A Game-Based Verification of Non-Repudiation and Fair Exchange Protocols

Steve Kremer Jean-François Raskin*

Département d'Informatique - Faculté des Sciences
Université Libre de Bruxelles, Belgium
skremer@ulb.ac.be Jean-Francois.Raskin@ulb.ac.be

Abstract. In this paper, we report on a recent work for the verification of non-repudiation protocols. We propose a verification method based on the idea that non-repudiation protocols are best modeled as games. To formalize this idea, we use alternating transition systems, a game based model, to model protocols and alternating temporal logic, a game based logic, to express requirements that the protocols must ensure. This method is automated by using the model-checker MOCHA, a model-checker that supports the alternating transition systems and the alternating temporal logic. Several optimistic protocols are analyzed using MOCHA.

1 Introduction

Non-repudiation protocols. During the last decade, open networks, above all the Internet, have seen an impressive growth. As a consequence, new security issues, like non-repudiation have to be considered. Repudiation is defined as the *denial* of an entity of having participated in all or part of a communication. Consider for instance the following scenario: Alice wants to send a message to Bob; after having sent the message, Alice may deny having sent it (repudiation of origin), or Bob may deny having received it (repudiation of receipt). Therefore specific protocols have been designed, generating both a non-repudiation of origin (NRO) evidence, destined to Bob, and a non-repudiation of receipt (NRR) evidence, intended to Alice. These evidences are based on digital signatures that provide proofs of the origin of a message or a receipt. In case of a dispute Alice or Bob can present their evidences to an adjudicator, who can take a decision in favor of one of the two entities without ambiguity. The major problem in these protocols is to handle the fact that at one moment an entity will come into an advantageous position. For instance, if Alice starts sending her message, Bob has received all expected information and may stop the protocol being in an advantageous position.

Different solutions have been proposed: they are generally divided into two classes, according to whether they use a trusted third party (TTP) or not. The

* This author was partially supported by a "Crédit aux chercheurs" granted by the Belgian National Fund for Scientific Research.

approach without TTP is either based on a gradual release of knowledge or on a probabilistic protocol. It generally requires that all involved parties have equivalent computational power. In most cases these protocols are inefficient due to the large number of messages that need to be sent. The second approach is the one using a TTP. Thus, the message is first sent to the TTP, who acts as an intermediary to assure delivery. The major problem of this approach is the network and communication bottleneck, created at the TTP. To avoid the performance decrease created by this bottleneck, Asokan et al. introduced the optimistic approach for fair exchanges [2]. In an optimistic protocol one supposes that in general the involved entities are honest and the network is well functioning. The rationale is that the TTP only intervenes in case of a problem (a cheating entity or a network failing a delivery at a critical moment). Afterwards Zhou et al. applied the optimistic approach to the non-repudiation protocols [16]. Optimistic protocols are the ones that received most of the attention in recent literature.

A non-repudiation protocol has to respect several properties. The first one is *fairness*[1]: fairness must ensure that if at least one entity is honest, either both entities receive the expected non-repudiation evidence or none of them receives it. This property can also be split in the following two properties. *Fairness for Alice* ensures that, if Alice is honest, Bob receives the non-repudiation of origin evidence only if Alice receives her non-repudiation of receipt evidence. *Fairness for Bob* ensures that, if Bob is honest, Alice receives her non-repudiation of receipt evidence only if Bob receives his non-repudiation of origin evidence. The fairness requirement is the conjunction of these two properties. Another property we require is *timeliness*: we want that the protocol finishes for each honest player after a finite amount of time. A third property that is desirable but not necessary is viability. A protocol is *viable* if two honest players always succeed in exchanging the expected evidences. In general viability can only be realized by strengthening the requirements we make on the communication channels. We define three classes of channels: unreliable channels, resilient channels and operational channels. No assumptions have to be made about *unreliable channels*: data may be lost. A *resilient channel* delivers data after a finite, but unknown amount of time. Data may be delayed, but will eventually arrive. When using an *operational channel* data arrive before a known, constant amount of time. Operational channels are however rather unrealistic in heterogeneous networks. A formal definition of both these properties and the channels, will be found later.

In comparison to other security issues, such as privacy or authenticity of communications, non-repudiation has not been studied intensively. However, the more studied authentication protocols have shown that the design of security protocols is an error prone process. Some flaws have only been found after years, e.g. the Needham-Schroeder public-key authentication protocol. Hence, a need for formal verification methods has been identified. These methods include an ad-

[1] The term *fairness* will be used in two different contexts in this paper that must not be confused: on one hand it is used to denote the fairness property that a protocol must respect, on the other hand we use it when discussing fair computations in the context of temporal logic.

equate specification, more precise than the traditional, informal and sometimes ambiguous one often used in literature. Another aim is the automatic verification. Several works have been engaged: believe logics, such as BAN [5], general purpose model checkers, e.g. FDR [9], theorem provers as ISABELLE for instance [12] and also special purpose verification tools as the NRL protocol analyzer [11] have successfully been applied to authentication protocols. To the best of our knowledge only four attempts have been made to verify non-repudiation and fair exchange protocols. First works have been done on non-repudiation protocols using CSP [13], where the proofs were generated by hand, and Zhou and Gollman briefly considered using the belief logic SVO [15]. Some work on fair exchange protocols has been realized using the model-checker Murφ [14] as well as the animation tool Possum [4].

Non-repudiation protocols as games. There are some fundamental differences between authentication protocols and exchange protocols, e.g. non-repudiation protocols. Generally one of the most difficult problems in authentication protocols is to deal with the presence of an intruder. In non-repudiation protocols we do not need to model an intruder, but we have to consider that either Alice or Bob, the two entities taking part in the protocol, may *cheat* (cf also [13] and [4]). The most important difference is that exchange protocols, above all optimistic ones, are not linear. Generally authentication protocols are ping pong protocols and only allow very few different traces. On the other hand, non-repudiation and fair exchange protocols are divided in several subprotocols (e.g. a main and a recovery protocol), making branching possible, although they are intended to be executed in a given order by a honest entity. Changing the order of execution could result in subtle errors. This is the reason why we propose a new method for the specification and the verification of exchange protocols. First, we want to model the actions that are possible in the course of the protocol and not stick to a given predefined order of execution. In that way, we give a malicious entity the potential not to follow the protocol, but to construct an attack against the honest entity. Second, we consider the execution of the protocol as a game: each entity (Alice, Bob, TTP) and each communication channel are players. We can think of designing a protocol as finding a *strategy*: the strategy proposed by the protocol has to defend a honest entity against all possible strategies of malicious parties that are trying to cheat. This point of view also allows us to express formally the required properties as strategies. For instance, a property such as fairness for Alice can be expressed as follows: "a coalition of Bob and all the communication channels does not have a strategy to obtain a non-repudiation of origin evidence without Alice having a strategy to obtain a non-repudiation of receipt evidence". Here, we have rephrased the property as the existence of a strategy. The main advantage of modeling such protocols as games is that we *directly* and *formally* take into account the possibility of *adversarial* behaviors. As the communication channels are also modeled as players, they can cooperate with protocol entities. This means that either the channels are not well-functioning or that they are controlled by a player of the coalition. For instance, when a message is lost on an unreliable channel, this can be due to a network failure, or a dishonest player who removed the message by cooperating with the communication channels. As a second example, consider the game view of the viability property: Alice, in cooperation with Bob must have a strategy against the coalition of the communication channels, such that both Alice and Bob possess their expected evidences at the end of the protocol. This example illustrates the facility using the strategies not only to express adversarial behaviors, but also *cooperative* behaviors between several players. The trust in the TTP is modeled by giving the TTP a unique strategy: even if he makes part of one coalition or the other, his behavior is deterministic and thus *cannot choose* to help anyone. By using alternating transition systems and alternating-time temporal logic of Alur et al. [1], we are able to formalize the non-repudiation protocols and their requirements in a direct way.

Structure of the paper. We organize the paper as follows. In section 2, we introduce the *alternating transition systems* and the *alternating-time temporal logic* of Alur et al. [1]. Section 3 shows how exchange protocols, and more specifically non-repudiation protocols, can be modeled *naturally* and *accurately* as games. In section 4, we report on results about the automatic verification of several non-repudiation protocols using the model-checker Mocha. In section 5, we compare our techniques to some related works. Finally, we report on plans for future works in section 6 and draw some conclusions.

2 A Formal Model of Games and its Logic

Alternating transition systems. The formalism that we use to model exchange protocols as games is the model of alternating transition systems, ATS for short. The formal definition of this model is given in [1]. Here we only give an intuitive introduction.

ATS are a game variant of usual Kripke structures. An ATS is composed of a set of states Q that represents the possible game configurations, a finite set of propositions P, a labeling function $L : Q \to 2^P$ that labels states with propositions, a set of players Σ and a game transition function δ. The game transition function defines for every player a and state q the set of choices $\delta(q, a) = \{Q_1, Q_2, \ldots, Q_n\}$, with $Q_i \subseteq Q$, that the player a can make in q. A choice is a set of possible next states. One step of the game at a state q is played as follows : each player $a \in \Sigma$ makes his choice Q_a and the next state q' of the game is the intersection (that is required to be a singleton) of the choices made by all the players of Σ, i.e. $\{q'\} = \bigcap_{a \in \Sigma} Q_a$. A computation is an infinite sequence $\lambda = q_0 q_1 \ldots q_n \ldots$ of states, such that for every $i \geq 0$, there exists $Q_{a_1}, \ldots, Q_{a_n}$ with $\Sigma = \{a_1, \ldots, a_n\}$ and such that for every j, $1 \leq j \leq n$, $Q_{a_j} \in \delta(q_i, a_j)$ and $q_{i+1} = \bigcap_{1 \leq j \leq n} Q_{a_j}$.

Alternating-time temporal logic. We now introduce the alternating-time temporal logic [1], ATL for short. For a set of players $A \subseteq \Sigma$, a set of computations Λ, and a state q, consider the following game between a protagonist and an antagonist. The game starts at state q. At each step, to determine the next state, the protagonist chooses among the choices controlled by the players in the set A, while the antagonist chooses among the remaining choices. If the resulting infinite computation belongs to the set Λ, then the protagonist wins. If the protagonist has a winning strategy, we say that the ATL formula $\langle\langle A\rangle\rangle\Lambda$ is satisfied in state q. Here, $\langle\langle A\rangle\rangle$ is a path quantifier, parameterized by the set A of players, which ranges over all computations that the players in A can force the game into, irrespective of how the players in $\Sigma \setminus A$ proceed. The set Λ is defined using temporal logic formulas. If the reader is familiar with the branching time temporal logics, he may see that the parameterized path quantifier $\langle\langle A\rangle\rangle$ is a generalization of the path quantifiers of CTL: the existential path quantifier $\exists$ corresponds to $\langle\langle\Sigma\rangle\rangle$, and the universal path quantifier $\forall$ corresponds to $\langle\langle\emptyset\rangle\rangle$. We now illustrate the expressive power of ATL. Consider the set of players $\Sigma = \{a, b, c\}$ and the following formulas with their verbal reading:

- $\langle\langle a\rangle\rangle\Diamond p$, player a has a strategy against players b and c to eventually reach a state where the proposition p is true;
- $\neg\langle\langle b, c\rangle\rangle\Box p$, the coalition of players b and c does not have a strategy against a to reach a point where the proposition p will be true for ever;
- $\langle\langle a, b\rangle\rangle\bigcirc\, (p \wedge \neg\langle\langle c\rangle\rangle\Box p)$, a and b can cooperate so that the next state satisfies p and from there c does not have a strategy to impose p for ever.

Those three formulas are a good illustration of the great expressive power of ATL to express cooperative as well as adversarial behaviors between players.

Fairness. As in the usual temporal logic setting, fairness can be used to rule out computations. For example, to evaluate a formula of the form $\langle\langle a\rangle\rangle\Diamond\phi$, we only consider computations where the antagonists, that is the agents in $\Sigma \setminus \{a\}$, respect their fairness constraints. A *fairness constraint* is a function $\gamma : Q \times \Sigma \to 2^{2^Q}$ such that for each $q \in Q$ and $a \in \Sigma$, we have $\gamma(q, a) \subseteq \delta(q, a)$. Given a computation $\lambda = q_0 q_1 \ldots q_n \ldots$, we say that γ is *a-enabled* at position $i \geq 0$ if $\gamma(q_i, a) \neq \emptyset$. We say that γ is *a-taken* at position $i \geq 0$ if there exists a set $Q' \in \gamma(q_i, a)$ such that $q_{i+1} \in Q'$. Finally given a set $A \subseteq \Sigma$, we say that λ is *weakly $\langle\gamma, A\rangle$-fair* [2] if for each agent $a \in A$, either there are infinitely many positions of λ at which γ is not a-enabled, or there are infinitely many positions of λ at which γ is a-taken. To check ATL formulas under weak fairness constraints, we use the script language of MOCHA and a symbolic adaptation of the techniques described in [1].

Game guarded command language. Instead of modeling protocols directly with ATS we will use a more user-oriented notation: a guarded command language "à la Dijkstra". The details about the syntax and semantics of this language (given in terms of ATS) can be found in [6]. Here follows an intuitive presentation. Each player $a \in \Sigma$ disposes of a set of guarded commands of the form

$$guard_\xi(X) \to update_\xi(X, Y') \tag{1}$$

[2] Note that there exists also, as in the usual temporal logic setting a notion of strong fairness, the interested reader is referred to [1] for the definition. We will not need the notion of strong fairness in this paper.

where X denotes the set of all variables and Y' the set of variables, controlled by a in the next state. A computation-step is defined as follows: each player $a \in \Sigma$ chooses one of his commands whose guard evaluates to true, and the next state is obtained by taking the conjunction of the effects of each update part of the commands selected by the players.

3 Formal Modelization of Non-Repudiation Protocols

Modeling of the protocols. Non-repudiation and fair exchange protocols have several particularities that allow us to make some simplifications. As also noted in [4] all messages are generally protected by digital signatures or an equivalent mechanism. Hence we consider that only well formed messages can be sent. Moreover each protocol execution can be uniquely identified by the identity of the participating entities and a special label. When the label contains some well chosen information related to an execution, as proposed for instance in [16], it is possible to show that different parallel executions cannot interfere. Therefore we do not need to verify the execution of several parallel runs.

The notation, classically used in literature to describe cryptographic protocols ($A \to B : m$, to denote that Alice sends a message m to Bob), has several drawbacks. The protocols are presented as a linear sequence of message exchanges, with a predefined order. In the case of optimistic exchange protocols often subprotocols can be invoked at different moments. But, running a subprotocol at a time not foreseen by the designer, may have unexpected side-effects, threatening the security of a protocol if one of the participating entities tries to cheat the other entities. We use the modeling language described above to model the exchange protocols. To execute a given action ξ, $guard_\xi$ must evaluate to true. The guard $guard_\xi$ is used to represent the elements (such as keys, messages, ...) necessary for a participating entity, that is a player in our modelization, to execute the action ξ. The variables controlled by a player represent his current knowledge and thus determine which actions he/she can execute. This specification takes into account all possible executions of subprotocols corresponding to a given initial knowledge, as we do not give any predefined order to these guarded commands. At each point in the protocol execution all messages that could possibly be sent are determined. Thus, the specification enables a malicious entity to choose a different order of execution and construct a possible attack. We model the two players Alice and Bob with this approach.

The TTP is a special player and has to be modeled in a particular way. The TTP must be *impartial*, it may not help one or the other player. To make sure that the TTP does not have a strategy to help one of the players to cheat, we model the TTP such that it is deterministic: at each stage of the execution of the protocol, the TTP executes the action requested by the protocol.

The communication channels are also modeled as players. Each transmission is modeled as a guarded command. We consider three different kinds of communication channels: unreliable, resilient, and operational. To specify *unreliable channels* we add an action that the channel can always take and has no effect, i.e.

this is simply an idle action. As a consequence, a message sent on an unreliable channel may never be delivered. A *resilient channel* can be specified in a similar way, but we have to add fairness conditions on the computations in order to force the transmission before a finite amount of time. More precisely, assume that a resilient channel c has to deliver a message m_1 when the proposition $\mathtt{SendM_1}$ is set to true and the delivery of the message is modeled by setting $\mathtt{M_1}$ to true. We impose the following (weak) fairness constraint on c: for each $q \in Q$ such that $q \models \mathtt{SendM_1}$, we have

$$\gamma(c, q) = \{Q' \mid Q' \in \delta(c, q) \text{ and } \forall q' \in Q' : q' \models \mathtt{M_1}\}.$$

This fairness constraint rules out trajectories where c can deliver a message and never delivers it. *Operational channels* do not have any additional action and thus immediately transmit the messages that have been sent on it. By immediately transmitting messages we alter the channel definition, which requires messages to be transmitted before a known constant time. This is however acceptable as the sender can always wait before sending the message to obtain the desired delay.

When starting to model an informally described protocol we apply the following method. First, we determine the initial knowledge of each player. Initial knowledge may for instance include nonces, public and shared keys as well as initially known messages. All these items are represented by boolean variables initialized to true. All other variables are set to false, indicating that there value is not known yet. Then we model all useful cryptographic operations that can be applied on the initial knowledge. By useful we mean that the operation generates an item that can be used later in the protocol. Now the following steps are applied to each message $\mathtt{X} \rightarrow \mathtt{Y} : \mathtt{M_i} = \mathtt{m_{i1}}, \mathtt{m_{i2}}, \ldots, \mathtt{m_{in}}$ of the protocol. To represent the sending of $\mathtt{M_i}$, a guarded command $\mathtt{m_{i1}} \wedge \cdots \wedge \mathtt{m_{in}} \rightarrow \mathtt{SENDm_i} := \mathtt{true}$ is added to player $\mathtt{X}$'s description. The guard contains the conjunction of all the elements of the message. This means that a player is able to send a message if and only if he has knowledge of all the required elements. The update relation expresses the intention to send the message. The transmission of the message is represented by a guarded command added to the description of the communication channel between $\mathtt{X}$ and $\mathtt{Y}$. The command will be of the form $\mathtt{SENDm_i} \rightarrow \mathtt{M_i} := \mathtt{true}$. Note that in order to delay this message a communication channel may choose to execute a different command. Reception of the message is modeled by adding the following command to $\mathtt{Y}$'s description $\mathtt{M_i} \rightarrow \mathtt{m_{i1}} := \mathtt{true}; \ldots; \mathtt{m_{in}} := \mathtt{true}$. For each element of $\mathtt{M_i}$ being set to true for the first time in $\mathtt{Y}$'s description, we also add all useful cryptographic operations that can be applied on it.

The above presented method describes how to model Alice, respectively Bob having arbitrary behavior. To restrict their behavior to the honest protocol execution, we can easily reinforce the guards so that the order of execution corresponds to the one dictated by the protocol. We get two different descriptions of Alice, as well as of Bob: the first description models arbitrary behaviors, the second one only the honest behavior.

Modeling of the requirements. We show here how the main requirements that an exchange protocol must fulfill, can be *naturally* rephrased as the existence of strategies for the participating entities to reach their goal. The logic ATL is used to formalize those requirements. We will concentrate here on properties of non-repudiation protocols. The properties described are general properties that do not apply to a given protocol. They may need to be instantiated when studying a protocol.

In a first approximation, we introduced fairness as the property that either both entities receive all their desired evidences or none of them receives any valuable evidence. We can now formulate this requirement as the existence of strategies. We say that the protocol is fair for Alice if "Bob and the communication channels do not have a strategy to reach a state where Bob has his proof of origin and Alice has no more a strategy to obtain her proof of receipt". This can be formally expressed by the following ATL formula:

$$\neg\langle\langle B, \mathsf{Com}\rangle\rangle\Diamond(\mathsf{NRO} \wedge \neg\langle\langle A\rangle\rangle\Diamond\mathsf{NRR}) \tag{2}$$

Here, Com denotes the set of all communication channels. NRR and NRO respectively denote the non-repudiation of receipt and origin evidences. This is a generic notation as the form of those evidences depends on a particular protocol. The expressive power of the logic ATL is well illustrated by this example. Cooperation and adversarity are naturally expressed. To better understand the advantages of using a game logic to formalize this requirement, let us consider the following CTL formula:

$$\neg\exists\Diamond(\mathsf{NRO} \wedge \neg\exists\Diamond\mathsf{NRR}) \tag{3}$$

that may look as an appropriate CTL candidate for the formalization of fairness for Bob. Note that we have obtained this formula by replacing the two teams in (2) by the entire set of players Σ. Note also that the formula (3) can be rewritten in the equivalent and more readable positive form:

$$\forall\Box(\mathsf{NRO} \rightarrow \exists\Diamond\mathsf{NRR}) \tag{4}$$

That says "on every state of every run of the protocol, if Bob has his evidence of origin then there should exist a continuation of the protocol on which Alice eventually receives her evidence of receipt". This way, we have lost however the ability of distinguishing between cooperative and adversarial behaviors. Let us show what are some consequences of this, so to say, "loss of precision". For example, it may be the case that the formula (4) is verified because in the course of the protocol a *resilient* channel helps Alice to obtain her proof by delivering a message within a given bound (which is not generally the case for such a channel). As this assumption of cooperative behavior of the resilient channels cannot be made in general, in order to be fair the protocol should guarantee to Alice that she obtains her proof even if the resilient channel does not deliver the message within a certain bound. This fact is not ensured by the CTL formalization but is ensured by our ATL formalization. In fact, in formula (2), it is explicitly stated that the communication channels play against Alice. A resilient channel has only the obligation to deliver the message after a finite amount of time and not within a given bound, even if that helps Alice to obtain her proof. This non-cooperative aspect is precisely formalized in ATL. Also formula (3) may be verified because there exists some paths, where Bob is honest and allows Alice to obtain her proof. Again, the protocol should not make the hypothesis that Bob is honest, and should ensure fairness for Alice even if Bob is trying to cheat her. This adversarial behavior is formally modeled by our ATL formula and can not be formalized in CTL. We can also have that the CTL formula is false, even if the ATL formula holds. Such a situation can occur if Alice's specification contains some non-determinism. For instance, if the specification does not contain at which moment Alice can stop the protocol, she may stop it at a "wrong" moment. The CTL formula will fail, while the ATL formula may hold, as not stopping the protocol at a given moment would have lead to a winning strategy. Using the notion of strategy, we "automatically" exclude behaviors of Alice that do not serve her objective. Note that this last comment also rules out the following CTL formula as a candidate for fairness:

$$\forall\Box(\mathsf{NRO} \rightarrow \forall\Diamond\mathsf{NRR}) \tag{5}$$

Situations where the protocol is not well defined, occur frequently when integrating the use of formal methods as an aid in the protocol designing process. In the first step while designing the protocol, we generally do not want to deal with all details: the aim of the verification is to determine whether the given specification contains a winning strategy that leads to a correct protocol. If Alice is specified in a deterministic way, one may use formula 5 to verify fairness for Alice. However, we believe that at a specification level, as well as during the design process, the fairness requirement should explicitly contain the adversarial as well as the cooperative behaviors.

Viability is expressed by the following formula:

$$\langle\langle A, B\rangle\rangle\Diamond(\mathsf{NRO} \wedge \mathsf{NRR}) \tag{6}$$

that says "Alice and Bob can cooperate, so they are honest (i.e. they are following the protocol), and in that case the protocol should allow them, even in presence of non-cooperating channels, to obtain their respective evidences". And finally timeliness is formalized by :

$$\langle\langle A\rangle\rangle\Diamond(\mathsf{stop}_A \wedge (\neg\mathsf{NRR} \rightarrow \neg\langle\langle B\rangle\rangle\Diamond\mathsf{NRO})) \tag{7}$$

which expresses that "Alice has a strategy to finish the protocol and if she does not have her evidence at that point, Bob will not be able to obtain his evidence neither". The same requirement can be expressed for Bob.

4 Verification with Mocha

We used the model-checker Mocha to verify several protocols. We report in detail on our verification of the Zhou-Gollmann optimistic protocol [16] and illustrate on this example our verification methodology. We also checked the Asokan-Shoup-Waidner certified mail protocol [3], the Kremer-Markowitch non-repudiation protocol [7], as well as the Markowitch-Kremer multi-party non-repudiation protocol [10]. For these protocols, due to lack of space, we only briefly show our results without giving a complete description of the protocols.

4.1 The ZG optimistic non-repudiation protocol

We first give an informal description of the protocol using the following notation:

- $X \to Y$: transmission from entity X to entity Y
- $X \leftrightarrow Y$: ftp get operation performed by X at Y
- $h()$: a collision resistant one-way hash function
- $E_k()$: a symmetric-key encryption function under key k
- $D_k()$: a symmetric-key decryption function under key k
- $S_X()$: the signature function of entity X
- m: the message sent from A to B
- k: the message key A uses to cipher m
- $c = E_k(m)$: the cipher of m under the key k
- $l = h(m, k)$: a label that in conjunction with the entities (A,B) identifies a protocol run
- f: a flag indicating the purpose of a message
- t: the time-out chosen by A

The protocol generates the following evidences:

- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, B, l, t, c)$: the evidence of origin of c
- $\mathsf{EOR} = S_B(f_{\mathsf{EOR}}, A, l, t, c)$: the evidence of receipt of c
- $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, B, l, t, k)$: the evidence of origin of k
- $\mathsf{EOR}_k = S_B(f_{\mathsf{EOR}_k}, A, l, t, k)$: the evidence of receipt of k
- $\mathsf{Sub}_k = S_A(f_{\mathsf{Sub}_k}, B, l, k)$: the evidence of submission of k
- $\mathsf{Con}_k = S_{\mathsf{TTP}}(f_{\mathsf{Con}_k}, A, B, l, t, k)$: the evidence of confirmation of k issued by the TTP

The protocol is divided into two subprotocols: a main protocol and a recovery protocol. As the protocol is based on the optimistic approach, the trusted third party (TTP) does only intervene in the recovery protocol. We start by describing the main protocol.

1. $A \to B$: $f_{\mathsf{EOO}}, B, l, t, c, \mathsf{EOO}$
2. $B \to A$: $f_{\mathsf{EOR}}, A, l, \mathsf{EOR}$
3. $A \to B$: $f_{\mathsf{EOO}_k}, B, l, k, \mathsf{EOO}_k$
4. $B \to A$: $f_{\mathsf{EOR}_k}, A, l, \mathsf{EOR}_k$

First Alice sends the digitally signed cipher $c = E_k(m)$ to Bob. In the second message Bob responds with the evidence of receipt for this cipher (EOR). If Alice does not receive the second transmission she stops the protocol, otherwise she sends the signed decryption key k to Bob. Bob answers by sending the receipt EOR_k for the key. The label l, present in each transmission, identifies the protocol run. The time-out t specified in message 1 is used in the recovery protocol. Alice may initiate the recovery protocol if Bob does not send the receipt for the key.

The steps of the recovery protocol are the following:

1. $A \rightarrow \mathsf{TTP}:\ f_{\mathsf{Sub}_k}, B, l, t, k, \mathsf{Sub}_k$
2. $B \leftrightarrow \mathsf{TTP}:\ f_{\mathsf{Con}_k}, A, B, l, t, k, \mathsf{Con}_k$
3. $A \leftrightarrow \mathsf{TTP}:\ f_{\mathsf{Con}_k}, A, B, l, t, k, \mathsf{Con}_k$

Alice sends the signed key, together with the deadline t to the TTP. If the key arrives after t, the TTP does not accept the recovery. Otherwise the TTP publishes the key together with a confirmation Con_k for the key in a read-only accessible directory, where both Alice and Bob can fetch the key as well as Con_k. Con_k serves to Bob as the evidence of origin of the key, and to Alice as the evidence of receipt of the key as it is accessible to Bob. The deadline t is necessary for Bob to know the moment when either the key is published or will not be published anymore. In this protocol, the non-repudiation of origin evidence NRO is composed of EOO and EOO_k or of EOO and Con_k. The non-repudiation of receipt evidence NRR is composed of EOR and EOR_k or of EOR and Con_k.

Zhou et al. suppose that the channels between Alice and Bob are unreliable and the channels between the TTP and both Alice and Bob are resilient. In our model we do not model the ftp get operation, but assume that the TTP takes the initiative to send messages 2 and 3 of the recovery protocol to Alice respectively Bob, as soon as these actions are possible.

To verify the protocol we instantiate the properties defined in section 3 for two versions of both Alice and Bob. The first version allows arbitrary behavior[3], the second one restricts the behavior to the honest protocol execution. The specification of Alice, respectively Bob, allowing arbitrary behavior will be denoted with A, respectively B, while the specification of the honest behavior of Alice and Bob will be denoted $\mathsf{A_h}$, respectively $\mathsf{B_h}$. When we want to check a property, for instance fairness for Alice, we first verify that Alice has a strategy to assure the fairness requirements, when she is allowed to behave in an arbitrary way. We check that

$$\neg\langle\langle \mathsf{B}, \mathsf{Com}\rangle\rangle\Diamond(\mathsf{EOO} \wedge (\mathsf{EOO_k} \vee \mathsf{Con_k}) \wedge \neg\langle\langle \mathsf{A}\rangle\rangle\Diamond(\mathsf{EOR} \wedge (\mathsf{EOR_k} \vee \mathsf{Con_k})))$$

holds. If the formula does not hold, even with arbitrary behavior of Alice, the protocol is flawed and can not be fixed without introducing new 'mechanisms'

[3] We call arbitrary behavior, all behavior that is *interesting* with respect to the protocol, i.e. behavior that results in sending or obtaining acceptable messages related to the protocol.

in the protocol. If the formula holds we additionally need to check that Alice cannot be cheated when following the protocol. It is possible that the formula holds because Alice adopted a strategy, that is not the one proposed by the protocol. Therefore we check the following formula:

$$\neg\langle\langle \mathsf{B}, \mathsf{Com}\rangle\rangle\Diamond(\mathsf{EOO} \wedge (\mathsf{EOO_k} \vee \mathsf{Con_k}) \wedge \neg\langle\langle \mathsf{A_h}\rangle\rangle\Diamond(\mathsf{EOR} \wedge (\mathsf{EOR_k} \vee \mathsf{Con_k})))$$

In this formula we restricted Alice's behavior to the actions that are dictated by the protocol. If the formula evaluates to false, it means that the protocol, as originally described, is flawed. However, it is possible to easily correct it, as a strategy was found by the model-checking algorithm while checking the previous formula. This means that we do not have to fundamentally change the protocol, by adding or changing messages.

While investigating the protocol, we succeeded in finding a flaw in the protocol, even if Alice's behavior is not restricted to the honest protocol execution. Remember that the communication channels between the TTP and both Alice and Bob are resilient and that a finite time-out must be chosen by Alice at the begin of the protocol. In our model, the time-out is a boolean variable controlled by a player *Clock*. We restrict the time-out to become true after a finite amount of time, using a similar technique as used for transmission on resilient channels. In fact, when the main protocol has been executed until step 3, Bob has received all of his evidences and may decide to stop the protocol in order to try to cheat Alice. At this point we check whether Alice in cooperation with the clock does have a strategy against Bob who is cooperating with the communication channels, to receive her non-repudiation of receipt evidences. As Alice cooperates with the clock, she entirely controls the triggering of the time-out. Intuitively, it means that Alice can choose the time-out to occur as late as she wants, which represents the fact, that she chooses the time-out at the first step of the protocol. As Alice does not receive her evidence, she has to initiate a recovery. For the recovery protocol to be successful, the request needs to arrive before the time-out. However, as communication channels can help Bob, this message can be delayed until the time-out occurs. Intuitively, a resilient channel has the capacity to delay messages: thus for each possible finite time-out value chosen by Alice, Bob (with the help of the communication channels) can delay long enough the delivery of the message in order for the TTP to reject the recovery request. However, when we change the communication channel to be operational (messages are transmitted immediately) the protocol becomes fair. Unfortunately, operational channels are rather unrealistic in practice.

We can also use our method to find the origin of a flaw. We can add players to the cooperation to detect the player(s) that make the property fail. As an example, take a look at the flaw we described in the previous paragraph. We asked whether Bob, together with all the communication channels, has a strategy to obtain the NRO evidence, such that Alice, in cooperation with the clock, does not have any strategy to receive the NRR evidence. Using MOCHA we showed the existence of a flaw, by validating this property. However, it may be difficult to determine the failure's origin. Thus we try to change the coalitions to find the player responsible for the flaw. By adding the communication channel between Alice and the TTP to Alice's coalition we succeeded in avoiding the property to be validated. Once we know that the communication channel is at the origin of the error it may be interesting to alter the channel's quality. In that way we see which modifications are needed to recover the fairness property. In this example we concluded that replacing the resilient channel by an operational channel was enough to prevent protocol failure. Hence, we have a rather easy and quick investigation method to locate and possibly prevent flaws.

4.2 Other verified protocols

We also verified the ASW certified e-mail protocol [3]. In this protocol we found errors due to the fact that sub-protocols can be executed out of order, in a way not foreseen by the authors. We verified a variant of the KM non-repudiation protocol [7], where the abort protocol has been removed. This results in a fair protocol, which however does not respect timeliness anymore and hence is not secure. Finally we analyzed the MK multi-party non-repudiation protocol [10]. We verified an instance with two recipients and showed that problems can arise due to a race condition, if we do not correctly handle a clock synchronization problem, as proposed in the original paper. These results can be found in an extended version of this paper, available as a technical report [8].

5 Related Works

First efforts to apply formal methods to the verification of non-repudiation protocols have been presented by Zhou et al. in [15]. SVO, a "BAN-like" belief logic has been used to study a non-repudiation protocol. The aim of that study differs however from our study. In the context of non-repudiation protocols, belief logics are useful to reason about the validity of the evidences. They deal with questions of what a judge has to believe when Alice or Bob present their respective evidences. Belief logics cannot be applied to verify properties such as fairness or timeliness. In [13] Schneider uses CSP to prove the correctness of a non-repudiation protocol. These proofs are not automated and require great efforts. Recently Boyd and Kearny [4] discussed a method using specification animation to analyse fair exchange protocols using the Possum animation tool. The tool gives the possibility to step through the protocol and examine the consequences of various actions. Boyd et al. use a high level abstraction that only allows to find errors due to sending messages out of order. However they succeed in finding errors even on these very simplified versions of the protocols.

The most extensive studies of fair exchange protocols using formal methods have been presented by Shmatikov and Mitchell in [14]. They use Murφ, a finite-state model-checker to analyze a fair exchange and two contract signing protocols. Their approach differs from ours on some major points. First, they use an intruder model. To model the fact that a party is malicious and could try to cheat, they make that party share all its knowledge with the intruder. The same intruder model has been used to analyse other security protocols. However we believe that there is no need to use an intruder model. Therefore we directly model Alice as well as Bob to be potentially malicious. The analysis using Murφ is based on invariant checking. Channel resilience is obtained by checking all invariants in final protocol states. Protocol runs where messages are lost do not reach the final protocol states and are discarded. To achieve resilience in our model we need to add fairness conditions. Invariant checking is sufficient to verify fairness as fairness is a *monotonic* property: if fairness is broken at one point of the protocol, the protocol will remain unfair. Checking only final states is thus sufficient to verify fairness. However with this approach timeliness cannot be verified. Timeliness is a liveness property guaranteeing that we can always reach a final state in the protocol. A protocol run, not respecting timeliness, would be discarded for the invariant check as the protocol would not be finished. Note that Shmatikov et al. did not aim to verify timeliness. In our model the timeliness property can easily be verified. As shown before timeliness is of crucial importance for the security of a protocol and should be verified.

6 Conclusions and Future Works

We have shown that exchange protocols, due to their particularities, are best modeled as games. First, the adversarial and cooperative behaviors that occur during the execution of those protocols are naturally and precisely expressed in term of strategies. Second, the main requirements that an exchange protocol must ensure are easily phrased as existence of strategies for the participating entities to reach well defined goals. We have proposed to use the framework of alternating transition systems and alternating-time temporal logic to formalize this view of exchange protocols as games. The great expressive power of alternating temporal logic in this context has been illustrated. The practical interest of the method has been demonstrated by analyzing several protocols. For the analysis, we have used the tool Mocha which is able to automatically verify alternating temporal formula on alternating transition systems.

In the future, we will try to show that this approach is applicable to the analysis and verification of more specific properties, such as abuse-freeness in contract signing protocols for instance. We will also investigate the problem of strategy synthesis. Strategy synthesis is a very powerful tool in protocol design and can also be used to synthesize attacks, showing the origin of flaws.

Acknowledgement. The authors would like to thank Giorgio Delzanno, Olivier Markowitch and Thierry Massart for carefully reading previous versions of this paper, as well as Freddy Mang for his assistance while using Mocha and the anonymous referees for their helpful comments.

References

1. R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1997.
2. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 6, 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.
3. N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, Oakland, CA, May 1998. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.
4. C. Boyd and P. Kearney. Exploring fair exchange protocols using specification animation. In *The Third International Workshop on Information Security - ISW2000*, Lecture Notes in Computer Science, Australia, Dec. 2000. Springer-Verlag.
5. M. Burrows, M. Abadi, and R. Needham. A logic of authentication, from proceedings of the royal society, volume 426, number 1871, 1989. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996*. 1996.
6. T. Henzinger, R. Manjumdar, F. Mang, and J.-F. Raskin. Abstract interpretation of game properties. In *SAS 2000: Intertional Symposium on Static Analysis*, Lecture Notes in Computer Science. Springer-Verlag, 2000.
7. S. Kremer and O. Markowitch. Optimistic non-repudiable information exchange. In J. Biemond, editor, *21th Symp. on Information Theory in the Benelux*, pages 139–146. Werkgemeenschap Informatie- en Communicatietheorie, Enschede, may 2000.
8. S. Kremer and J.-F. Raskin. A game-based verification of non-repudiation and fair exchange protocols. Technical Report 451, ULB, 2001.
9. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. *Lecture Notes in Computer Science*, 1055:147–157, 1996.
10. O. Markowitch and S. Kremer. A multi-party optimistic non-repudiation protocol. In D. Won, editor, *Proceedings of The 3rd International Conference on Information Security and Cryptology (ICISC 2000)*, volume 2015 of *Lecture Notes in Computer Science*, Seoul, Korea, 2000. Springer-Verlag.
11. C. A. Meadows. Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches. *Lecture Notes in Computer Science*, 1146:351–365, 1996.
12. L. C. Paulson. Proving properties of security protocols by induction. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, June 1997.
13. S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 54–65, Washington - Brussels - Tokyo, June 1998. IEEE.
14. V. Shmatikov and J. Mitchell. Analysis of abuse-free contract signing. In *Financial Cryptography '00*, Anguilla, 2000.
15. J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In *Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380, Canberra, Australia, Sept. 1998. Springer.
16. Y. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.