A Probabilistic Attacker Model for Quantitative Verification of DoS Security Threats

Stylianos Basagiannis Panagiotis Katsaros Andrew Pombortsis

Nikolaos Alexiou

Department of Informatics, Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
tel.: +30-2310-998532, fax: +30-2310-998419
{basags, katsaros, apombo, nalexiou}@csd.auth.gr

Abstract

This work introduces probabilistic model checking as a viable tool-assisted approach for systematically quantifying DoS security threats. The proposed analysis is based on a probabilistic attacker model implementing simultaneous N zombie participants, which subvert secure authentication features in communication protocols and electronic commerce systems. DoS threats are expressed as probabilistic reachability properties that are automatically verified through an appropriate Discrete Time Markov Chain representing the protocol participants and attacker models. The overall analysis takes place in a mature probabilistic model checking toolset called PRISM. We believe that the applied quantitative verification approach is a valuable means for comparing protocol implementations with alternative parameter choices, for optimal resistance to the analyzed threats.

Key Words- Denial of Service, model checking, security

1. Introduction

Formal techniques for the analysis of security protocols assume the existence of an intruder model that allows studying the possibility of secrecy and authentication failures. All these approaches adopt the basic assumptions of the general Dolev and Yao intruder model [4] that are summarized as follows: (i) *The encryption method used is unbreakable* (ii) *The intruder can prevent any message from reaching its destination* and (iii) *The intruder can create messages of his own*. However, existing formal approaches focus only on secrecy and authentication guarantees and do not address the need for quantitative verification of potential availability threats. In this article, we review the few quantitative analyses found in the related bibliography and subsequently we introduce our own approach.

Our proposal is based on a probabilistic attacker model that combines appropriate attack actions, which make it possible to reveal potential Denial of Service (DoS) threats. Moreover, the applied probabilistic analysis exploits pre-assigned cost values that quantify the resource expenditure for the associated operations. This allows comparing the total cost to the attacker against the cost to the legitimate participants in case of an existing DoS threat. Thus, it is possible to determine the minimum resource requirements for an attacker to accomplish the attack and this is useful for developing measures for optimal resistance to the analyzed threat.

The DoS threat is expressed as a probabilistic reachability property that is automatically verified (according to [8] and [9]) with respect to an appropriate Discrete Time Marcov Chain (DTMC) representing the protocol participants and attacker models. The analysis takes place in a probabilistic model checking toolset called PRISM [13].

Our approach is described in terms of the performed analysis for the Host Identity Protocol (HIP) base-exchange. We realized that an attacker model embedding three basic attack tactics [3] that successfully incarnate simultaneous N zombie participants breaks the employed DoS resistance mechanism. Appropriate queries expressed in Probabilistic Computation Tree Logic (PCTL) provide illuminating probabilistic estimates together with the attacker and protocol participants' costs for the analyzed DoS threat.

In section 2 we review the related work. Section 3 provides a brief introduction to probabilistic model checking and defines the DoS resistance property, in terms of a general probabilistic attacker model. Section 4 introduces the PRISM model for the used attacker. Section 5 provides a description of the HIP Base Exchange and outlines its implementation into the PRISM model checker. In section 6 we present the results of the performed PCTL queries that provide estimates for the analyzed DoS threat. We conclude with a summary of the overall analysis approach and a comment on its usability and its potential impact.

2. Related Work

The importance of enabling availability analysis for a given cryptographic protocol was first shown in [12]. In that work the author examines DoS in the context of the resource intensive task of authentication and develops a framework for weighting the cost to the defender against the cost to the attacker.

Recently, the approach of [12] formed the basis for the analysis framework of [15] that according to the authors provides a more accurate representation of computational cost. However, quantitative evaluation takes place by simulation of the developed Timed Colored Petri Net model, without having exploited the formal analysis capabilities of the employed toolset.

An interesting stochastic modelling and analysis approach for quantifying the availability of software systems under DoS threats is the one introduced in [10]. In that work, the authors formulate the analyzed system in terms of an appropriate semi-Markov process (SMP). The whole approach requires stochastic modeling and analysis competence, since it is not carried out within an automated analysis tool like PRISM. Furthermore, the performed system-level analysis does not take into account any resource expenditure for the considered states and thus it is not possible to evaluate the message processing costs for DoS threats upon a security protocol model.

The most closely related work found in the bibliography is the one published in [1]. In that work, the authors specify in probabilistic rewriting logic a DoS resistant 3-way handshaking in TCP. In the VESTA toolset, the developed algebraic specification generates a timed probabilistic model, which is then analyzed by Monte Carlo simulation using a sequence of interrelated statistical hypothesis tests, to check on the generated sample if the quantitative property of interest is satisfied. This reflects the so-called statistical model checking approach. Compared to the probabilistic model checking analysis that is proposed in our work, this approach does not produce the same accurate results [14]. Moreover, the aforementioned analysis does not take into account message processing costs as we do and for this reason it is not possible to weight the cost to the honest participants against the cost to the attacker. For the resource intensive authentication features of modern security protocols this may be a significant analysis limitation.

3. Preliminaries of Probabilistic Model Checking

Probabilistic model checking is based on labelling transitions between model states with information about the likelihood that they will occur ([8], [9]).

In the PRISM language, a probabilistic model is defined as a set of m modules $M_1, \ldots, M_m$, where module M_i is a pair (Var_i, C_i) with Var_i a set of integer-valued local variables with finite range and C_i a set of commands. We denote by Var the set of all local variables in the model, i.e.

$$Var = \bigcup_{i=1}^{m} Var_i$$

Each variable $v \in Var$ has an initial value $\bar{v}$. Each command $c \in C_i$ takes the form $(g, (\lambda_1, u_1), \ldots, (\lambda_{nc}, u_{nc}))$, comprising a guard g and a set of pairs (λ_j, u_j) where $\lambda_j \in IR > 0$ and u_j is an update for each $1 \le j \le n_c$. A guard g is a predicate over the set of all local variables Var and each update u_j corresponds to one possible transition of module M_i. If Var_i contains n_i local variables $v_1, \ldots, v_{ni}$, then an update takes the form $(v_1' = \mathrm{expr}_1) \wedge \ldots \wedge (v_{n_i}' = \mathrm{expr}_{n_i})$, where each $expr_j$ is an expression in terms of the variables in Var. When in an update the values of some variables in Var_i remain unchanged, the model description may omit this information. In a DTMC specification, the values λ_j determine the probability of the corresponding transition and for this reason $\lambda_j \in (0, 1]$ for $1 \le j \le n_c$ and $\sum_{j=1}^{n_c} \lambda_j = 1$.

Definition 1. A DTMC is a tuple $(S, \bar{s}, P, L)$ where:

- S is a finite set of states
- $\bar{s} \in S$ is the initial state
- $P: S \times S \to [0, 1]$ is the transition probability matrix such that $\sum_{s' \in S} P(s, s') = 1$ for $s \in S$
- $L: S \to 2^{AP}$ is a labelling function mapping states to sets of atomic propositions from a set AP with the properties of interest

Terminating states are modelled by a single transition going back to the same state with probability 1.

In order to be able to determine the probability that paths in a DTMC are taken, a probability measure $Prob_s$ is formally defined (definition is omitted) on the set $Path_s$ of all infinite paths starting in state $s \in S$. In this way, it is possible to quantify the probability that a DTMC behaves in a specified fashion by identifying the set of paths which satisfy the property specification and assuming that this probability is measurable using the measure $Prob_s$.

The DTMC model corresponding to a PRISM language description is constructed as the parallel composition of its modules by computing the reachable state space of the model and discarding any unreachable states ([8], [9]). In every state, there is a set of commands (belonging to any of the modules) which are enabled. The choice between which command is performed is probabilistic, with *each enabled command selected with equal probability*. PCTL property specifications [13] are checked by applying appropriate model checking algorithms on the model by induction over their syntax. The underlying computation in PRISM involves a combination of (i) Graph-theoretical algorithms, for reachability analysis and qualitative probabilistic model checking; (ii) Numerical computation (iterative solvers), for quantitative probabilistic model checking that in the case of a DTMC implies the solution of linear equation systems.

Definition 2. The syntax of PCTL is as follows:

$\Phi ::= \texttt{true} \mid \alpha \mid \neg\Phi \mid \Phi \wedge \Phi \mid P_{\sim p}[\varphi]$, for state formulae and $\varphi ::= \mathtt{X}\Phi \mid \Phi\ \mathtt{U}^{\leq k}\ \Phi$, for path formulae

that are evaluated over states and paths of a DTMC respectively, where α is an atomic proposition, $\sim\ \in \{<, \leq, \geq, >\}$, $p \in [0, 1]$ and $k \in \aleph \cup \{\infty\}$.

To specify a property, we always use a state formula: path formulae only occur as the parameter of the $P_{\sim p}[\cdot]$ operator. In a DTMC, a state s satisfies $P_{\sim p}[\varphi]$ if the probability of taking a path from s satisfying φ is in the interval specified by $\sim p$. This is quantified by the probability measure $Prob_s$ that is defined over $Path_s$. As path formulae we allow the $\mathtt{X}$ ('next') and $\mathtt{U}^{\leq k}$ ('bounded until' implying property compliance within k time-steps) operators which are standard in temporal logic. The unbounded until is obtained by taking k equal to ∞, i.e. $\Phi\ \mathtt{U}\ \Psi = \Phi\ \mathtt{U}^{\leq\infty}\ \Psi$. Path formulae may also contain the operators $\Diamond$ (eventually) and $\Box$ (always) in their bounded and unbounded variants:

$$P_{\sim p}[\Diamond^{\leq k}\ \Phi] \equiv P_{\sim p}[\texttt{true}\ \mathtt{U}^{\leq k}\ \Phi]$$
$$P_{\sim p}[\Diamond\ \Phi] \equiv P_{\sim p}[\texttt{true}\ \mathtt{U}^{\leq\infty}\ \Phi]$$
$$P_{\sim p}[\Box^{\leq k}\ \Phi] \equiv P_{\overline{\sim}\ 1-p}[\Diamond^{\leq k}\ \neg\Phi]$$
$$P_{\sim p}[\Box\ \Phi] \equiv P_{\overline{\sim}\ 1-p}[\Diamond\neg\Phi]$$

where $\overline{<} \equiv >$, $\overline{\leq} \equiv \geq$, $\overline{\geq} \equiv \leq$ and $\overline{>} \equiv <$.

Apart from quantitative assertions, in PCTL we can also express properties which evaluate to a numerical value. These properties are specified in the form $P_{=?}[\varphi]$.

In addition to the aforementioned features, a reward structure for a DTMC allows the specification of two distinct types of rewards, namely state or cumulative rewards and transition or instantaneous rewards. Also, the logic PCTL is extended to allow specification of reward properties. In our problem, rewards represent costs, i.e. consumption of an exhaustible resource that depending on the modelled DoS threat it may be communication capacity (bandwidth), memory or processing power.

4. The Probabilistic Attacker Model

The strong assumptions of the typical Dolev - Yao attacker have been effective in the analysis of security guarantees that are formulated as safety properties (secrecy and authentication), but they are not entirely suitable for the analysis of security guarantees like DoS resistance that involves liveness [5]. A typical Dolev - Yao attacker has full control over the communication channels between the protocol participants and it is treated as a nondeterministic process that may attempt any possible attack. A protocol is considered secure if no possible interleaving of actions results in a security breach. However, the presence of non-determinism means that certain liveness properties cannot be established unless fairness is assumed. From this perspective, as far as fairness can be viewed as an abstraction of a probabilistic behaviour, it seems more natural to invest on a probabilistic model checking approach and to not adopt fairness assumptions, which are not valid for all attacker abilities considered in typical Dolev - Yao style analyses. Some other facts that make the use of probabilistic model checking a preferable choice are:

- The requirement to model the ability of an attacker to send randomly chosen messages or to model some sophisticated (yet probabilistic polynomial-time) computation to derive an attack from eavesdropped messages.
- The need to model probabilistic selection of implementation parameters, whose values affect the protocol's security.
- The need to capture the DoS faithfully, i.e. in terms of relative probabilities of certain observations by the attacker that depend solely on potentially probabilistic behaviours of the protocol participants (like for example the probability for a participant to

resubmit a service request that was previously dropped from the server's queue).

Our proposal for the DoS attacker model uses the open-ended attack tactics base we proposed in [3], from where the analyst selects the right set of abilities for his DoS problem. The selected attack actions are combined into a single PRISM module and the analyst assigns to the performed operations cost values that depend on the operation's resource expenditure, as well as on some resource constraint for the attacker. A similar cost assignment approach is applied to the honest protocol participants. In all cases, the assigned values refer to the same exhaustible resource, which can be either, participants' bandwidth (communication capacity), memory or processing power.

A DoS attacker uses a fixed number *N* of compromised machines that are commonly called zombie machines, because they are identical to the machines used by honest protocol participants. The zombie participants create bogus protocol messages capable of tricking some honest protocol participant into fully expediting resources, before determining that the opened protocol sessions are bogus. Alternatively, instead of considering *N* zombie machines we may consider a powerful attacker with identity spoofing abilities that allow him to incarnate the collective behaviour of *N* zombie participants. We assume the same resource expenditure for the attacker and the honest participants for the same operations and we assign the corresponding costs by taking into account the resource constraints assumed for the modelled protocol participants.

Definition 3. The *DoS resistance property* is defined as the low probability for an attacker representing the modelled DoS threat to eventually prevent - with disproportionaly low cost- honest participants from using the protocol's services.

In this sort of analysis it is very important to discover appropriate designs or parameter choices, such that every time a legitimate participant takes part in some action that requires the use of significant amount of resources, the attacker cannot fraudulently cause him to reach that step without spending a significant amount of its own resources.

DoS protection is usually based on an appropriate cookie-based [7] or client puzzle mechanism [2], where a protocol participant passes a "cookie" (e.g. an unforgeable keyed hash value of the information identifying the connection) or a puzzle to another participant, in order to establish mutual trust, possibly in the form of some shared-secret. The idea is that the responder should remain stateless (protection against memory exhaustion) and refuse to perform expensive cryptographic operations (protection against processing power exhaustion), until it has verified the honesty of the initiator. In this setting, there are three key strategies by which an attacker can implement a DoS threat:

- *Counterfeiting*: The attacker sends invalid cookies, puzzles or puzzle solutions.
- *Time Shifting*: The attacker is prepared for an attack by computing fake shared secrets (either by solving puzzles or manipulating cookies), in order to expend them in a massive DoS attack.
- *Message replays*: The attacker may send the same valid cookie or puzzle solution many times.

The aforementioned strategies assume that the attacker model performs three basic operations, i.e. message interception, message projection and message concatenation, mentioned in decreasing order of processing demands. The analyst implements a DoS threat by selecting the right set of attack tactics (deflection, message integrity violation and straight replay) from the ones formalized by us in [3] and composes them into a single PRISM module with appropriate cost values for the performed operations.

Figure 1 provides a high level view of the analyzed DoS threat upon the HIP base-exchange. The attacker (*At*) intercepts the message traffic between the initiator (*I*) and the responder (*R*) and alters the puzzle contained in msg_R by simple message concatenations, in order to create *N* zombie messages that are subsequently sent to the initiator. The attack tactics used in the attacker model are the message integrity violation combined with multiple straight message replays.

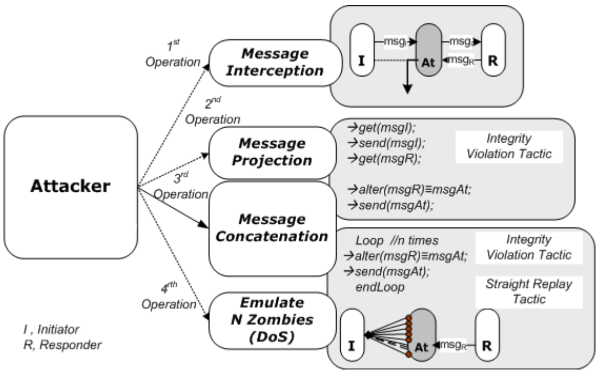


Figure 1. A DoS threat with message counterfeiting for N zombie participants

5. The HIP base exchange

The main goal of HIP [6] is the separation of host identifiers from locations in the IPv4 and IPv6

Internet. HIP also plays the role of a security protocol that defines host identifiers for naming the communication endpoints and performs authentication and IPsec security associations between them. The HIP base-exchange is built around a classic authenticated Diffie - Hellman key exchange, in an attempt to establish session keys between the communication endpoints.

Table 1: HIP Base Exchange notation

HITI	*I* identity tag
HITR	*R* identity tag
gR	pre-computed part of R1
sigR	signature of *R*
sigI	signature of *I*
C	puzzle nonce
K	puzzle difficulty
J	puzzle solution
LSB_k	returns the k least significant bits
K_e,K_s	generated Diffie-Hellman keys
E_x	message x encrypted with K_e
HK_s	cryptographic hash with key K_s
HMAC	HMAC based message authentication code with key K_s

In HIP, the *host identity* (HI) of the protocol participants plays the role of a public key: the used identifier can be used to verify signatures without access to certificates or a public-key infrastructure. It is usually represented by the *host identity tag* (HIT), which is a 128-bit hash of the HI. As shown in Figure 2 and Tale 1, the HIP base-exchange includes four messages that are supposed to provide a certain degree of DoS protection. The Initiator first sends the message I1 with the *HITI* and the *HITR* tags, to the Responder. We note that all messages contain the Initiator and Responder identity tags (*HITI*, *HITR*) in the header.

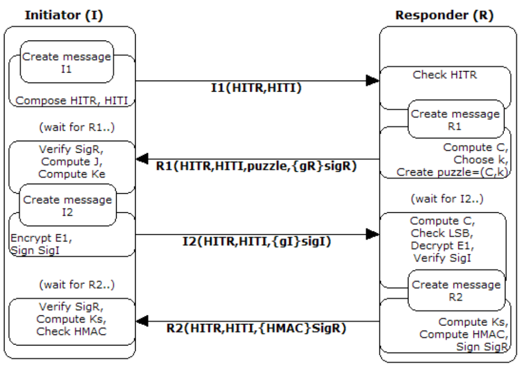


Figure 2. The HIP base-exchange

Message R1 is partially pre-computed by the Responder, even before the receipt of I1. The pre-computed part (gR) includes (i) the HITR, (ii) the Responder's Diffie-Hellman key, (iii) the Responder HI, (iv) the proposed cryptographic algorithms for the next steps of the base-exchange, (v) the proposed Encapsulating Security Payload (ESP) transforms and (vi) an echo request field.

The Responder signs the pre-computed part of R1 with sigR. All other parts of R1, i.e. the cryptographic puzzle and the HITI fields are populated after receiving an I1 and they are not protected by the signature sigR. A host may receive more than one R1 messages, either due to having sent multiple I1s or due to a replay of an old R1.

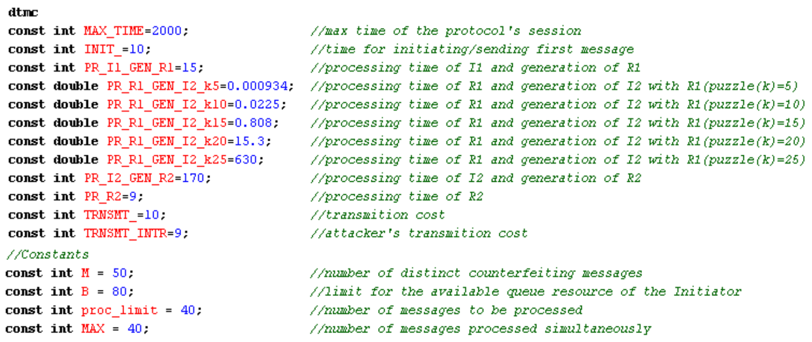
The used puzzle has three components: the puzzle nonce C, the difficulty level *k* and the corresponding solution J. The puzzle solution is verified as follows: we compute the SHA-1 hash of the concatenation of C, HITI, HITR and J and then we check that the k low-order bits of the hash are all zeros.

$$LSBk(SHA\text{-}1(C \mid HITI \mid HITR \mid J), k) == 0$$

While the Initiator performs a brute-force search for J that takes $O(2^k)$ trials, the Responder verifies the solution by computing a single hash (protection against processing power exhaustion). On receiving R1 the Initiator checks that it has sent a corresponding I1 and verifies the signature using the Responder HI. Then, it solves the puzzle and creates the message I2 that includes HITI, HITR and a signed part gI. The signed part contains (i) the puzzle and its solution, (ii) the Initiator's Diffie-Hellman key Ke, (iii) the HIP and ESP transforms proposed by the Initiator, (iv) the Initiator HI (public key) encrypted using Ke shown as E1, (v) the HIP and ESP transforms proposed by the Initiator, (vi) a security parameter index for the Responder-to-Initiator security associations and (vii) the echo response generated for the received echo request.

On receiving I2, the Responder verifies the puzzle solution, decrypts E1 that contains the Initiator HI, verifies the signature on I2 and computes the session key Ks. For the Initiator, the HIP base-exchange is concluded by the receipt of R2, which allows the verification of the HMAC and the signature. If a host decides to drop a security association, it deletes the corresponding HIP state, including the keying material.

Our PRISM model includes four modules: (i) the Medium (*m*), representing the communication channel used by the protocol participants to exchange messages, (ii) the Initiator (*I*), (iii) the Responder (*R*) and (iv) the Attacker (*At*). In their parallel composition, the aforementioned modules interact by updates to their local variables. These updates correspond to the modelled state transitions.

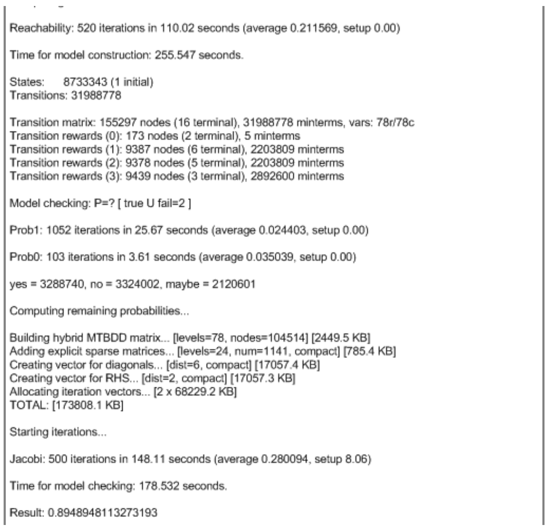


```
dtmc
const int MAX_TIME=2000;                    //max time of the protocol's session
const int INIT_=10;                         //time for initiating/sending first message
const int PR_I1_GEN_R1=15;                  //processing time of I1 and generation of R1
const double PR_R1_GEN_I2_k5=0.000934;      //processing time of R1 and generation of I2 with R1(puzzle(k)=5)
const double PR_R1_GEN_I2_k10=0.0225;       //processing time of R1 and generation of I2 with R1(puzzle(k)=10)
const double PR_R1_GEN_I2_k15=0.808;        //processing time of R1 and generation of I2 with R1(puzzle(k)=15)
const double PR_R1_GEN_I2_k20=15.3;         //processing time of R1 and generation of I2 with R1(puzzle(k)=20)
const double PR_R1_GEN_I2_k25=630;          //processing time of R1 and generation of I2 with R1(puzzle(k)=25)
const int PR_I2_GEN_R2=170;                 //processing time of I2 and generation of R2
const int PR_R2=9;                          //processing time of R2
const int TRNSMT_=10;                       //transmition cost
const int TRNSMT_INTR=9;                    //attacker's transmition cost
//Constants
const int M = 50;                           //number of distinct counterfeiting messages
const int B = 80;                           //limit for the available queue resource of the Initiator
const int proc_limit = 40;                  //number of messages to be processed
const int MAX = 40;                         //number of messages processed simultaneously
```

Figure 3. Global variables of the HIP model

Figure 3 introduces the global declarations of the model. Assuming that the communication channel acts as a message buffer, we modelled (omitted module code) three possible states for module *m*, namely (i) no messages, (ii) message origin *I* or *At* and (iii) message origin *R* or *At*. Two variables are related to HIP implementation parameters: (i) *proc_limit* represents the number of messages that are served simultaneously by the Initiator and (ii) *B* denotes the maximum number of messages in Initiator's admission queue. Finally, *M* expresses the number of distinct counterfeited messages created by *At* in the way shown in Figure 1. The global declarations of Figure 3 also include the message processing costs for the considered participants. The shown values are based on the considered message processing demands (data taken from related HIP performance studies [11]) and on the relative differences between the interacting participants regarding their processing capacity (number of instructions processed per second).

The PRISM module incarnating the analyzed DoS threat (omitted code) exploits the fact that the puzzle sent to *I* is not included in the pre-computed signed part of R1. The puzzle is generated on demand based on a random nonce and a parameter *k* that adjusts the puzzle difficulty and in effect influences I's cost to compute the solution. The protocol requirement for generation of fresh puzzles protects it from time shifting and message replay DoS threats, but at the same time makes it vulnerable to the counterfeiting DoS threat shown in Figure 1.



```
Reachability: 520 iterations in 110.02 seconds (average 0.211569, setup 0.00)

Time for model construction: 255.547 seconds.

States:      8733343 (1 initial)
Transitions: 31988778

Transition matrix: 155297 nodes (16 terminal), 31988778 minterms, vars: 78r/78c
Transition rewards (0): 173 nodes (2 terminal), 5 minterms
Transition rewards (1): 9387 nodes (6 terminal), 2203809 minterms
Transition rewards (2): 9378 nodes (5 terminal), 2203809 minterms
Transition rewards (3): 9439 nodes (3 terminal), 2892600 minterms

Model checking: P=? [ true U fail=2 ]

Prob1: 1052 iterations in 25.67 seconds (average 0.024403, setup 0.00)

Prob0: 103 iterations in 3.61 seconds (average 0.035039, setup 0.00)

yes = 3288740, no = 3324002, maybe = 2120601

Computing remaining probabilities...

Building hybrid MTBDD matrix... [levels=78, nodes=104514] [2449.5 KB]
Adding explicit sparse matrices... [levels=24, num=1141, compact] [785.4 KB]
Creating vector for diagonals... [dist=6, compact] [17057.4 KB]
Creating vector for RHS... [dist=2, compact] [17057.3 KB]
Allocating iteration vectors... [2 x 68229.2 KB]
TOTAL: [173808.1 KB]

Starting iterations...

Jacobi: 500 iterations in 148.11 seconds (average 0.280094, setup 8.06)

Time for model checking: 178.532 seconds.

Result: 0.8948948113273193
```

Figure 4. Probability to eventually reach a state where the Initiator is not available (DoS attack)

6. Verification Results

Figure 4 illustrates the probability to reach a state where the Initiator is not available. If *At* eventually brings the system into a state such that *I* processes simultaneously *proc_limit* counterfeited messages with wrong puzzles and its admission queue includes *B* messages in total then any valid R1 message will be dropped. According to definition 3, if *At* brings the system into this situation with high probability and the cumulated processing cost is disproportionaly low compared to the cost to *I*, we have proved an outstanding case of DoS threat. For the generated DTMC, the PCTL query

Q1: P=? [true U fail=2]

evaluates the probability of taking a path that eventually reaches such a state that implies variable *fail* having value 2. The results shown in Figure 4 reveal an unacceptably high probability (P=0.895) to eventually reach a state that reflects the discussed situation.

A more illuminating view of the variation of the probability to reach a state, where *I* is not available, is given in Figure 5 with the experiment:

Q2: P=? [true U msgs_in_service=proc_limit & msgs_in_queue=B]

where *proc_limit* is varied between 0 and 200 with step 50 and *B* is varied between 0 and 400 with step 50.

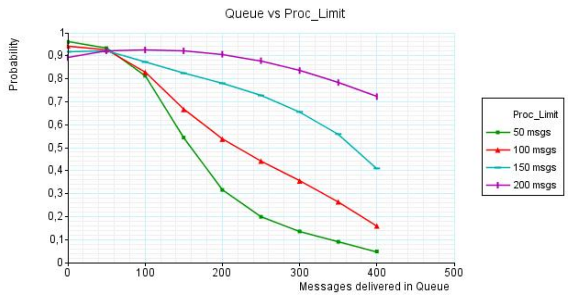


Figure 5. Probability for reaching a state, where I is unavailable for different values of proc_limit and B

We note that the system's DoS resistance is improved for allowed queue lengths of 250 messages or more, under the condition that *I* will not process simultaneously more than 50 messages.

However, a complete view for the protocol's DoS resistance is obtained only when having the results of the following reward queries regarding *I* and *At* processing costs. Query `Q3` provides for different values of puzzle difficulty *k* the expected cumulated processing cost to *I*, when having reached some state in which it is not available any more. We note that in any protocol implementation the puzzle difficulty is originally selected by *R*.

Q3: R[Initiator_cost]=? [true U msgs_in_queue=B & k=puz_dif]

Figure 6 shows the obtained results for `B` varied between 0 and 100 with step 20 and assigned puzzle difficulty *k* selected from the values 1, 10, 15, 20, 25 (`puz_dif`). For allowed queue lengths between 0 and 60 the cumulated processing cost for *I* can be dramatically increased, especially when the used puzzle difficulty is a number greater than 20.

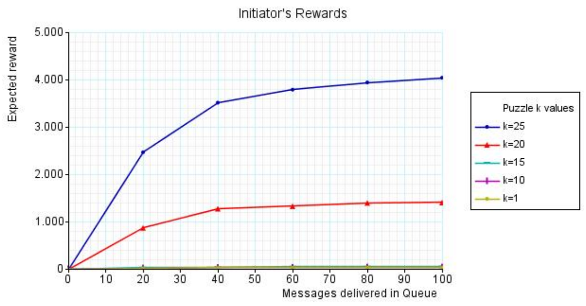


Figure 6. Expected cumulated processing cost for I when becoming unavailable

Reward queries `Q4` and `Q5` complete the picture for the system's DoS resistance according to definition 3. The graphs shown in Figures 7 and 8 provide a comparative view of the expected cumulated processing cost to *At* against the cost incurred to the *I* when becoming unavailable:

Q4: R[Initiator_cost]=? [true U msgs_in_queue=B & k=puz_dif]
Q5: R[Attacker_cost]=? [true U msgs_in_queue=B & k=puz_dif]

As before, `B` varied between 0 and 100 with step 20 and we generated results for puzzle difficulty 15 and 25 (`puz_dif`).

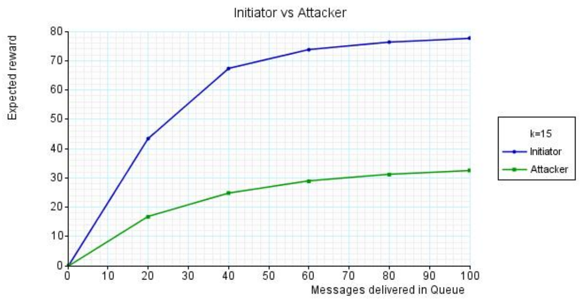


Figure 7. Expected processing costs for I and At for k=15

The shown results provide valuable insight regarding the severity of the demonstrated DoS threat for different combinations of puzzle difficulty, the number of messages simultaneously processed by *I* and its allowed queue length. Our *At* module counterfeits partially signed messages dispatched by *R* and replays the counterfeited messages, in order to trick *I* into solving wrong puzzles. If the bogus puzzles are based on previously eavesdropped puzzles, then an efficient approach for deamplification of the demonstrated attack is the use of puzzle difficulty values between 10 and 18, together with appropriate adjustments in the allowed queue length for *I* and the number of messages simultaneously processed.

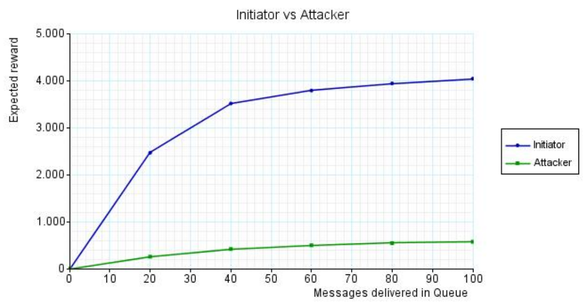


Figure 8. Expected processing costs for I and At for k =25

An alternative countermeasure with considerable implementation cost is the use of an R1 generation counter per host identity ([6]). This monotonically increasing counter will indicate the current generation of puzzles and a host will accept puzzles from the same generation, but it may be also possible to accept puzzles from earlier generations. Again, our quantitative verification approach is a valuable means in the design of such countermeasures.

7. Conclusion

Secure authentication features of modern communication and electronic commerce protocols have the potential to be turned into denial-of-service (DoS) exploits. This article introduces probabilistic model checking as a viable tool-assisted approach for systematically quantifying DoS security threats. Our approach is based on a general probabilistic attacker model encompassing the most common DoS attack strategies. We formulated the DoS resistance property as a quantifiable measure that depends (i) on the probability to reach a state where some protocol participant becomes unavailable and (ii) the requirement for the attacker to cause this event with disproportionaly low cost compared to the cost incurred to the victim.

We developed a probabilistic model for the HIP base-exchange protocol that uses a client puzzle mechanism for protection against DoS. The obtained results provide valuable insight regarding the severity of the demonstrated DoS threat in different protocol implementation cases. We believe that the proposed approach can be utilized in the design of new security protocols and protocol implementations.

8. References

[1] Agha, G., Greenwald, M., Gunter, C. A., Khanna, S., Meseguer, J., Sen, K., Thati, P. "Formal modelling and analysis of DoS using probabilistic rewrite theories", Proc. IEEE Work. on Foundations of Computer Security (FCS '05), Chicago, 2005.

[2] Aura, T., Nikander, P., Leiwo, J., "DOS-resistant authentication with client puzzles", Proc. Security Protocols Worksh., Cambridge, Springer LNCS 2133, 170-181, 2001.

[3] Basagiannis, S., Katsaros, P. and Pombortsis, A. "Intrusion Attack Tactics for the model checking of e-commerce security guarantees", Proc. 26th Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP), Springer LNCS 4680, 238-251, 2007.

[4] Dolev, D. and Yao, A. "On the security of public-key protocols", IEEE Trans. on Information Theory, 2 (29), 198-208, 1983.

[5] Gärtner, F. "Revisiting liveness properties in the context of secure systems", Proc. 1st Int. Conf. Formal Aspects on Security, Springer LNCS 2629, 203-211, 2003.

[6] IETF - Network Working Group. Host Identity Protocol. Internet Draft, Feb. 2007.

[7] Karn, P. and Simpson, A. Photuris: Session-key management protocol, RFC 2522, IETF Network Working Group, 1999.

[8] Kwiatkowska, M., Norman, G., Parker, D. "Stochastic model checking", In: Formal Methods for the Design of Comp., Comm. & Software Systems: Performance Evaluation, Springer LNCS 4486, 220-270, 2007.

[9] Kwiatkowska, M. "Quantitative verification: Models, Techniques and Tools", Proc. 6th joint meeting of the Europ. Software Engineering Conf. and ACM SIGSOFT Symp. on the Foundations of Soft. Engineering (ESEC/FSE), ACM Press, 449-458, 2007.

[10] Madan, B. B., Goseva-Popstojanova, K., Vaidyanathan, K., Trivedi, K. S. "Modeling and quantification of security attributes of software systems", Proc. IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN 02), IEEE Computer Society, 2002.

[11] InfraHIP Project Web Site, http://infrahip.hiit.fi/, (last access: 21st of December 2007).

[12] Meadows, C. "A cost-based framework for analysis of DoS in networks", J. Comp. Security, 9, 143-164, 2001

[13] The PRISM Model Checker Web Site, http://www.prismmodelchecker.org/

[14] Sen, K., Viswanathan, M., Agha, G. "On statistical model checking of stochastic systems", Pro. 17th Int. Conf. on Computer Aided Verification (CAV'05), Springer LNCS 3576, 266-288, 2000.

[15] Tritilanunt, S., Boyd, C., Foo, E., Gonzalez Neto, J. M. "Using Coloured Petri Nets to simulate DoS-resistant protocols", Proc. 7th Worksh. on Practical Use of Coloured Petri Nets & the CPN Tools, Un. of Aarhus, Denmark, 2006.