A Game Approach to the Verification of Exchange Protocols

Application to Non-repudiation Protocols *

Steve Kremer Jean-François Raskin
Département d'Informatique
Faculté des Sciences
Université Libre de Bruxelles, Belgium

1 Introduction

Non-repudiation Protocols. During the last decade open networks, above all the Internet, have known 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 in order to generate evidences for non-repudiation of origin (NRO) (for Bob), and non-repudiation of receipt (NRR) (for Alice). In case of a dispute Alice or Bob will present their evidences to an adjudicator, who will take a decision in favor of one of the two entities without ambiguity.

One solution consists in using a trusted third party (TTP) as an intermediary to ensure delivery. The major problem of this approach is the network bottleneck, represented by the TTP. To avoid the decrease of performance created by this bottleneck, Asokan et al. introduced the optimistic approach for fair exchange [ASW97]. In an optimistic protocol one supposes that in general the involved entities are honest and that the network is well functioning. The rational is that the TTP only intervenes in case of a problem. Afterwards, Zhou et al. applied the optimistic approach to the non-repudiation protocols [ZG97].

A non-repudiation protocol has to respect several properties. The most important one is *fairness*: fairness must ensure that if at least one entity is honest, either both receive the expected non-repudiation evidence or none of them receives it. 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. We consider three classes of channels: unreliable channels, resilient channels and operational channels. No assumptions can 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 after a known, constant amount of time.

In comparison to other security issues such as privacy or authenticity of communications,

---

*An extended version of this abstract has been accepted for publication in FMCS 2000 [KR00]

non-repudiation has not been extensively investigated. However, the more studied authentication protocols have shown that the design of security protocols is an error prone process.

Exchange protocols as games. There are fundamental differences between authentication protocols and exchange protocols (e.g. non-repudiation protocols). Generally, the most difficult problem 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*. Another difference is that exchange protocols, and above all optimistic ones, are not linear: they are divided into several subprotocols (making branching possible) although they are intended to be executed in a given order. Changing the order of execution could result in subtle errors. This is the reason why we want to introduce a new method that is more adequate for 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. Second, we consider the execution of the protocol as a game: each entity (Alice, Bob, TTP) and each communication channel is a player. This point of view allows us to express with more precision the required properties as *strategies*. The main advantage of modeling such protocols as games is that we *directly* and *formally* take into account the possibility of *adversarial* as well as *cooperative* behaviors between entities. The trust in the TTP is modeled by giving the TTP a unique strategy. By using alternating transition systems and their alternating-time temporal logic of Alur et al. [AHK97], we are able to formalize the non-repudiation protocols and their requirements in a *direct* and *natural* way.

2 A Formal Model of Games and its Logic

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 [AHK97], 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 labelling 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 and the next state of the game q' is the intersection (that is required to be a singleton) of the choices made by all the players of Σ, i.e. $q' = \cap_{a\in\Sigma}\delta(q, a)$. A computation is an infinite sequence $\lambda = q_0q_1 \ldots q_n \ldots$ of states.

We now introduce the alternating-time temporal logic [AHK97], 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 the choices controlled by the players in the set A, while the antagonist chooses 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.

Instead of modeling protocols directly with ATS we will use a more user-oriented notation: a guarded command language a la Dijkstra. The details about the syntax and semantics of this language (given in terms of ATS) can be found in [HMMR00]. Here follows an intuitive presentation. Each player $a \in \Sigma$ disposes of a set of guarded commands of the form:

$$guard_\xi \rightarrow update_\xi \tag{1}$$

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 Model of Non-Repudiation Protocols

Modeling of protocols. The notation typically used to describe cryptographic protocols ($A \rightarrow B : m$, 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 subprotocols can be invoked at different moments. Unfortunately, running a subprotocol at a time not foreseen by the designer, may have unexpected side-effects, threatening the security of the protocol. Therefore, we use the modeling language described above to model the exchange protocols. The guard $guard_\xi$ is used to represent the elements (such as keys, messages, ...) necessary for a participating entity (a player) to execute the action ξ. The variables controlled by a player represent his current knowledge and thus determine which actions he 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 between the guarded commands. Cryptographic operations are also modeled using guarded commands. 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 in a deterministic way: at each stage of the execution of the protocol, the TTP executes the action requested by the protocol.

The communication channels can also be modeled as players. Each transmission is modeled as a guarded command.

Modeling of 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.

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 its 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 can not be made in general, in order to be fair the protocol should guarantee Alice to obtain her proof even if the resilient channel does not deliver the message within 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. Now let us take the other possibility: the CTL requirement can be false even if the ATL formula is verified. This can occur in the following situation: as we consider all reachable states where NRO is true, we may consider states where Alice, by its own fault, has reached a situation where she is not able to get her evidence anymore. This may happen if, for example, Alice has sent the origin evidence to Bob and decides to quit the protocol. This problem does not occur in our ATL formalization as by using the notion of strategy, we "automatically" exclude behaviors of Alice that does not serve her objective. Note that this last comment also rules out the following CTL formula as a candidate for fairness:

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

To complete the fairness requirement, we express fairness for Bob as follows:

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

and take the conjunction of (2) and (6). Viability is expressed by the following formula:

$$\langle\langle A, B\rangle\rangle\Diamond(\mathsf{NRO} \wedge \mathsf{NRR}) \tag{7}$$

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 :

$$\langle\langle A\rangle\rangle\Diamond(\mathsf{stop}_A \wedge (\neg\mathsf{NRR} \rightarrow \neg\langle\langle B\rangle\rangle\Diamond\mathsf{NRO})) \tag{8}$$

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 either". The same requirement can be expressed for Bob.

4 Verification with Mocha

We have applied the proposed verification technique, to the optimistic Zhou-Gollman protocol. A detailed description of this protocol can be found in [ZG97]. We have used the model-checker Mocha [AHM+98] that supports the logic ATL.

Using Mocha we made a detailed formal analysis of the protocol and succeeded in showing a flaw, that has recently been found [KM00]. Zhou et al. suppose that the communication channels between the TTP and both Alice and Bob are resilient. However the TTP accepts a recovery request only if it arrives before a given, finite time-out. In our model, the time-out is a boolean variable controlled by a player *Clock*. The fact that Alice chooses this time-out at the beginning of the protocol, is modeled by a coalition of Alice with the Clock. However as the communication channels can help Bob, the recovery request can always be delayed until the time-out has occurred. 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. Note, that if we are using the CTL formula (3) this flaw is not detected. In fact the resilient channel can deliver the request before the time-out and thus a path satisfying this formula exists. However, as the resilient channel is cooperating with Bob, Bob can *force* this path never to be taken.

If a requirement is not respected, we may try to add players to the cooperation to detect the player(s) being at the origin of the protocol failure. It may also be interesting to alter the channel's quality. In that way we see which modifications are needed to recover the fairness property. More details on the verification using Mocha can be found in [KR00].

5 Future Works

In the future, we will try to show that this approach is applicable to the analysis and verification of other, more complex, exchange protocols (fair exchange, digital contract signing, certified mail, ...). We will also try to model the possibility for several concurrent protocol runs. Moreover we will investigate more theoretical implication related to the fact that we see those protocols as games. In particular, we will investigate the relevance of the notion of partial knowledge in the modeling of exchange protocols and its implication on the automatic verification of those protocols. Finally we will look at the possibilities for protocol synthesis: given an ATL formula the model checking algorithm can synthesize strategies to verify the given property and so produce a valid protocol automatically.

Acknowledgement. The authors would like to thank Giorgio Delzanno for carefully reading a previous version of this abstract and the members of the group "Vérification" of ULB, FUNDP and UMH for their comments on that work.

References

[AHK97] R. Alur, T.A. 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.

[AHM+98] R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Tasiran. MOCHA: modularity in model checking. In A.J. Hu and M.Y. Vardi, editors, *CAV 98: Computer-aided Verification*, Lecture Notes in Computer Science 1427, pages 521–525. Springer-Verlag, 1998.

[ASW97] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In Tsutomu Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 6, 8–17, Zurich, Switzerland, April 1997. ACM Press.

[HMMR00] T.A. Henzinger, R. Manjumdar, F.Y.C. 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, (to appear) 2000.

[KM00] Steve Kremer and Olivier Markowitch. Optimistic non-repudiable information exchange. In *21th Symp. on Information Theory in the Benelux*. Werkgemeenschap Informatie- en Communicatietheorie, Enschede, may 2000. to appear.

[KR00] S. Kremer and J.-F. Raskin. Formal verification of non-repudiation protocols - a game approach. In *Procceedings of the Workshop on Formal Methods and Computer Security*, 2000. (to appear).

[ZG97] J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.