Formal Automatic Verification of Authentication Cryptographic Protocols

M. Debbabi M. Mejri N. Tawbi I. Yahmadi

Computer Science Department,
Laval University, Quebec, G1K 7P4,
Canada.
{debabi,mejri,tawbi,yahmadi}@ift.ulaval.ca

Abstract

We address the formal analysis of authentication cryptographic protocols. We present a new verification algorithm that generates from the protocol description the set of possible flaws, if any, as well as the corresponding attack scenarios. This algorithm does not require any property or invariant specification. The algorithm involves three steps: extracting the protocol roles, modeling the intruder abilities and verification. In addition to the classical known intruder computational abilities such as encryption and decryption, we also consider those computations that result from different instrumentations of the protocol. The intruder abilities are modeled as a deductive system. The verification is based on the extracted roles as well as the deductive system. It consists in checking whether the intruder can answer all the challenges uttered by a particular role. If it is the case, an attack scenario is automatically constructed. The extracted proof system does not ensure the termination of deductions. For that purpose, we present a general transformation schema that allows one to automatically rewrite the non-terminating proof system into a terminating one. The transformation schema is shown to be correct. To exemplify the usefulness and efficiency of our approach, we illustrate it on the Woo and Lam authentication protocol. Abadi and Needham have shown that the protocol is insecure and they proposed a new corrected version. Thanks to this method we have discovered new unknown flaws in the Woo and Lam protocol and in the corrected version of Abadi and Needham.

Keywords: *Algorithm, Automatic Formal Verification, Cryptographic Protocols, Authentication, Proof System, Flaws, Attack Scenario, Termination.*

1 Motivations and Background

Authentication is a major security concern in distributed systems. It is meant to help principals (persons, hosts, computers, etc.) to prove their identities to each other. Cryptographic protocols are used to ensure authentication and related purposes.

The design of authentication cryptographic protocols is known to be error prone. Several protocols have been shown flawed by computer security researchers. Consequently, a surge of interest is being devoted to the development of formal methods and tools for the specification, design and analysis of cryptographic protocols. A complete survey and a comparative study of these methods can be found in [1, 2, 4, 5, 6].

A new approach for the analysis of authentication cryptographic protocols is presented in this paper. This approach is fully automatic, formal and does not need any specification of protocol properties or invariants. The analysis of a given cryptographic protocol is performed in three main steps. First, starting from a classical protocol description, the roles of the different agents acting in the protocol are extracted. A role is, actually, a protocol abstraction where the emphasis is put only on one principal. Such an abstraction is useful to point out the individual interpretation of the exchanged protocol messages by a specific principal. Second, the intruder abilities are automatically extracted. In addition to the well known abilities, we consider all the computational abilities provided by the protocol itself. In our approach, such computation abilities are captured as a finite proof system in which the inference is based on narrowing i.e. rewriting modulo unification. Third, the roles together with the proof system are combined to perform the protocol verification. The latter consists in checking whether one can instrument a particular role by answering all

the challenges uttered by this role.

The main contributions of this work are:

- A new approach for the analysis of cryptographic protocols. This approach is formal, fully automatic and does not necessitate any specification of any protocol property or invariant.
- An illustration of the usefulness and efficiency of our approach, on the analysis of the Woo and Lam authentication protocol.
- A new transformation schema that allows one to rewrite some non-terminating proof systems into terminating ones.

We consider here a subclass of authentication cryptographic protocols. This class consists of protocols that use symmetric-key cryptography and at most one server. We assume also that the exchanged messages are only constructed over nonces, principal identities and symmetric-keys. The rest of this paper is structured as follows: In Section 2, we recall briefly the one-way authentication protocol of Woo and Lam. Section 3 is devoted to an informal presentation of the verification algorithm. Section 4 is concerned with termination issues of proof systems. A few concluding remarks and a discussion of further research are ultimately sketched as a conclusion in Section 5.

2 The Woo and Lam Protocol

In what follows, we recall the one-way Woo and Lam authentication protocol as presented in [7, 8]. This protocol relies on symmetric-key cryptography. The protocol is given in Table 1. Here N_b is a nonce i.e. a random number generated by B specially 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 protocol runs as follows when the principal A wants to prove his identity to the principal B: (1) A initiates the protocol and claims his identity to B; (2) B replies by sending the nonce N_b and asking A to encrypt it under k_{as} in order to prove what he claimed; (3) A returns the nonce N_b encrypted under k_{as}; (4) B forwards the response encrypted, together with A's identity, under k_{bs} for verification; (5) S decrypts the received message using B's key, extracts the encrypted component, decrypts it using A's key and re-encrypts under B's Key. If S replies by $\{N_b\}_{k_{bs}}$, then B will find N_b after decrypting it and he should be convinced that A is really running this session with him. The next section is devoted to an informal presentation of our verification algorithm. The latter is illustrated over the Woo and Lam authentication protocol described above.

3 Informal Presentation

Starting from a protocol description, the algorithm operates in three main steps:

(a) Role extraction
(b) Proof system generation
(c) Verification

3.1 Role Extraction

Roles are protocol abstractions where the emphasis is put each time on a particular principal. A role reflects the way by which some principal perceives the protocol. For instance, in the case of the Woo and Lam protocol of Table 1, three roles could be extracted: A, B and S. The principal, playing the role A, participates in the protocol through three main steps: First, A sends his identity to the principal B. Second, he receives a nonce N_b from B. Third, he sends the message $\{N_b\}_{k_{as}}$ to B. Hence, the role associated to A could be written as the following sequence of actions:

$$Role(A) = \langle !, A, B\rangle\langle ?, N_b, B\rangle\langle !, \{N_b\}_{k_{as}}, B\rangle$$

An action is a triple of the form $\langle dir, m, P\rangle$ where dir is a direction symbol (either ? meaning input or ! meaning output), m is a message and P is a principal identifier. Similarly, the roles associated with B and S could be written as follows:

$$\begin{aligned} Role(B) &= \langle ?, A, A\rangle\langle !, N_b, A\rangle \\ &\quad\ \langle ?, \{N_b\}_{k_{as}}, A\rangle \\ &\quad\ \langle !, \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}, S\rangle \\ &\quad\ \langle ?, \{N_b\}_{k_{bs}}, S\rangle \\ Role(S) &= \langle ?, \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}, B\rangle\langle !, \{N_b\}_{k_{bs}}, B\rangle \end{aligned}$$

The second step of our algorithm is the proof system generation which is explained in the next section.

3.2 Proof System Generation

In cryptographic protocol design, one must always assume the presence of an intruder that has a complete control over the communication network. According to this assumption, the intruder is able to intercept, modify, store, retrieve and send any circulating message in the network. Moreover, we assume that the intruder has the usual encryption and decryption abilities. In this work, we consider another valuable source

$Message\ 1.$	A	$\longrightarrow$	B	:	A
$Message\ 2.$	B	$\longrightarrow$	A	:	N_b
$Message\ 3.$	A	$\longrightarrow$	B	:	$\{N_b\}_{k_{as}}$
$Message\ 4.$	B	$\longrightarrow$	S	:	$\{A, \{N_b\}_{k_{as}}\}_{k_{bs}}$
$Message\ 5.$	S	$\longrightarrow$	B	:	$\{N_b\}_{k_{bs}}$

Table 1: The Woo and Lam Authentication Protocol

of information available to the intruder: the protocol itself. Actually, the protocol may be viewed by the intruder as a computation resource. The latter may be used to synthesize and deliver some particular information that makes possible a fatal attack. Here, we capture all these intruder abilities by a deductive proof system. We associate to each protocol step an inference rule. Each inference rule captures a class of possible instrumentations of the protocol. The general form of an inference rule is:

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

where $p_1, \ldots, p_n, m$ are messages and c is a boolean formula. Here is the way such an inference rule should be read: the intruder could supply the protocol with the messages $p1, \ldots, p_n$ and get the message m from the protocol provided that side condition c holds. In fact, the side condition refers to those freshness conditions dictated by the protocol. Furthermore, we will endow each inference rule with a sequence of protocol steps (a scenario) showing *how* the intruder could instrument the protocol with the information $p_1, \ldots, p_n$ so as to get the message m.

Now, let us see how this applies to the Woo and Lam protocol of Table 1. For the lack of space, we present here the inference rules together with the associated scenarios without explaining how they are generated.

The first step in the Woo and Lam protocol of Table 1 is:

$$1.\ A \longrightarrow B : A$$

The inference rule associated with this protocol step is:

$$\frac{}{A}$$

meaning that the intruder could get from the protocol the identity of any principal wishing to initiate a protocol session. Here is the scenario that makes this possible:

$$1.\ A \longrightarrow I(B) : A$$

This sequence stipulates that the intruder could get the identity of a principal A by intercepting the message sent by this principal when wishing to initiate a new protocol session.

The second step in the Woo and Lam protocol of Table 1 is:

$$2.\ B \longrightarrow A : N_b$$

The inference rule associated with this protocol step is:

$$\frac{A}{N_b}\ Fresh(N_b)$$

meaning that the intruder could get a fresh nonce (generated by B) from the protocol just by supplying it with an agent identity. The side condition $Fresh(N_b)$ stipulates the freshness[1] of the information N_b. Here is the scenario that illustrates such a situation:

1.	$I(A)$	$\longrightarrow$	B	:	A
2.	B	$\longrightarrow$	$I(A)$	:	N_b

The third step in the Woo and Lam protocol of Table 1 is:

$$3.\ A \longrightarrow B : \{N_b\}_{k_{as}}$$

The inference rule associated with this protocol step is:

$$\frac{X}{\{X\}_{k_{as}}}$$

meaning that the intruder could supply the protocol with any information X and get it encrypted under the secret key shared between the server S and a principal

[1] The symbol $Fresh$ denotes a unary predicate that holds whenever its argument is fresh.

A. At this level, we would like to pinpoint one important feature in our verification algorithm. The reader should notice that we used X in the inference rule instead of N_b. The rationale underlying this choice is that the principal A, at the third step of the protocol, replies with the message received at the second step (i.e. N_b) encrypted by k_{as}. Notice that N_b is a nonce generated and uttered by B and sent to A. Now, let us mention the following facts:

- The principal A has no preliminary knowledge on the effective value of the expected message at the second step of the protocol. Accordingly, A has no means to verify the value of the received message at that step.
- The freshness of N_b 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 message received at the second step of the protocol.
- 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 second 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 second step of the protocol. Consequently, A will accept any message at that step and will reply by sending the encrypted version of this message under the key k_{as}. This explains the rationale underlying the use of a variable message X instead of N_b. Here is the scenario that exhibits how the intruder could instrument the protocol so as to get the message $\{X\}_{k_{as}}$ provided that he supplies the protocol with X:

$$\begin{array}{llcll} 1. & A & \longrightarrow & I(B) & : & A \\ 2. & I(B) & \longrightarrow & A & : & X \\ 3. & A & \longrightarrow & I(B) & : & \{X\}_{k_{as}} \end{array}$$

Now, let us give some general comments regarding the inference rules and the associated scenarios. First, the agent identifiers as well as the introduced variables are implicitly universally quantified. Second, the generated scenarios constitute *proofs* of the *correction* of the inference rules. Actually, the scenarios are propagated during the deduction process.

The fourth step in the Woo and Lam protocol of Table 1 is:

$$4.\ B \longrightarrow S : \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}$$

The inference rule associated with this protocol step is:

$$\frac{A \quad X}{\{A, X\}_{k_{bs}}}$$

meaning that the intruder could supply the protocol with a principal identifier, say A, and a known message X so as to get an encrypted version of the composition of these two messages under the key k_{bs}. Here is the scenario that illustrates such a situation:

$$\begin{array}{llcll} 1. & I(A) & \longrightarrow & B & : & A \\ 2. & B & \longrightarrow & I(A) & : & N_b \\ 3. & I(A) & \longrightarrow & B & : & X \\ 4. & B & \longrightarrow & I(S) & : & \{A, X\}_{k_{bs}} \end{array}$$

The fifth step in the Woo and Lam protocol is:

$$5.\ S \longrightarrow B : \{N_b\}_{k_{bs}}$$

The inference rule associated with this protocol step is:

$$\frac{\{A, \{X\}_{k_{as}}\}_{k_{bs}}}{\{X\}_{k_{bs}}}$$

meaning that the intruder could supply the protocol with a message of the form $\{A, \{X\}_{k_{as}}\}_{k_{bs}}$ so as to get the message $\{X\}_{k_{bs}}$. Here is the scenario that illustrates such a situation:

$$\begin{array}{llcll} 4. & I(B) & \longrightarrow & S & : & \{A, \{X\}_{k_{as}}\}_{k_{bs}} \\ 5. & S & \longrightarrow & I(B) & : & \{X\}_{k_{bs}} \end{array}$$

3.3 Verification

In this section, we show how to combine roles and inference rules so as to perform the verification of a given cryptographic authentication protocol.

A principal A wishing to prove his identity to a principal B, has to answer all the challenges uttered by B. Hence, the intruder can impersonate the principal A if he can answer all those challenges uttered by B to A. Thus, the verification principal consists in checking whether the intruder, with all its computation abilities, can answer such challenges. Actually, the intruder computation abilities consist of:

- An initial knowledge generally made of the keys that the intruder shares with other principals, the server identity and other principal identities. Notice that this initial knowledge must be given explicitly within the protocol specification.
- The usual encryption, decryption, composition and decomposition abilities. Indeed, the intruder could encrypt and decrypt any message under known keys. In addition, he has the ability to compose (catenate) and decompose messages.

- Those computation abilities given by the protocol itself and captured by the deductive proof system.

Now we come to the explanation of the verification procedure. Starting from the protocol description, we extract the roles. Now, for each role, the intruder will attempt to answer all the challenges uttered so as to perform a masquerade. The intruder will progressively follow the role in a stepwise way. For a given step, if the role sends a message, the intruder will merely store it and hence extending his knowledge. If the role is waiting for receiving some particular message, the intruder will attempt to generate it using his current computation abilities. In other words, the intruder will try to synthesize the required message using his current knowledge modulo some composition/decomposition and encryption/decryption steps, and also up to the use of the deductive proof system.

Now, let us see how this applies to the unidirectional Woo and Lam authentication protocol of Table 1. Notice that the only concerned role is B.

Actually, before starting to take up the various challenges uttered by B, we have first to proceed to what we call the *generalization* of the role B. This operation aims to replace some messages by message variables as done in the deductive proof system. The rationale underlying such substitutions is that these messages could not be verified from the content, structure, freshness and type standpoint. The role B is rewritten into:

$$\begin{aligned} Role(B) \quad = \quad & \langle ?, A, A\rangle \\ & \langle !, N_b, A\rangle \\ & \langle ?, X, A\rangle \\ & \langle !, \{A, X\}_{k_{bs}}, S\rangle \\ & \langle ?, \{N_b\}_{k_{bs}}, S\rangle \end{aligned}$$

Here, we replace the message $\{N_b\}_{k_{as}}$ by a message variable X since B does not know the key k_{as}, hence he could accept any other information.

First of all, let us initialize the knowledge of the intruder, noted KI, with the intruder initial knowledge. In other words, KI is set to $\{k_{is}, A, B, S\}$. This means that the intruder initially knows the identity of the server S, his shared key with S, the role A and the role B.

In the first step of the role B, the principal is waiting for a message identifier, say A. To take up this challenge, the intruder has to generate such an identifier from KI modulo some composition/decomposition and encryption/decryption steps, and also up to the use of the deductive proof system. Such a constraint is written as:

$$(E_1) \qquad KI \models_R A$$

where R stands for the set of inference rules previously mentioned together with the usual composition/decomposition and encryption/decryption rules.

At the second step, B sends the nonce N_b over the network. Hence, the message becomes available to the intruder who extends his current knowledge to become $KI \cup \{N_b\} = \{k_{is}, A, B, S, N_b\}$.

At the third step, the intruder must supply B with X. Accordingly, this second constraint will be written as:

$$(E_2) \qquad KI \cup \{N_b\} \models_R X$$

At the fourth step, B offers the message $\{A, X\}_{k_{bs}}$ to the intruder. The latter updates his knowledge to become $KI \cup \{N_b, \{A, X\}_{k_{bs}}\}$.

Finally, the intruder has to supply the role B with $\{N_b\}_{k_{bs}}$ in order to achieve the masquerade. This last constraint will be written as:

$$(E_3) \qquad KI \cup \{N_b, \{A, X\}_{k_{bs}}\} \models_R \{N_b\}_{k_{bs}}$$

At this level, we are ready to check whether the intruder can or cannot impersonate the role A. This amounts to check the existence of solutions to the constraint set $\{E_1, E_2, E_3\}$.

In the case of the constraint set $\{E_1, E_2, E_3\}$, the algorithm produces many flaws and attack scenarios. Most of these attacks are new and completely different from those known up to now. In what follows, we present in Table 2 one among the many elegant attacks yielded by the algorithm.

This attack could be read as follows: First, the intruder begins a session with B in order to prove to B that its identity is A. Second, the intruder I intercepts the nonce N_b generated by B and sent to A. Third, I responds by a message *anything* which may be any message since B cannot do any verification. Fourth, B must react according to the protocol by sending the message $\{A, anything\}_{K_{bs}}$. This message will be intercepted by the intruder. Meanwhile, the server S has no idea about what is happening. To finish this protocol run, the intruder ultimately needs the message $\{N_b\}_{k_{bs}}$ which will be sent to B at the step (1.5). The goal of the sessions 2,3 and 4 is to generate that final required message. Finally, the intruder finishes the first session with B by sending the message $\{N_b\}_{k_{bs}}$ as if it is the response of S. Hence, B is convinced that the initiator of the session one is A.

$$
\begin{array}{llcll}
1.1 & I(A) & \longrightarrow & B & : A \\
1.2 & B & \longrightarrow & I(A) & : N_b \\
1.3 & I(A) & \longrightarrow & B & : anything \\
1.4 & B & \longrightarrow & I(S) & : \{A, anything\}_{k_{bs}} \\
\quad 2.1 & C & \longrightarrow & I(D) & : C \\
\quad 2.2 & I(D) & \longrightarrow & C & : N_b \\
\quad 2.3 & C & \longrightarrow & I(D) & : \{N_b\}_{k_{cs}} \\
\quad\quad 3.1 & C & \longrightarrow & I(E) & : C \\
\quad\quad 3.2 & I(E) & \longrightarrow & C & : C, \{N_b\}_{k_{cs}} \\
\quad\quad 3.3 & C & \longrightarrow & I(E) & : \{C, \{N_b\}_{k_{cs}}\}_{k_{cs}} \\
\quad\quad\quad 4.1 & I(C) & \longrightarrow & B & : C \\
\quad\quad\quad 4.2 & B & \longrightarrow & I(C) & : N'_b \\
\quad\quad\quad 4.3 & I(C) & \longrightarrow & B & : \{C, \{N_b\}_{k_{cs}}\}_{k_{cs}} \\
\quad\quad\quad 4.4 & B & \longrightarrow & S & : \{C, \{C, \{N_b\}_{k_{cs}}\}_{k_{cs}}\}_{k_{bs}} \\
\quad\quad\quad 4.5 & S & \longrightarrow & I(B) & : \{C, \{N_b\}_{k_{cs}}\}_{k_{bs}} \\
\quad\quad\quad\quad 5.4 & I(B) & \longrightarrow & S & : \{C, \{N_b\}_{k_{cs}}\}_{k_{bs}} \\
\quad\quad\quad\quad 5.5 & S & \longrightarrow & I(B) & : \{N_b\}_{k_{bs}} \\
1.5 & I(S) & \longrightarrow & B & : \{N_b\}_{k_{bs}}
\end{array}
$$

Table 2: An Attack of the Woo and Lam Authentication Protocol

4 Handling Termination

In this section, we pinpoint the termination problem at the level of the proof systems extracted from protocol specifications. We illustrate the problem on a concrete example and present a transformation schema that allows one to generate a terminating proof system from a non-terminating one. We establish the soundness of such a transformation.

Actually, if S is a proof system and g a goal, then the deduction using backward chaining induced by SLD-resolution may not terminate. Here is an illustration of such a phenomenon. Let S_1 be the following proof system:

$$S_1 = \{R_1 = \frac{ff(x_1)}{f(x_1)}, R_2 = \frac{x_2}{fff(x_2)}, R_3 = \frac{\Box}{a}\}$$

Let $f(b)$ be a goal. The resolution of this goal using backward chaining does not terminate even though it is obvious that $f(b)$ is not a theorem. The non-termination is due to the rule R_1 that increases the complexity of the goal with respect to the well-founded ordering induced by term structure.

Now, if we try to establish the same goal using the proof system S_2 defined hereafter:

$$S_2 = \{R_1 = \frac{x}{f(x)}, R_2 = \frac{\Box}{a}\}$$

we will succeed despite that the two systems S_1 and S_2 are equivalent in the sense that they denote the same theory i.e. each theorem in S_1 is also a theorem in S_2 and vice-versa.

In the sequel, we establish a significant result: we present a new method that allows us to generate from a non-terminating[2] proof system, an equivalent terminating one. Notice that the method works for almost all the extracted proof systems and is proved to be sound. The proof is not given in this paper for the lack of space. The following definitions and notations are adopted along this section.

Let S be an inference system, we denote by $\mathcal{U}^i(S)$ any tuple of i arbitrary rules of S where i is a positive integer. Let R be an inference rule, we denote by $\mathcal{P}(R)$ the set of R premises and by $\mathcal{C}(R)$ the R conclusion. We assume that all the variables appearing in R are universally implicitly quantified. We assume also that the premises do not involve boolean connectives.

The next definition allows us to view the set of premises of an inference rule as a tuple. Such a repre-

[2]The termination here is related to the deduction strategy. We use here backward chaining. A proof system is said to be non-terminating whenever it is impossible to find a well-founded ordering that makes the conclusion of each rule less than each premise of the same rule.

sentation is adopted for the sake of convenience in the transformation process of an inference system.

Definition 4.1 (minimal tuple) *Let $\mathcal{A}$ be an algebra and $T = \{t_1, t_2, \ldots, t_n\}$ a set of n terms in $\mathcal{A}$. Let $\preccurlyeq_*$ be a total ordering on $\mathcal{A}$. We define $Ord_{\preccurlyeq_*}(T) = (t_{i1}, t_{i2}, \ldots, t_{in})$ such that $t_{ij}, t_{ik} \in T$ and if $j < k$ then $t_{ij} \prec_* t_{ik}$.*

The definition hereafter establishes the connection between premises and conclusions that are unifiable in a proof tree.

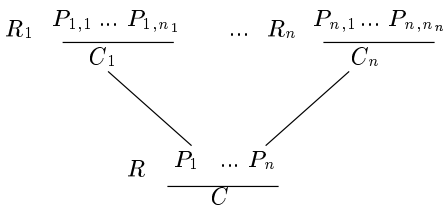
Definition 4.2 (n-conjunction) *We denote by n-conjunction, the conjunction of n pairs of terms $(P_1, C_1), (P_2, C_2), ..., (P_n, C_n)$, written $(P_1, C_1) \wedge (P_2, C_2) \wedge ... \wedge (P_n, C_n)$.*

The next definition extends the usual definition of the most general unifier to an n-conjunction.

Definition 4.3 (common most general unifier of an n-conjunction) *Let $A = (P_1, C_1) \wedge (P_2, C_2) \wedge ... \wedge (P_n, C_n)$ be an n-conjunction. The common most general unifier (cmgu) of A, written $cmgu(A)$, is a substitution σ that satisfies the following conditions:*
(1) $(P_i, C_i) \in A$ then $\sigma(P_i) = \sigma(C_i)$
(2) if $\sigma\prime$ satisfies (1) then there exists $\sigma\prime\prime$ such that: $\sigma\prime\prime \circ \sigma = \sigma\prime$

Notice that if $n = 1$ the common most general unifier is merely the most general unifier.

The following definition introduces a rule combinator written $\Uparrow$. Such a combinator generates a new rule from an inference rule, say R, and a tuple of inference rules $(R_1, R_2, ..., R_n)$. The new rule generated corresponds to the contraction (abstraction) of those proof trees in which R and $R_1, R_2, ..., R_n$ are involved. In the case of the following diagram:



the new generated rule is:

$$\frac{\sigma(P_{1,1}), ..., \sigma(P_{1,n_1}), ..., \sigma(P_{n,1}), ..., \sigma(P_{n,n_n})}{\sigma(C)}$$

where $\sigma = cmgu((P_1, C_1) \wedge ... \wedge (P_n, C_n))$

Definition 4.4 (composition of rules) *Let $R = \frac{\{p_1, p_2, ..., p_n\}}{c}$ be an inference rule such that: $Ord_{\preccurlyeq_*}\{p_1, p_2, ..., p_n\} = (p_{i_1}, p_{i_2}, ..., p_{i_n})$. Let $(R_1, R_2, ..., R_n)$ be a tuple of n rules.*

We define the composition of R with $(R_1, R_2, ..., R_n)$, denoted by $R \Uparrow (R_1, R_2, ..., R_n)$, as the following inference rule:

$$\frac{\sigma(\mathcal{P}(R_1)) \cup \sigma(\mathcal{P}(R_2)) ... \cup \sigma(\mathcal{P}(R_n))}{\sigma(c)}$$

where $\sigma = cmgu(p_1, \mathcal{C}(R_1)) \wedge (p_2, \mathcal{C}(R_2)) \wedge \ldots \wedge (p_n, \mathcal{C}(R_n))$. This composition is defined only if σ exists.

The next definition is devoted to the extension of the previous rule combinator to sets of inference rules. Indeed, we introduce the operator $\Uparrow\!\!\!\!\rightarrow$ that performs the composition of two sets of inference rules. Actually, thanks to such an operator, we can rewrite progressively the original non-terminating proof system until saturation. Once such a state reached, we can exclude all those inference rules that are problematic from the termination standpoint. This transformation schema is made sound (theory preserving) thanks to the soundness theorem stated in the sequel.

Definition 4.5 (partial closure composition) *Let $S_1 = \{R_1^1, R_2^1, ..., R_{n_1}^1\}$ and $S_2 = \{R_1^2, R_2^2, ..., R_{n_2}^2\}$ two sets of inference rules. We define the partial closure composition of S_1 by S_2, written $S_1 \Uparrow\!\!\!\!\rightarrow S_2$ as follows:*

$$S_1 \Uparrow\!\!\!\!\rightarrow S_2 = S_1 \cup S_2 \cup \left(\bigcup_{R_i^2 \in S_2} \left(\bigcup_{\substack{u \in \mathcal{U}^{|\mathcal{P}(R_i^2)|}(S_1) \\ \wedge R_i^2 \Uparrow u \text{ is defined}}} \{R_i^2 \Uparrow u\} \right) \right)$$

In the following, we introduce a set of definitions that formalize the notion of proof tree or merely proof. In addition, we present an equivalence notion between inference systems. These definitions will be used in both the statement and the proof of the soundness theorem, which constitutes the major result of this section.

Definition 4.6 ($\mathcal{R}$-tree) *A tree A is said to be an $\mathcal{R}$-tree if each node is a pair (R, σ) where R is an inference rule and σ is a substitution.*

The following definitions are associated with an $\mathcal{R}$-tree.

- If T is an $\mathcal{R}$-tree of root (R, σ) then $\sigma(\mathcal{C}(R))$ is denoted by $Res(T)$.

- We define an equivalence relation over $\mathcal{R}$-trees, written $\equiv_T$ as follows: $T_1 \equiv_T T_2$ if $Res(T_1) = Res(T_2)$.
- For each node s in the tree we associate indexes 1,2, ..., n to the edges linking s to its sons (where n is the number of sons of s) starting from 1 which is associated to the leftmost son. Hence, it is easy to identify the position of a node in the tree as a list of numbers representing the path followed from the root to this node. The empty list ϵ is associated to the root position. We define an order relation $>$ over the positions such that $p_1 > p_2$ if p_1 is lexicographically greater than p_2 i.e. $p_1 = n_1.n_2...n_{n_1} > p_2 = m_1.m_2...m_{n_2}$, if there exists i such that $\forall\ j,\ 1 \leq j < i,\ n_j = m_j$ and $n_i > m_i$.
- Let S be a set of rules and T be an $\mathcal{R}$-tree. We define $T(S) = \{((R_i, \sigma_i), p_i) \mid R_i \in S$ and (R_i, σ_i) occurs at position p_i in $T\}$ and $\overline{T(S)} = \{((R_i, \sigma_i), p_i) \mid R_i \notin S$ and (R_i, σ_i) occurs at position p_i in $T\}$.
- $((R_q, \sigma_q), p_q) \in T(S)$ is an upper bound of $T(S)$ if $\forall\ ((R, \sigma), p) \in T(S) \backslash \{(R, \sigma), p)\}$ we have $p_q > p$.
- Let T be an $\mathcal{R}$-tree. We define $T_{|p}$ as the subtree of T such that its root occurs at position p.

Definition 4.7 (well-formed $\mathcal{R}$-tree) *Let T be an $\mathcal{R}$-tree. The tree T is said to be a well-formed $\mathcal{R}$-tree if it satisfies one of the following conditions:*

1. *T has only one node (R, σ) such that $\mathcal{P}(R) = \emptyset$.*
2. *If (R, σ) is a node of T such that $Ord_{\preccurlyeq_*}(\mathcal{P}(R)) = (P_1, P_2, ..., P_n)$ then $\forall\ i,\ 1 \leq i \leq n$ $T_{|i}$ is a well-formed $\mathcal{R}$-tree and $Res(T_{|i}) = \sigma(P_i)$.*

Definition 4.8 (proof) *Let S be an inference system and T be a well-formed $\mathcal{R}$-tree. T is a proof in S if $\overline{T(S)} = \emptyset$. $Res(T)$ is a theorem in S, T being its proof.*

The following definition formalizes the equivalence notions between two inference systems.

Definition 4.9 (comparison of inference systems) *Let S_1 and S_2 be two inference systems:*

1. *$S1 \leq_S S2$ if $\forall\ Th$, Th is a theorem in S_1 then Th is a theorem in S_2.*
2. *$S1 \geq_S S2$ if $\forall\ Th$, Th is a theorem in S_2 then Th is a theorem in S_1.*
3. *$S_1 \equiv_S S_2$ if $S_1 \leq_S S_2$ and $S_1 \geq_S S_2$.*

The next definition introduces the operator $\Downarrow$ that is devised to simplify an inference system by eliminating those redundant rules.

Definition 4.10 (simplified inference system) *Let S be an inference system, we define the simplified inference system associated with S, written $S_\Downarrow$, as follows: let $R \in S$ then $R \in S_\Downarrow$ if the two following conditions hold:*

1. *$\forall\ P \in \mathcal{P}(R), P \neq \mathcal{C}(R)$*
2. *There is no rule $R' \in S$, $(R' \neq R)$ with a substitution σ such that: $\sigma(\mathcal{P}(R')) \subseteq \mathcal{P}(R)$ and $\sigma(\mathcal{C}(R')) = \mathcal{C}(R)$*

The next definition characterizes the transformation schema mentioned previously. Actually, it allows us to construct a series of inference systems. Whenever such a series converges, it yields as a limit an equivalent (to the original one) and terminating inference system.

Definition 4.11 (derivator and derivated series) *We define a derivator as a triplet $(S, \looparrowright, \preccurlyeq)$ where S is an inference system, $\looparrowright$ is a partial closure relation and $\preccurlyeq$ is a total ordering over the terms. With each derivator $(S, \looparrowright, \preccurlyeq)$, we associate a derivated series, written $(S^n)_{n \geq 0}$ defined as follows:*

$$\begin{cases} S^0 = S \\ S^n = (S^{n-1}_{\prec} \looparrowright S^{n-1}_{\succcurlyeq})_\Downarrow \quad n \geq 1 \end{cases}$$

Here is the theorem that states the soundness of the transformation schema. The reader should notice that this result goes beyond cryptographic proof systems. Actually, the transformation schema as well as the soundness theorem apply to any inference system.

Theorem 4.12 *Let $S = S_1 \cup S_2$ an inference system such that $\forall\ R \in S_2$ we have $\mathcal{P}(R) \neq \emptyset$. If $(S_1 \looparrowright S_2)_\Downarrow = (S_1 \cup S_2)_\Downarrow$ then $S \equiv_S S_1$*

The intention hereafter is to explain the effective use of the transformation schema. Starting from a non-terminating inference system S, we have first to choose a total ordering $\prec_*$ over the terms. This ordering induces a natural partition over S. In fact, $S = S_{\prec_*} \cup S_{\succcurlyeq_*}$ where $S_{\prec_*}$ contains the rules of S that satisfy the following condition: $R \in S_{\prec_*}$ then $\forall\ P \in \mathcal{P}(R)$ we have $P \prec_* \mathcal{C}(R)$, whereas $S_{\succcurlyeq_*} = S \backslash S_{\prec_*}$. Second, we compute the elements of derivated series $(S^n)_{n \geq 0}$:

$$\begin{cases} S^0 = S \\ S^n = (S^{n-1}_{\prec_*} \looparrowright S^{n-1}_{\succcurlyeq_*})_\Downarrow \quad n \geq 1 \end{cases}$$

associated to the derivator $(S, \hookrightarrow, \preccurlyeq_*)$ until we reach an inference system S^i such that $(S^i)_\Downarrow \equiv_S (S^{i-1})_\Downarrow$. Hence, we conclude that $S \equiv_S S^{i-1}_{\prec_*}$ and we declare that the series converges.

Generally speaking, the series do not necessarily converge. Nevertheless, as far as we have experimented with this transformation schema, it seems that this method is very efficient in handling termination and benefits from a good convergence acceleration. Also, convergence is achieved in almost all the inference systems we have tested. Notice that at each iteration, it is mandatory to rename the variables so as to avoid clashes between variable rules.

In the sequel, we will denote by $\preccurlyeq_{nf}$ the total ordering defined as follows: Let A be an algebra and t is a term in A. We denote by $Fnct(t)$ the number of function symbols used in t. If t_1, t_2 are two terms in A, $t_1 \preccurlyeq_{nf} t_2$ whenever $Fnct(t_1) \leq Fnct(t_2)$. In the following, we will illustrate the transformation schema using this ordering. However, the reader may refer to [3] for other suitable orderings.

The second row in 4 reports the proof system $S^1 = (S^0_{\prec_{nf}} \hookrightarrow S^0_{\succcurlyeq_{nf}})_\Downarrow$. The third row of 4 reports the result of the composition $S^1_{\prec_{nf}} \hookrightarrow S^1_{\succcurlyeq_{nf}})_\Downarrow$. Now, by the computation of $S^2 = (S^1_{\prec_{nf}} \hookrightarrow S^1_{\succcurlyeq_{nf}})_\Downarrow$ we can eliminate the following redundant rules:
1. $R_{12} \Uparrow R_5$ because it is an α-renaming of R_{13}.
2. $R_{13} \Uparrow (R_1, R_3)$ because it is an α-renaming of R_3.
3. $R_{13} \Uparrow (R_1, R_4)$ because it is an α-renaming of R_4.
4. $R_{13} \Uparrow (R_1, R_6)$ because it is an α-renaming of R_6.

At this level, we have $(S^1_{\prec_{nf}} \hookrightarrow S^1_{\succcurlyeq_{nf}})_\Downarrow = S^1$. Consequently, by using the theorem above, we conclude that $S^0 \equiv_S S^1_{\prec_{nf}}$. The reader should notice that, albeit that $S^0 \equiv_S S^1_{\prec_{nf}}$, the proof system $S^1_{\prec_{nf}}$ is terminating whereas S^0 is not.

5 Conclusion

We reported in this paper a new algorithm for the formal automatic verification of authentication protocols. This algorithm does not necessitate any specification of any protocol property or invariant. It takes as parameter the protocol specification and generates the set of flaws, if any, as well as the corresponding attack scenarios. The algorithm involves three steps. First, protocol roles are extracted from the protocol specification. Second, the intruder abilities to perform communications and computations are generated from the protocol specification. In addition to the classical known intruder computational abilities, we also consider those computations that result from different instrumentations of the protocol. The intruder abilities are modeled as a deductive proof system. Third, the extracted roles as well as the deductive system are combined to perform the verification. The latter consists in checking whether the intruder can answer all the challenges uttered by a particular role. We illustrated the algorithm on Woo and Lam authentication protocol. Thanks to this algorithm, we discovered new unknown flaws in the Woo and Lam protocol and even in the corrected version of Abadi and Needham. A prototype of this verification algorithm has been implemented in BNR-Prolog. A significant contribution of this paper is a new transformation schema that allows one to rewrite some non-terminating proof systems into terminating ones. The soundness of such a schema is established and is applied to those inference systems generated by our algorithm. As future work, we intend to lift this verification algorithm to deal with key-exchange cryptographic protocols, timestamps and hash functions.

References

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

[2] John Clark and Jeremy Jacob. A Survey of Authentication Protocol Literature. Unpublished article available at `http://dcpu1.cs.york.ac.uk/~jeremy`.

[3] N. Dershowitz. Termination of rewriting. *Symbolic Computation*, 3(1&2):69–115, 1987.

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

[5] Catherine Meadows. Formal verification of cryptographic protocols: A survey. In *Proceedings of Asiacrypt 96*, 1996.

[6] 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.

[7] T. Y. C. Woo and S. S. Lam. Authentication for Distributed Systems. *Computer*, 25(1):39–52, January 1992.

[8] T. Y. C. Woo and S. S. Lam. A Lesson on Authentication Protocol Design. *Operating Systems Review*, pages 24–37, 1994.

$S^i_{\prec_{nf}}$	$S^i_{\succcurlyeq_{nf}}$
$R_1 : \frac{\square}{\widehat{x_0}}$ $\quad R_2 : \frac{\square}{K(\widehat{I},\widehat{S})}$ $R_3 : \frac{x_1}{\{x_1\}_{K(\widehat{x_2},\widehat{S})}}$ $\quad R_4 : \frac{\widehat{x_3}, x_4}{\{x_3.x_4\}_{K(\widehat{x_5},\widehat{S})}}$ $R_5 : \frac{x_6, x_7}{x_6.x_7}$ $\quad R_6 : \frac{x_8, K(\widehat{x_9},\widehat{S})}{\{x_8\}_{K(\widehat{x_9},\widehat{S})}}$	$R_7 : \frac{\{\widehat{x_{10}}.\{x_{11}\}_{K(\widehat{x_{10}},\widehat{S})}\}_{K(\widehat{x_{12}},\widehat{S})}}{\{x_{11}\}_{K(\widehat{x_{12}},\widehat{S})}}$ $R_8 : \frac{\widehat{x_{13}}}{N_b} fresh(N_b)$ $R_9 : \frac{x_{14}.x_{15}}{x_{14}}$ $\quad R_{10} : \frac{x_{16}.x_{17}}{x_{17}}$ $R_{11} : \frac{\{x_{18}\}_{K(\widehat{x_{19}},\widehat{S})}, K(\widehat{x_{19}},\widehat{S})}{x_{18}}$
$R_8 \Uparrow R_1 : \frac{\square}{N_b} fresh(N_b)$	$R_7 \Uparrow R_3 : \frac{\widehat{x_{10}}.\{x_{11}\}_{K(\widehat{x_{10}},\widehat{S})}}{\{x_{11}\}_{K(\widehat{x_{12}},\widehat{S})}}$ $R_7 \Uparrow R_4 : \frac{\widehat{x_{10}}, \{x_{11}\}_{K(\widehat{x_{10}},\widehat{S})}}{\{x_{11}\}_{K(\widehat{x_{12}},\widehat{S})}}$ $R_7 \Uparrow R_6 : \frac{\widehat{x_{10}}.\{x_{11}\}_{K(\widehat{x_{10}},\widehat{S})}, K(\widehat{x_{12}},\widehat{S})}{\{x_{11}\}_{K(\widehat{x_{12}},\widehat{S})}}$ $R_9 \Uparrow R_5 : \frac{x_{14}, x_{15}}{x_{14}}$ $\quad R_{10} \Uparrow R_5 : \frac{x_{16}.x_{17}}{x_{17}}$ $R_{11} \Uparrow (R_3, R_2) : \frac{x_{18}}{x_{18}}$ $R_{11} \Uparrow (R_4, R_2) : \frac{x_3, x_4}{x_3.x_4}$ $R_{11} \Uparrow (R_6, R_2) : \frac{x_{18}, K(\widehat{x_{19}},\widehat{S})}{x_{18}}$

Table 3: Transformation Schema Applied to the Woo and Lam Protocol: Part I

$S^i_{\prec_{nf}}$	$S^i_{\succcurlyeq_{nf}}$
$R_1 : \frac{\square}{\widehat{x_0}}$ $\quad R_2 : \frac{\square}{K(\widehat{I},\widehat{S})}$ $R_3 : \frac{x_1}{\{x_1\}_{K(\widehat{x_2},\widehat{S})}}$ $\quad R_4 : \frac{\widehat{x_3}, x_4}{\{x_3.x_4\}_{K(\widehat{x_5},\widehat{S})}}$ $R_5 : \frac{x_6, x_7}{x_6.x_7}$ $\quad R_6 : \frac{x_8, K(\widehat{x_9},\widehat{S})}{\{x_8\}_{K(\widehat{x_9},\widehat{S})}}$ $R_{14} = R_8 \Uparrow R_1 : \frac{\square}{N_b} fresh(N_b)$	$R_7 : \frac{\{\widehat{x_{10}}.\{x_{11}\}_{K(\widehat{x_{10}},\widehat{S})}\}_{K(\widehat{x_{12}},\widehat{S})}}{\{x_{11}\}_{K(\widehat{x_{12}},\widehat{S})}}$ $R_8 : \frac{\widehat{x_{13}}}{N_b} fresh(N_b)$ $R_9 : \frac{x_{14}.x_{15}}{x_{14}}$ $\quad R_{10} : \frac{x_{16}.x_{17}}{x_{17}}$ $R_{11} : \frac{\{x_{18}\}_{K(\widehat{x_{19}},\widehat{S})}, K(\widehat{x_{19}},\widehat{S})}{x_{18}}$ $R_{12} = R_7 \Uparrow R_3 : \frac{\widehat{x_{20}}.\{x_{21}\}_{K(\widehat{x_{20}},\widehat{S})}}{\{x_{21}\}_{K(\widehat{x_{22}},\widehat{S})}}$ $R_{13} = R_7 \Uparrow R_4 : \frac{\widehat{x_{20}}, \{x_{21}\}_{K(\widehat{x_{20}},\widehat{S})}}{\{x_{21}\}_{K(\widehat{x_{22}},\widehat{S})}}$
$R_{13} \Uparrow (R_1, R_3) : \frac{x_{21}}{\{x_{21}\}_{K(\widehat{x_{22}},\widehat{S})}}$ $R_{13} \Uparrow (R_1, R_4) : \frac{\widehat{x_3}, x_4}{\{x_3.x_4\}_{K(\widehat{x_5},\widehat{S})}}$ $R_{13} \Uparrow (R_1, R_6) : \frac{x_8, K(\widehat{x_9},\widehat{S})}{\{x_8\}_{K(\widehat{x_9},\widehat{S})}}$	$R_{12} \Uparrow R_5 : \frac{\widehat{x_{20}}, \{x_{21}\}_{K(\widehat{x_{20}},\widehat{S})}}{\{x_{21}\}_{K(\widehat{x_{22}},\widehat{S})}}$

Table 4: Transformation Schema Applied to the Woo and Lam Protocol: Part II