# An approach to handle Real Time and Probabilistic behaviors in e-commerce: Validating the SET Protocol

G. Díaz[†], K. Larsen[‡], J. Pardo[†], F. Cuartero[†] and V. Valero[†]

[†]Department of Computer Science
University of Castilla-La Mancha
Campus Universitario, Avd. España s/n
02071, Albacete, Spain
(gregorio,jpardo,fernando,valentin)@info-ab.uclm.es

[‡]Department of Computer Science
Aalborg University
Fr. Bajersvej 7E
9220 Aalborg East, DENMARK
kgl@cs.auc.dk

## ABSTRACT

In this work we describe an approach to deal with systems having at the same time probabilistic and real-time behaviors. The main goal in the paper is to show the automatic translation from a real time model based on UPPAAL tool, which makes automatic verification of Real Time Systems, to the RAPTURE tool, which makes verification of probabilistic systems.

Furthermore, this approach allows us to use the best techniques developed in both tools (abstraction, refinement, state space reduction, etc). Finally, this translation is applied to verify a case study, the SET internet protocol.

## Categories and Subject Descriptors

K.4.4 [**Electronic Commerce**]: [Security]; C.2.2 [**Network Protocols**]: [Protocol verification]; D.2.4 [**Software / Program verification**]: [Model Checking].

## General Terms

Authentication Protocols and Model Checking.

## Keywords

e-Commerce, Security, Authentication Protocols, System Verification and Model Checking.

## 1. INTRODUCTION

UPPAAL [11, 10, 2, 8] is a suitable tool to make automatic verification of real time systems. But sometimes, it is necessary not only to capture the timed behavior of systems. We may need to introduce the quantified possibility for an event to occur. Then, we need to describe the probabilistic behavior of the systems. Actual tools to verify probabilistic systems have shown that it is necessary to develop new algorithms to deal with real time problems. In this way, a tool making automatic probabilistic verification is RAPTURE [5, 6], which uses a model abstraction in order to deal with the state explosion problem and a refinement technique in order to improve the abstraction. But RAPTURE cannot deal directly with clocks. Thus, it is not possible to use this tool to handle at the same time with probabilistic and real time behaviors.

Thus, an integration of both tools (UPPAAL and RAPTURE) would be helpful to solve this problem. A way to do it could be to translate the verified UPPAAL models to RAPTURE, and then we could verify its probabilistic behavior. However, as it is pointed out in [7], once the translation is made, we lose the timed references in the model. Thus, we need a way to keep in our translated model these references. As it is shown in d'Argenio et. al. in [9], an integer semantics can be sometimes adequate to verify probabilistic real time systems, if we are using forward reachability algorithms. Thus, we could translate the clocks in a very easy way to Probabilistic Transition Systems, because we could consider these clocks as integer variables. But this is not enough; we need to simulate the delay transitions occurring inside a location, i.e. the time that we can spend in a location.

A tool that makes a direct translation between UPPAAL and RAPTURE is P_UPPAAL [7]. We will see in this paper the theory of this translation, between Real Time Model and Probabilistic Transaction Model, keeping the temporal references. We also apply it to an interesting case study, the internet protocol known as "Secure Electronic Transaction" or SET for short.

To apply the Model Checking techniques to these kind of protocols we need to fix the set of properties that the protocol must satisfy. For example, we can specify properties such as "the system will never reach an erroneous situation". However, it is often vital that additional quantitative properties are established in order that the system can be considered as correct. Such properties include real-time requirements such as "a desired state will be reached within 105 seconds" and probabilistic properties of the type "a desired state will be reached with probability of at least 99% ". But the goal is to verify properties joining real-time and probabilistic requirements at the same time, such as "a


*SAC'05* March 13-17, 2005, Santa Fe, New Mexico, USA

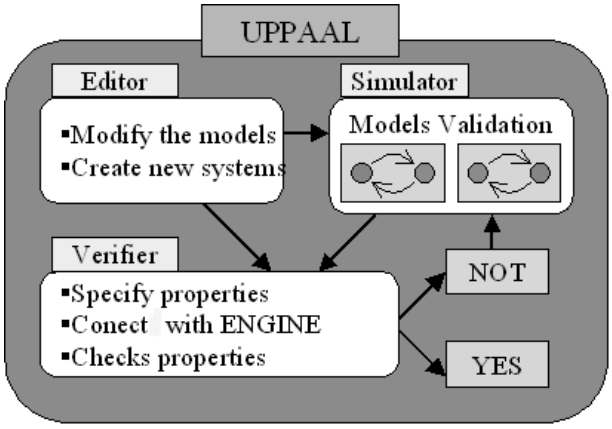


**Figure 1: The UPPAAL architecture.**

desired state will be reached within 105 seconds and with probability of at least 99%".

Thus, this kind of Model checking will allow us to check the probabilistic and real-time behavior of security protocols such as SET, without losing temporal references.

In the next section we will see the functionality of UPPAAL and RAPTURE. After that, we will deal with the problem of translating from UPPAAL models (Hybrid Networks of Timed Automata) to RAPTURE models (Probabilistic Transitions Systems) keeping the temporal references. Finally we will see the SET protocol and how we can check its probabilistic real-time behaviors.

## 2. UPPAAL AND RAPTURE

In this section, we will first see the architecture of UPPAAL and, afterwards, we will present the RAPTURE architecture.

### 2.1 UPPAAL

The architecture of UPPAAL, shown in figure 1, is divided in three main parts, i.e. the Editor, the Simulator and the Verifier.

The **Editor** allows us to model the studied system. The system is modelled as a network of Hybrid Timed Automata. An Hybrid Timed Automaton consists of States and transitions between states, furthermore the model has variables and clocks, allowing us to capture timed behaviors. However, it is necessary to add some constraints and synchronizations. The constraints in UPPAAL are fixed by invariants establishing how much time we can spend in a state, and by guards enabling a transition. Finally, the synchronization opens a channel between two transitions and allows us to make these two transitions at the same time.

The **Simulator** makes simulations throw the modelled system. With it we can check that our system satisfies the expected behaviors.

The **Verifier** allows us to validate the model. In this way, the user must first specify the properties that the model must hold, by using a temporal logic. With this temporal logic we specify properties such as "The model could reach the finished state within five time units" or "We always reach the finished state within five time units" or "The model is deadlock free". After the specification, we can check the properties and, also, if the model does not satisfy a specific property, we obtain a counterexample, i.e. a path where the specific property does not hold.

### 2.2 RAPTURE

The system RAPTURE is described in terms of Markov decision process [13], also called probabilistic transition systems (PTS). This model allows us to combine probabilistic and non-deterministic steps providing a natural extension to traditional non-deterministic models.

RAPTURE is focused on a restricted class of reachability properties. These properties allow to specify that the probability of reaching a particular condition $\phi_f$ from any reachable state satisfying a given initial condition $\phi_i$ is smaller (or greater) than a given probability $p$ regardless of how non-deterministic choices of the model are resolved.

RAPTURE is based on automatic abstraction and refinement [6]. The basic idea is to use abstractions in order to reduce the high cost of the numerical analysis involved in computing the minimum and maximum reachability probabilities for PTSs. The abstractions considered are obtained via successive refinements, starting from an initial coarse partitioning of the state space derived from the property under study. For a given refinement the property is checked on the induced abstract model, hopefully settling the property. However, the verdict may be inconclusive, when threshold probability $p$ happens to be between the calculated minimum and the maximum abstract probabilities. In this case, the abstraction is further refined and the property checked again. This process is successively repeated until either the property is settled, or no further refinement is possible.

## 3. TRANSLATION

In this section we will see how we can translate timed automata from UPPAAL to Probabilistic Transition Systems for RAPTURE without losing the temporal references.

By definition, a timed automata is a standard finite-state automaton extended with a finite collection of real valued clocks. Clocks are assumed to proceed at the same rate and their values may be compared with natural numbers or reset to 0. It is possible to extend the notion of timed automata to include integer variables, i.e. integer valued variables that may be compared to natural numbers or assigned to any value of the form $ax + b$ where $a, b \in \mathbf{Z}$ and $x$ is the variable being reassigned. The model also allows clocks not only to be reset, but also to be set to any non-negative integer value.

For further information see [10] where the UPPAAL model: "Hybrid Networks of Timed Automata" are described in detail.

### 3.1 Probabilistic Transition Systems

As it is pointed out in [6], Probabilistic transition systems (PTS for short) generalize the well-known transition systems with probabilistic information. In a PTS, a transition does not lead to a single state but to a probability space whose sample space is a set of states. The model we define is widely used (see e.g. [1]) and it is also known as Markov decision processes [13].

Let $Distr(\Omega)$ denote the set of all probability distributions over the sample space $\Omega$.

Definition 1. Probabilistic Transition Systems. *A probabilistic transition system (PTS for short) is a structure*

*$T = (S, \rightarrow)$ where $S$ is a set of states, and $\rightarrow \subseteq S \times Distr(S)$ is the transition relation. We write $s \rightarrow \pi$ for $(s, \pi) \in \rightarrow$. A PTS is said to be a fully probabilistic transition systems (FPTS for short) if $s \rightarrow \pi \wedge s \rightarrow \rho \Rightarrow \pi = \rho$. A rooted PTS (resp. FPTS) $(T, s_0)$is PTS (resp. FPTS) equipped with an initial state $s_0 \in S$. A PTS may be equipped with a proposition assignment $p : S \rightarrow \mathcal{P}(AP)$, where $AP$ is a finite set of atoms and $\mathcal{P}(AP)$the set of propositional formula on $AP$. We define $\models \subseteq S \times \mathcal{P}(AP)$ by $s \models g$ iff $p(s) \Rightarrow g$ is a tautology.*

We write $s \rightarrow$ whenever there is a $\pi$ such that $s \rightarrow \pi$; otherwise, we write $s \nrightarrow$. We let $sup(\pi) = \{s' | \pi(s') > 0\}$. We call $s$ a *sink* state if $s \nrightarrow$.

Let $T = (S, \rightarrow)$ be a PTS. A *simple path* in $T$ is a finite sequence of states $\sigma = s_0 s_1 s_2 \dots s_n$, where for each $0 \leq i < n$ there exists $\pi_i \in Distr(S)$ such that $s_i \rightarrow \pi_i$ and $\pi_i(s_{i+1}) > 0$. Let $\sigma(i)$ denote the state in the $i$-th position. Let $|\sigma|$ be the length of $\sigma$ and let $first(\sigma) = \sigma(1)$ and $last(\sigma) = \sigma(|\sigma|)$. A *simple path starting from* $s \in S$ is a simple path $\sigma$ with $\sigma(1) = s$. A state $t$ is *reachable* from another state $s$ in $T$ if there is a simple path in $T$ with $s = first(\sigma)$ and $t = last(\sigma)$.

A *full path in* $T$ is a sequence of states $\sigma$ being either a simple path with $last(\sigma) \nrightarrow$, or a infinite sequence. We denote by $s - paths(T)$ and $f - paths(T)$ the sets of simple paths and fullpaths in $T$ starting from $s$. Let $reach(T, s)$ denote the set of all states reachable from $s$ in $T$.

We now define a probability measure on the full paths of a FPTS $F$. For any simple path $\sigma \in s - paths(F)$, we define $\sigma_\uparrow = \{\pi \in f - paths(F) | \sigma \leq \pi\}$ where $\leq$ is the classical prefix order on sequences. Let $\mathcal{F}(F)$ be the smallest $\sigma$-fied on $f - paths(F)$ which contains $\sigma_\uparrow$ for each $\sigma \in s - paths(F)$. Then for any state $s$ of $F$, $P_{F,s}$ is the uniquely defined probability measure on $\mathcal{F}(F)$ such that for any $\sigma = s_0 s_1 \dots s_n \in s - paths(F)$ such that $s_i \rightarrow_F \pi_i$ for all $i, 0 \leq i < n$:

$$\mathrm{P}_{F,s}(\sigma_\uparrow) \triangleq \textbf{if } (s = s_0) \textbf{then} \pi_0(s_1) \cdot \pi_1(s_2) \cdot \ldots \cdot \pi_{n-1}(s_n) \textbf{else } 0$$

We will write $\mathrm{P}_{F,s}(\sigma)$ to denote $\mathrm{P}_{F,s}(\sigma_\uparrow)$. Intuitively, $\mathrm{P}_{F,s}(\sigma)$ is the probability of $\sigma$ in $F$ starting from $s$.

## 3.2 Keeping temporal references

As we have seen before, the main difference between both models are the clocks and probabilistic extensions over the well-known transition system. Thus, in order to translate from Hybrid Networks of Timed Automata to Probabilistic Transition Systems, it is necessary to remove the clocks from UPPAAL models, as it is pointed out in [7]. But in this case, it is not possible to check properties consisting of probabilistic and temporal references because PTS cannot deal with clocks.

A possible solution is to translate the clocks to a common part of both models, and both models work with integer variables. Thus, the clocks could be considered as integer variables, and then, we could reset or modify these clocks, and also, we could use them in guards and in invariants as before. From the semantic point of view this is correct, but furthermore we must consider the operational point of view, i.e. how the automaton may evolve. The system may evolve using a delay transition or an action transition. There are no problems when the system evolves by making action transitions, despite we must check the guards that may use clocks. But, if the system evolves by making a delay transition, the clocks cannot evolve, i.e. no time is elapsed, owing to the fact that they are now integer variables.

Therefore, we must consider a technique that allows the system to simulate the passage of time. An approach to deal with this problem, is to add a loop transition in each node to increase the clocks. These new transitions do not have any guards or synchronizations. But, each loop have an assignment that increase all clocks, regardless of they are global or local to one automaton.

Actually, the technique used in the translation is a "Integer Semantic", which is valid as a basic model to be extended with probabilities. The obtained output from the translation process will be the input for the probabilistic tools, after adding the probabilistic extensions. Then, we can specify and check properties with probabilistic and temporal references.

# 4. AN E-COMMERCE STUDY CASE: THE SET PROTOCOL

Electronic commerce became a buzzword as the information society developed rapidly throughout the 1990s. Internet has made e-commerce available to a wider user group, notably smaller enterprises and households. Amongst the business community the search for increased productivity and efficiency is expected to lead to even more enterprises adopting e-commerce as a way of doing business in the future. Whilst an ever growing awareness of the opportunities, technological developments in infrastructure and access devices, and falling access costs will facilitate this, fears about security and a lack of skills could hold it back. Therefore an important goal in e-commerce is *security* [3, 14] and in order to ensure it, the current society is spending an increasing amount of research resources.

According to EITO estimates, e-commerce on the Internet was valued at 172 billion EUR in 2001 in European Union, close to 2% of GDP (Gross Domestic Product). In that way safety errors on e-commerce could be very expensive. But we can use *system validation* in order to avoid it [4, 15]. *System validation* is the process of determining the correctness of specifications, designs and products. Furthermore, it is a technique to support the quality control of the system design. A technique that implements *system validation* is **Model Checking** [12, 8]. Now, we will see how we can apply this technique to a security protocol, but first we will describe this protocol.

Secure Electronic Transaction (SET) is a system for ensuring the security of financial transactions on the Internet. It was supported initially by Mastercard, Visa, Microsoft, Netscape, and others. With SET, a user is given an electronic wallet (digital certificate) and a transaction is conducted and verified using a combination of digital certificates and digital signatures among the purchaser, a merchant, and the purchaser's bank in a way that ensures privacy and confidentiality. SET makes use of Netscape's Secure Sockets Layer (SSL), Microsoft's Secure Transaction Technology (STT), and Terisa System's Secure Hypertext Transfer Protocol (S-HTTP). SET uses some but not all aspects of a public key infrastructure (PKI). SET works in the following way:

Let us assume that a customer has a SET-enabled browser, such as Netscape or Microsoft's Internet Explorer, and that

the transaction provider (bank, store, etc.) has a SET-enabled server.

1. The customer opens a Mastercard or Visa bank account. Any issuer of a credit card is some kind of bank.
2. The customer receives a digital certificate. This electronic file functions as a credit card for online purchases or other transactions. It includes a public key with an expiration date. It has been through a digital switch to the bank to ensure its validity.
3. Third-party merchants also receive certificates from the bank. These certificates include the merchant's public key and the bank's public key.
4. The customer places an order over a Web page.
5. The browser of the customer receives and confirms that the merchant is valid from its certificate.
6. The browser sends the order information. This message is encrypted with the merchant's public key, the payment information, which is encrypted with the bank's public key (which cannot be read by the merchant), and information that ensures the payment can only be used with this particular order.
7. The merchant verifies the customer by checking the digital signature on its certificate. This may be done by referring the certificate to the bank or to a third-party verifier.
8. The merchant sends the order message to the bank. This includes the bank's public key, the customer's payment information (which the merchant cannot decode), and the merchant's certificate.
9. The bank verifies the merchant and the message. The bank uses the digital signature on the certificate with the message and verifies the payment part of the message.
10. The bank digitally signs and sends authorization to the merchant, who can then fill the order.

The set of participants in a SET protocol are represented in figure 2.

**Cardholder** A cardholder uses a payment card that has been issued by the Issuer. SET ensures that the interactions of cardholder with the merchant and the payment card information remain confidential.

**Issuer** It is a financial institution. It establishes an account for the cardholder. It also issues the payment card. The Issuer also guarantees payment for authorized transactions using the payment card.

**Merchant** The merchant offers goods or services in exchange for a payment. In order to accept payment cards the merchant must have a relationship with an Acquirer.

**Acquirer** A financial institution that establishes an account with a merchant and process payment card authorizations and payments.

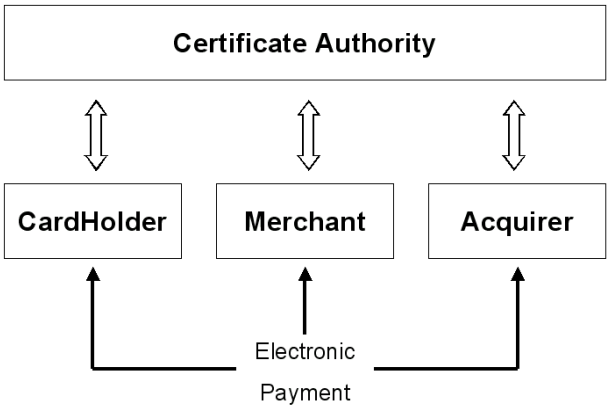


**Figure 2: Participants in a SET protocol & their Interactions.**

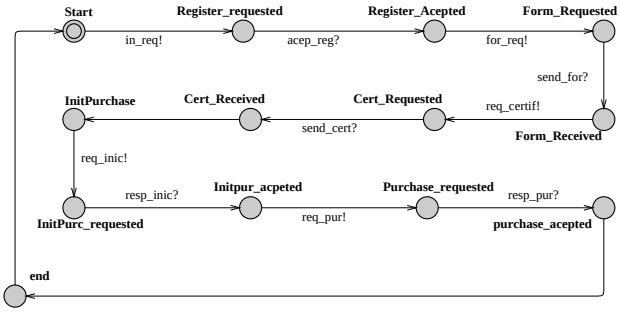


**Figure 3: CardHolder Template.**

**Payment Gateway** It is a device operated by an Acquirer or a designated third party that processes merchant payment messages, including payment instructions from cardholders.

**Brand** Financial Institutions came up with payment card brands in order to protect and advertise the brand. It creates an atmosphere conducive to establishing and enforcing rules for use and acceptance of their respective payment cards. It also provides a network to interconnect the financial institutions.

**Third Parties** Issuers and Acquirers sometime assign processing of payment card transactions to these third-party processors. [16]

Once we have described the protocol, we can validate the protocol with UPPAAL and RAPTURE, after translating it.

## 4.1 Modelling, Validating, Translating and Verifying in UPPAAL and RAPTURE

To **model** this protocol in UPPAAL we have used the formal description that is available at http://www.setco.org [16, 17, 18]. In figures 3,4,5 and 6 we can see the main processes involved in the protocol.

Then, we start by checking that the model we have constructed satisfies the expected behavior. This task is accom-

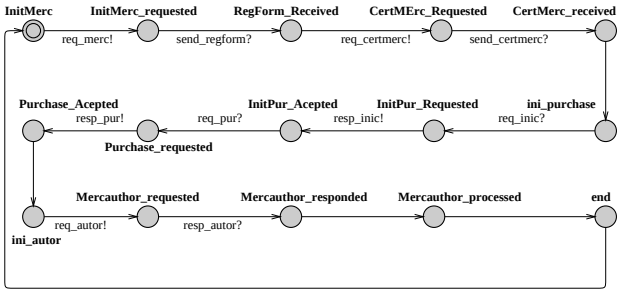


**Figure 4: Merchant Template.**

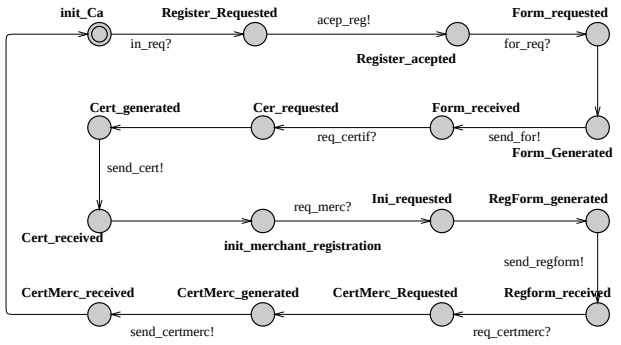


**Figure 5: Certificate Authority Template.**

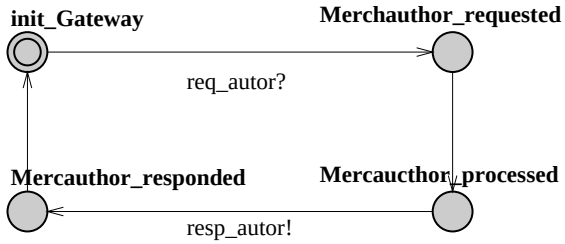


**Figure 6: Payment Gateway Template.**

plished by simulation. Simulations are made by choosing different transitions and delays along the system evolution. At any moment during the simulation, we can see the variable values and the enabled transitions. Thus, we can choose the next transition to be executed. But we have also the possibility to make an automatic execution, and thus, the system evolves by executing transitions and delays which are selected randomly. Then, with respect to our model of the SET protocol, our main goal in the validation phase is to check the correctness of the message flow, taking into account the protocol definition. We have made a number of simulations; and we have concluded that the system design satisfies the expected behavior in terms of the message flow between the Client and the Server.

To **verify** the protocol in UPPAAL we must establish the properties that the model must fulfill. We have divided these properties into three classes: Safety, Liveness and Deadlocks. These properties are specified by means of a *Temporal Logic*. The temporal Logic used by UPPAAL is described in [10].

**Safety Properties** allow us to check if our model satisfies some security restrictions. For example, if we have two trains that have to cross the same bridge, a security property is that both trains cannot cross at the same time the bridge which is specified by: $\forall\Box\neg(Train1.crossing \wedge Train2.crossing)$ or
$\neg\exists\Diamond(Train1.crossing \wedge Train2.crossing)$

In our case study the main safety properties are:

- A Merchant cannot start the identification process if CardHolder has not started the identification process:

$$\forall\Box CardHolder.RegisterRequested \Rightarrow Merchant.InitMercRequested \quad (1)$$

- A Certificate Authority sends the registration form immediately after the Merchant starts the identification:

$$\forall\Diamond(Merchant.InitMercRequested \Rightarrow CertificateAuthority.RegFormReceive) \quad (2)$$

- A Payment Gateway cannot finish authorization if the Merchant has not requested:

$$\forall\Box Merchant.InitAuthority \Rightarrow PaymentGateway.MerchAuthorResponded \quad (3)$$

**Liveness Properties** check that our model can evolve in the right order. Returning to the train example, if a train approaches the bridge, some time later, the train could cross it: $Train.approach \rightarrow Train.crossed$

Liveness Properties for our model are simple. If a CardHolder sends the message *Register Request*, some time later, we must obtain the message *Purchase Accepted*. Translating this into Temporal Logic we obtain:

$$CardHolder.RegisterRequested \longrightarrow CardHolder.PurchaseAccepted \quad (4)$$

On the other hand, if a Merchant sends a registered request, some time later, the Certificate Authority sends a certificate.

$$CertificateAuthority.RegisterRequested \longrightarrow CertificateAuthority.CertMercReceived \quad (5)$$

**Deadlocks** are clear restrictions, which can be checked by:

$$\forall\Box\neg Deadlock \quad (6)$$

To **translate** the UPPAAL model to RAPTURE we have used the technique that we have explained before and we have modified it by adding probabilities. Finally, we have verified a property that merge probabilistic and real time references. This property is: "Is it possible to reach the state *Purchase Accepted* from *Register Requested* within 1340 time units with a probability greater than 70%?". For this question we have obtained a positive answer.

## 5. CONCLUSION

In this paper we have presented the translation between the models used in the UPPAAL ant the RAPTURE tools. We have seen the necessity to use new techniques in order to make this possible. With this translation we have some advantages and some drawbacks. The main advantage is that the used integer semantics allows us to check probabilistic and temporal properties together. And the main drawback is that the integer semantics is not adequate for some systems. Therefore, we must use these techniques in the proper manner.

Furthermore, we have presented the validation and verification of the SET Protocol. The properties verified ensure the correct message flow. The verification techniques used in this paper, not the translation, are not particularly novel, and they have been used in some industrial and theoretical use cases and protocols. Then we may ask ourselves why incorrect protocols and software are still appearing. One reason seems to be that protocol developers and developers in general, are failing to learn from the mistakes of others. Another reason and perhaps the most significant one, is that they are usually unaware of formal techniques of verification, or even they are applied incorrectly.

As future work we will develop a new tool in order to model, validate and verify protocols and other systems. This tool is called P_UPPAAL [7]. It includes a new feature with respect to UPPAAL, P_UPPAAL models have capability to work directly with Probabilistic Real-Time Systems, but as shown before, it is necessary to implement new verification techniques yet.

## 6. ACKNOWLEDGMENTS

This work has been supported by spanish grants CYCIT (ref. TIC 2003-07848-C02-02) and JCCM (ref PAC-03001).

## 7. REFERENCES

[1] K. Larsen B. Jonsson and W. Yi. Probabilistic process algebra. E-HPA, chapter 4: Extensions, pages 685–710. 2001.

[2] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, Yi Wang, and C. Weise. New Generation of Uppaal. In *Int. Workshop on Software Tools for Technology Transfer*, June 1998.

[3] D. Bolignano. Towards the formal verification of electronic commerce protocols. In *Proc. 10th IEEE Computer Security Foundations Workshop*, 1997.

[4] P. Broadfoot and G. Lowe. On distributed security transactions that use secure transport protocols, 2003.

[5] P. D'Argenio, B., H. Jensen, and K. Larsen. Reduction and refinement strategies for probabilistic analysis. In *Process Algebra and Probabilistic Methods - Performance Modelling and Verification, PAPM-PROBMIV 2002*, volume 2399 of *LNCS*, Copenhagen (Denmark), July 2002.

[6] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *PAPM-PROBM 2001*, volume 2165 of *LNCS*, Aachen (Germany).

[7] G. Díaz, D. Cazorla, F. Pelayo, F. Cuartero, and V. Valero. Verifying and capturing probabilistic bechaviours of real-time systems. In *19th Annual UK Performance Engineering Workshop*, 2003.

[8] G. Díaz, F. Cuartero, V. Valero, and F. L. Pelayo. Automatic verification of the tls handshake protocol. In *Proceedings of the 19th ACM Symposium on Applied Computing (SAC 2004)*, Nicosia (Cyprus).

[9] B. Jeannet, P. D'Argenio, and K. Larsen. RAPTURE: A tool for verifying markov decision processes. In *Tools Day, International Conference on Concurrency Theory, CONCUR'02*, Brno (Czech Republic), August 2002. Technical Report, Faculty of Informatics at Masaryk University Brno.

[10] K. Larsen, P. Pettersson, and Wang Yi. Uppaal in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, October 1997.

[11] F. Larsson, K. Larsen, P. Pettersson, and Wang Yi. Efficient Verification of Real-Time Systems: Compact Data Structures and State-Space Reduction. In *Proc. of the 18*th *IEEE Real-Time Systems Symposium*, 1997.

[12] G. Lowe. Towards a completeness result for model checking of security protocols. In *Proc. of The 11th Computer Security Foundations Workshop*, 1999.

[13] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Willey series in probability and mathematical statistics. Jhon Wiley and Sons, 1994.

[14] A. W. Roscoe. Proving security protocols with model checkers by data independence techniques. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 1998.

[15] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2001.

[16] Mastercard & VISA. Set secure electronic transaction specification: Business description, 1997.

[17] Mastercard & VISA. Set secure electronic transaction specification: Formal protocol definition, 1997.

[18] Mastercard & VISA. Set secure electronic transaction specification: Programmer's guide, 1997.