

Logical aspects of user authentication protocols

László Aszalós* and Philippe Balbiani

Institut de recherche en informatique de Toulouse
Irit-CNRS, Université Paul Sabatier
118 route de Narbonne, 31062 Toulouse Cedex 4, France
aszalos@math.klte.hu balbiani@irit.fr

Abstract. The core of our paper is a general purpose logical system for reasoning about user authentication protocols. Proposed as an extension of the propositional epistemic logic by dynamic operators, the potential usefulness of our calculus for protocol verification is illustrated with examples.

1 Introduction

To protect data from exposure, it is desirable to encipher plaintext information under keys which allow users to send and receive messages over an insecure network. As a result, the users must find secure methods for exchanging the keys either by themselves or by means of a system key manager. Authentication protocols emerged from numerous works of computer scientists and their use has become common in the science and study of methods of exchanging keys. They are basically sequences of message exchanges, whose purpose is to assure users that communications do not leak confidential data. Indeed, there is a wide variety of protocols that have been specified and implemented, from protocols with trusted third party, to protocols with public key and, even more generally, hybrid protocols. The one drawback is that many of them have been shown to be flawed, from which one may explain the great deal of attention devoted to the formal verification of security properties of protocols. Examples of protocols can be found in [2].

In the literature, the most popular logic-based formal approach to the analysis of authentication protocols is perhaps the modal BAN calculus introduced by Burrows, Abadi and Needham [1]. From the point of view of computer science, a virtue of BAN is that it allows static characterization of epistemic concepts. In spite of its success in finding flaws or redundancies in some well-known protocols, the effectiveness of BAN as a formal method for the analysis of authentication protocols has been a source of debate, see [11] for details. The problem with the BAN logic is that it explicitly excludes time. On the other hand there is no way to represent actions performed by users. Communication, by its nature, refers to time, and its properties are naturally expressed in terms of actions like sending and receiving messages. When devising a protocol, we usually think of

* On leave from the university of Debrecen, Hungary.

some property that we want the protocol to satisfy. We are mainly interested in the correctness of a protocol with respect to epistemic properties between two users like the arranging of a secret key known only to them. Therefore, our emphasis is on the interplay between knowledge and action. This leads us to consider a language that allows to express notions of knowledge and actions in a straightforward way: the language of modal logic.

Many-dimensional modal logics are logics arising from the study of formal languages that are capable of characterizing different aspects of a domain, from time, to space, and, even more generally, intensional concepts like knowledge, action, obligation, etc. They form a part of the field of modal logic and have applications in artificial intelligence and computer science. Their study can be found in [5, 6, 10]. To illustrate the truth of this, many-dimensional modal logics allow an intuitive and attractive approach to the analysis of the behavior of multi-agent distributed systems by means of a formalism containing both epistemic operators and temporal operators, see [4] for details. This paper uses epistemic operators and dynamic operators to develop a formalized language, focusing on the fundamental notions of knowledge and action.

This paper presents what in our opinion constitutes the basis of authentication protocol verification. In section 2, we provide some necessary background on cryptography and data security. In addition to the basic definitions, we present a number of protocols, in particular the Needham-Schroeder public key protocol. A formalized language for the analysis of authentication protocols, proposed as an extension of the propositional epistemic logic by dynamic operators, is introduced in section 3. We also see examples illustrating its potential usefulness for the analysis of how knowledge evolves when protocols are executed. The semantic presentation, based on the notion of a global state is given in section 4. Finally, section 5 concludes the paper through a number of open questions.

2 Cryptography and data security

A cryptosystem or cipher system is a method of disguising messages so that only certain people can see through the disguise. Cryptography is the art of creating and using cryptosystems. The original message is called a plaintext. The disguised message is called a ciphertext. Encryption means any procedure to convert plaintext into ciphertext. Decryption means any procedure to convert ciphertext into plaintext. At symmetric encryption, we use the same key for encryption and decryption. At public key encryption, there are pairs of keys. If we use one of the keys to encrypt then we can use the other key to decrypt and vice versa. Usually one of the keys is known only to the owner (private key) and the other is known to everybody (public key). In this article we denote by k_{AB} the symmetric key shared by A and B whereas we denote by k_A and k_A^{-1} the public and private key of user A , respectively. At the public key schemes, the encryption and the decryption is a very lengthy procedure whereas symmetric key encryption can be done more efficiently. Hence the public key cryptosystems usually generate symmetric “session” keys and are using this key. We assume

that users communicate over a network and hence they need to exchange the session key over the network. At this exchange we require that

- After the exchange the sender and receiver can perform encryption and decryption using the session key.
- Intruders cannot decrypt messages, only the receiver (confidentiality)
- Receiver knows that the message was encrypted by a given entity and not someone else (authentication)
- Intruders cannot modify messages (integrity)

The cryptosystem consists of protocols. One protocol is an ordered list of messages. Let us see a simple example of a protocol:

Protocol 1

1. $A \rightarrow B: k_A^{-1}(k_B(k_{AB}))$.

Here we have one message, the sender is A and the receiver is B . The message contains the symmetric key generated by A , and this key at first is encrypted with the public key of user B , and after this ciphertext is encrypted again, but now with the private key of A . All the users know the public key of A , so all of them can extract $k_B(k_{AB})$, but this ciphertext can be opened (decrypt) with the private key of B , and this is the secret of B . Hence only B can acquire the symmetric key k_{AB} .

The second protocol is a variant of the previous one.

Protocol 2

1. $A \rightarrow B: k_B(k_A^{-1}(k_{AB}))$

Here we switch the order of encryptions. Hence by using previous reasoning, we get that only user B can decrypt this message. Here, it is obvious that only user A could generate the ciphertext $k_A^{-1}(k_{AB})$. From this it follows that A knows the symmetric key, which is not true at the first protocol, as we shall see in the next section. Finally we give a more complicated example, the celebrated Needham-Schroeder public key protocol:

1. $A \rightarrow B: k_B(N_A, A)$
2. $B \rightarrow A: k_A(N_A, N_B,)$
3. $A \rightarrow B: k_B(N_B)$

This protocol use *nonces*. A nonce is a randomly generated number, and at the successive runs of a protocol nonces never get the same value. This can help to exclude several flaws, see [3, 9].

3 A formalized language

When analyzing protocols run by users over a network that is vulnerable to many attacks, we want to focus on the communication aspects. As a result, the notion of message is basic. Suppose we fix a finite set USE of users' names,

with typical member denoted i , a countably infinite set PLA of plaintexts, with typical member denoted P , a countably infinite set SYM of symmetric keys, with typical member denoted k and a countably infinite set NON of nonces, with typical member denoted n . We assume a countably infinite set VAR of variables, which we usually write as x, y, z , etc. The set of all messages is inductively defined as follows:

$$m := i \mid x \mid \langle m_1, m_2 \rangle \mid \text{left}(m) \mid \text{right}(m) \mid k_i(m) \mid k_i^{-1}(m) \mid E(x, m)$$

i.e. we start with users' names and variables and we form more complicated messages by closing off under the pairing operators $\langle \cdot, \cdot \rangle$, $\text{left}(\cdot)$ and $\text{right}(\cdot)$ and the ciphering operators $k_i(\cdot)$, $k_i^{-1}(\cdot)$ and $E(\cdot, \cdot)$. The expression $E(x, m)$ denotes the encryption of m with symmetric key x . We will assume that:

- $USE = \{1, \dots, n\}$ for some positive integer $n \geq 2$,
- PLA consists of all grammatically correct sentences in some plaintext space,
- SYM is made up of all symmetric keys in some key space and
- NON is composed of all nonces in some nonce space.

As for variables, they will take values corresponding to the universes discussed by the users: USE , PLA , SYM and NON . The pairing operator $\langle \cdot, \cdot \rangle$ is used to form the concatenation $\langle m_1, m_2 \rangle$ of messages m_1 and m_2 whereas left and right are used to obtain respectively the left part m_1 and the right part m_2 of compound message $\langle m_1, m_2 \rangle$. The ciphering operators $k_i(\cdot)$ and $k_i^{-1}(\cdot)$ denote respectively the public enciphering transformation of user i and the private enciphering transformation of that user. We will always assume that for every user i , its public k_i is distributed to all users before protocols are run while its private k_i^{-1} is known only to that user. Because we have supposed that k_i and k_i^{-1} denote the public and private enciphering transformations of user i , this means that messages like $k_i(k_i^{-1}(m))$ and $k_i^{-1}(k_i(m))$ must be identified with m . The ciphering operator $E(\cdot, \cdot)$ denote some enciphering transformation in a symmetric cryptosystem. This means that messages like $E(x, E(x, m))$ must be identified with m . From all this it follows that we have to consider the equational theory \mathcal{E} over our set of terms defined by the following equations:

- $\text{left}(\langle m_1, m_2 \rangle) = m_1$ and $\text{right}(\langle m_1, m_2 \rangle) = m_2$,
- $k_i(k_i^{-1}(m)) = m$ and $k_i^{-1}(k_i(m)) = m$ and
- $E(x, E(x, m)) = m$.

\mathcal{E} defines a congruence relation on the set of all messages and we say that messages m_1 and m_2 are \mathcal{E} -equal, abbreviated by $m_1 \cong_{\mathcal{E}} m_2$, iff m_1 and m_2 are in this congruence. We shall say that equation $m_1 = m_2$ is matchable iff there is a substitution σ such that $\text{dom}(\sigma) \cap \text{var}(m_1) = \emptyset$ and $m_1 \cong_{\mathcal{E}} \sigma m_2$. An authentication protocol is an abstraction where users interact by making actions. Within our framework, we will have to consider atomic actions like sending and receiving messages. Further actions with several atomic actions in sequence are typically described by means of dynamic constructs like sequential composition,

nondeterministic choice, nondeterministic iteration and test. As a result, the set of all actions is inductively defined as follows:

$$\alpha := \lambda \mid \text{exp?} \mid \text{text}(x) \mid \text{key}(x) \mid \text{nonce}(x) \mid \text{send}(m) \mid \text{rec}(m) \mid \\ \alpha_1; \alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \alpha^*$$

where exp ranges over the set of all expressions. λ is the nullary action “do nothing”. Roughly speaking, expressions allow us to reason about messages, users and the messages that users have. We formally defined them in this section. It must be remarked that programs are formed by starting with tests like exp? , atomic actions like $\text{text}(x)$, $\text{key}(x)$, $\text{nonce}(x)$, $\text{send}(m)$ and $\text{rec}(m)$, and closing off under dynamic constructs. With these definitions in hand, we can now define what it means for a user to execute an action:

- When i performs the atomic action $\text{text}(x)$, it chooses an element in the plaintext space PLA the value of which is given to x .
- When i executes the atomic action $\text{key}(x)$, it picks an element in the key space SYM the value of which is given to x .
- When i does the atomic action $\text{nonce}(x)$, it finds an element in the nonce space NON the value of which is given to x .
- When i makes the action $\alpha_1; \alpha_2$, it performs α_1 and then α_2 .
- When i carries out the action $\alpha_1 \cup \alpha_2$, it makes either α_1 or α_2 nondeterministically.
- When i performs the action α^* , it repeats α some finite number of times.

Remark that the dynamic constructs of our language come from the standard language of propositional dynamic logic [7, 8]. As for test, when user i does the action exp? , it evaluates whether the current global state satisfies exp . It continues if exp is true, otherwise it fails. We needed expressions to define actions. We can now define formally expressions:

$$\text{exp} := \text{has}_i(m) \mid m = m' \mid \neg \text{exp} \mid \text{exp}_1 \vee \text{exp}_2 \mid K_i \text{exp}$$

Atomic expression $\text{has}_i(m)$ is interpreted to mean that user i can compute m from:

- The pairing operators $\langle \cdot, \cdot \rangle$, *left* and *right*.
- The public keys k_1, \dots, k_n .
- Its private key k_i^{-1} .
- The plaintexts, the symmetric keys and the nonces he has already computed.
- The messages it has already received.

We read $K_i \text{exp}$ as “user i knows that exp is true”. We should consider, for instance, the expression $K_i \text{has}_j(m)$ which represents the fact that user i knows that user j can compute m . Our language should allow us to express the notion of a user gaining knowledge over time as it receives messages from the network. To make this idea precise, we start with the primitive formulas $K_i \text{exp}$ and we

form more complicated formulas by closing off under negation, disjunction, and the dynamic operators $[\alpha_1 \parallel \dots \parallel \alpha_n]$. This is expressed in one line as:

$$\phi := K_i \text{exp} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid [\alpha_1 \parallel \dots \parallel \alpha_n]\phi \mid K_i\phi$$

with the formula $[\alpha_1 \parallel \dots \parallel \alpha_n]\phi$ being read “after actions α_1 to α_n terminate in parallel, ϕ ” or “after every terminating execution of actions α_1 to α_n in parallel, ϕ is true”. Now consider the protocol 1

1. $1 \rightarrow 2: k_1^{-1}(k_2(k_{12}))$

where 1 sends $k_1^{-1}(k_2(k_{12}))$ to 2. We assume that k_{12} is a symmetric key picked by 1 in the key space SYM . As a result, 1 executes first the action $key(x)$, and performs then the action $send(k_1^{-1}(k_2(x)))$. As for user 2, it does not know in advance the symmetric key it will receive. As a result, 2 does the action $k_1^{-1}(k_2(y))$. Thus, in protocol 1, users 1 and 2 are making respectively the actions β_1 and β_2 :

$$\begin{aligned}\beta_1 &= \mathbf{key}(x); \mathbf{send}(k_1^{-1}(k_2(x))) \\ \beta_2 &= \mathbf{rec}(k_1^{-1}(k_2(y)))\end{aligned}$$

Consider the protocol 2:

1. $1 \rightarrow 2: k_2(k_1^{-1}(k_{12}))$

where 1 sends $k_2(k_1^{-1}(k_{12}))$ to 2. We assume again that k_{12} is a symmetric key picked by 1 in the key space SYM . We can formulate protocol 2 in a similar way:

$$\begin{aligned}\gamma_1 &= \mathbf{key}(x); \mathbf{send}(k_2(k_1^{-1}(x))) \\ \gamma_2 &= \mathbf{rec}(k_2(k_1^{-1}(y)))\end{aligned}$$

As for the Needham-Schroeder public key protocol, users 1 and 2 are doing respectively the actions δ_1 and δ_2 :

$$\begin{aligned}\delta_1 &= \mathbf{nonce}(x); \mathbf{send}(k_2(\langle x, 1 \rangle)); \mathbf{rec}(k_1(\langle x, y \rangle)); \mathbf{send}(k_2(y)); \\ \delta_2 &= \mathbf{rec}(k_2(\langle u, 1 \rangle)); \mathbf{nonce}(v); \mathbf{send}(k_1(\langle u, v \rangle)); \mathbf{rec}(k_2(v))\end{aligned}$$

4 Semantical presentation

Now that we have described the syntax of our logic, we need a semantics to determine whether a given formula is true or false. Following the line of reasoning suggested by Fagin et al. [4], the first step consists of defining the notion of local state and the notion of global state. In our framework, a user’s knowledge is determined by its local state whereas the global state describes the system at a given point of time. The user i ’s local state consists of two substitutions: θ_i and τ_i . Roughly speaking, θ_i is about the values given to variables by i when

it executes atomic actions like $\mathbf{text}(x)$, $\mathbf{key}(x)$ or $\mathbf{nonce}(x)$ whereas τ_i is about the values given to variables by i when it executes the atomic action $\mathbf{rec}(m)$. Since a user's local state reflect the knowledge it has acquired, we assume that i 's local state is getting longer over time. Once we associate a local state to each user, we have to associate to the whole system a global state. A global state, at a given point of time, must contain the n -tuple:

$$\begin{pmatrix} \theta_1 \dots \theta_n \\ \tau_1 \dots \tau_n \end{pmatrix}$$

where θ_i and τ_i constitute user i 's local state. It must also contain the list of all messages that has been sent up to this time point as well as markers indicating that part of the list such and such user has already received. Take the case of the global state:

$$\begin{pmatrix} \begin{pmatrix} (x/m) & (w/m') \\ (y/m') & (z/m) \end{pmatrix} & k_2(m, 1)k_1(m, m') & k_2(m') \\ & \uparrow & \uparrow \\ & 1 & 2 \end{pmatrix}$$

At this point of time, the above global state indicates that:

- User 1 has given the value m to variable x while executing some atomic action like $\mathbf{text}(x)$, $\mathbf{key}(x)$ or $\mathbf{nonce}(x)$.
- User 1 has given the value m' to variable y while receiving some message.
- User 2 has given the value m' to variable w while executing some atomic action like $\mathbf{text}(w)$, $\mathbf{key}(w)$ or $\mathbf{nonce}(w)$.
- User 2 has given the value m to variable z while receiving some message.
- Three messages have already been sent: $k_2(m, 1)$, $k_1(m, m')$ and $k_2(m')$.
- User 1 has not read the third message yet, while user 2 has read all messages.

Consider an expression \mathbf{exp} , a formula ϕ and a global state s . The relation “ \mathbf{exp} is true at global state s ”, denoted $\mathbf{exp} \in T(s)$, and the relation “ ϕ is true at global state s ”, denoted $s \models \phi$, are defined inductively on the formation of \mathbf{exp} and on the formation of ϕ as follows:

- $\mathbf{has}_i(m) \in T(s)$ iff, according to the information it has at its local state s_i , user i can compute m .
- $m = m' \in T(s)$ iff $m \cong_{\mathcal{E}} m'$.
- $\neg \mathbf{exp} \in T(s)$ iff $\mathbf{exp} \notin T(s)$.
- $\mathbf{exp} \vee \mathbf{exp}' \in T(s)$ iff $\mathbf{exp} \in T(s)$ or $\mathbf{exp}' \in T(s)$.
- $K_i \mathbf{exp} \in T(s)$ iff, for every global states t , if $s \equiv_i t$ then $\mathbf{exp} \in T(t)$.
- $s \models K_i \phi$ iff, for every global states t , if $s \equiv_i t$ then $(\theta_i \cup \tau_i) \mathbf{exp} \in T(t)$.
- $s \models \neg \phi$ iff $s \not\models \phi$.
- $s \models \phi \vee \psi$ iff $s \models \phi$ or $s \models \psi$.
- $s \models [\alpha_1 \parallel \dots \parallel \alpha_n] \phi$ iff, for all global states t , if $s R_{\alpha_1, \dots, \alpha_n} t$ then $t \models \phi$.
- $s \models K_i \phi$ iff, for all available global states t , if $s \equiv_i t$ then $t \models \phi$.

The notation of available global states is defined later in this section. The binary relations \equiv_i are the relations between global states defined by $s \equiv_i s'$ iff:

- $\theta_i = \theta'_i$
- $\tau_i = \tau'_i$
- User i has send in l and l' the same messages in the same order.
- User i has received in l and l' the same messages in the same order.

where

$$s = \left(\left(\begin{array}{c} \theta_1 \dots \theta_i \dots \theta_n \\ \tau_1 \dots \tau_i \dots \tau_n \end{array} \right) l \right)$$

and

$$s' = \left(\left(\begin{array}{c} \theta'_1 \dots \theta'_i \dots \theta'_n \\ \tau'_1 \dots \tau'_i \dots \tau'_n \end{array} \right) l' \right).$$

Remark that \equiv_i is an equivalence relation between global states for every user ' i '. The binary relations $R_{\alpha_1, \dots, \alpha_n}$ are the relations between global states defined by:

$$\begin{aligned} R_{\alpha_1, \dots, \alpha_{i_1} \cup \alpha_{i_2}, \dots, \alpha_n} &= R_{\alpha_1, \dots, \alpha_{i_1}, \dots, \alpha_n} \cup R_{\alpha_1, \dots, \alpha_{i_2}, \dots, \alpha_n} \\ R_{\pi_1; \alpha_1, \dots, \pi_i; \alpha_i, \dots, \pi_n; \alpha_n} &= \\ &R_{\pi_1, \lambda, \dots, \lambda} \circ R_{\alpha_1, \dots, \pi_i; \alpha_i, \dots, \pi_n; \alpha_n} \cup \dots \\ &R_{\lambda, \dots, \lambda, \pi_i, \lambda, \dots, \lambda} \circ R_{\pi_1; \alpha_1, \dots, \alpha_i, \dots, \pi_n; \alpha_n} \cup \dots \\ &R_{\lambda, \dots, \lambda, \pi_n} \circ R_{\pi_1; \alpha_1, \dots, \pi_i; \alpha_i, \dots, \alpha_n} \end{aligned}$$

where π_i are atomic actions, namely **text**(\cdot), **key**(\cdot), **nonce**(\cdot) **send**(\cdot) or **rec**(\cdot).

$$s R_{\lambda, \dots, \text{exp?}, \dots, \lambda} t \text{ iff } s = t \text{ and } t \models K_i \text{exp}$$

$$\left(\left(\begin{array}{c} \theta_1 \dots \theta_i \dots \theta_n \\ \tau_1 \dots \tau_i \dots \tau_n \end{array} \right) l \right) R_{\lambda, \dots, \text{key}(x), \dots, \lambda} \left(\left(\begin{array}{c} \theta_1 \dots \theta_i \cdot (x/c) \dots \theta_n \\ \tau_1 \dots \tau_i \dots \tau_n \end{array} \right) l \right)$$

At **nonce** and **text** we have a similar condition.

$$\left(\left(\begin{array}{c} \theta_1 \dots \theta_i \dots \theta_n \\ \tau_1 \dots \tau_i \dots \tau_n \end{array} \right) l \right) R_{\lambda, \dots, \text{send}(m), \dots, \lambda} \left(\left(\begin{array}{c} \theta_1 \dots \theta_i \dots \theta_n \\ \tau_1 \dots \tau_i \dots \tau_n \end{array} \right) l, (\theta_i \cup \tau_i)m \right)$$

Here $(\theta_i \cup \tau_i)m$ is the result of applying substitutions for θ_i and τ_i to the term m . The $(\theta_i \cup \tau_i)m$ must be ground.

$$\begin{aligned} &\left(\left(\begin{array}{c} \theta_1 \dots \theta_i \dots \theta_n \\ \tau_1 \dots \tau_i \dots \tau_n \end{array} \right) \dots \begin{array}{c} m' \dots \\ \uparrow \\ i \end{array} \dots \right) R_{\lambda, \dots, \text{rec}(m), \dots, \lambda} \\ &\left(\left(\begin{array}{c} \theta_1 \dots \theta_i \dots \theta_n \\ \tau_1 \dots \tau_i \cdot \mu(m, m') \dots \tau_n \end{array} \right) \dots \begin{array}{c} m' \dots \\ \uparrow \\ i \end{array} \dots \right) \end{aligned}$$

The substitution $\mu(m, m')$ matches the message m to the message m' .

The global state

$$s_0 = \left(\left(\begin{array}{c} \lambda \dots \lambda \\ \lambda \dots \lambda \end{array} \right) \begin{array}{c} \uparrow \\ 1 \end{array} \dots \begin{array}{c} \uparrow \\ n \end{array} \lambda \right)$$

is called the initial global state. A global state s is said to be *available* when there exist programs $\alpha_1, \dots, \alpha_n$ such that $s_0 R_{\alpha_1, \dots, \alpha_n} s$. Let us see one run of protocol 1. Here at the beginning there are no substitutions and there are no messages. The initial global state with two agents, is equal to:

$$\left(\left(\begin{array}{cc} \lambda & \lambda \\ \lambda & \lambda \end{array} \right) \begin{array}{c} \uparrow \uparrow \\ 1 \ 2 \end{array} \lambda \right)$$

As user 1 executes the **key**(x) action and produces the key k , the global state became the following:

$$\left(\left(\begin{array}{cc} (x/k) & \lambda \\ \lambda & \lambda \end{array} \right) \begin{array}{c} \uparrow \uparrow \\ 1 \ 2 \end{array} \lambda \right)$$

In this step there are no messages yet. But when user 1 send the message $k_1^{-1}(k_2(k))$, we move to the following global state:

$$\left(\left(\begin{array}{cc} (x/k) & \lambda \\ \lambda & \lambda \end{array} \right) \begin{array}{c} \uparrow \uparrow \\ 1 \ 2 \end{array} k_1^{-1}(k_2(k)) \right)$$

When user 2 receives this message, its marker moves forward, and user 2 realizes that the variable y gets the value k :

$$\left(\left(\begin{array}{cc} (x/k) & \lambda \\ \lambda & (y/k) \end{array} \right) \begin{array}{c} \uparrow \\ 1 \end{array} \begin{array}{c} k_1^{-1}(k_2(k)) \\ \uparrow \\ 2 \end{array} \right)$$

We are interested in users' knowledge, for example at this global state user 2 can deduce that user 1 knows (or using our terminology has) the key k . Because private key k_1^{-1} is known only by user 1, it is sure that user 1 has the message $k_2(k)$, because user 1 is the only use able to produce $k_1^{-1}(k_2(k))$. In this case we had only one message, so user 2 cannot receive the message $k_2(k)$ from a third user. Hence user 1 has generated it. And in this case user 1 really has the key k .

What is the situation if we have more users, e.g. three, and user 2 executes its own program? Is it true, that in each case user 1 has the key k ? We can restate this question as: for all programs ζ_1 and ζ_3

$$\left(\left(\begin{array}{ccc} \lambda & \lambda & \lambda \\ \lambda & \lambda & \lambda \end{array} \right) \begin{array}{c} \uparrow \uparrow \uparrow \\ 1 \ 2 \ 3 \end{array} \lambda \right) \models [\zeta_1 \parallel \beta_2 \parallel \zeta_3] K_2 \mathbf{has}_1(y)?$$

It can be check easily that with

$$\begin{aligned} \zeta_1 &= \mathbf{rec}(x); \mathbf{send}(k_1^{-1}(x)) \\ \zeta_3 &= \mathbf{key}(z); \mathbf{rec}(k_2(z)) \end{aligned}$$

we have a case where user 1 do not own the key. If we use protocol 2, this cannot happen, because only user 1 can generate $k_1^{-1}(k)$, so by using protocol 2 user 2 can be sure that user 1 knows the key. (But in this case user 2 cannot be sure that user 1 directed him this message.)

To prove, that

$$\left(\left(\begin{array}{cc} \lambda & \lambda \\ \lambda & \lambda \end{array} \right) \begin{array}{c} \uparrow \uparrow \\ 1 \ 2 \end{array} \lambda \right) \models [\delta_1 \parallel \delta_2] K_1 \mathbf{has}_2(y)$$

we need to check the alternative states of the final state

$$\left(\left(\begin{array}{cc} (x/m) & (w/m') \\ (y/m') & (z/m) \end{array} \right) \begin{array}{c} k_2(m,1)k_1(m,m') \\ \uparrow \uparrow \\ 1 \quad 2 \end{array} \begin{array}{c} k_2(m') \\ \uparrow \\ 2 \end{array} \right) \quad (1)$$

according to user 1. Such states are one of the following forms:

$$\left(\left(\begin{array}{cc} (x/m) & (w/m') \\ (y/m') & (z/m) \end{array} \right) \begin{array}{c} k_2(m,1)k_1(m,m') \\ \uparrow \uparrow \\ 1 \ 2 \end{array} \begin{array}{c} k_2(m') \dots \\ \uparrow \\ 2 \end{array} \right) \quad (2)$$

or

$$\left(\left(\begin{array}{cc} (x/m) & (w/m') \\ (y/m') & (z/m) \end{array} \right) \begin{array}{c} k_2(m,1)k_1(m,m') \\ \uparrow \\ 1 \end{array} \begin{array}{c} k_2(m') \dots \\ \uparrow \\ 2 \end{array} \right). \quad (3)$$

In both case user 1 received the message $k_1(m, m')$. This message contains m , so user 2 has received the first message and has extracted this nonce. There are no messages between the first and the second message, hence only user 2 could send it. This message contains m' , so user 2 knows this nonce.

The global state (1) has no other alternatives according to user 2. By using a similar reasoning as before we can conclude that

$$\left(\left(\begin{array}{cc} \lambda & \lambda \\ \lambda & \lambda \end{array} \right) \begin{array}{c} \uparrow \uparrow \\ 1 \ 2 \end{array} \lambda \right) \models [\delta_1 \parallel \delta_2] K_2 \mathbf{has}_1(w).$$

As state (2) is alternative of state (1) according to user 1, state

$$\left(\left(\begin{array}{cc} (x/m) & (w/m') \\ (y/m') & (z/m) \end{array} \right) \begin{array}{c} k_2(m,1) \\ \uparrow \\ 1 \end{array} \begin{array}{c} k_1(m,m') \\ \uparrow \\ 2 \end{array} \begin{array}{c} k_2(m') \\ \uparrow \\ 2 \end{array} \right) \quad (4)$$

is alternative of state (2) according to user 2, and at state (4) it is not true that $\mathbf{has}_1(m')$, hence

$$\left(\left(\begin{array}{cc} \lambda & \lambda \\ \lambda & \lambda \end{array} \right) \begin{array}{c} \uparrow \uparrow \\ 1 \ 2 \end{array} \lambda \right) \not\models [\delta_2 \parallel \delta_2] K_1 K_2 \mathbf{has}_1(y).$$

5 Conclusion

In this paper, we have presented a formal language devoted to the verification of authentication protocols. Through tree examples of protocols, we have seen how the exchanges of messages between users can be expressed. The axiomatization and the decidability of the set of formulas true at all global states are open problems.

Acknowledgments

The research of the first author is partly supported by the French ministry of education whereas the research of the second author is supported by the COST action “Theory and Applications of Relational Structures as Knowledge Instruments”.

References

- [1] Michael Burrows, Martín Abadi, and Roger Needham. A logic for authentication. In *Proceedings of the Royal Society of London*, volume 426, pages 233–271, 1989.
- [2] John Clark and Jeremy Jacob. A survey of authentication protocol literature. Unpublished report.
- [3] Dorothy Elizabeth Robling Denning. *Cryptography and Data Security*. Addison-Wesley Pub Co, 1982.
- [4] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about knowledge*. MIT Press, Cambridge, MA, 1995.
- [5] Dov M. Gabbay, Ágnes Kurucz, Frank Wolter, and Michael Zakharyashev. Many-Dimensional Modal Logics: Theory and Applications. To appear.
- [6] Dov M. Gabbay and Valentin B. Shehtman. Products of modal logics. I. *Logic Journal of the IGPL. Interest Group in Pure and Applied Logics*, 6(1):73–146, 1998.
- [7] Robert Goldblatt. *Logics of Time and Computation*, volume 7 of *Lecture Notes*. Center for the Study of Language and Information, 1987.
- [8] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.
- [9] Gavin Lowe. An attack on the Needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56:131–136, 1995.
- [10] Maarten Marx and Yde Venema. *Multi-dimensional modal logic*, volume 4 of *Applied Logic Series*. Kluwer Academic Publishers, Dordrecht, 1997.
- [11] Aviel D. Rubin and Peter Honeyman. Formal methods for the analysis of authentication protocols. Unpublished report.