# Verification of Payment Protocols via MultiAgent Model Checking

M. Benerecetti[1], M. Panti[2], L. Spalazzi[2], and S. Tacconi[2]

[1] Dept. of Physics, University of Naples "Federico II"
Napoli, Italy
bene@na.infn.it

[2] Istituto di Informatica, University of Ancona
Ancona, Italy
{panti,spalazzi,tacconi}@inform.unian.it

**Abstract.** The paper presents a logic of belief and time (called MATL) that can be used to verify electronic payment protocols. This logic encompasses its predecessors in the family of logics of authentication. According to our approach, the verification is performed by means of MultiAgent Model Checking Checking, an extension of traditional model checking to cope with time and beliefs. In this framework, principals are modeled as concurrent processes able to have beliefs about other principals. The approach is applied to the verification of the Lu and Smolka protocol, a variant of SET. The results of our analysis show that the protocol does not satisfy some important security requirements, which make it subject to attacks.

## 1 Introduction

In this paper we show how *MultiAgent Model Checking* [6] (an extension of traditional model checking, see e.g. [10]) can be used for the verification of electronic payment protocols using a *logic of belief and time*. This work extends to payment protocols our previous work on authentication protocols [4,5].

The application of model checking to payment protocol verification is not new (e.g., see [13,15,17]). However, in the previous work, payment protocols are verified by introducing the notion of intruder and, then, by verifying whether the intruder can attack a given protocol. This approach makes it possible to directly find a trace of a possible attack, but it may not be clear what the protocol flaw really is. This work usually employs temporal logics or process algebras.

A different approach makes use of logics of belief or knowledge to specify and verifying both authentication protocols (see, e.g. [8,1]) and payment protocols (e.g., see [7,11,14]). The use of such logics requires no models of intruder, and allows one to find what the protocol flaw is, allowing to specify (and check) security properties in a more natural way. However, in this approach, usually verification is performed proof-theoretically.

Our approach can be seen as a combination of the above two: we employ a logic called MATL (MultiAgent Temporal Logic) which is able to represent both time and belief (thus it follows the line of the work based on logics of belief or knowledge and does not use any model of the intruder); but verification is performed by means of a symbolic model checker (called NuMAS [3], a model checker based on NuSMV [9]). NuMAS is built on the work described in [6], where model checking is applied to BDI attitudes (i.e., Belief, Desire, and Intention) of agents.

Our work aims at the use of MATL for modeling payment protocols and uses NuMAS for their verification. This goal is fulfilled in three steps. First, we capture traditional logics of authentication (e.g., as [1,8,18]) in MATL. Second, we extend the above work in order to capture typical issues of electronic payment protocols. MATL is expressive enough to fulfill both the previous steps. Third, we model *principals* participating to a payment protocol session as (concurrent finite state) processes able to have beliefs within the NuMAS system.

The specification of a principal has two orthogonal aspects: a temporal aspect, and a belief aspect. When we consider the temporal evolution of a principal we treat belief atoms (namely, atomic formulae expressing belief) as atomic propositions. The fact that these formulae talk about beliefs is not taken into consideration. When we deal with the beliefs of a principal $P$, we model its beliefs about another principal $Q$ as the fact that $P$ has access to a representation of $Q$ as a process. Then, any time it needs to check the truth value of some belief formula about $Q$, e.g., $B_Q \phi$, $P$ simply tests whether $\phi$ holds in its (appropriate) representation of $Q$. Beliefs are essentially used to control the "jumping" among processes. This operation is iterated in the obvious way in case of nested beliefs.

The paper is structured as follows. In Section 2 we briefly introduce the Lu and Smolka Protocol (a variant of the well-known Secure Electronic Transaction (SET) Protocol), as a running example. Section 3 describes MATL and its use as a logic of authentication. The use of MATL as a logic for payment protocols is described in Section 4. Section 5 describes the formal specifications for the usual security requirements of payment protocols. The results of the verification of the Lu and Smolka protocol are reported in Section 6. Finally, some conclusions are drawn in Section 7.

## 2 The Lu-Smolka Variant of the SET Protocol

The Secure Electronic Transaction (SET [16]) is an electronic commerce protocol jointly developed by Visa and Mastercard in order to guarantee secure transactions over open networks. SET is not a monolithic protocol, but a suite comprising seventeen subprotocols, each devoted to make secure a specific phase of a commercial transaction. The Lu-Smolka variant of this protocol [15], reported in Figure 1, is a simplified version of the subprotocol involved in the payment phase only. This subprotocol is supposed to be invoked during a web-based commercial transaction. In other words, a client (the cardholder) after selecting the goods/services that it wishes to purchase/request, the shipping address, and the billing address (if any), uses the protocol to perform the on-line payment. In Figure 1, $C$ denotes the cardholder, $M$ the merchant, and $P$ the payment

$$\begin{array}{lll}
(1) & C \rightarrow M & : TIDR \quad \textit{Initiate Request} \\
(2) & M \rightarrow C & : S_{K_M^{-1}}\{TID\} \quad \textit{Initiate Response} \\
(3) & C \rightarrow M & : S_{K_C^{-1}}\{TID\}, \{TID, PA\}_{K_m}, S_{K_C^{-1}}\{\{TID, PY, CA\}_{K_p}\} \\
 & & \quad \textit{Purchase Request} \\
(4) & M \rightarrow P & : S_{K_C^{-1}}\{\{TID, PY, CA\}_{K_p}\}, S_{K_M^{-1}}\{TID\}, \{TID, AA, MA\}_{K_p} \\
 & & \quad \textit{Authorization \& Capture Request} \\
(5) & P \rightarrow M & : S_{K_P^{-1}}\{TID, Tr\} \quad \textit{Authorization \& Capture Response} \\
(6) & M \rightarrow C & : S_{K_P^{-1}}\{TID, Tr\} \quad \textit{Purchase Response}
\end{array}$$

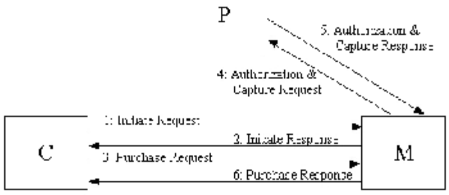

**Fig. 1.** The Lu-Smolka variant of SET

gateway. The notation $\{m\}_{K_x}$ denotes a message $m$ encrypted with the public key of $X$, whereas $S_{K_x^{-1}}\{m\}$ denotes a message $m$ signed by $X$ with its private key. $TIDR$ represents a record containing initialization data for the protocol. $TID$ represents the unique identifier of the transaction, it can be considered as a nonce. $PA$ is the amount that the cardholder is supposed to pay, $PY$ is the amount that the cardholder is willing to pay, and $AA$ is the amount of charge that the merchant requests for authorization. $CA$ and $MA$ are the identifiers of the cardholder's account and of the merchant's account, respectively. Finally, $Tr$ denotes the answer of the gateway that can be either an authorization or a negation of authorization.

Intuitively, the protocol works as follows. In step (1), $C$ sends the request for a unique transaction identifier to $M$. $M$ assigns the $TID$ to the transaction, signs it, and returns it to $C$ in step (2). At this point, in step (3), $C$ sends the Ordering Information (OI, i.e., $TID$ and $PA$) to $M$, together with the Payment Instruction (PI, i.e., $TID$,$PY$,$CA$). $PY$ may be lower than $PA$, in case the cardholder is trying to deceive the merchant. Moreover, the $PI$ is for the payment gateway, while $OI$ is for the merchant. In step (4), $M$ decrypts $OI$, checks if it is correct, then encrypts a new $OI$ ($OI'$ composed by $TID$,$AA$,$MA$) with the payment gateway's public key. Finally, $M$ sends, in a single message, an authorization and capture request to $P$. Notice that $AA$ may be higher than $PY$. In step (5), $P$ authorizes $C$'s payment card, checks for consistency between $OI'$ and $PI$, performs appropriate account operations, and returns the transaction result $Tr$ to $M$. In step (6), $M$ reads the response, and forwards the message to $C$.

## 3 MATL as a Logic of Authentication

In this section we briefly introduce MATL and show how MATL can be used for security protocols. The intuitive idea underlying MATL is to model principals engaged in a protocol session as finite state processes. We build the notion of principal incrementally over the notion of process. Suppose we have a set $I$ of principals. Each principal is seen as a process having beliefs about (itself and) other principals. We adopt the usual syntax for beliefs: $B_i\phi$ means that principal $i$ believes $\phi$, and $\phi$ is a belief of $i$. $B_i$ is the belief operator for $i$. The idea is then to associate to each (level of) nesting of belief operators a *process evolving over time*, each of which intuitively correspond to a "view" about that process.

**View Structure**. Let $B = \{\text{B}_1, ..., \text{B}_n\}$, where each index $1, ..., n \in I$ corresponds to a principal. Let $B^*$ denote the set of finite strings of the form $\text{B}_1, ..., \text{B}_n$ with $\text{B}_i \in B$. We call any $\alpha \in B^*$, a *view*. Each view in $B^*$ corresponds to a possible nesting of belief operators. We also allow for the empty string, $\epsilon$. The intuition is that $\epsilon$ represents the view of an external observer (e.g., the designer) which, from the outside, "sees" the behavior of the overall protocol.

*Example 1.* Figure 2 depicts the structure of views for the Lu and Smolka protocol. The beliefs of principal $C$ correspond to the view $\text{B}_C$ and are modeled by a process playing the cardholder's role in the protocol. The beliefs of principals $M$ (the view $\text{B}_M$) and $P$ (the view $\text{B}_P$) are modeled similarly. The beliefs that $C$ has about (the behavior of) principal $M$ correspond to the view $\text{B}_C\text{B}_M$ and are modeled by a process playing $M$'s role in the protocol. Things work in the same way for any arbitrary nesting of belief operators.

**Language**. We associate a language $\mathcal{L}_\alpha$ to each view $\alpha \in B^*$. Intuitively, each $\mathcal{L}_\alpha$ is the language used to express what is true (and false) about the process of view $\alpha$. We employ the Computational Tree Logic (CTL) [12], a well known propositional branching-time temporal logic widely used in formal verification. For each $\alpha$, let $P_\alpha$ be a set of propositional atoms. Each $P_\alpha$ allows for the definition of a different language, called a MATL language (on $P_\alpha$). A MATL language $\mathcal{L}_\alpha$ on $P_\alpha$ is the smallest CTL language containing the set of propositional atoms $P_\alpha$ and the belief atoms $B_i\phi$, for any formula $\phi$ of $\mathcal{L}_{\alpha B_i}$. In particular, $\mathcal{L}_\epsilon$ is used to speak about the whole protocol. The language $\mathcal{L}_{B_i}$ ($\mathcal{L}_{B_j}$) is the language adopted to represent $i$'s ($j$'s) beliefs. $i$'s beliefs about $j$'s beliefs are specified by the language of the view $\text{B}_i\text{B}_j$. Given a family $\{P_\alpha\}$ of sets of propositional atoms, the family of MATL languages on $\{P_\alpha\}$ is the family of CTL languages $\{\mathcal{L}_\alpha\}$. We write $\alpha : \phi$ (called labeled formula) to mean that $\phi$ is a formula of $\mathcal{L}_\alpha$. For instance, the formula $\mathsf{AG}\,(p \rightarrow B_i\neg q) \in \mathcal{L}_\epsilon$, (denoted by $\epsilon : \mathsf{AG}\,(p \rightarrow B_i\neg q)$), intuitively means that in every future state (the CTL operator $\mathsf{AG}$), if $p$ is true then principal $i$ believes $q$ is false.

The next step is the definition of an appropriate $\{P_\alpha\}$ in order to represent the usual propositions of a logic of authentication as in [1,8,18]. First of all, a logic of authentication is a logic of belief, i.e. it has formulae as $P$ *believes* $\phi$. Such formulae have a one-to-one mapping with MATL formulae as $B_P\phi$ (see

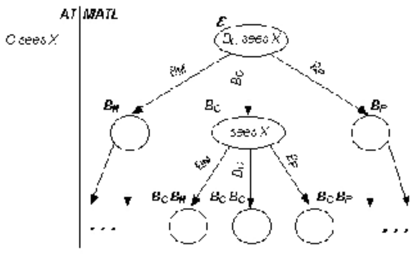

**Fig. 2.** The structure of views for the Lu and Smolka protocol and the proposition *C sees X* in MATL

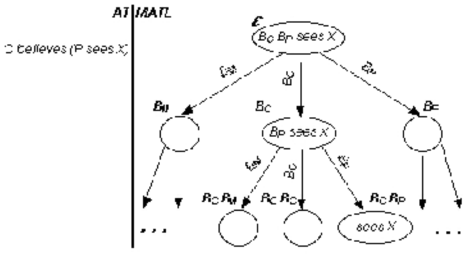

**Fig. 3.** The structure of views for the Lu and Smolka protocol and the proposition *C believes P sees X* in MATL

Figure 3). Furthermore, a logic of authentication has propositions about which (fragments of) messages a principal (say $P$) sends or receives. For instance, in [1] they are expressed by propositions as $P$ *said* $X$ and $P$ *sees* $X$ (where $X$ is a fragment of message). In MATL, such notions can be easily expressed by formulae as $B_P said\ X$ and $B_P sees\ X$ (see Figure 2). This means that we need to introduce the propositional atoms *said* $X$ and *sees* $X$ in $P_{B_P}$. A logic of authentication also has propositions as $fresh(X)$ that expresses the freshness of (fragments of) messages. Intuitive meaning is that $X$ has been generated during the current protocol session. In MATL, we introduce the propositional atom $fresh(X)$. Usually, a logic of authentication also has propositions such as $P$ *says* $X$ to express that principal $P$ has sent $X$ recently. This can be expressed in MATL by the formula $B_P says\ X$. Propositions of the form $pubk_P\, K$ and $prik_{P,Q}\, K^{-1}$ mean that $K$ is the public key of $P$ and $K^{-1}$ the corresponding secret key. They can be directly added as propositional atoms to the languages of MATL.

*Example 2.* We can set the atoms $P_\alpha$ for the views $B_C$, $B_M$ and $B_P$ as follows:

$$P_{B_C} = \left\{\begin{array}{l} said\, TIDR, \\ sees\, S_{K_M{}^{-1}}\{TID\}, \\ said\, S_{K_C{}^{-1}}\{TID\}, \{TID, PA\}_{K_m}, S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}, \\ fresh\, S_{K_C{}^{-1}}\{TID\}, \{TID, PA\}_{K_m}, S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}, \\ fresh\, TID, \\ pubk_M\, K_m, \\ \ldots \end{array}\right\}$$

$$P_{B_M} = \left\{\begin{array}{l} sees\, TIDR, \\ said\, S_{K_M{}^{-1}}\{TID\}, \\ sees\, S_{K_C{}^{-1}}\{TID\}, \{TID, PA\}_{K_m}, S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}, \\ fresh\, S_{K_C{}^{-1}}\{TID\}, \{TID, PA\}_{K_m}, S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}, \\ pubk_P\, K_p, \\ \ldots \end{array}\right\}$$

$$P_{B_P} = \left\{ \begin{array}{l} \textit{sees}\, S_{K_C^{-1}}\{\{TID, PY, CA\}_{K_p}\}, S_{K_M^{-1}}\{TID\}, \{TID, AA, MA\}_{K_p}, \\ \textit{said}\, S_{K_P^{-1}}\{TID, Tr\}, \\ \textit{fresh}\, TID, \\ \textit{prik}_P\, K_P^{-1}, \\ ... \end{array} \right\}$$

For instance, the atom *said* $TIDR$ in view $B_C$ represents $C$ sending $TIDR$ to $M$ (Message 1 of the Lu and Smolka protocol). The atomic propositions of the other views can be defined similarly. Since each view $\alpha B_i$ (with $i = C, M, P$) models the (beliefs about the) behavior of principal $i$, the set of atomic propositions will be that of view $B_i$ (see [5]).

**Semantics**. To understand the semantics of the family of languages $\{\mathcal{L}_\alpha\}_{\alpha \in B^*}$ (hereafter we drop the subscript), we need to understand two key facts. On the one hand the semantics of formulae depend on the view. For instance, the formula $p$ in the view $B_i$ expresses the fact that $i$ believes that $p$ is true. The same formula in the view $B_j$ expresses the fact that $j$ believes that $p$ is true. As a consequence, the semantics associates *locally* to each view $\alpha$ a set of pairs $\langle m, s\rangle$, where: $m = \langle S, J, R, L\rangle$ is a CTL structure, with $S$ a set of states, $J \subseteq S$ the set of *initial states*, $R$ the transition relation, and $L : S \to \mathcal{P}(P)$ the *labeling function*; and $s$ is a *reachable state* of $m$ (a state $s$ of a CTL structure is said to be reachable if there is a path leading from an initial state of the CTL structure to state $s$). On the other hand there are formulae in different views which have the same intended meaning. For instance $B_j p$ in view $B_i$, and $p$ in view $B_i B_j$ both mean that $i$ believes that $j$ believes that $p$ is true. This implies that only certain interpretations of different views are *compatible* with each other, and these are those which agree on the truth values of the formulae with the same intended meaning. To capture this notion of compatibility we introduce the notion of chain.

**Definition 1 (Chain).** *Let $\alpha$ be any view, a $\alpha-$chain $c$ is a finite sequence $\langle c_\epsilon, ..., c_\beta, ..., c_\alpha\rangle$, where $c_\beta = \langle m, s\rangle$ is an interpretation for $\mathcal{L}_\beta$ and $\beta$ is a prefix of $\alpha$ (i.e., $\alpha = \beta\gamma$ for some index $\gamma \in B^*$). A* compatibility relation $C$ on $\{\mathcal{L}_\alpha\}$ *is a set of $\alpha-$chains, for every $\alpha$.*

Intuitively, $C$ will contain all those $c$'s whose elements $c_\alpha$, $c_\beta$ (where $\alpha, \beta$ are two views in $B^*$) assign the same truth values to the formulae with the same intended meaning.

*Example 3.* Figure 4 shows some possible chains of the MATL structure for the Lu and Smolka protocol. The boxes in each view represent interpretations of the language associated with the corresponding view. The figure shows an interpretation for view $\epsilon$, two interpretations for view $B_M$ and two interpretations for view $B_M B_P$. Links connecting boxes in different views represent $B_M B_P-$chains. Figure 4 shows three $B_M B_P-$chains, $c = \langle c_\epsilon, c_{B_M}, c_{B_M B_P}\rangle$, $c' = \langle c'_\epsilon, c'_{B_M}, c'_{B_M B_P}\rangle$ and $c'' = \langle c''_\epsilon, c''_{B_M}, c''_{B_M B_P}\rangle$, where $c_\epsilon = c'_\epsilon = c''_\epsilon$, $c'_{B_M} = c''_{B_M}$ and $c_{B_M B_P} = c''_{B_M B_P}$. Let us assume that each interpretation satisfies the formula written close to it in figure. Therefore, the interpretation labeled $c_{B_M}$ satisfies the formula $B_P$*said* $X$. The intended meaning of this formula in view $B_M$ is that $M$

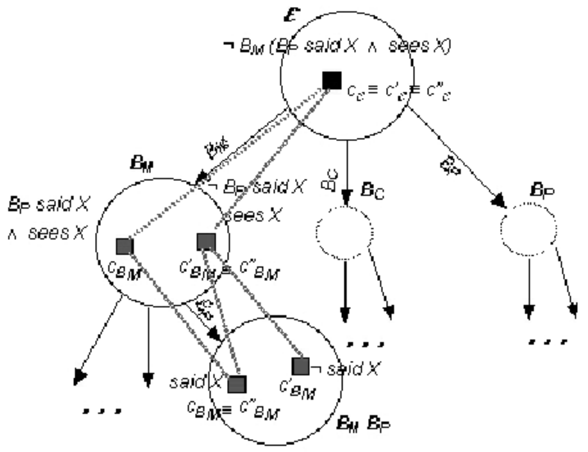

**Fig. 4.** Some chains of the Lu and Smolka protocol

believes that $P$ believes it has sent message $X$ to $M$. The formula *said* $X$ in view $_{B_M B_P}$ has the same intended meaning, as $_{B_M B_P}$ models the beliefs of $P$ seen from (the beliefs of) $M$. Therefore any $_{B_M B_P}-$chain passing through the interpretation $c'_{B_M}$ must reach an interpretation in $_{B_M B_P}$ which satisfies the argument *said* $X$, as shown in the figure.

Let us now define the notion of satisfiability. We start with satisfiability local to views (first step) and suppose that for each view $\alpha$ there is a satisfiability relation between CTL structures and formulae of $\mathcal{L}_\alpha$. With an abuse of notation, we denote all these satisfiability relations with the same symbol $\models$. The context always makes clear which relation we mean. The second step is to define (global) satisfiability taking into account chains. To do this we need some further notation. Let $\models$ denote satisfiability also on chains. For any $\alpha-$chain $c$ and for any formula in $\mathcal{L}_\beta$, satisfiability relation $\models$ is defined only when either $\alpha$ is a prefix of $\beta$ or $\beta$ is a prefix of $\alpha$. (i.e., when either $\alpha = \beta\gamma$ or $\beta = \alpha\gamma$). If $\alpha = \beta\gamma$ then $c_\beta \models \phi$ iff $\phi$ is true in $c_\beta$. If $\beta = \alpha\gamma$ then $c_\beta \models \phi$ for any $\phi$. In other words, if a chain stops at a given level (e.g. $\alpha$), then it satisfies every formula of the views (e.g., $\alpha\gamma$) which are below that level in the tree. Let us extend the satisfiability relation to sets of formulae: $x \models Y$ if and only if for any $y \in Y$, $x \models y$.

We are now ready to define the notion of model for MATL (called MATL structure), and then that of satisfiability between MATL structures and formulae of a view.

**Definition 2 (MATL structure).** *A nonempty compatibility relation $C$ for a family of MATL languages on $\{P_\alpha\}$ is a* MATL structure *on $\{P_\alpha\}$ if for any $\alpha\beta-$chain $c \in C$,*

1. $c_\alpha \models B_i\phi$ *iff for every $\alpha\gamma-$chain $c' \in C$, $c'_\alpha = c_\alpha$ implies $c'_{\alpha B_i} \models \phi$;*

*2. if $c_\alpha = \langle m, s\rangle$, then for any state $s'$ of $m$, there is a $\alpha\beta-$chain $c' \in C$ such that $c'_\alpha = \langle m, s'\rangle$.*

Briefly: the nonemptyness condition for $C$ guarantees that the external observer has a consistent view of the world. The *only if* part in condition 1 guarantees that each view has correct beliefs, i.e., any time $B_i\phi$ holds at a view then $\phi$ holds in the view one level down in the chain. The *if* part is the dual property and ensures the completeness of each view. Condition 2 can be understood on the basis of two crucial observations, concerning the mutual nesting of CTL operators and belief operators. The first, concerning the nesting of CTL operators inside belief operators is that $c_\alpha \models \phi$ is computed using the notion of satisfiability in a CTL structure. Therefore, a chain links the fact that a belief atom holds in a state of a CTL structure in one view with the fact that its argument holds in a state of a CTL structure in the view below. The second observation concerns the nesting of belief operators inside temporal operators (temporal operators which involve no belief atoms are treated as in CTL structures, i.e., without jumping among views). Consider for instance the formula $\mathsf{EX}\, B_i p$. To assess the truth of $\mathsf{EX}\, B_i p$ we need to be able to assess the truth of $B_i p$ in some (reachable) next state $s'$ (the CTL operator $\mathsf{EX}$) of the CTL structure we are considering, e.g., $\langle\langle S, J, R, L\rangle, s\rangle$. The only way to establish this is to request that in $s'$ we have a chain $c'$ which gives access to a CTL structure in the view below. Given the fact that chains connect CTL structures only for what holds in their (reachable) states, the only solution is to request that $s'$ is the state component of the structure $c'_\epsilon = \langle\langle S, J, R, L\rangle, s'\rangle$ with $c' \in C$. Given the fact that temporal operators allow us to state facts about all the states in a CTL structure, this operation must be repeated for each state $s \in S$. But this is exactly what Item 2 of Definition 2 says.

*Example 4.* Consider again Figure 4. The formula $\neg B_M(B_P \mathit{said}\ X \wedge \mathit{sees}\ X)$ is satisfied by the interpretation in view $\epsilon$. By Item 1 of Definition 2 (only if direction), there must be a chain starting from the interpretation in view $\epsilon$ whose component for view $B_M$ does not satisfy the argument of the belief, i.e., $B_P \mathit{said}\ X \wedge \mathit{sees}\ X$. This is indeed the case for both $c'$ and $c''$, as $c'_{B_M} = c''_{B_M}$ and both of them do not satisfy $B_P \mathit{said}\ X$. This is due to the fact that there exists a chain whose component for view $B_M B_P$ does not satisfy the argument of the belief, i.e., $\mathit{said}\ X$. This is indeed the case for $c'$ as $c'_{B_M B_P}$ does not satisfy $\mathit{said}\ X$. Assume now that $c$ is the only chain passing through interpretation $c_{B_M}$. The component for view $B_M B_P$ of chain $c$ passing through that interpretation, i.e., $c_{B_M B_P}$, satisfies the argument $\mathit{said}\ X$. Thus, by Item 1 of Definition 2 (if direction), $c_{B_M}$ must satisfy $B_P \mathit{said}\ X$, as shown in Figure 4.

Given a MATL structure $C$, a formula $\phi$ and a view $\alpha$, $C \models \alpha : \phi$ is read as $\phi$ is true in $C$ (or equivalently, $\phi$ holds in $C$, or $\phi$ is satisfied by $C$) at view $\alpha$, and it is defined as follows:

$$C \models \alpha : \phi \text{ iff for all } c \in C \text{ s.t. } c_\alpha = \langle\langle S, J, R, L\rangle, s\rangle \text{ and } s \in J,\ c_\alpha \models \phi \qquad (1)$$

The intuition is that in order to check the satisfiability of $\phi$ at the view $\alpha$ we need to check all the interpretations of $\mathcal{L}_\alpha$ allowed by the compatibility imposed by the chains we are considering.

For any set of labeled formulae $\Gamma$, let $\Gamma_\alpha$ denote the set $\{\phi \mid \alpha : \phi \in \Gamma\}$.

**Definition 3 (MATL Logical consequence).** *A set of labeled formulae $\Gamma$ logically entails $\alpha : \phi$, in symbols $\Gamma \models \alpha : \phi$, if for every MATL structure $C$ and every $\alpha\beta-$chain $c \in C$, if for every $\gamma$ prefix of $\alpha$, $c_\gamma \models \Gamma_\gamma$ then $c_\alpha \models \phi$. A labeled formula is* valid *if it is a logical consequence of the empty set.*

Notice that MATL structures define a logic where each $B_i$ has the same strength as a modal operator in the multimodal logic $K(m)$, where $m$ is the number of principals.

**Axioms**. MATL is expressive enough to be used as a logic of authentication. Furthermore, it has a temporal component that usually is not present in the other logics of authentication (e.g., see [1,8,18]). In order to show how the properties of security protocols can be expressed within MATL, we shall now impose some constraints to the models in order to capture the intended behavior of a protocol. These constraints can be formalized with a set of sound axioms. This is similar to what happens with several logics of authentications (see for example [1,18]). Indeed MATL encompasses such logics. Here, for the sake of readability, we show how it is possible to translate in MATL some of the most significant axioms that has been proposed in most logics of authentication.

As a first example, let us consider the *message meaning* axioms. Usually, such axioms correspond to the following schema:

$$shk_{P,Q}K \wedge P\ sees\{X\}_K \rightarrow Q\ said\ X$$

Intuitively, it means that when a principal $P$ receives a (fragment of) message encrypted with $K$, and $K$ is a key shared by $P$ and $Q$, then it is possible to conclude that the message comes from $Q$. The above axiom schema can be formulated in MATL as follows:

$$P : shk_{P,Q}K \wedge sees\{X\}_K \rightarrow B_Q said X \tag{2}$$

where with $P : Ax$ we also emphasize which view ($P$) the axiom $Ax$ belongs to. Message meaning is often used with the *nonce verification*, that has the following schema:

$$Q\ said\ X \wedge fresh(X) \rightarrow Q\ says\ X$$

This schema expresses that when a principal $Q$ has sent $X$ (i.e., $Q$ *said* $X$) recently (i.e., $fresh(X)$), then we can assert that $Q$ *says* $X$. In MATL, this becomes

$$P : B_Q said X \wedge fresh(X) \rightarrow B_Q says X \tag{3}$$

As a consequence, it is important to establish whether a fragment of message is fresh. The following axioms help on this task:

$$fresh(X_i) \rightarrow fresh(X_1, \ldots, X_n)$$
$$fresh(X) \rightarrow fresh(\{X\}_K)$$

Intuitively, they mean that when a fragment is fresh, then the message that contains such a fragment (the encryption of the fragment) is fresh as well. In MATL, they can be inserted in the appropriate views without modification. Another important set of axioms establishes how a message can be decomposed. For instance, in [1] we have the following schemata:

$$P\ sees(X_1 \ldots X_n) \rightarrow P\ sees\ X_i$$
$$P\ sees\ \{X\}_K \wedge P\ has\ K \rightarrow P\ sees\ X$$

For instance, the intuitive meaning of the second schema is that a principal can decrypt a message encrypted with a given key when it knows such a key. Once again, in MATL the above axiom schemata can be inserted in a view without modification.

## 4 MATL as a Logic for Payment Protocols

In this section we discuss the main differences between our logic and previous logics of authentication. We especially take into account those differences which arise when the focus is on payment protocols. For what concerns atomic propositions, we introduce atoms of the form $rec\, X$ and $send_P\, X$ (where $X$ can be a full message of a given protocol but not a fragment of message) that represent the communicative act of receiving $X$ and sending $X$ to $P$, respectively. This allows us to take into account the temporal aspects of a protocol. Indeed, such propositions represent when a principal actually receives or sends a message during a session. Therefore, we are able to recognize when such events occur by looking at the sequence of states. This is different from the notion of what are the fragments of messages that a principal has (atom *sees*) or uses when composing its messages (atom *said*). Furthermore, the atom *sees* represents both the notion of *possessing* (what a principal has because it is initially available or newly generated) and *seeing* (what has been obtained from a message sent by another principal). Notice that, for instance, Abadi and Tuttle [1] make for keys a different choice. Indeed, when they have a key $K$, they distinguish the proposition $P\ sees\ K$ by $P\ has\ K$. In our opinion this difference does not make much sense, as in their logic a principal happens to have all the keys it sees (e.g., see [18] where the authors introduce the axiom schema: $P\ sees\ K \leftrightarrow P\ has\ K$). The above facts have the following consequences in the corresponding axiom system. The following axiom schemata relate sent (received) messages to what a principal says (sees).

$$P : send_Q X \rightarrow said\ X \tag{4}$$
$$P : rec\ X \rightarrow sees\ X \tag{5}$$

The next axiom schemata capture the idea that a principal sees what it previously said or what it says.

$$P : said\ X \rightarrow sees\ X \tag{6}$$
$$P : says\ X \rightarrow sees\ X \tag{7}$$

The following schema represents the fact that a principal can compose a new message starting from the (fragment of) messages it already sees.

$$P : sees\ X_1 \wedge \ldots \wedge sees\ X_n \rightarrow sees(X_1 \ldots X_n) \tag{8}$$

For what concerns payment protocols, we have to take into account hash functions and signatures, as well. First of all, we assume a signature as a reliable schema without considering how such a schema is[1]. This allows us to focus on whether the protocol is trustworthy. This is similar to what is usually assumed for encryption algorithms in such a kind of verification. The corresponding axiom schemata are the following:

$$P : fresh(X) \rightarrow fresh(H(X)) \tag{9}$$
$$P : fresh(X) \rightarrow fresh(S_{K_Q^{-1}}\{X\}) \tag{10}$$
$$P : sees\ X \rightarrow sees\ H(X) \tag{11}$$
$$P : sees\ X \wedge sees\ {K_P}^{-1} \wedge prik_P\ {K_P}^{-1} \rightarrow sees\ S_{K_P^{-1}}\{X\} \tag{12}$$
$$P : sees\ X \wedge sees\ K_Q \wedge pubk_Q\ K_Q \rightarrow sees\ \{X\}_{K_Q} \tag{13}$$

where $Q$ and $P$ can be both substituted with the same principal or with different ones. The above schemata are the obvious extension to hash functions and signatures of the axioms about the freshness and the capability of composing messages.

The following axiom schema represents the capability of extracting the original message from its signed version when the principal has the corresponding public key.

$$P : sees\ S_{K_Q^{-1}}\{X\} \wedge sees\ K_Q \rightarrow sees\ X \tag{14}$$

The next schema corresponds to the *message meaning* axiom schema for signed messages.

$$P : sees\ S_{K_Q^{-1}}\{X\} \wedge pubk_Q\ K_Q \rightarrow B_Q said X \tag{15}$$

Intuitively, it means that when a principal sees a message signed with the key of $Q$, then it believes that Q said such a message.

## 5 Specifications for Payment Protocol

Payment protocols are intended to protect business transactions, therefore they must achieve various security requirements. In literature, there's no common agreement about this issue. There are indeed different definitions of security requirements for payment protocols. In this work, we follow the taxonomy introduced in [2], where requirements are classified with respect to the security needs of each actor involved in an electronic payment. Indeed, [2] proposes a number of requirements expected by the customer, by the merchant, and by the payment gateway. When we formalize security requirements with MATL, it is

[1] The most common schema is the following: $S_{K_Q^{-1}}\{X\} = (X, H(X)_{K_Q^{-1}})$

indeed quite natural to state a specification for a principal as a formula in the view of the principal itself.

Customer Requirements

*Proof of Transaction Authorization by Payment Gateway.* The customer must have a proof that the payment gateway authorized the transaction.

*Example 5.* In the case of the Lu and Smolka protocol, the above requirement can be written as follows:

$$C : \mathsf{AG}\,(\mathit{sees}\ \mathit{Tr} \rightarrow B_P \mathit{says}\ \mathit{Tr}) \tag{16}$$

Intuitively, it asserts that in the view of the customer $C$, in every future state, if $C$ receives *Tr*, then $C$ must believe that $P$ has recently sent this information.

*Receipt from Merchant.* The customer must have a proof that the merchant who has made the offer has received payment and promised to deliver the good/service.

*Example 6.* For the Lu and Smolka protocol, such a requirement becomes:

$$C : \mathsf{AG}\,(\mathit{sees}\ \mathit{Tr} \rightarrow B_M \mathit{says}\ \mathit{Tr}) \tag{17}$$

This means that, in the view of $C$, in every future state, if $C$ receives the gateway answer *Tr*, then it must believe that the merchant $M$ has recently sent it (remember that in the Lu-Smolka protocol the answer is forwarded to the customer by the merchant).

Merchant Requirements

*Proof of Transaction Authorization by Payment Gateway.* The merchant needs an unforgeable proof that the payment gateway has authorized the payment.

*Example 7.* For the Lu and Smolka protocol, we have

$$M : \mathsf{AG}\,(\mathit{sees}\ S_{K_P{}^{-1}}\{\mathit{TID}, \mathit{Tr}\} \rightarrow B_P \mathit{says}\ \mathit{Tr}) \tag{18}$$

Intuitively, this formula means that in the view of $M$, in every future state, if the merchant $M$ receives $S_{K_P{}^{-1}}\{TID, Tr\}$, then it must believe that the gateway $P$ has recently sent $Tr$.

*Proof of Transaction Authorization by Customer.* The merchant needs an unforgeable proof that the customer has authorized the payment before that the merchant receives the transaction authorization from the payment gateway.

*Example 8.* For the Lu and Smolka protocol, such a requirement becomes:

$$\begin{aligned} M : \mathsf{A}\,((B_C \mathit{says}\ PA)\ \mathcal{B}& \\ (\mathit{send}_P\ S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}, S_{K_M{}^{-1}}\{TID\}, \{TID, AA, MA\}_{K_p}))& \end{aligned} \tag{19}$$

Intuitively[2], the specification above asserts that the merchant $M$ must believe that, in every future path, the customer $C$ will have recently sent $PA$ before the merchant $M$ sends the *Authorization & Capture Request* message (message 3 of the protocol).

[2] $\mathsf{A}\,(p\ \mathcal{B}\, q)$ means that in every future path, $p$ will be true before $q$ holds.

*Confidentiality of Customer Sensitive Information.* The customer desires privacy of information related to its account.

*Example 9.* In order to formalize the fact that the merchant must not know the Customer Account (in particular its Credit Card Number) in the Lu and Smolka protocol, we write:

$$M : \mathsf{AG}\,(\neg \mathit{sees}\ \mathit{CA}) \tag{20}$$

The above formula states that in the view of the merchant $M$, in every future state, $M$ never sees the *Customer Account*.

Payment Gateway Requirements

*Proof of Transaction Authorization by Customer.* When the payment gateway debits a certain amount to the customer, the payment gateway must have an unforgeable proof that the customer has authorized this payment.

*Example 10.* For the Lu-Smolka protocol, such a requirement becomes:

$$P : \mathsf{AG}\,(\mathit{sees}\ S_{K_P{}^{-1}}\{\mathit{TID}, \mathit{Tr}\} \rightarrow B_C \mathit{says}\ \mathit{PY}) \tag{21}$$

This formula asserts that in the view of the gateway $P$, in every future state, if $P$ receives $S_{K_P{}^{-1}}\{TID, Tr\}$ then it must believe that the customer $C$ has recently sent $PY$.

*Proof of Transaction Authorization by Merchant.* When the payment gateway authorizes a payment to a certain merchant, the payment gateway must have an unforgeable proof that this merchant has required that this payment must be made to him.

*Example 11.* For the Lu-Smolka protocol, we have:

$$P : \mathsf{AG}\,(\mathit{sees}\ S_{K_P{}^{-1}}\{\mathit{TID}, \mathit{Tr}\} \rightarrow B_M \mathit{says}\ \mathit{AA}) \tag{22}$$

This formula asserts that in the view of the gateway $P$, in every future state, if $P$ receives $S_{K_P{}^{-1}}\{TID, Tr\}$ then it must believe that the merchant $M$ has recently sent $AA$.

## 6 Verification of the Lu and Smolka Protocol

The verification of the Lu and Smolka protocol with NuMAS requires to model each view as a finite state machine, to specify the security requirements in the appropriate views (see Section 5), and to check the specifications by means of the model checker. Modeling a view amounts at establishing what are the propositional atoms and the beliefs of the a view, and how they vary over time. The temporal evolution of the propositional atoms $\mathit{send}_P\, X$ and $\mathit{rec}\, X$ can be derived directly from the protocol description. The behavior of the other atoms derives from the axioms described in Section 3. The reader can refer to [4,5] for a description of how the views can be modeled in NuMAS and to [3] for a description of the symbolic model checking algorithm for MATL. Here we report

**Table 1.** The verification of the Lu and Smolka protocol

| View | Specification | Result |
|---|---|---|
| C | (16) | False |
| C | (17) | True |
| M | (18) | False |
| M | (19) | True |
| M | (20) | True |
| P | (21) | False |
| P | (22) | False |

the results of the verification (that required 0.3 sec. on a PC equipped with a Pentium III and 512 MB RAM). The results are summarized in Table 1. Notice that some expected security requirements concerning *transaction authorizations* are not satisfied. This means that the protocol suffers from weaknesses which make it vulnerable to attacks by dishonest entities. Indeed, following we describe a possible attack reported in [17] (see Figure 5). We write $X \rightarrow I(Y)\ :\ m$ when the dishonest entity intercepts a message $m$ in transit from $X$ to $Y$, preventing it from being received; we write $I(X) \rightarrow Y\ :\ m$ when the dishonest entity sends a message $m$ to $Y$, impersonating $X$. The attack proceeds as follows. A

$(\alpha.1)$ $C \rightarrow\ I\ :\ TIDR$
  $(\beta.1)$ $I(C) \rightarrow\ M\ :\ TIDR$
  $(\beta.2)$ $M \rightarrow\ I(C)\ :\ S_{K_M{}^{-1}}\{TID\}$
$(\alpha.2)$ $I \rightarrow\ C\ :\ S_{K_I{}^{-1}}\{TID\}$
$(\alpha.3)$ $C \rightarrow\ I\ :\ S_{K_C{}^{-1}}\{TID\}, \{TID, PA\}_{K_i}, S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}$
  $(\beta.3)$ $I(C) \rightarrow\ M\ :\ S_{K_C{}^{-1}}\{TID\}, \{TID, PA\}_{K_m}, S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}$
  $(\beta.4)$ $M \rightarrow\ P\ :\ S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}, S_{K_M{}^{-1}}\{TID\}, \{TID, AA, MA\}_{K_p}$
  $(\beta.5)$ $P \rightarrow\ M\ :\ S_{K_P{}^{-1}}\{TID, Tr\}$
  $(\beta.6)$ $M \rightarrow\ I(C)\ :\ S_{K_P{}^{-1}}\{TID, Tr\}$

**Fig. 5.** An attack on the protocol Lu-Smolka protocol

dishonest merchant $I$ waits for a buyer $C$ to start a session $\alpha$ with it. At this point, $I$ opens a parallel session $\beta$, impersonating the client $C$ towards another merchant $M$. The $TID$ provided by $M$ in step $(\beta.2)$ is sent by $I$ to $C$ as the transaction identifier of session $\alpha$. In this way $I$ obtains a message having the two components with the $TID$ signed by $C$ in step $(\alpha.3)$. By means of these signed fragments, $I$ can send the message in step $(\beta.3)$ to $M$, masquerading as $C$. After receiving the above message, in step $(\beta.4)$, $M$ requires the payment authorization to the gateway $P$. For this purpose, it sends a message with the same fake $S_{K_C{}^{-1}}\{\{TID, PY, CA\}_{K_p}\}$ it received in the previous step. Notice that $M$ blindly forwards this fragment, since it is encrypted with $P$'s public key.

The gateway is unable to detect the cheat and thus authorizes the payment in favor of $M$, sending the message in step ($\beta$.5). Finally, $M$ forwards the received message to $C$, but $I$ intercepts and removes it, so that $C$ has no suspects of the fraud. In this way, the session $\beta$ ends, allowing $I$ to masquerade as $C$ towards $M$. As a consequence of the above attack on a commercial transaction, a dishonest buyer $I$ succeeds in debiting to another buyer $C$ a purchase that $C$ in fact has never performed. Moreover, if the good is delivered via Internet (e.g., the content of a web-page) or $I$ has altered the shipping address in the (often insecure) phase before the invocation of the protocol, $I$ is able to obtain that good without paying it. As a final remark, notice that when $M$ receives the message in step (3), it is not able to deduce the identity of the intended receiver. This problem has implications on the non-repudiation requirement, and exists in the original version of SET as well [19]. However, in SET, the presence of the merchant identifier in message (4) allows the gateway $P$ to deduce the real identity of the intended merchant and, thus, to avoid the above fraud. This is however not true of the Lu and Smolka protocol, where Specification (21) is not satisfied and the attack in Figure 5 can occur.

## 7 Conclusions

In this paper we have described MATL, a logic of belief and time that can be used for the verification of payment protocols as well as other kinds of security protocols. The verification is fulfilled by means of NuMAS, a symbolic model checker based on MATL. This kind of verification has been applied to the Lu and Smolka protocol, a variant of SET. Even if this protocol was considered secure, we have discovered that the protocol does not satisfy some important requirements as the proof of transaction authorization. This kind of verification has been applied to SSL 3.0 and 3KP as well. For these protocols, all the security requirements are satisfied. Such verifications require few seconds with a normally equipped PC. For lack of space, in this paper we have described only the verification of the Lu and Smolka protocol.

## Acknowledgments

We would like to thank Fausto Giunchiglia for his support to the present research. We also owe a lot to all the past and current members of the Information Systems Group at the Istituto di Informatica – University of Ancona. In particular, we would like to mention Johann Dimarti, Loïc Jay, and Luca Roganti who participated to the verification of the protocol described in this paper.

## References

1. M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216, 1991. 311, 312, 314, 315, 319, 320

2. M. Bellare, J. A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP-a family of secure electronic payment protocols. In *Proc. of the First USENIX Workshop of Electronic Commerce*, Berkeley, CA, USA, 1995. USENIX Assoc. 321
3. M. Benerecetti and A. Cimatti. Symbolic Model Checking for Multi–Agent Systems. In *CLIMA-2001, Workshop on Computational Logic in Multi-Agent Systems*, Paphos, Cyprus, December 1st 2001. Co-located with ICLP'01. 312, 323
4. M. Benerecetti and F. Giunchiglia. Model checking security protocols using a logic of belief. In *Proc. of the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, Berlin, Germany, March 27th - April 1st 2000. 311, 323
5. M. Benerecetti, F. Giunchiglia, M. Panti, and L. Spalazzi. A Logic of Belief and a Model Checking Algorithm for Security Protocols. In *Proc. of IFIP TC6/WG6.1 International Conference FORTE/ PSTV 2000*, Dordrecht, The Netherlands, 2000. Kluwer Academic Publisher. 311, 316, 323
6. M. Benerecetti, F. Giunchiglia, and L. Serafini. Model Checking Multiagent Systems. *Journal of Logic and Computation*, 8(3):401–423, June 1998. 311, 312
7. D. Bolignano. Towards the Formal Verification of Electronic Commerce Protocols. In *Proceedings of* $10^{th}$ *Computer Security Foundations Workshop*, pages 133–146, 1997. 311
8. Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990. 311, 312, 314, 319
9. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new Symbolic Model Verifier. In *Proceedings of the International Conference on Computer-Aided Verification (CAV'99)*, Trento, Italy, July 1999. 312
10. E. Clarke, O. Grumberg, and D. Long. Model Checking. In *Proc. of the International Summer School on Deductive Program Design*, Marktoberdorf, Germany, 1994. 311
11. E. Clarke, S. Jha, and W. Marrero. A machine checkable logic of knowledge for specifying security properties of electronic commerce protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1998. 311
12. E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072, Amsterdam, The Netherland, 1990. Elsevier Science Publishers. 314
13. N. Heintze, J. D. Tygar, J. Wing, and H. C. Wong. Model Checking Electronic Commerce Protocols. In *Proc. of the* $2^{nd}$ *USENIX Workshop on Electronic Commerce*, 1996. 311
14. R. W. Lichota, G. L. Hammonds, and S. H. Brackin. Verifying Cryptographic Protocols for Electronic Commerce. In *Proc. of the* $2^{nd}$ *USENIX Workshop on Electronic Commerce*, pages 53–65, 1996. 311
15. S. Lu and S. A. Smolka. Model Checking the Secure Electronic Transaction (SET) Protocol. In *Proceedings of* $7^{th}$ *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 358–365. IEEE Computer Society, 1999. 311, 312
16. Mastercard and Visa. *SET Secure Electronic Transaction Specification.* Mastercard & Visa, May 1997. Available at http://www.setco.org. 312
17. M. Panti, L. Spalazzi, and S. Tacconi. Verification of Security Properties in Electronic Payment Protocols. In *IFIP WG 1.7 Workshop on Issues in the Theory of Security (WITS '02)*, Portland, Oregon, January 2002. Co-located with POPL'02. 311, 324

18. P. Syverson and P. C. van Oorschot. On Unifying Some Cryptographic Protocol Logics. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 14–28, Oakland, CA, May 1994. IEEE Computer Society Press. 312, 314, 319, 320
19. E. Van Herreweghen. Non-repudiation in SET: Open Issues. In *Proceedings of the 4$^{th}$ Conference on Financial Cryptography*, 2000. 325