Examining the DoS Resistance of HIP

Suratose Tritilanunt[1], Colin Boyd[2],
Ernest Foo[2], and Juan Manuel González Nieto[2]

Information Security Institute
Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia
1)s.tritilanunt@student.qut.edu.au,
2){c.boyd, e.foo, j.gonzaleznieto}@qut.edu.au

Abstract. We examine DoS resistance of the Host Identity Protocol (HIP) and reveal a technique to deny legitimate services. To demonstrate the experiment, we implement a formal model of HIP based on Timed Petri Nets and use a simulation approach provided in CPN Tools to achieve a formal analysis. By integrating adjustable puzzle difficulty, HIP can mitigate the effect of DoS attacks. However, the inability to protect against coordinated adversaries on a hash-based puzzle causes the responder to be susceptible to DoS attacks at the identity verification phase. As a result, we propose an enhanced approach by employing a time-lock puzzle instead of a hash-based scheme. Once the time-lock puzzle is adopted, the effect of coordinated attacks will be removed and the throughput from legitimate users will return to the desirable level.

1 Introduction

Many key exchange protocols have been developed for dealing with denial-of-service (DoS) attacks, especially resource exhaustion attacks. Host Identity Protocol (HIP) [14] is an interesting example of a DoS-resistant protocol which has been developed to deal with this kind of DoS attack. The concept behind this implementation is that HIP does not commit the responder's resource before the responder ensures the identity of the initiator. HIP achieves this concept by adopting stateless connection [3] and reachability testing by using a *client puzzle* [4, 12] incorporated via a cookie [17] to protect the responder from SYN flooding attacks [7] at the beginning phase. Moreover, the responder can authenticate the initiator by starting with the cheap computation using a client puzzle and then increase the level of authentication to the expensive computation using a digital signature for ensuring the identity of the initiator.

HIP is a promising protocol for protecting the responder from DoS attacks. However, lack of formal analysis in the design phase of HIP might introduce other kinds of vulnerability. Moreover, the instruction on how to adjust the client puzzle difficulty is not clearly specified and examined in the HIP specification [14]. In this paper, we implement a formal model of HIP using the formal specification language of Timed Petri Nets. In order to achieve a formal analysis, we use a

simulation technique provided in CPN Tools for analysing HIP model. The purpose of the simulation in the cryptographic protocol is to identify vulnerabilities in the system that might be difficult to explore in the design phase.

Simulation approaches are well-known not only for exploring vulnerabilities in cryptographic protocols, but guaranteeing security services of such protocols as well. Using simulation approaches has several benefits over mathematical analysis. For instance, they can provide *flexibility* and *visualization* during protocol analysis and verification. In our experiment, we set up the simulation of HIP for exploring unbalanced computational steps that cause a responder to spend more computations than an initiator does. In addition, our experimental result provides a measurement of successful legitimate traffic as proposed by Beal and Shepard [6] in different situations under DoS attacks. This factor can be used as a parameter for justifying the effectiveness of HIP to resist DoS attacks. In order to set up an experiment, we allow four kinds of adversary and the honest client to participate with the same responder during the protocol run. We set up two experiments; 1) the responder can choose only a fixed value of a puzzle difficulty no matter what the workload is, and 2) the responder has an ability to flexibly adjust puzzle difficulty by using the workload condition as criterion.

The main contributions of this paper are:

1. A simulation and analysis of HIP in Timed Coloured Petri Nets.
2. Identification of four scenarios of resource exhaustion attack on HIP.
3. A proposed technique to deal with adversaries who try to overwhelm the responder's resource by computing a puzzle solution in parallel.

1.1 Host Identity Protocol (HIP)

HIP has been developed by Moskowitz [14]. Later, Aura et al. [2] found some vulnerabilities and proposed guidelines to strengthen its security. HIP is a four-packet exchange protocol which allows the initiator I and responder R to establish an authenticated communication. Both I and R hold long-term keys to generate signatures $Sig_I(\cdot)$ and $Sig_R(\cdot)$ respectively. It is assumed that both principals know the public key PK_I of the initiator and PK_R of the responder represented in the form of host identifiers (HI) in advance. HIT represents the host identity tag created by taking a cryptographic hash H over a host identifier.

H_{K_s} represents a keyed hash function using session key K_s to generate a hashed-MAC ($HMAC$). The value s is a periodically changing secret only known to the responder. LSB takes as input a string t and a parameter k and returns the k least significant bits of t. 0^k is a string consisting of k zero bits. $E_{K_e}(\cdot)$ and $D_{K_e}(\cdot)$ denotes a symmetric encryption and decryption respectively under session key K_e. In order to generate session keys K_e and K_s, HIP employs Diffie-Hellman key agreement. Diffie-Hellman parameters used to generate these keys consist of large prime numbers p and q, a generator g, a responder's secret value r, and an initiator's secret value i.

HIP adopts a proof-of-work scheme [11] for countering resource exhaustion attacks. In a proof-of-work, HIP extends the concept of a *client puzzle* [4, 12]

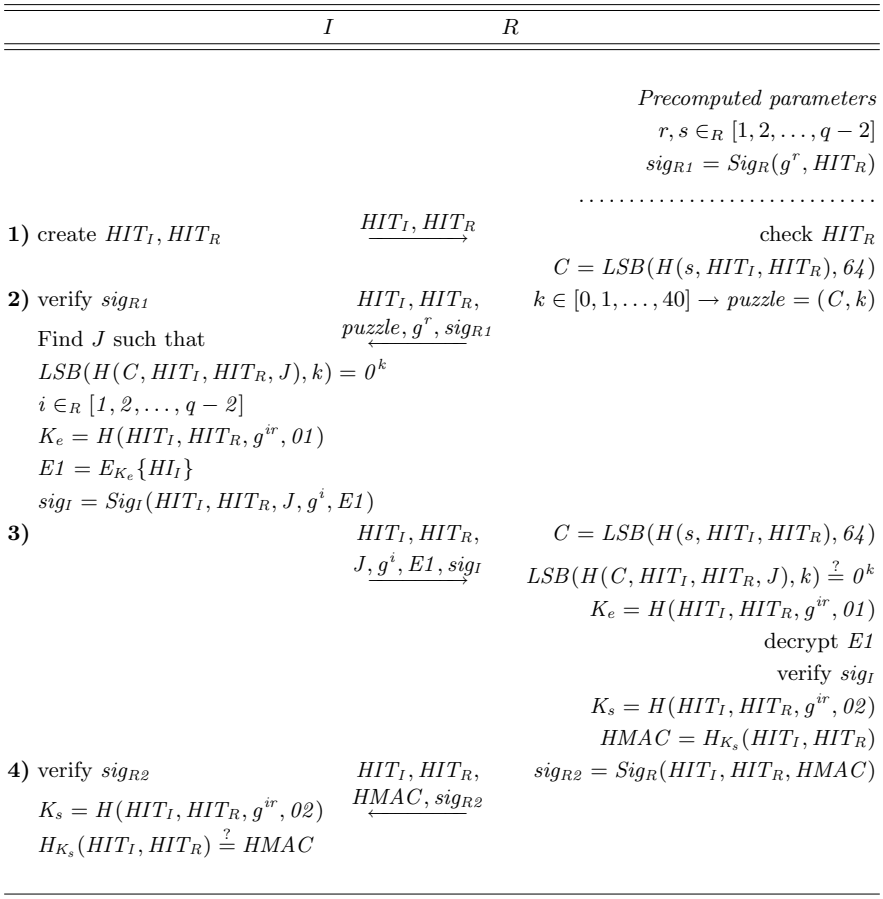


Fig. 1. HIP Protocol [14]

for protecting the responder against DoS attacks. HIP uses the client puzzle to delay state creation [3] in the responder until the checking of the second incoming message and the authentication has been done in order to protect the responder against resource exhaustion attacks.

1.2 Previous Work

Over many years, cryptographic and security protocols have been modeled and verified using Coloured Petri Nets (CPNs). Doyle [8] developed a model of three-pass mutual authentication and allowed an adversary to launch multiple iteration and parallel session attacks. Han [10] adopted CPNs for constructing a reachability graph to insecure states and examining the final states in OAKLEY. Al-Azzoni [1] developed a model of Needham-Schroeder public-key authentication protocol and Tatebayashi-Matsuzaki-Neuman (TMN) key exchange protocol.

Beal and Shepard [6] constructed a model of HIP protocol using a mathematical equation for analysing the effect of puzzle difficulty under the steady-state attack. In order to deamplify the arrival rate of incoming requests, Beal and Shepard have set up two strategies; 1) forcing a sustainable arrival rate, and 2) limiting service disruption. Beal and Shepard have modeled adversaries in which the ability of adversaries has been limited to disrupt the service of legitimate initiators by flooding bogus requests.

To the best of our knowledge, there is no implementation of CPNs focusing on an exploration of vulnerabilities based on unbalanced computation that might lead to resource exhaustion attacks in key exchange protocols. Moreover, Beal and Shepard's mathematical model has a few limitations including 1) they do not allow the responder to dynamically adjust puzzle difficulty, and 2) there is only one attacking technique to overwhelm the responder's resources.

2 Experimental Results and Analysis

In our model, we have allowed a system to consist of honest clients, individual type of adversaries, and a responder. The responder has to deal with different strategies of adversaries and amounts of packets which consist of both legitimate and bogus messages. We allow three different packet rates for both honest clients and adversaries in order to measure the toleration of HIP under DoS attacks.

- *Honest Clients* (`hc`): can initiate requests at 80%, 100%, and 150% of the responder's capacity (R).
- *Adversaries*: can flood bogus requests at 100%, 200%, and 1000% of the responder's capacity (R).

Apart from honest clients who initiate the legitimate traffic, we allow four types of adversary who have the similar goal to deny the service of the responder by overwhelming CPU usage and connection queue of the responder. While other adversarial strategies are certainly possible, the defined adversaries cover the most obvious attacks at all stages of the protocol execution. To our knowledge, no previous formal analysis of DoS-resistant protocols has included such a comprehensive adversary definition.

Type 1 adversary (`ad1`) computes a valid first message (may be pre-computed in practice), and then takes no further action in the protocol.

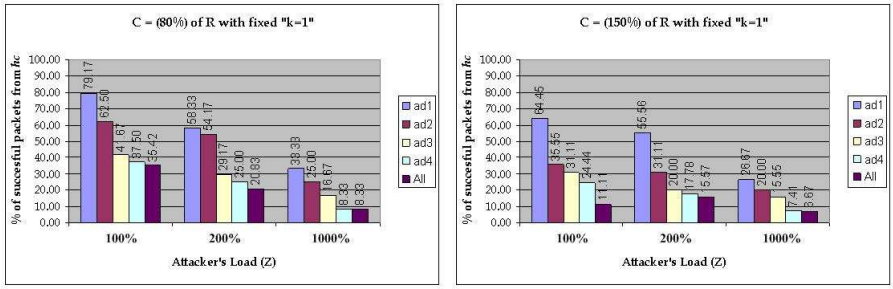
Type 2 adversary (`ad2`) completes the protocol normally until the third message is sent and takes no further action after this. The computations of this adversary include searching a correct client puzzle solution J, generating a session key K_e and encrypting a public key PK_I, and finally computing a digital signature Sig_I.

Type 3 adversary (`ad3`) completes the protocol step one and two with the exception that the adversary does not verify the responder signature sig_{R1}. The adversary searches for a correct client puzzle solution J but randomly chooses the remaining message elements: an encrypted element $K_e\{HI_I\}$ and a digital signature sig_I. The adversary takes no further action in the protocol.

Type 4 adversary (ad4) is like an adversary type 3, except that the client puzzle solution J is now also chosen randomly.

In the simulation, we initially set up the responder's capacity for handling incoming requests. The hc initiates a request only once and keep waiting to process next steps. If its requests is rejected, hc gives up. For adversaries, there are two different situations in which the responder rejects bogus messages; 1) the responder detects the bogus messages during the verification steps, and 2) the responder does not have enough resources for serving requests. In order to evaluate the system performance, the rate of successful legitimate traffic under different attacks has been measured as the percentage of throughput. Some sample results of our experiment are demonstrated in the following subsection.

Experiment 1: non-adjustable client puzzle
The purpose of experiment 1 is to investigate the behaviour of the protocol under four different attacks in the non-adjustable client puzzle. We initially fix k=1, i.e. the easiest value[1]. Under normal circumstances, hc prefers to spend nothing expensive for establishing a connection. The experiments were run both with a single and all adversary types running together defined as All.



(a) $hc = 80\%$ (b) $hc = 150\%$

Fig. 2. Percentage of throughput from honest clients with k=1

From figure 2, when adversaries increase the number of bogus messages , the percentage of successful messages from hc to obtain a service will drop drastically. Comparing ad1 and ad4, even though both of them craft random messages, ad4 can achieve the goal at higher rate than ad1 because the responder can process the incoming request at step 1 and clear a queue faster than at step 3. At step 1, the responder only participates in the protocol by choosing the puzzle difficulty (k) and pre-computed information, and returns it to ad1. Although, ad1 can re-send bogus messages after receiving replied messages, this does not cause the responder to reject a large number of messages because HIP mitigates

[1] If we choose k=0, we cannot see the difference of costs between ad3 and ad4.

such problem by adopting a stateless-connection. On the other hand, the task of `ad4`, to fill-up the responder's queue at step 3, can be achieved more easily than `ad1` because the process of checking a puzzle solution and a digital signature takes longer than a whole process at step 1.

Comparing `ad2` and `ad3` who attempt to deny service at phase 3 by computing the puzzle solution, the results show that `ad3` succeeds at higher proportion than `ad2`. This is because `ad3` can flood attack messages faster than `ad2` who must engage in the correct generation of message two. Nonetheless, both adversaries can force the responder to engage in the signature verification. In the case of `ad4`, although, they flood large number of messages at step 3 as well as `ad2` and `ad3`, `ad4` cannot force the responder to engage in expensive operations because the responder is able to detect the message forgery at the cheap puzzle verification process. However, without the assistance of puzzle difficulty, the percentage of successful messages in the case of `hc` and `ad4` is lower than the others because `ad4` floods message three at the highest rate. As a result, the most effective technique to deny services on the responder would be the fourth scenario that attacks the verification phase. Most key agreement protocols incorporate verification tasks that would be susceptible to resource exhaustion attacks.

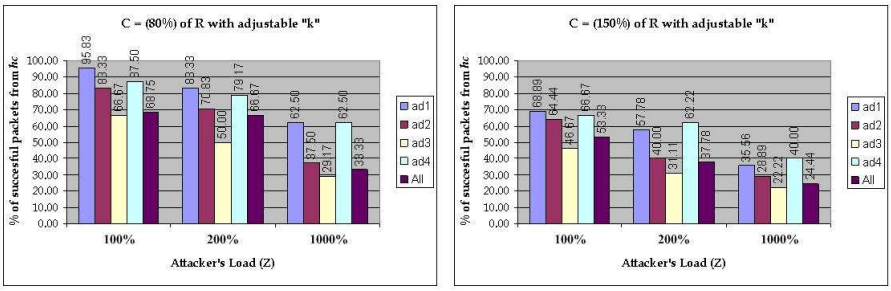
The result of the combination of all attack techniques shows that when the responder has to deal with all types of adversary, the percentage of legitimate users served by the responder will fall significantly with increase of bogus messages. Now we have identified the most effective scenario, we will apply this technique to the experiment 2 for investigating the usefulness of puzzle difficulty.

Experiment 2: adjustable client puzzle

The purpose of the second experiment is to observe and evaluate how a client puzzle can mitigate the problem of DoS attacks on the responder's machine. By calibrating several ranges of puzzle difficulty to obtain an optimal throughput, we anticipate to find a simple and flexible technique for dynamically adjusting puzzle difficulty to suit all DoS-attack scenarios.

To adjust the puzzle difficulty, we allocate two possible values for the responder to determine. Under normal circumstance, the responder selects k=1, which means the easiest puzzle solution is required from the initiator. Once the responder receives more requested packets than its maximum capacity to handle, the responder raises the puzzle difficulty. In the experiments described here, we choose $k = 10$. Because this puzzle technique is a hash-based puzzle, this value will help the responder to slow down the incoming rate by requiring a work of the initiator to solve a puzzles at the factor of 2^{10}.

From the experimental result in figure 3, the number of attacking machines that the responder can tolerate is increased to a higher proportion compared to the result of experiment 1. Another interesting result is that the successful rate of an honest client's message in the case of `ad4` is higher than for the fixed value k=1. The reason is that `ad4` does not compute the puzzle solution, so, no matter what the puzzle difficulty is, `ad4` can flood the bogus messages at the similar speed as experiment 1. However, at that amount of bogus messages, there are only messages from `ad4` (no legitimate traffic because `hc` has to spend some



(a) $hc = 80\%$ (b) $hc = 150\%$

Fig. 3. Percentage of throughput from `hc` with k is adjustable between 1 and 10

amount of time to solve the puzzle solution), or just only a few messages from `hc` that arrive to the connection queue before the responder increases puzzle difficulty. As a result, the responder can validate the puzzle solution before the next group of messages has arrived. Undoubtedly, these bogus messages from `ad4` will be rejected at the first step of verification which requires only short period and removes such attack from the connection queue. However, this situation does not occur in the case of `ad3` because they have to spend some amount of time to solve the puzzle as well as `hc`.

In experiment 2, the most effective scenario is from `ad3`. Comparing to experiment 1, if adversaries can flood messages at the same speed as `ad4` and force the responder to participate in expensive verification as for `ad3`, those adversaries would obtain higher satisfied outcome. A possible adversarial technique to obtain higher rejected rate is that if `ad3` can solve a puzzle more quickly and flood these solutions as fast as `ad4`. These packets will be accumulated in the connection queue longer than those from `ad4` because the responder has to participate and spend more computational time to verify `ad3`'s messages due to the signature verification. To achieve this technique, suppose that we use SHA-1 for generating hash output, so the result is 160-bits long. When the responder chooses $k = 10$, it means that the 10 left-most significant bits must be zero but the remaining bits can be either 1 or 0. Therefore, the chance of a user to get the result which have 10-zero bits at the beginning of the output would be 2^{-10}. As a result, if `ad3` shares value J, which is 64-bits long, to coordinated attackers (`Co-ad`) trying to find a solution, they can save time in this process depending on the number of participating machines in `Co-ad`. They can achieve this technique because a puzzle construction based on hash function can be computed faster in a parallel fashion. This attack technique is defined in terms of a coordinated attack [18]. Another attack on hash-based puzzles has been introduced by Price [15]. The experiment and results of are demonstrated in Section 3.

3 A New Approach

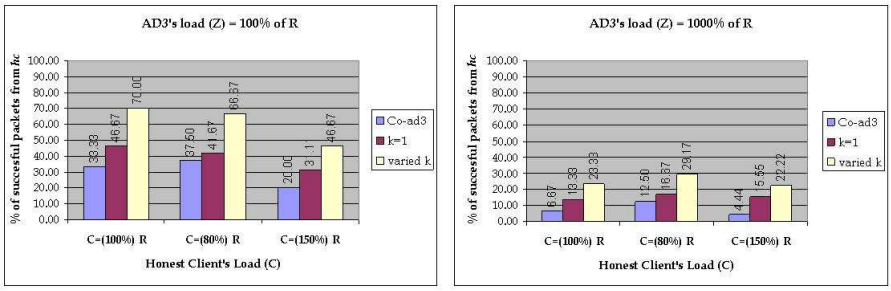
Vulnerabilities based on unbalanced computations between an initiator and a responder have been revealed in Section 2. This vulnerability leads to the risk of the responder's machine to be overwhelmed by the coordinated adversaries. This section propose a technique to mitigate this problem. The results show that the proposed technique can be help to deal with such attack.

In the experiment, we re-construct a HIP model by adopting the concept of a time-lock puzzle [16] which has been developed by Rivest et al. The fundamental property of time-lock puzzles is that they require a precise amount of time to be solved and can not be solved in parallel computation. Therefore, the responder can select the predetermined time period for a puzzle for delaying the incoming requests when the responder has heavy load to serve.

In order to generate a time-lock puzzle, the responder has to determine the amount of time for the client to spend in solving the puzzle (T) and estimate the initiator capacity in calculating repeated squaring per second (S). Next, the responder computes the number of repeated squaring $t = T \cdot S$ that must be computed by the initiator in order to find a solution of a time-lock puzzle. Finally, the responder forces the initiator to calculate $b = a^{2^t} (\text{mod } n)$, where n is the product of two large primes p and q. Because the responder knows the factors p and q, he can compute b much faster by first computing $2^t \ mod \ \phi(n)$.

We setup simulation for evaluating a system with corresponding to coordinated adversaries type 3 (`Co-ad3`). When we insert a time-lock puzzle into HIP model at step two of an initiator, the result for `hc` and `Co-ad3` will be improved to the higher percentage of successful packets approximately equal to the experiment 2. In the experimental results, graphs represented with `Co-ad3` term are simulated by using a hash-based puzzle with adjustable k, while graphs represented with `k=1` and `varied-k` are simulated by using a time-lock puzzle with fixed k=1, and adjustable k, respectively. From figure 4, if we compare the graph of `ad3` at workload `hc` = 80% of `R` in Figure 3(a) with `Co-ad3` in Figure 4(a), the throughput falls from 66.67% to 37.50%. Once we employed time-lock puzzle as shown in the graph `k=1` and `varied-k` of figure 4, the throughput will increase to approximately the same as experiment 2 (shown in figure 3).

The reason is that `Co-ad3` has been forced to spend time specified by the responder until the time-lock puzzle has been solved. This period is similar to the period in experiment 2 in which normal `ad3` spends time to search for a correct solution of a hash-based puzzle. As a result, when the responder constructs a time-lock puzzle, the responder can control time required for the initiator to solve a puzzle more precisely. Figure 4 displays results from the simulation which adopts the time-lock puzzle technique into the system. We see that use of a hash-based puzzle against a coordinated adversary results is less throughput than no puzzle at all (k=1). Use of a time-lock puzzle with varied k effectively increases the percentage of successful packets from the `hc`.



(a) Co-ad3 = 100% (b) Co-ad3 = 1000%

Fig. 4. Percentage of throughput from *hc* and `Co-ad3` comparing between a *hash-based* puzzle and a *time-lock* puzzle

4 Conclusion and Future Work

We have explored unbalanced computational vulnerabilities on HIP which cause the responder to deplete resources and then terminate all processes. Even though the experimental result demonstrates that puzzle difficulty can mitigate the problem of resource exhaustion attacks, `Co-ad` can employ an alternative technique to attack hash-based puzzles and force the responder to verify the signature. In order to prevent such attacks, we replace hash-based puzzles with time-lock puzzles.

According to a comparison by Feng et al. [9], the most interesting property of time-lock puzzle is non-parallelizability that prevents `Co-ad` to speed up the process of searching a solution by distributing a puzzle to other high-performance machines. Moreover, time-lock puzzles also provide fine-grained control in order to precisely adjust puzzle difficulty by the responder. Although the integration of time-lock puzzles mitigates the problem of `Co-ad3`, the underlying computation for constructing time-lock puzzle is a major concern because the puzzle generation is limited by the calculation of modular exponentiation which has greater magnitude than hash-based puzzle. Some example uses of time-lock puzzles have been evaluated and identified by Mao [13], Back [5], and Feng et al. [9] which all suffer from the same problem.

We are currently working on a new technique to construct a client puzzle satisfying desirable properties identified by Aura et al. [4]. In particular, puzzles should be inexpensive for the responder to generate and verify, and impossible to precompute a solution by the initiator. Two additional properties which enhance DoS-resistant protocols for preventing `Co-ad3` should be included:

1. *the puzzle should not be solvable in parallel for obtaining an output in less than a specific time.*
2. *the responder should be able to precisely control puzzle difficulty in a linear manner.*

References

1. I. Al-azzoni. The Verification of Cryptographic Protocols using Coloured Petri Nets. Master of Applied Sciences Thesis, Department of Software Engineering, McMaster University, Ontario, Canada, 2004.
2. T. Aura, A. Nagarajan, and A. Gurtov. Analysis of the HIP Base Exchange Protocol. In *Proceedings of 10th Australasian Conference on Information Security and Privacy (ACISP 2005)*, pages 481 – 493, Brisbane, Australia, Jun 2005.
3. T. Aura and P. Nikander. Stateless Connections. In *International Conference on Information and Communications Security*, pages 87–97, Beijing, China, Nov 1997.
4. T. Aura, P. Nikander, and J. Leiwo. DoS-resistant authentication with client puzzles. In *Security Protocols Workshop 2000*, pages 170–181, Apr 2000.
5. A. Back. Hashcash - A Denial of Service Counter-Measure, 2002. `http://citeseer.ist.psu.edu/back02hashcash.html`.
6. J. Beal and T. Shepard. Deamplification of DoS Attacks via Puzzles. Available: `http://web.mit.edu/jakebeal/www/Unpublished/puzzle.pdf`, 2004.
7. Computer Emergency Response Team (CERT). SYN Flooding Attack. [Online]. Available: `http://www.cert.org/advisories/CA-1996-21.html`, 1996.
8. E. M. Doyle. Automated Security Analysis of Cryptographic Protocols using Coloured Petri Net Specification. Master of Science Thesis, Department of Electrical and Computer Engineering, Queen's University, Ontario, Canada, 1996.
9. W. Feng, A. Luu, and W. Feng. Scalable, Fine-grained Control of Network Puzzles. Technical report 03-015, OGI CSE, 2003.
10. Y. Han. Automated Security Analysis of Internet Protocols using Coloured Petri Net Specification. Master of Science Thesis, Department of Electrical and Computer Engineering, Queen's University, Ontario, Canada, 1996.
11. M. Jakobsson and A. Juels. Proofs of work and bread pudding protocols. In *the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS 99)*, Sep 1999.
12. A. Juels and J. Brainard. Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks. In *the 1999 Network and Distributed System Security Symposium (NDSS '99)*, pages 151–165, San Diego, California, USA, Feb 1999.
13. W. Mao. Time-Lock Puzzle with Examinable Evidence of Unlocking Time. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 95–102, London, UK, 2000. Springer-Verlag.
14. R. Moskowitz. The Host Identity Protocol (HIP). Internet Draft, Internet Engineering Task Force, Jun 2006. `http://www.ietf.org/internet-drafts/draft-ietf-hip-base-06.txt`.
15. G. Price. A General Attack Model on Hash-Based Client Puzzles. In *9th IMA International Conference on Cryptography and Coding*, pages 319 – 331, Cirencester, UK, 16-18 Dec 2003. Springer-Verlag.
16. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock Puzzles and Timed-release Crypto. Technical Report TR-684, Massachusetts Institute of Technology, Cambridge, MA, USA, 10 Mar 1996.
17. W. A. Simpson. IKE/ISAKMP Considered Harmful. *USENIX*, 24(6), dec 1999.
18. J. Smith, J. M. González Nieto, and C. Boyd. Modelling Denial of Service Attacks on JFK with Meadows's Cost-Based Framework. In *Fourth Australasian Information Security Workshop (AISW-NetSec'06)*, volume 54, pages 125–134, 2006.