Probabilistic Attack Scenarios to Evaluate Policies over Communication Protocols

Samir Ouchani, Yosr Jarraya, Otmane Ait Mohamed and Mourad Debbabi
Computer Security Laboratory (CLS), Hardware Verification Group (HVG)
Concordia Universiy, Montreal, Canada
Email: {s_oucha,y_jarray,ait,debbabi}@ece.concordia.ca

Abstract— **Security is an important non-functional requirement that should be analyzed in any system or software that is potentially exposed to security threats. Since we can't manage what we don't measure, it is not enough to address only the qualitative assessment of security. In this paper, we propose a novel approach that leads to a qualitative and quantitative analysis of communication protocols. Our approach is based on probabilistic model-checking and probabilistic attack scenarios. To the best of our knowledge, the present work is the first initiative that combines these two techniques in the verification of security of communication protocol. Considering that security attacks are random in nature, we quantify this randomness using probability values denoting the likelihoods of attacks to occur. The composed model formed by the attack scenario and the system model is then analyzed using the probabilistic model-checker PRISM against a set of security and performance requirements. As a case study, we demonstrated the applicability of our approach on Secure Real-time Transport Protocol over Real-Time Streaming Protocol (RTSP/SRTP).**

I. Introduction

The application-level protocols are mainly used for control and ensure the delivery of data with real-time properties that respect protocol policies. They are widely used by many applications such as streaming video, IP telephony, video conferences, internet radio and distance learning. Currently, they provide an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video based applications. The main problem that they suffer from is security which affects the application performance. In this paper, we address the issue of security evaluation of a communication protocol based on their behavioural models to analyze how well a protocol is meeting its security requirements. Furthermore, we would like to prove that the protocol is free from deadlocks [1] then improve network performance criteria such as communication quality due to the fact that a "5% rate inefficiency causes a significant degradation in audio/video quality" [2].

From a security perspective, a strong system is one in which the cost of an attack is greater than the potential gain to the attacker. Conversely, a weak system is one where the cost of an attack is less than the potential gain. The cost of an attack should take into account not only money, but also time of recovery and potential for criminal punishment [3]. Current research initiatives focus mainly on qualitative model checking to ensure the correctness of the protocol under study, while security evaluation of a communication protocol is much less common.

Following the characteristic of a communication protocol, we address security issues by running an attack scenario composed of a set of proposed attacks against the protocol model using Model Checking technique. Model Checking [4], [5] is a formal verification technique that can detect design faults that are difficult to discover otherwise. In addition, it is a counterexample-based technique. Basically, it verifies a properties/constraints against a model through exhaustive state-space search exploration, and generates a trace of states called a counterexample when the property is violated.

In this work, we use the PRISM model checker [6] to quantify the protocol security by modeling the protocol as a probabilistic timed automata [7] (PTA) that interacts with the attacks which are also expressed as PTAs. Then, security properties are expressed using probabilistic temporal logic PCTL [5]. We selected PTAs as the best formalism to express the behaviour of a communication protocol because it's easy to exhibit the main protocol features such as non-determinism and probabilistic choice. PTAs can be seen as an extension to probabilistic automata and finite state machine. In addition, PTAs supports the notion of *time* which provides the real time flavor for the protocol. In this paper, in addition to the verification contribution, we provide a formal definition for PRISM as well as a formal semantic of an attack scenario. To our knowledge, this is the first time this has been done.

The remainder of this paper is organized as follows: Section II presents the related work. Section III describes our approach. The attack scenario is formally presented in Section IV and the probabilistic verification technique is detailed in Section V. The PRISM semantic is given in the Section VI. Section VII presents the results of the application of our approach on Secure Real-time Transport Protocol over Real-Time Streaming Protocol (RTSP/SRTP). Finally, we conclude the paper in Section VIII and present our future works.

II. Related Work

In this section, we cite different works related to our work and the ones concerning RTSP modeling and network attacks in RTSP.

Chaki and Data [8] presented ASPIER, a model-checking based framework dedicated to security protocol. The tool analysis authentication and secrecy properties. The framework supports two phases: protocol compilation and verification. In the first, the C program describing the protocol is translated into ASPIER protocol language by extracting its corresponding Control Flow Graph. Secondly, CEGAR approach is applied to verify the ASPIER model by involving predicates to abstract the real behaviour. But, this tool is limited only to two security requirements are secrecy and authentication.

Clarke et al. [9] presented BRUTUS, a tool for verifying properties of security protocols. First, a honest principal with adversaries are modeled. Next, the protocol requirements are specified by using first-order logic that includes past-time modal operator. BRUTUS explores the state space using depth-first search with integration of partial-order with symmetry reductions. This tool don't cover security quantification and how adversaries are modeled.

Akbarzadeh and Azgomi [10] use security protocol language to describe a protocol with a possible attack. This presentation is transformed into a variety of stochastic Petri nets called the colored stochastic activity networks (CSAN) supported by PDETool. They use PRISM to verify the generated CSAN model but they didn't show how the possible attacks are selected and how security can be evaluated.

Norman and Shmatikov [11] use PRISM model checker to analyze property as a contract signing protocol: fairness, timeliness, Rabins probabilistic protocol for fair commitment exchange. The probabilistic contract signing protocol of Ben-Or, Goldreich, Micali, and Rivest; (BGMR) and a randomized protocol for signing contracts of Even, Goldreich, and Lempel. This work is restricted in a specific protocol which is contract signing protocol.

Xin et al. [12] proposed a performance analysis framework for RTSP-based applications. It is mainly based on four modules which include: a protocol identification module, an application layer session management module, an attack model module and an attack analysis module. The measured performance is the time delay between the occurrence of an attack and its detection. The authors didn't show how those modules work and interact. Lei and Dejian [13] implemented a Distributed Denial-of-Service tool for streaming applications. The tool measures streaming media applications performance by taking into consideration evaluated metrics such as: memory cost, time cost per disk read/write, CPU cost and current send rate. This proposed tool is limited to the performance instead of security. Bilien et al. [14] studied the possibility of establishing a secure VoIP telephone call using SIP. The security was introduced either by SRTP or IPSec and measured with both TCP and TLS. The measurements of the delays shows that the call establishment delay will not be significantly affected by introducing these security protocols. We observe that their work focuses only on delay. Turki and Abdul Waheed [14] showed the verification of RTSP without security features by modeling its behaviour using the PROMELA modeling language. The obtained model is then fed to the SPIN model checker. Since, the results given by SPIN are qualitative, it's difficult to evaluate the security risk of a protocol.

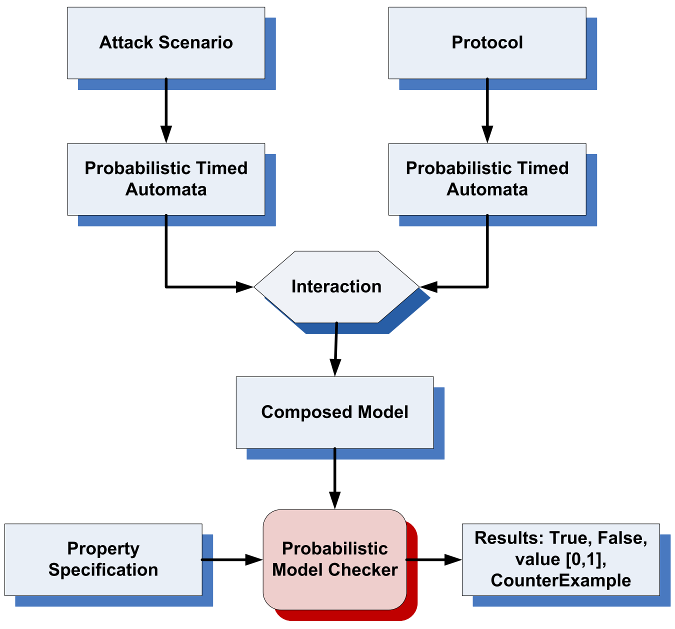
Attack Scenario
Protocol
Probabilistic Timed Automata
Probabilistic Timed Automata
Interaction
Composed Model
Property Specification
Probabilistic Model Checker
Results: True, False, value [0,1], CounterExample

Figure 1. Security Policies Evaluation Approach

The cited works in contrast to our approach do not handle the models' interaction with probabilistic attacks and security estimation based on probabilistic symbolic checkers for streaming/communication protocols.

III. Approach

We present in this section our approach for the security policies verification and performance analysis of a communication protocol designed as probabilistic timed automata (PTA). In Figure 1, the proposed approach is illustrated. The approach consists in mapping the studied protocol along with its related attack scenario (studied in next section) to a corresponding composed PTA. Also, it can be extended to other mathematical formalisms such as priced probabilistic timed automata. In addition, it is supported by the most known probabilistic model checker such as UPPAAL [15] and PRISM [6]. From our analysis, we found that PRISM[1] is more scalable than UPPAAL while it is probabilistic symbolic-based model checker.

The combined PTA of the resulting model is encoded into the PRISM input language. The policies requirements are expressed in PCTL in order to evaluate the security requirements, the probabilities for best/worst cases.

IV. Attack Scenario

In the present section, we present in detail the attack behavior and show its impact on a specific application

[1]22486 downloads of PRISM to October 11, 2011

such as RTSP. As defined in [16], an attack is an attempt to gain unauthorized access to an Information System's (IS) services, resources, or information or the attempt to compromise an ISs integrity, availability, or confidentiality, as applicable. Formally, the behaviour of an attack scenario can be described by a compostion of PTAs and can be expressed with the following BNF syntax form:
$Att = \epsilon \,|\, att_i \vee_p att \,|\, att_j \wedge att \,|\, att_k \cdot att$
where:

- ϵ is the empty attack
- $att_i \vee_p att$ means the attack att_i can be executed by a probability (p) else att will be launched by a probability (1-p)
- $att_j \wedge att$ means the attack att_j executes in parallel with att
- $att_k \cdot att$ means the attack att execute when att_j will finish

The behaviour of each attack is a PTA where the atomic action of an attack are: send a message($!m$), receive a message ($?m$), or either modify a message ($m = y$). In the definition 4.1, we present the formal definition of a PTA.

Definition 4.1 (Probabilistic Timed Automata):
A probabilistic timed automata (PTA) is a tuple $M = (\mathrm{S}, \overline{s}, \alpha_{\mathrm{M}}, \delta_{\mathrm{M}}, \mathrm{L})$ where:

- S is a finite set of states,
- $\overline{s}$ is an initial state,
- α_{M} is a finite alphabet,
- $\delta_{\mathrm{M}} \subseteq \mathrm{S} \times \alpha_{\mathrm{M}} \times Dist\,(S)$ is a probabilistic transition relation,
- $\mathrm{L} : \mathrm{S} \rightarrow 2^{AP}$ is a labelling function mapping each state to a set of atomic propositions taken from a set AP constrained with time.

The composition of more than one PTA is defined as follow:

Definition 4.2 (Parallel Composition of PTAs):
Let $M_1 = (\mathrm{S}_1, \overline{s_1}, \alpha_{\mathrm{M}_1}, \delta_{\mathrm{M}_1}, \mathrm{L}_1)$ and $M_2 = (\mathrm{S}_2, \overline{s_2}, \alpha_{\mathrm{M}_2}, \delta_{\mathrm{M}_2}, \mathrm{L}_2)$ are two PTAs. Their parallel composition $M_1 \parallel M_2$ is a PTA $M = (\mathrm{S}_1 \times \mathrm{S}_2, (\overline{s_1}, \overline{s_2}), \alpha_{\mathrm{M}_1} \cup \alpha_{\mathrm{M}_2}, \delta_{\mathrm{M}_1 \parallel \mathrm{M}_2}, \mathrm{L})$ where $\delta_{\mathrm{M}_1 \parallel \mathrm{M}_2}$ is defined such that $(s_1, s_2) \xrightarrow{a} \mu_1 \times \mu_2$ if and only if the following holds:

- $s_1 \xrightarrow{a} \mu_1$ and $s_2 \xrightarrow{a} \mu_2$ and $a \in \alpha_{\mathrm{M}_1} \cap \alpha_{\mathrm{M}_2}$
- $s_1 \xrightarrow{a} \mu_1$ and $\mu_2 = \eta_{s_2}$ and $a \in \alpha_{\mathrm{M}_1} \backslash \alpha_{\mathrm{M}_2} \cup \tau$
- $s_2 \xrightarrow{a} \mu_2$ and $\mu_1 = \eta_{s_1}$ and $a \in \alpha_{\mathrm{M}_2} \backslash \alpha_{\mathrm{M}_1} \cup \tau$
- $L(s_1, s_2) = L(s_1) \cup\ L(s_2)$

We can define, a probabilistic choice between at least two PTA's (PTA_1 and PTA_2) as a PTA with a special characteristic. The initial state of the global PTA has two probabilistic paths. One with a probability (p) and followed by PTA_1, the second one with a probability (1-p) and followed by PTA_2. The formal composition with a probabilistic choice is given in the Definition 4.4.

Definition 4.3 (Probabilistic choice between PTAs):
Let $M_1 = (\mathrm{S}_1, \overline{s_1}, \alpha_{\mathrm{M}_1}, \delta_{\mathrm{M}_1}, \mathrm{L}_1)$ and $M_2 = (\mathrm{S}_2, \overline{s_2}, \alpha_{\mathrm{M}_2}, \delta_{\mathrm{M}_2}, \mathrm{L}_2)$ are two PTAs. The probabilistic choice between M_1 and M_2 is $M_1 \vee_p \ M_2$ in a given state s_{ref} is a PTA $M = ((\mathrm{S}_1 \cup \overline{\mathrm{s}_1}) \times\ (\mathrm{S}_2 \cup \overline{\mathrm{s}_2}), \overline{s_{ref}}, \alpha_{\mathrm{M}_1} \cup \alpha_{\mathrm{M}_2}, \delta_{\mathrm{M}_1 \parallel \mathrm{M}_2}, \mathrm{L})$ where $\delta_{\mathrm{M}_1 \vee_{\mathrm{p}} \mathrm{M}_2}$ is defined such that:

- $s_{ref} \xrightarrow{p} \overline{s_1}$
- $s_{ref} \xrightarrow{1-p} \overline{s_2}$

The sequential Composition of at least two PTA's (PTA_1 and PTA_2) is a PTA where the final state of PTA_1 is replaced by the first state of PTA_2. Its formal definition is given in the Definition 4.4.

Definition 4.4 (Sequential Composition of PTAs):
Let $M_1 = (\mathrm{S}_1, \overline{s_1}, \alpha_{\mathrm{M}_1}, \delta_{\mathrm{M}_1}, \mathrm{L}_1)$ and $M_2 = (\mathrm{S}_2, \overline{s_2}, \alpha_{\mathrm{M}_2}, \delta_{\mathrm{M}_2}, \mathrm{L}_2)$ are two PTAs. The probabilistic choice between M_1 and M_2 is $M_1 . M_2$ in a given state s_{ref} is a PTA $M = (\mathrm{S}_1 \times\ (\mathrm{S}_2 \cup \overline{\mathrm{s}_2}), \overline{s_1}, \alpha_{\mathrm{M}_1} \cup \alpha_{\mathrm{M}_2}, \delta_{\mathrm{M}_1 \parallel \mathrm{M}_2}, \mathrm{L})$ where $\delta_{\mathrm{M}_1 \vee_{\mathrm{p}} \mathrm{M}_2}$ is defined such that:

- $Final(M_1) \rightarrow \overline{s_2}$

In the following we discuss different attacks related to the RTSP protocol. For each attack represented as a probabilistic automata, a time constraint in the form of $(t < value)$ is added to each transition[2].

A. Flooding attack

This attack aims at depleting the resources of RTSP services so that they become unavailable for processing legitimate requests or it forces the service's processing time to be considerably extended. In Figure 2, we present the behaviour of this attack for the RTSP server. The attacker has two choices to flood the server either by DESCRIBE messages with a probability of measure p_1 or by SETUP messages with a probability of measure $1-p_1$. If the attacker fail, it can repeat sending the same message with a probability of p_2 for DESCRIBE messages or p_3 for SETUP messages. Also, it can end the attack by a probability of $1 - p_2$ (or $1 - p_3$ for the second one).

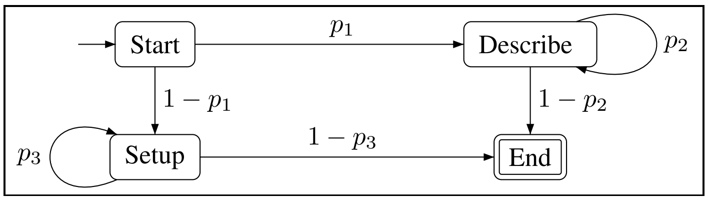

Figure 2. Behaviour of Flooding attack

B. Session Hijacking attack

An attacker builds a message such that it is able to masquerade as an authorized message from a trusted principal. As a result, consumers of these messages can be manipulated into responding or processing the deceptive message. The attacker can receive a session ID by a probability (P) then modify it by a probability (P1) and send it by a probability (P2). Finally, it can have all the rights that a legal client has and access the server resource.

[2]for the clarity of the figure, those constraints are not shown

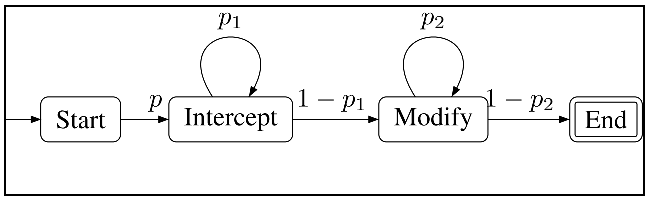

Figure 3. Behaviour of Session Hijacking attack

C. TEARDOWN attack

TEARDOWN attack terminates the communication session of the client. We show in Figure 4 the behaviour of a TEARDOWN attack against an RTSP server. The attacker starts by preparing to send a TEARDOWN message. In the case where the intruder has no knowledge regarding this attack's success, the attacker could repeat sending the TEARDOWN messages with a probability p or the messages may stop which would end the cycle by a probability of $1 - p$.

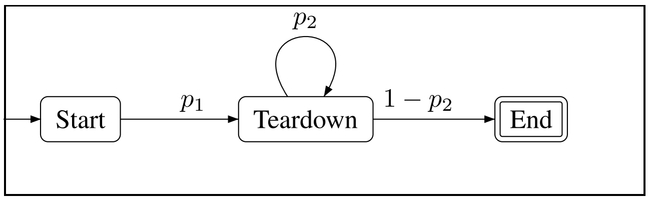

Figure 4. Behaviour of TEARDOWN attack

D. Example of an Attack Scenario

Basing on the previous attacks, we construct the attack scenario related to RTSP protocol as presented by in Figure 5. It is composed from four attacks are: 1) Describe flooding attack, 2) Setup hijacking session attack, 3) intercept RTSP message followed by teardown attack, and 4) play or pause flooding attack. Each attack can be launched by a uniform distribution 0.25 each.

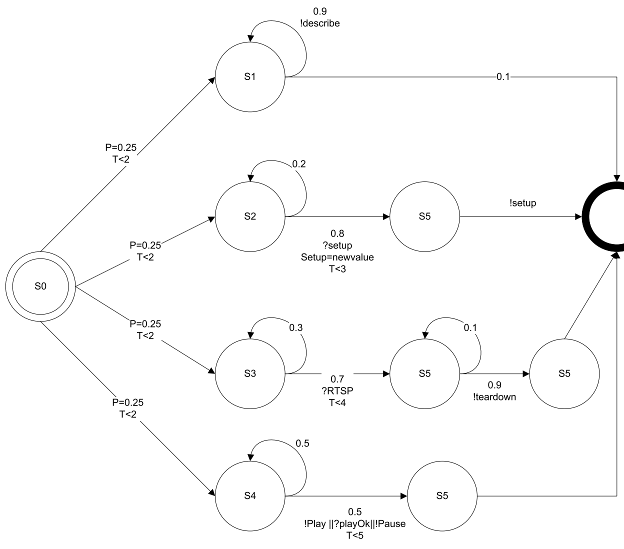

Figure 5. RTSP Attack Scenario

V. Probabilistic Verification

The actual probabilistic model checkers such as PRISM are mainly based on the stochastic version of the classical shortest path problem. This problem firstly is formulated by Eaton and Zadeh [17] who called it the problem of *pursuit*.

In this section, we introduce probability computation in a symbolic model checker. It proceeds by induction on the parse tree of the formula, as in the case of CTL model checking [4]. To show that, we select the MDP as a special [18] formalism of probabilistic automata that exhibit both probabilistic and nondeterministic behaviour. It is defined in the Definition 5.1.

Definition 5.1 (Markov Decision Process): A Markov decision process (MDP) is a tuple $M = (\mathrm{S}, \overline{s}, \alpha_{\mathrm{M}}, \delta_{\mathrm{M}}, \mathrm{L})$ where:

- S is a finite set of states,
- $\overline{s}$ is an initial state,
- α_{M} is a finite alphabet,
- $\delta_{\mathrm{M}} : \mathrm{S} \times \alpha_{\mathrm{M}} \rightarrow Dist\,(S)$ is a (partial) probabilistic transition function,
- $\mathrm{L} : \mathrm{S} \rightarrow 2^{AP}$ is a labeling function mapping each state to a set of atomic propositions taken from a set AP.

To reason formally about MDPs, we need a probabilistic space over it. And, as it is a nondeterministic behaviour, the *adversary* notion is introduced to decide which action should be chosen in any state of the MDP. In general, the choice is made depending on the history execution of the MDP. The Definition 5.2 describes the adversary function.

Definition 5.2 (Adversary): An adversary of an MDP $M = (\mathrm{S}, \overline{s}, \alpha_{\mathrm{M}}, \delta_{\mathrm{M}}, \mathrm{L})$ is a function $\sigma : FPath_M \rightarrow Dist(\alpha_M)$ that maps every finite path of the system onto a distribution where:

- $\sigma(\rho)(a) > 0$ only if $a \in \ A(last(\rho))$,
- $FPath_M$ is a finite set of nodes (states),
- $Dist(\alpha_M)$ is a labeled function assigning to each node of the automaton the set of atomic propositions that are true in that node.

Reachability analysis is the kernel of a model checker, and the probabilistic reachability refers to the minimum/maximum probability with which a given set of states of a probabilistic system ($T \subseteq \ S$) can be reached from a particular state (s). To this end, $reach_s\,(T)$ is the set of paths that starts from s and contains a state from T.

$reach_s\,(T) = \{\pi \in IPath_{M,s} | \pi\,(i) \in T and i \in \mathbb{N}\}$
$= \bigcup_{\rho \in\, I} \{\pi \ \in \ IPath_{M,s} | \pi\, has\, prefix\, \rho\}$, where I is the (countable) set of all finite paths from s ending in T, and each element of this union is measurable. And, this is equivalent to compute the probabilistic bounds of the reached paths:

$$P^{min}_{M,s}(reach_s\,(T)) \quad = \quad inf_{\sigma \in Adv_M} Prob^{\sigma}_{M,s}(s, \psi) \qquad (1)$$
$$P^{max}_{M,s}(reach_s\,(T)) \quad = \quad sup_{\sigma \in Adv_M} Prob^{\sigma}_{M,s}(s, \psi) \qquad (2)$$

In fact, find the probability $x_s = P^{min}_{M,s}(reach_s\,(T))$, $s \in S$ is the unique solution of the following linear programming problem:

$\max \Sigma_{s \in S} x_s$
$x_s = 1 \ \& \forall s \in S^{s=1}_{\min}$
$x_s = 0 \ \& \forall s \in S^{s=0}_{\min}$
$x_s \leq \sum \delta_M(s,a)(s') \cdot x_{s'} \ \forall s \notin (S^{s=1}_{min} \cup S^{s=0}_{min})$

In the case of $x_s = P^{max}_{M,s}(reach_s(T))$, $s \in S$ it is similar to the previous problem following linear programming problem:

$\min \Sigma_{s \in S} x_s$
$x_s = 1 \& \forall s \in S^{s=1}_{\min}$
$x_s = 0 \& \forall s \in S^{s=0}_{\min}$
$x_s \leq \sum \delta_M(s,a)(s') \cdot x_{s'} \ \forall s \notin (S^{s=1}_{min} \cup S^{s=0}_{min})$

Bertsekas and Tsitsiklis [17], prove that this minimum is the unique solution for Bellman's equation and the successive approximation methods converge to the optimal vector. This yield to the fact that the linear programming problem can be solved as an equation system problem. This mean, find the probability $x_s = P^{min}_{M,s}(reach_s(T))$, $s \in S$ is the unique solution of Bellman's equation:

$$X_s = \begin{cases} 1 & \text{if } s \in S^{s=1}_{\min} \\ 0 & \text{if } s \in S^{s=0}_{\min} \\ \min_{a \in A(s)} \sum \delta_M(s,a)(s') \cdot X_{s'} & \text{otherwise} \end{cases} \quad (3)$$

The Computing reachability probabilities can be achieved through three ways: value iteration, the linear programming problem or policy iteration. The first one is the most used in practice due to its approximate algorithm based on an iterative solution method which corresponds to fixed point computation. From a practice experience, the second approach is more scalable than the first one.

To complete the model checker process, a property should be specified to verify if it holds or not, in other case by which percent it can be true. For that, different mechanism are dedicated such as temporal logic and special automata. In our work, we selected a probabilistic extension of CTL temporal logic called PCTL. It is supported by the most tools and its BNF grammar is expressed as follow:

Definition 5.3 (PCTL Syntax): The syntax of PCTL is as follows:

$\phi ::= true | a \mid \phi \wedge \phi \mid \neg\phi \mid P_{\bowtie p}[\psi]$
$\psi ::= X\phi | \phi U^{\leq k} \phi | \phi U \phi$

where a is an atomic proposition, $k \in N$,$p \in [0,1]$, and $\bowtie \in \{<, \leq, >, \geq\}$.

To specify a satisfaction relation of a PCTL formula in a state s, a class of adversaries (Adv) is defined [18]. It is true if it is satisfied under all adversaries of a given MDP.

The satisfaction relation ($\models_{Adv}$) of PCTL is defined [18] inductively as follow:

- $s \models_{Adv} True \quad Always$
- $s \models_{Adv} a \Leftrightarrow a \in L(s)$
- $s \models_{Adv} \phi_1 \wedge \phi_2 \Leftrightarrow s \models_{Adv} \phi_1 \wedge s \models_{Adv} \phi_2$
- $s \models_{Adv} \neg\phi \Leftrightarrow s \not\models_{Adv} \phi$
- $s \models_{Adv} P_{\bowtie p}[\psi] \Leftrightarrow$
- $\pi \models_{Adv} X\phi \Leftrightarrow \pi(1) \models_{Adv} X\phi$
- $\pi \models_{Adv} \phi_1 U^{\leq k} \phi_2 \Leftrightarrow \exists i \geq k.(\pi(i) \models_{Adv} \phi_2 \wedge \pi(j) \models_{Adv} \phi_1 \ \forall j < i)$
- $\pi \models_{Adv} \phi_1 U \phi_2 \Leftrightarrow \exists k \geq 0. \ \pi \models_{Adv} \phi_1 U^{\leq k} \phi_2$

From the basic PCTL syntax, several other useful operators can derived with a logical equivalences, such as:

1) Future: $F\phi \equiv true \ U \ \phi$ and $F^{\leq k}\phi \equiv true \ U^{\leq k} \phi$.
2) Generally: $G\phi \equiv \neg(F\neg\phi)$ and $G^{\leq k}\phi \equiv \neg(F^{\leq k}\neg\phi)$.

Here , we will consider the basic PCTL operators "Next and "Until to compute minimum of probability to reach states that satisfy a formula ψ of type $X\phi$ and $\phi_1 U^{\leq k}\phi_2$. In the case of $\psi = X\phi$, we have: $P^{min}_S(X\phi) = \min_{a \in A(s)} \sum_{s' \in Sat(\phi)} \delta_M(s,a)(s') \cdot s'$

In the case of $\psi = \phi_1 U^{\leq k}\phi_2$, we have:

$$x^l_s = \begin{cases} 1 \text{ if } s \in Sat(\phi_2) \\ 0 \text{ if } s \notin (Sat(\phi_1) \cup Sat(\phi_2)) \\ 0 \text{if } s \in Sat(\phi_1) \setminus Sat(\phi_2) \ and \ l = 0 \\ \min_{a \in A(s)} \sum_{s' \in S} \delta_M(s,a)(s') \cdot x^{l-1}_{s'} \text{otherwise} \end{cases} \quad (4)$$

VI. PRISM SEMANTIC

In this section, we define the PRISM model and its semantic. A system described as PRISM model comprises a set of n modules, the state of each one is defined by an evaluation of a set of finite-ranging local variables. The global state of the system is the evaluation of the union of all local variables V_l in addition tothe global ones V_g, which we denote $V = V_g \cup V_l$. The behaviour of each module is defined by a set of guarded commands and a set of invariants in the case of probabilistic timed automata (PTA) representing the clock constraints.

In MDP as in PA and PTA formalisms, a command takes the following form: [act] guard $\rightarrow p_1 : u_1 + ... + p_m : u_m$, which mean, for the action "act" if the condition "guard" is true, then, the updates "u_i" of the behaviour can be changed by a probability "p_i". Its formal definition is given in the Definition 6.1 to be used next.

Definition 6.1 (PRISM Command): A PRISM command is a tuple $cmd = (act, guard, update)$ where:

- act: is an action label,
- guard: is a predicate over V,
- $update = \{(p_i, u_i) | i < m, \sum_{i=1}^{m} p_i = 1$ and $u_i \in \{(V, \mathbb{N})\}\}$ where $f : V \rightarrow \mathbb{N}$.

A module that describes the behaviour of a sub-part from a system is defined formally in the Definition 6.3.

Definition 6.2 (PRISM Module): A PRISM system is a tuple $M = (var, init, com)$ where:

- var is a finite set of module local variables,
- $init$ is the initial values of $var(M)$,
- $com = \{cmd\}$ is a set of commands that define the behaviour of the module.

A system that contains n sub-parts that each one is described by a module and their relation is described by an algebraic expression. The supported algebraic expression in PRISM are:

1) M1||M2 : It is a parallel composition of modules. M1 and M2 synchronize on only actions appearing in both M1 and M2,
2) M1|||M2: asynchronous parallel composition of M1 and M2 (fully interleaved, no synchronization),
3) M1|$[a, b, ...]$|M2: restricted parallel composition of modules M1 and M2 (synchronizing only on actions from the set a, b,...),
4) M/a,b,... : hiding of actions a, b, ... in module M,
5) Ma¡-b,c¡-d,... : renaming of actions a to b, c to d, etc. in module M.

Finally, the system containing n modules is defined formally in the Definition 6.3.

Definition 6.3 (PRISM System): A PRISM model is a tuple $P = (var, sys, M_1, \dots, M_n)$ where:

- $var(P) = \bigcup_{i=1}^{n} V_{Gi}$ is a finite set of system variables. It is the union of all modules global variables (V_{Gi}).
- sys is algebraic expression that defines the models' communication,
- $M_1, \dots, M_n$ is a countable set of modules.

VII. Real Time Streaming Protocol (RTSP)

The Real Time Streaming Protocol (RTSP) [19] is a client-server application-level protocol for control over the delivery of data with real-time features. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. To deliver the continuous RTSP streams, it is intended to control multiple data delivery sessions, and provide a means for choosing delivery mechanisms based upon RTP; an alternative mechanism is the Secure RTP Profile (SRTP) [20]. SRTP profile is designed to support confidentiality and authentication suitable for use with links that may have relatively high loss rate, and that require header compression for efficient operation. It provides confidentiality of RTP data packets by encrypting the payload part. Furthermore, it supports message integrity protection by appending a message authentication tag to the end of the packet and it supports source origin authentication by using the TESLA authentication algorithm (Timed Efficient Stream Loss-tolerant Authentication).

Secure RTP profile when built within RTSP offers secure retrieval of media from the media server, the invitation of a media server to a conference, and adds additional media to an existing presentation. RTSP requests can be transmitted in several different ways:

- Persistent transport connections used for several request-response transactions
- Transactions with one connection per request/response
- Connectionless mode transactions.

The main methods used to define RTSP vocabulary are:

- DESCRIBE: a request includes an RTSP URL, and the type of reply data that can be handled.
- SETUP: causes the server to allocate resources for a stream and start an RTSP session.

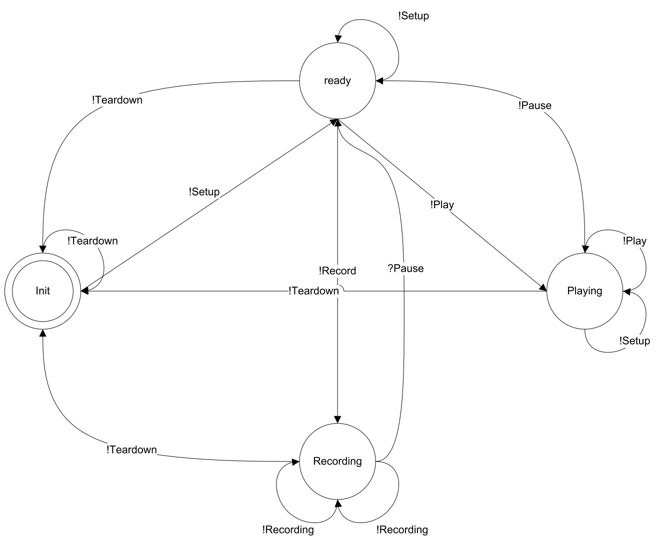

Figure 6. Client State Machine

- PLAY and RECORD: starts data transmission on a stream allocated via SETUP.
- PAUSE: temporarily halts a stream without freeing server resources.
- TEARDOWN: frees resources associated with the stream. The RTSP session ceases to exist on the server.
- SUCCESS and ERROR: server's response to client requests.

From RTSP [19] and SRTP [20] RFC's, we have extracted and designed the behaviour of RTSP upon Secure RTP profile as a state machine. Therefore, Figure 7 presents the main behaviour of server end and Figure 6 shows the client side.

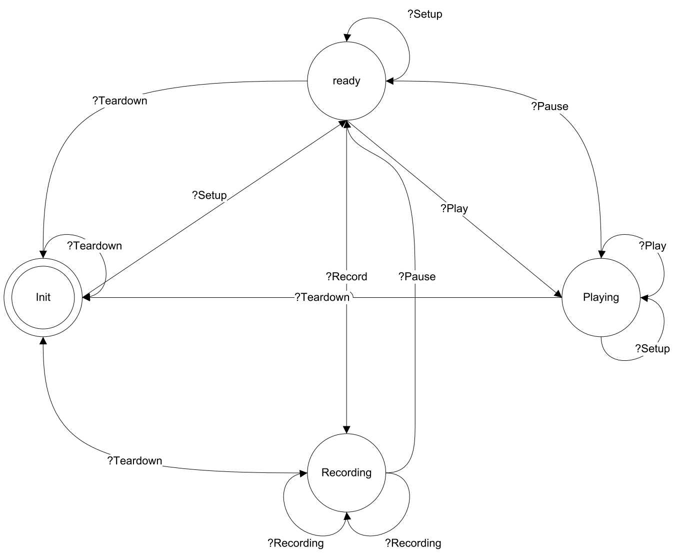

Figure 7. Server State Machine

A. Properties

Here, we propose a set of properties to be verified on the model resulting from the interaction between

Property	1	2	3	4	5	6	7	8	9	10
Result	0	0.02	1	1	1	0.06	0.6	0.6	0.82	0.8

TABLE I.
VERIFICATION RESULT

Client, Server and Attacker models. To achieve that, we express the presented properties by using PCTL temporal logic given by this grammar: $\phi ::= true|a\,|\,\phi \wedge \phi\,|\,\neg\phi\,|\,\mathcal{P}_{\bowtie p}[\psi]\,|\,\mathcal{R}_{\sim r}[F\phi]$ and $\psi ::= \phi|\psi_1\mathcal{U}^t\psi_2|\psi_1\mathcal{U}\psi_2|\mathcal{X}\psi|\psi_1 \wedge \psi_2|\neg\psi$ where: a is an atomic proposition, $t \in N$,$p \in [0,1]$,$\bowtie\in \{<,\leq,>,\geq\}$, and $\mathcal{R}$ represents the rewards operators. The main operators used to express our proposed properties are a mix of propositional logic (!:Not, |:Or, &: And,→: Imply), temporal logic (A: All, E: Exists, X: neXt, G: Globaly, F: Finally, U: Until) and probabilistic temporal logic. The proposed properties are both functional and security related in nature, such as:

1) Deadlock: “The maximum probability to have a deadlock in any state of the model”.
PCTL: Pmax=?[GF”deadlock”]
2) Losing messages: “The maximum probability of losing at least one message?”
PCTL: Pmax=?[F(bs_pos≥5)]
3) “Measure the probability to interrupt viewing media”.
PCTL: Pmin=?[G(r_rtp⇒(X(endclient)))]
4) “What’s the probability that when an attack send pause message the client should not see the media”.
PCTL: Pmin=?[when_client_playing⇒(as_tear)]
5) “ What‘s the probability to hijack a session”.
PCTL: Pmin=? [G(when_client_playing⇒(F(SendSetup) U endclient))]
6) “Measure the ability to intercept an RTP packet”.
PCTL: Pmax=? [F(receivepacket)]
7) “ Estimate the probability of the ability of an attacker to pause the media viewed by one client”.
PCTL: Pmin=? [((when_client_playing)⇒(F(SendPause)))⇒(F(stop))]
8) “Measure the minimum probability to inhibit a client from reading the media”.
PCTL: Pmin=? [G (r_setupok⇒(X (endclient)))]
9) Calculate the minimum probability that an attacker premature a client to disconnect.
PCTL: Pmin=? [G(SendTeardown⇒(X(endclient)))]
10) “Find the minimum probability that an attacker enfore a client to pause viewing”.
PCTL: Pmin=?[G(s_play⇒(F(SendPause)U endclient))]

B. Numerical Results and Discussion

The verification of the above properties are done using the Jacobi method integrated within the PRISM model checker version 3.3.1 to produce a MTBDD (Multi-Terminal Binary Decision Diagram) model with 9524 transitions and 2383 states including the initial state. The attack model takes part in 76 states of the original MTBDD and includes 187 transitions. These techniques reveal the results summarized in Table I, from where we conclude that the model is free from deadlock and does not cause any significant degradation in audio/video quality (first and second property). The RTSP server can be down with 82% by using flooding attack. Furthermore, the attackers probability of intercepting a message is 6% and the client can lose his session’s connection by a probability of 80%.
We conclude by providing a brief description of a number of verification experimental results by tuning three main parameters: the number of messages, the sever memory size and the media size. These parameters are sampled uniformly as illustrated in Figure 8.
We observe that the probability of packet interception evaluated by the sixth property is constant even by changing the number of messages or the media size as depicted in Figure 10. Furthermore, the evaluation of the seventh property by tuning the media size with the message number parameter is showed in Figure 11. We remark that the probability measured doesn’t change as a function of the # of messages and the media size.

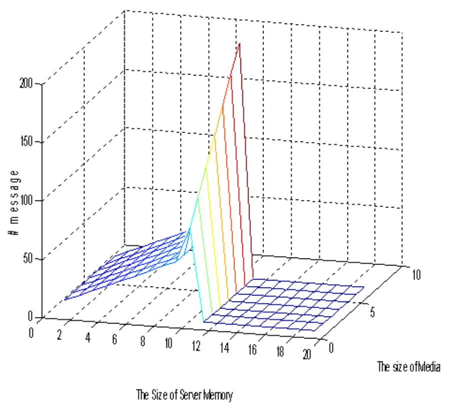

Figure 8. Parameters sampling.

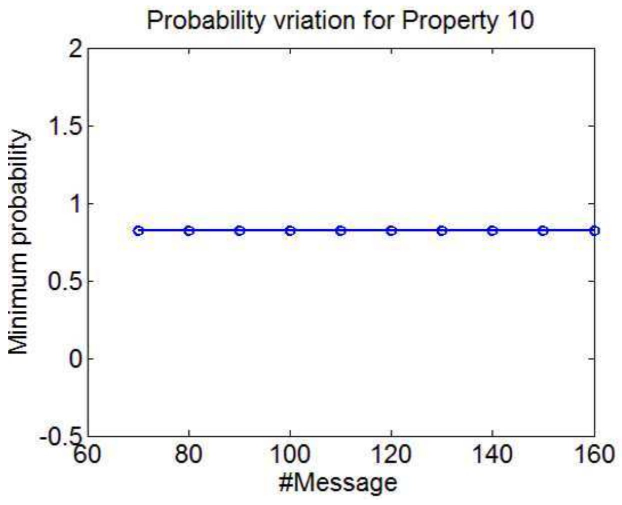

Figure 9. The probability variation for property #10

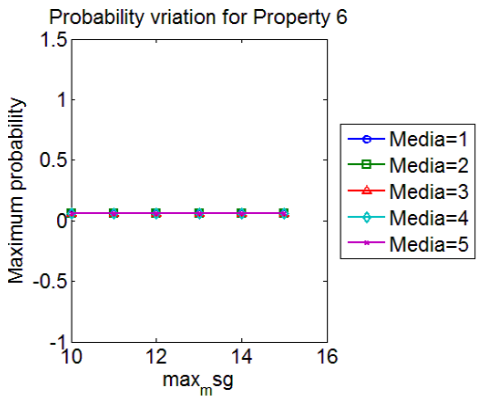

Figure 10. The probability variation for property #6

In Figure 9, we show the probability evaluation of the tenth property as a function of the message size variation. This evaluation is always fixed even by changing the number of messages. Finally, The entire code is downloadable from this foot link [3] within the list of the above listed properties.

VIII. Conclusion

In this paper, we proposed a technique for the probabilistic formal verification of a communication protocols taking into consideration their communication capabilities. To this end, we map the semantic model of the protocol with its related attack scenario in the form of probabilistic timed automata into the input language of the probabilistic model checker PRISM. As application, we apply our methodology on real time streaming protocol. It helps in reducing development cost by allowing detection of flaws and measuring security at the earlier stage of software life-cycle. As a future work, we intend to improve the scalability of our approach by developing reduction techniques that take into consideration only the affected part in the model for such property.

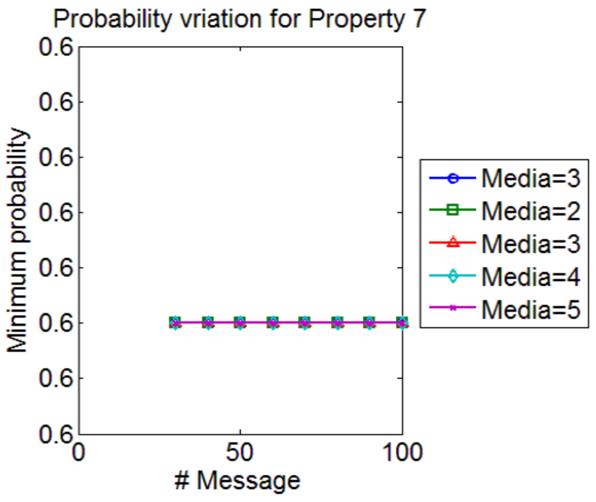

Figure 11. The probability variation for property #7

[3]http://users.encs.concordia.ca/∼s_oucha/

References

[1] G. J. Holzmann, *Design and Validation of Computer Protocols*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[2] C. Perkins, *Rtp: audio and video for the internet*. Addison-Wesley Professional, 2003.

[3] L. OGorman, “Comparing Passwords, Tokens, and Biometrics for User Authentication,” *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2021–2040, 2003.

[4] E. M. C. Jr., O. Grumberg, and D. A. Peled, *Model Checking*. The MIT Press, 1999.

[5] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, may 2008.

[6] P. Team, “PRISM - Probabilistic Symbolic Model Checker,” http://www.prismmodelchecker.org, last visited: June 2011.

[7] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, “Automatic verification of real-time systems with discrete probability distributions,” *Theor. Comput. Sci.*, vol. 282, no. 1, pp. 101–150, 2002.

[8] S. Chaki and A. Datta, “Aspier: An automated framework for verifying security protocol implementations,” in *Computer Security Foundations Workshop*, 2009, pp. 172–185.

[9] E. M. Clarke, S. Jha, and W. Marrero, “Verifying security protocols with brutus,” *ACM Trans. Softw. Eng. Methodol.*, vol. 9, pp. 443–487, October 2000.

[10] M. Akbarzadeh and M. Azgomi, “A framework for probabilistic model checking of security protocols using coloured stochastic activity networks and pdetool,” in *Telecommunications (IST), 2010 5th International Symposium on*, dec. 2010, pp. 210 –215.

[11] G. Norman and V. Shmatikov, “Analysis of probabilistic contract signing,” *Journal of Computer Security*, vol. 14, no. 6, pp. 561–589, 2006.

[12] Y. Xin, Y. Shao-hua, L. Yan, and M. Jian-hua, “Streaming media intrusion detection through interacting protocol state machines,” *Future Generation Communication and Networking*, vol. 1, pp. 441–444, 2008.

[13] C. Lei and Y. Dejian, “Dos and ddos attack’s possibility verification on streaming media application,” *Information Science and Engieering, International Symposium on*, vol. 2, pp. 63–67, 2008.

[14] J. Bilien, E. Eliasson, J. Orrblad, and J. olov Vatn, “Secure voip: call establishment and media protection,” in *In 2nd Workshop on Securing Voice over IP*, 2005.

[15] U. Team, “UPPAAL - ,” http://http://www.uppaal.org, last visited: June 2011.

[16] M. D. Abrams, “Nims information security threat methodology,” MITRE, Center for Advanced Aviation System Development,McLean, Virgini, Mitre Technical Report MTR 98 W000009, August 1998.

[17] D. P. Bertsekas and J. N. Tsitsiklis, “An analysis of stochastic shortest path problems,” *Math. Oper. Res.*, vol. 16, pp. 580–595, August 1991.

[18] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker, “Automated Verification Techniques for Probabilistic Systems,” in *Formal Methods for Eternal Networked Software Systems (SFM’11)*, ser. LNCS, M. Bernardo and V. Issarny, Eds. Springer, 2011, to appear.

[19] H. Schulzrinne, A. Rao, and R. Lanphier, *Real Time Streaming Protocol (RTSP)*, United States, 1998.

[20] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, *The Secure Real-time Transport Protocol (SRTP)*, United States, 2004.