N. Tawbi, and I. Yahmadi. Formal Automatic Verification of Authentication Cryptographic Protocols. In *Proceedings of the First IEEE International Conference on Formal Engineering Methods, Hiroshima, International Convention Center, Japan*. IEEE Press, November 1997.

[18] M. Debbabi, M. Mejri, N. Tawbi, and I. Yahmadi. From Protocol Specifications to Flaws and Attack Scenarios: An Automatic and Formal Algorithm. In *Proceedings of the Second International Workshop on Enterprise Security, Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts, USA*. IEEE Press, June 1997.

[19] M. Debbabi, M. Mejri N. Durgin, and J. Mitchell. Security by Typing. In *the International Journal on Software Tools For Technology Transfer (STTT); DOI 10.1007/s10009-002-0100-7.*, pages 1–24. Springer Verlag, December 2002.

[20] N. Dershowitz. Termination of Rewriting. *J. of Symbolic Computation*, 3(1&2):279–301, 1987.

[21] A. J. J. Dick. Eqution reasoning and the knuth-bandix algorithm-an informal introduction. Technical report, Imperial College, Reserch Report DOC 84/21, March 1984.

[22] D. Dolev and a. Yao. On the Security of Public Key Protocols. In *IEEE Transactions on Information Theory*, pages 198–208, March 1983.

[23] K. Gaarder and E. Snekkenes. Applying a formal analysis technique to the ccitt x.509 strong two-way authentification protocol. *Journal of cryptology, 3(2)*, pages 81–98, 1991.

[24] P. Gardiner, D. Jackson, J. Hulance, and B. Roscoe. Security Modelling in CSP and FDR: Deliverable Bundle 2. Technical report, Formal Systems (Europe) Ltd, April 1996.

[25] P. Gardiner, D. Jackson, and B. Roscoe. Security Modelling in CSP and FDR: Deliverable Bundle 3. Technical report, Formal Systems (Europe) Ltd, July 1996.

[26] J. I. Glasgow, G. H. MacEwen, and P. Panangaden. A Logic for Reasoning About Security. *ACM Transactions on Computer Systems*, 10(3):226–264, August 1992.

[27] V. D. Gligor and R. Kailar. On Belief Evolution in Authentication Protocols. In *Proceedings of the IEEE Computer Security Foundations Workshop IV, Franconia*, pages 103–116, June 1991.

[28] L. Gong, R. Needham, and R. Yahalom. Reasoning About Belief in Cryptographic Protocols. In Deborah Cooper and Teresa Lunt, editors, *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, May 1990.

[29] C. A. R. Hoare. *Communating Sequential Process*. Prentice Hall, 1985.

[30] C. Iliano, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Relating Strands and Multiset Rewriting for Security Protocol Analysis. In *Proceedings of the Thirteenth IEEE Computer Security Foundations Workshop- CSFW'00 (P. Syverson, editor)*, pages 35–51. IEEE Computer Society Press, Cambridge, UK, July 2000.

[31] R. Kemmerer, C. Meadows, and J. Millen. Three Systems for Cryptographic Protocol Analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[32] R. A. Kemmerer. Using Formal Verification Techniques to Analyse Encryption Protocols. In *Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy*, pages 134–139. IEEE Computer Society Press, 1987.

[33] R. A. Kemmerer. Analysing Encryption Protocols Using Formal Verification Techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, 1989.

[34] A. Liebl. Authentication in Distributed Systems: A Bibliography. *Operating Systems Review*, 27(4):122–136, October 1993.

[35] G. Lowe. An Attack on the Needham-Schroeder Public Key Authentication Protocol. *Information Processing Letters*, 56(3):131–136, November 1995.

[36] G. Lowe. Breaking and fixing the needham schroeder public-key protocol using fdr. In *Proceedings of TACAS*, volume 1055, pages 147–166. Springer Verlag, 1996.

[37] G. Lowe. Some new attacks upon security protocols. In *Proceedings of the Computer Security Foundations Workshop VIII*. IEEE Computer Society Pres, 1996.

[38] G. Lowe. Splice-as: A case study in using csp to detect errors in security protocols. Technical report, Programming Research Group, Oxford, 1996.

[39] W. Mao and C. Boyd. Towards the Formal Analysis of Security Protocols. In *Proceedings of the Computer Security Foundations Workshop VI*, pages 147–158. IEEE Computer Society Press, 1993.

[40] C. Meadows. Formal Verification of Cryptographic Protocols: A Survey. In *Proceedings of Asiacrypt 96*, 1996.

[41] R. Milner. The polyadic π-calculus: A tutorial. Technical report, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1991.

[42] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. Technical report, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1989.

[43] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[44] J. C. Mitchell, M. Mitchell, and U. Stern. Automated Analysis of Cryptographic Protocols Using Murphi. In *Proceedings 1997 IEEE Symposium on Security and Privacy*, pages 141–153. IEEE Computer Society, May 1997.

[45] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-State Analysis of SSL 3.0. In *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols, September 3-5, 1997, DIMACS Center, CoRE Building, Rutgers University, New Jersey, USA*, pages 1–20, September 1997.

[46] L. E. Moser. A Logic of Knowledge and Belief about Computer Security. In J Thomas Haigh, editor, *Proceedings of the Computer Security Foundations Workshop III*, pages 57–63. IEEE, Computer Society Press of the IEEE, 1989.

[47] P. V. Rangan. An Axomiatic Basis of Trust in Distributed System. In *In Proceedings of the 1988 Symposium on Security and Privacy*, pages 204–211. IEEE Computer Society Press, April 1988.

[48] B. Roscoe and P. Gardiner. Security Modelling in CSP and FDR: Final Report. Technical report, Formal Systems Europe, October 1995.

[49] A. D. Rubin and P. Honeyman. Formal Methods for the Analysis of Authentication Protocols. Technical Report Technical report 93–7, Technical Report, Center for Information Technology Integration, 1993. University of Michigan. Internal Draft.

[50] C. s. The NRL Protocol Analyser: An Overview. *Journal of Logic Programming*, 1994.

[51] S. Schneider. Security Properties and CSP. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 174–187. IEEE Computer Society Press, may 1996.

[52] V. Shmatikov and J. C. Mitchell. Analysis of a Fair Exchange Protocol. *Seventh Annual Symposium on Network and Distributed System Security, San Diego*, pages 119–128, 2000.

[53] E. Snekkenes. Authentication in Open Systems. *In 10th IFIP WG 6.1 Symposium on Protocol Specification, Testing and Verification*, pages 313–324, June 1990.

[54] E. Snekkenes. *Formal Specification and Analysis of Cryptographic Protocols*. PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, Norwegian Defence Research Establishment, P.O. Box 25, N-2007, Kjeller, Norway, January 1995.

[55] P. Syverson. The Use of Logic in the Analysis of Cryptographic Protocols. In Teresa F. Lunt and John McLean, editors, *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pages 156–170. IEEE Computer Society, May 1991.

[56] P. Syverson. Knowledge, Belief, and Semantics in the Analysis of Cryptographic Protocols. *Journal of Computer Security*, 1(3):317–334, 92.

[57] P. Syverson, C. Meadows, and C. Iliano. Dolev-Yao is no better than Machiavelli. In *Proceedings of the First Workshop on Issues in the Theory of Security - WITS'00, (P. Degano, editors)*, pages 87–92. Geneva, Switzerland, July 2000.

[58] P. Syverson and P. C. V. Oorshot. On Unifing some Cryptographic Protocol Logics. In *In IEEE 1994 Computer Society Symposium on Security and Privacy*, pages 14–28. IEEE Computer Society, May 1994.

[59] The commission of the European Communities CEC DG-XIII. Security Investigation Final Report. Technical Report S2011/7000/D010 7000 1000, CEC, September 1993.

[60] V. Varadharajan. Formal Specification of a Secure Distributed System. In *Proceedings of the 12th National Computer Security Conference*, pages 146–171, October 1989.

[61] V. Varadharajan. Verification of Network Security Protocols. *Computers and Security*, 8, December 1989.

[62] V. Varadharajan. Use of Formal Technique in the Specification of Authentication Protocols. *Computer Standards and Interfaces*, 9:203–215, 1990.