

Cost-based and Time-based Analysis of DoS-resistance in HIP

Suratose Tritilanunt Colin Boyd Ernest Foo Juan Manuel González Nieto

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

Abstract

We develop a formal model of the Host Identity Protocol (HIP) based on Timed Coloured Petri Nets (Timed CPNs) and use a simulation approach provided in CPN Tools to achieve a formal analysis. We aim to examine unbalanced computation that leads to resource exhaustion attacks in key exchange protocols comparing among a legitimate initiator, four types of adversary who attempt to deny the service at different stages of the protocol execution, and a responder. By adopting the key idea of Meadows' cost-based framework and refining the definition of operational costs during the protocol execution, our simulation provides an accurate cost estimate of protocol execution comparing between those principals. Under four defined attack strategies, however, Meadows' cost-based framework generates a different outcome compared with the simulation approach from Timed CPNs. Analysis of our experimental results reveals a limitation of Meadows' cost-based framework for addressing DoS threats.

Keywords : Host Identity Protocol (HIP), Cost-based Framework, Timed Coloured Petri Nets

1 Introduction

Denial of service (DoS) attacks are serious threats to computer networks since they attempt to prevent the responder and legitimate users to establish connections. DoS attacks might attempt to overwhelm either network bandwidth or responder's resources, including memory and central processor unit (CPU). Although both of these targets are important, in this paper we focus on how to protect the responder from exhausting resources based on vulnerabilities of the underlying key exchange protocol.

Most key exchange protocols aim to establish and exchange cryptographic parameters. These protocols require the associated responder to spend some expensive computations, such as modular exponentiation in the case of Diffie-Hellman and RSA digital signature algorithm, while the initiator requires computations which are cheaper than for the responder. As a result, these protocols are susceptible to denial-of-service (DoS) attacks because adversaries can launch DoS attacks to overwhelm resources of the responder before the responder can detect the attack and authenticate identity of the launchers.

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at Thirtieth Australasian Computer Science Conference (ACSC2007), Ballarat, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 62. Gillian Dobbie, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

In order to design cryptographic protocols, researchers have developed general-purpose verification tools to use in modeling and verifying them over many years. One example formal method is Petri Nets. Although Petri Nets have been developed since 1962 by Petri (1962), they have been recently used to verify cryptographic and security protocols. There are two forms of Petri Nets: ordinary Petri Nets and high level Petri Nets. In this paper, we focus on high level Petri Nets, i.e. Colored Petri Nets, because they have several benefits in the analysis and verification of cryptographic protocols as stated by Jensen (1998b).

To develop a formal framework of cryptographic protocols, we use CPN Tools (2004) which is an interaction tool for modeling and analysing Coloured Petri Nets. Our formal model is developed based on Timed Coloured Petri Nets (Timed CPNs) and uses a simulation approach provided in CPN Tools to achieve a formal analysis. The goals of simulation of this protocol are to explore vulnerabilities of DoS-resistant protocols by examining unbalanced computation that leads to resource exhaustion attacks in key exchange protocols. Simulation approaches are valued in the research community not only for exploring vulnerabilities in cryptographic protocols, but also guaranteeing security services of such protocols similar to a mathematical approach. Using simulation approaches has several benefits over mathematical analysis; for instance, simulation provides *flexibility* to the developer to choose, examine and adjust parameters for evaluating the system. In addition, simulation provides *visualization* to users who can see and learn what is happening during the simulation of cryptographic protocols to gain more understanding for evaluating the correctness of those protocols.

In this work, we adopt the key idea of Meadows' cost-based framework (2001) and refine the definition of operational costs by assigning an accurate estimate of various cryptographic operations during the protocol execution. Meadows introduced a systematic framework to analyse DoS-resistant protocols by computing and comparing the cost incurred by both parties at each step in a key exchange protocol. Meadows analysed the STS protocol (a protocol without special DoS resistance properties) and later Ramachandran (2002) and Smith (2006) used Meadows' framework to analyse the JFK protocol (2002) in order to demonstrate its DoS prevention capabilities.

Surprisingly, there has been little interest in the research community in applying Meadows' framework to different protocols. Moreover, the limited application so far has suffered from two significant shortcomings which make the results of restricted value.

1. The cost analysis has only taken into account *honest* runs of the protocol. In principle, the adversary (typically the client in a client-server protocol) can deviate arbitrarily from the protocol in order to achieve an attack. By only taking

into account honest behaviour it is quite likely that logical attacks will be missed. While Meadows certainly recognised this fact, no research has yet examined the effectiveness of the framework in detecting such potential attacks.

2. Meadows used only a coarse measure of computational cost, with three levels denoted as cheap, medium or expensive. In practice it can be quite difficult to classify and compare operations in such a limited way. For example, in Meadows’ classification digital signature generation and verification are counted as of equal cost, yet in practice an RSA signature generation may take 2 or 3 orders of magnitude more effort than RSA signature verification.

Motivated by the above two limitations, this paper provides a refinement of Meadows’ cost-based framework. For our sample protocol we use the Host Identity Protocol (HIP) (Moskowitz 2006), which has built-in DoS resistance. Using coloured Petri nets as our formalism, we provide a formal specification of HIP to allow automatic searching of adversary and victim cost under four different adversarial attack strategies. Moreover, we examine the tolerance of HIP under different levels of puzzle difficulty as well as investigate the most effective attack strategies by measuring the successful throughput in order to compare the result with the prediction from Meadows’ framework. Although our range of adversarial actions is limited, we believe that we have demonstrated that formal specification and analysis is an effective way to extend the scope and value of the Meadows’ framework which paves the way for more detailed application of the framework.

The main contributions of this paper are:

- a refinement of Meadows’ cost-based framework to more accurately represent the cost of typical cryptographic algorithms;
- the first formal specification and automatic analysis of Meadows’ framework;
- simulation and analysis of HIP under normal conditions and four scenarios of DoS attacks;
- a time-based analysis that reveals a limitation of Meadows’ cost-based framework for addressing DoS threats.

2 Background and Previous Work

The purpose of this section is to provide the background on the Meadows’ cost-based framework and HIP, as well as the previous work on the analysis of security protocols using Coloured Petri Nets.

2.1 Meadows’ Cost-Based Framework

Meadows’ framework (2001, 2003) works by comparing cost to the attacker and cost to the defender, defined using a *cost set*¹. To model the protocol framework, we need to calculate the cost of the sequence of protocol actions, comparing between an attacker and a defender. However, each action could use different CPU or memory resources during a protocol execution. Therefore, we need to find an alternative way to calculate the precise total cost of these actions and the technique for comparing them.

Considering the characteristic of DoS attacks, there are two possible ways mentioned by Meadows (2001) to cause the defender to waste resources. First,

the defender may process a bogus instance of a message inserted by the attacker into a protocol. The cost to an attacker is the cost of creating and inserting the bogus message, while the cost to the defender is the cost of processing the bogus message until an attack is detected. Second, the defender participates in a bogus instance of the protocol with the attacker. The cost of this situation is equivalent to the cost of running the entire protocol until the defender can detect the attack or the attack stops.

At this stage, we limit the abilities of adversaries during the protocol execution to take one of a small number of possible actions when the protocol specifies that a message should be sent: adversaries either continue normally with the protocol or partially complete the protocol. Intuitively these are the most obvious ways for adversaries to make the defender use unwanted resources. In our examples adversaries send messages at two points in the protocol, so we limit adversaries to either attack the first message by flooding a large number of random messages to overwhelm the resources of the responder, or to attack the second message by faking its packets to waste the responder resources for verifying it.

2.2 Host Identity Protocol

The host identity protocol (HIP) has been developed by Moskowitz (2006). Later, Aura et al. (2005) found some vulnerabilities and proposed guidelines to strengthen the security of HIP. The base exchange of HIP is illustrated in Figure 1.

HIP is a four-packet exchange protocol which allows the initiator I on the IP address IP_I and responder R on the IP address IP_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 over HI .

A one-way hash function $H(\cdot)$ is used to form the puzzle, while H_{K_s} represents a keyed hash function using session key K_s to generate a hash-MAC (HMAC). The value s is a periodically changing secret only known to the responder. $LSB(t, k)$ 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 . To generate session keys K_e and K_s , HIP employs Diffie-Hellman key agreement protocol. 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 proposed by Jakobsson and Juels (1999) for countering resource exhaustion attacks. In a proof-of-work, HIP extends the concept of a *client puzzle*, first proposed by Juels and Brainard (1999), and later implemented by Aura et al. (2000) for protecting the responder against DoS attacks in authentication protocols. Moreover, HIP allows the additional feature of the client puzzle that helps the responder to delay state creation (Aura & Nikander 1997) until the checking of the second incoming message and the authentication has been performed in order to protect the responder against resource exhaustion attack.

2.3 Coloured Petri Nets (CPNs)

CPNs (Jensen 1998a, Jensen 1997) are one type of high-level nets based on the concept developed back

¹More details are in Appendix A.

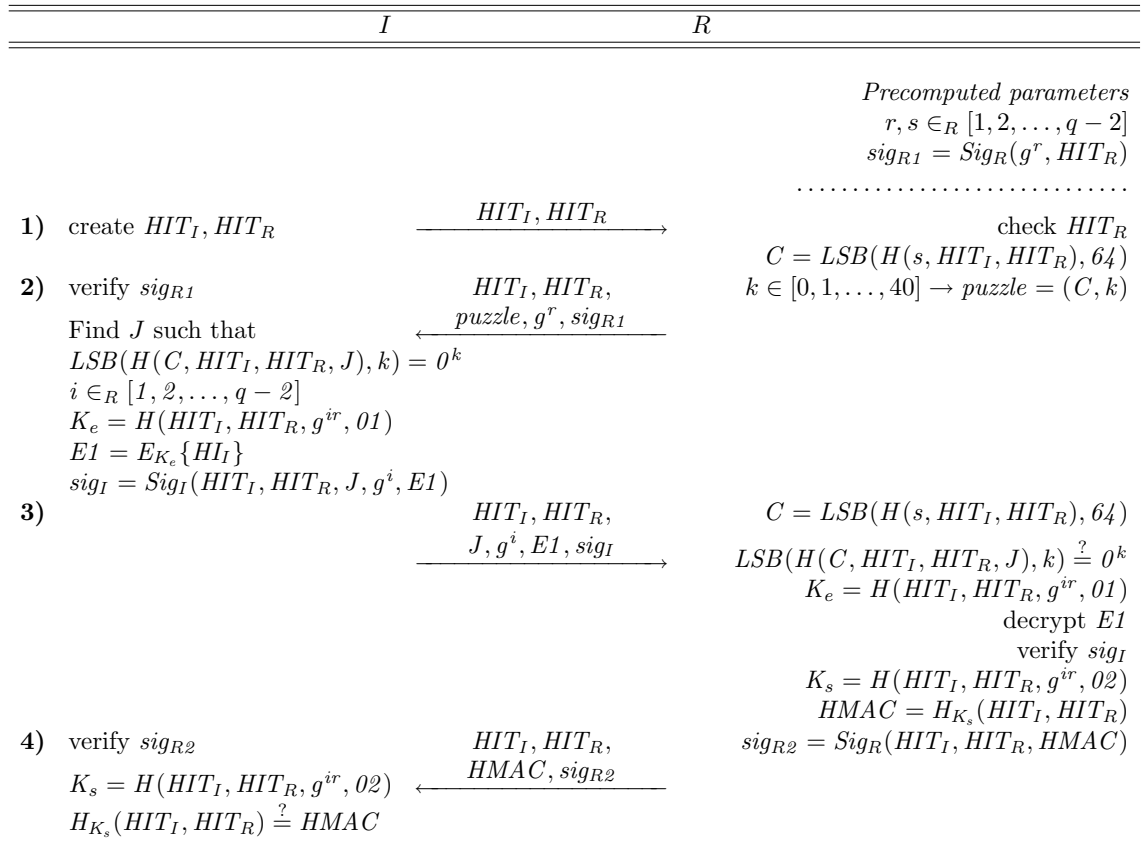


Figure 1: HIP Protocol

in 1962 by Petri (1962). CPNs is a state and action oriented model including places, transitions, and arcs.

Over many years, cryptographic and security protocols have been modeled and verified using CPNs. Neih and Tavares (1993) implemented models of cryptographic protocols in the form of Petri Nets. To explore vulnerabilities of such protocols, they allowed an implicit adversary with limited abilities to launch attacks and then examined the protocol using exhaustive forward execution. Doyle (1996) developed a model of three-pass mutual authentication and allowed an adversary to launch multiple iteration and parallel session attacks. Han (1996) adopted CPNs for constructing a reachability graph to insecure states and examining the final states in OAKLEY. Al-Azzoni (2004) has developed a model of Needham-Schroeder public key authentication protocol and Tatebayashi-Matsuzaki-Neuman (TMN) key exchange protocol.

To the best of our knowledge, there is no implementation of CPNs focusing on an exploration of unbalanced computational threats that lead to resource exhaustion attacks in key exchange protocols.

3 Cost-based Framework

This section provides a cryptographic benchmark of some specific algorithms which is an alternative technique to measure CPU usage for representing more specific costs instead of an original representation by Meadows. Finally, we present HIP by means of a cost-based specification.

3.1 Refinement of Meadows' Framework

An obvious limitation of the original formulation of the framework is that the computational costs are not

defined precisely, but consist instead of a small number of discrete values. Indeed Meadows (2001) herself called this a "crude and ad hoc cost function". In order to obtain a more useful cost comparison we need to obtain a more accurate estimate of the computational and/or storage costs required to complete the protocol steps. How to do this is not as obvious as it may seem at first.

When comparing efficiency of different cryptographic protocols it is customary to count the number of different types of cryptographic operations. For protocols that use public key operations it is common to ignore most operations and count only the most expensive ones, which typically are exponentiations in different groups (traditionally written as multiplications in elliptic curve computations). However, for the protocols that interest us this is definitely not acceptable. As mentioned above, one common technique in DoS prevention is to demand that clients solve *puzzles* which require the client to engage in some computational effort, such as to iterate a hash function a large number of times. Although one hash computation takes minimal time in comparison with a public key operation, ignoring a large number of hash computations could make the cost function ignore completely the DoS prevention mechanism when a puzzle is used. Therefore we need to be able to compare directly the cost of all different kinds of cryptographic operations.

Comparing operations like hashing and exponentiations directly seems very hard to do since they are based on completely different types of primitive instructions. Therefore we have resorted to an empirical comparison which compares benchmark implementation on common types of processors. While we acknowledge that the detailed results may differ considerably for different computing environments (CPU,

Table 1: Computational Cost of CPU and Time Usage of Specific Algorithms

Hash	kCycle/Block	nsec/bit	Symmetrical Crypto	kCycle/Block	nsec/bit
SHA-1 (512bits/block)	1.89	1.84	DES (64bits/block)	0.75	5.86
HMAC/MD5 (512bits/block)	0.59	0.58	AES (128bits/block)	0.53	2.05
Public-Key Crypto	kCycle/ops	nsec/bit	Key Exchange	kCycle/ops	nsec/bit
RSA Encryption / RSA Verification	383.66	187.08	Diffie-Hellman Key-Pair Generation	4605.65	2245.80
RSA Decryption / RSA Signature	9985.47	4869.11	Diffie-Hellman Key Agreement	8100.69	3950.05

memory, and so on) we believe that the obtained figures are indicative of the true cost in typical environments and allow reasonable comparisons to be made.

For our cost estimates, we use the cryptographic protocol benchmarks of Wei Dai (2004). These include tested speed benchmarks for some of the most commonly used cryptographic algorithms using Crypto++ library² version 5.2.1 on a Pentium 4 2.1 GHz processor under Windows XP SP 1. More cryptographic benchmarking has been done by Gupta (2002) and Tan (2004) on the specific processors; however, they did not test public-key encryption.

Table 1 only presents the results for some specific cryptographic algorithms available for negotiating during the HIP based exchange defined in HIP specification. The units that we use in Table 1 are kilocycles per block (note that block size varies for different algorithms). This allows direct comparison of CPU usage and may be expected to be similar on processors with different clock speeds. This entails conversion from the original data which uses direct time measurements.

From the table, we are able to estimate the CPU usage in cycles per block for common hash functions and the symmetric key encryption, and cycles per operations for the 1024-bit key lengths of public-key encryption and Diffie-Hellman key exchange algorithm. Once we get a result, we scale it down by a factor of 1000 (kilo) and apply these costs in our formal specification and analysis. Before we can export these values into CPN Tools, we round them into an integer representation because CPN Tools limits only integers in the simulation process.

3.2 HIP Cost-Based Specification

We shall use HIP as an example in order to provide an analysis using Meadows' cost-based framework. By employing the concepts and definition from Appendix A, we are able to interpret HIP based exchange between two principals by means of cost-based specification as shown in Figure 2.

For the first message the initiator (I) performs $create_HIT_I, create_HIT_R$ and sends a requested message to the responder (R). Meanwhile, R performs the actions $pre_create_r, pre_create_s, pre_exp_g^r, pre_sign_sig_{R1}$ before a requested message arrives to the connection queue. These *pre-actions* are pre-computed in order to protect R from wasting some amount of resources at the initial state. Once R receives this request, it

spends only a cheap operation to check HIT_R for completing the first message. Considering a second message, R performs $compute_hash_C, create_k$ and sends $HIT_I, HIT_R, puzzle, sig_{R1}$ to I . When I receives this message, I then performs $verifysig_sig_{R1}, accept_1$. If $verifysig_sig_{R1}$ fails, the process stops. Otherwise, I accepts the message and then continues to brute-force search a puzzle solution $compute_hashsolution_J$.

To construct the third message, I performs $computekey_K_e, encrypt_{HI}, sign_sig_I$ appended with the signature sig_I and sends it to R . Having received the message from I , R begins to check C by recovering the message $R1$ counter and then hashing received HIT_I, HIT_R with the server secret s . If this process is verified correctly, R validates the puzzle solution by performing $verify_solution$. If invalid, the process stops. Otherwise, R performs $computekey_K_e, decrypt_{E1}$ for recovering HI_I . After that, R finally verifies the signature of I by performing $verifysig_sig_I$. Also, the output will be either success which causes the responder to continue to the next steps, or failure which causes the responder to reject and clear the connection queue. In the fourth message, R performs $computekey_K_s, compute_hash_{MAC}, sign_sig_{R2}$ and sends the message $HMAC, Sig_R[HIT_I, HIT_R, H_{K_s}(MAC)]$ to I . Finally, I begins to compute $computekey_K_s, verify_hash_{MAC}, verifysig_sig_{R2}, accept_3$ for key confirmation.

By using the original cost functions, the tolerance relation for the first message would be the pair (Cheap, Cheap) which is an acceptable result, while for the third message would include the pair (Medium, Cheap), or (Medium, Medium), or (Expensive, Expensive) which is also a reasonable result. However, these results are too coarse for achieving formal analysis of the computational costs. As a result, we replace such original cost measures suitable for hand analysis with more precise values from Table 1 in our simulation.

4 Automated Simulation by CPN Tools

This section provide the detail of the HIP cost-based and time-based model as well as the experimental results of these constructions.

4.1 The Construction of HIP in CPNs

HIP Cost-based Construction: In this model, we insert a special place to every cryptographic transition called a *cost-place* for capturing and displaying the computational cost of individual steps. This

²Available at <http://www.eskimo.com/~weidai/cryptlib.html> and <http://sourceforge.net/projects/cryptopp/>

MSG	Cost-based Function
1)	$I \rightarrow R :$ $create_HIT_I, create_HIT_R \parallel$ $HIT_I, HIT_R \parallel$ $pre_create_r, pre_create_s, pre_exp_g^r, pre_sign_sig_{R1}, verify_HIT_R$
2)	$R \rightarrow I :$ $compute_hash_C, create_k \parallel$ $HIT_I, HIT_R, puzzle, sig_{R1} \parallel$ $verifysig_sig_{R1}, accept_1$
3)	$I \rightarrow R :$ $compute_hashsolution_J, computekey_K_e, encrypt_{HI}, sign_sig_I \parallel$ $HIT_I, HIT_R, J, g^i, K_e\{HI_I\}, Sig_I[HIT_I, HIT_R, J, g^i, K_e\{HI_I\}] \parallel$ $verify_C, verify_solution_J, computekey_K_e, decrypt_{E1}, verifysig_sig_I, accept_2$
4)	$R \rightarrow I :$ $computekey_K_s, compute_hash_{MAC}, sign_sig_{R2} \parallel$ $HIT_I, HIT_R, HMAC, Sig_R[HIT_I, HIT_R, HMAC] \parallel$ $verifysig_sig_{R2}, computekey_K_s, verify_hash_{MAC}, accept_3$

Figure 2: HIP Protocol in the Cost-Based Framework Notation

process is performed by adding the CPU usage data from Table 1. Measurement of CPU usage indicates individual steps as well as the total cost compared between an initiator, every single type of adversary, and a responder, when such principals complete a single round of the protocol simulation.

HIP Time-based Construction: We attempt to design a more realistic model by adopting the concept of time into the HIP time-based simulation and analysis. That means every cryptographic process requires some amount of time calculated by using cryptographic benchmarks of Crypto++ library (Dai 2004). In the Timed CPNs, the concept of the *simulated time* or the *model time*³, which is represented by the system clock in the tool, has been introduced. Once we have attached the system time into tokens, we can see the sequence or action of states that tokens move to as a temporal analysis. It means only tokens which hold the current time as presented on the clock can be traversed to the next position, while the others have to wait until the system clock reaches their value.

Furthermore, the concept of the responder’s resource is used in this evaluation. This resource represents the number of available connections in the queue to process the incoming requests. Once the responder has to deal with requests, the responder spends one unit of resource for each individual request. It means that if incoming packets exceed the responder capacity, the responder then rejects further incoming packets until he has either processed the legitimate traffic or detected bogus messages and removed them from the storage. Note that the process to order incoming packets arriving to the connection queue is non-deterministic because these packets have been arranged randomly by CPN Tools.

4.2 Adversary’s Ability

Apart from the honest client (**hc**) who initiates the legitimate traffic, we allow four types of adversary who have the similar goal to deny the service of the responder by overwhelming the responder’s resource.

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 for 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.

To clarify the description of adversaries’ ability, the major goal of Type 1 adversary is to overwhelm the responder’s storage by sending a large number of requested packets, for example, a denial-of-service attack via ping (CERT 1996a) and SYN flooding attack (CERT 1996b), while the major goal of Type 2, 3, and 4 adversary is to force the responder to waste computational resources up to the final step of the digital signature verification and digital signature generation which are expensive operations.

4.3 Experiment

To obtain experimental results, we set up two main different simulations;

Experiment 1: The purpose of the first experiment is to compare computational cost of the protocol execution based on the key concept of Meadows’ cost-based analysis between all principals with some possible ranges of puzzle difficulty (k) including $k = 0$ (which means that no puzzle is required), easiest value $k = 1$ for contrasting the difference between **ad3** and **ad4**, intermediate values $k = 10$, $k = 20$, and $k = 40$ for a hardest value as instructed in HIP specification. In this simulation, we allow individual initiators to initiate a request token only once, while the responder is able to flexibly adjust the puzzle difficulty within initially defined values. Once the

³More formal descriptions are available on the official website of CPN Tools, <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>

Table 2: Comparison of Computational Cost of HIP with $k=1$ and $k=10$

Authentication Protocol	Initiator		Responder			
	$k=1$	$k=10$	$J, E1, sig_I$ valid	only J valid	Everything invalid	
HIP	<i>hc</i>	19973	22017	19591	-	-
	<i>ad1</i>	0	0	-	-	2
	<i>ad2</i>	14982	17026	19591	-	-
	<i>ad3</i>	4	2048	-	4998	-
	<i>ad4</i>	0	0	-	-	6

simulation has arrived to the final step, we record the total computational cost of individual user comparing to the responder on specified ranges of k .

Experimental Result: During the protocol execution, the initiator sends a request message to the responder using the host identity tag (*HIT*) which is a hash of the host identifier (*HI*) used in HIP payload and to index the corresponding state in the end hosts. Therefore, the initiator only employs cheap operations at the beginning step. We assume that the computation at this step can be precomputed, so the cost at the first operation would be negligible. Once the responder receives the requested message, the responder requires a hash operation and some values from the precomputation for returning to the initiator in the second step. This operation is a cheap operation similar to the initiator's.

When the initiator receives the replied message, only honest clients participate in the verification of *HIT* and responder's signature, so the cost is equal to the *HIT* verification plus signature verification. In the case of *ad1*, it does not take any further actions after the first message, therefore the computational cost is zero for the second stage. The operations in message three of the initiator include the brute-force search to find the puzzle solution, and the key generation. The cost of solving a puzzle depends on the value of k in the puzzle message field. However, only an *hc*, *ad2*, and *ad3* is required to solve the puzzle solution. Like *ad1*, the *ad4* does not attempt to solve the puzzle. As a result the puzzle difficulty does not affect computational cost on this type of adversary. Another important thing to note is that, the cost of the adversary to spoof, insert, or interrupt the message has not been defined in this phase. So, we set the cost of generating randomly chosen messages in the case of *ad3* and *ad4* to be zero.

Considering the task on the responder's machine when it receives the third-step message from the initiator, the responder begins to validate the puzzle solution which is defined as a cheap operation because the responder performs only one hash calculation. If it is invalid, the process will stop and the responder will drop the corresponding packet from the connection queue (the system will return a resource to the responder). Otherwise, the responder performs the decryption to obtain an initiator's public key. The responder finally verifies the signature by using the initiator's public key obtained in the previous step. The result would be either valid or invalid. After the authentication has been completed, the responder and the initiator will perform a key confirmation and start to exchange information. Table 2 summarizes the computational cost when the puzzle difficulty is set to $k=1$ or $k=10$ comparing between every principal (honest client and adversaries) and the responder. The experimental result shows that the most effective adversary is *ad3* (the greatest different threshold be-

tween *ad3* and the responder) because *ad3* can force the responder to engage in the expensive tasks, i.e. digital signature verification.

Figure 3 illustrates the computational cost of *hc*, *ad2*, and *ad3*, respectively. In the comparison charts, we measure the cost of those users who engage in solving the puzzle of the difficulty level $k = 0, 1, 10, 20, 40$.

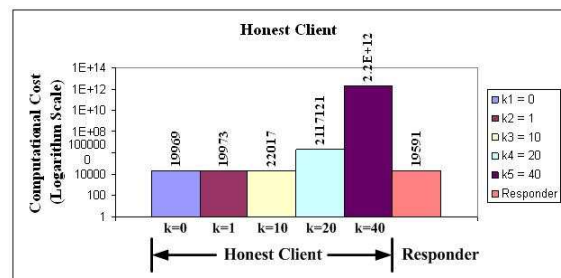
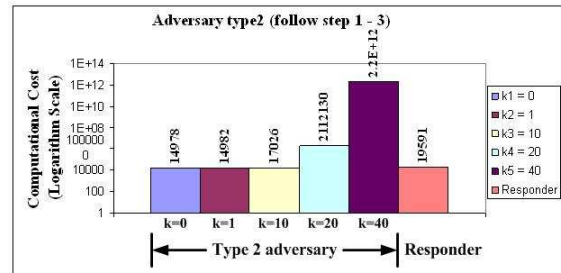
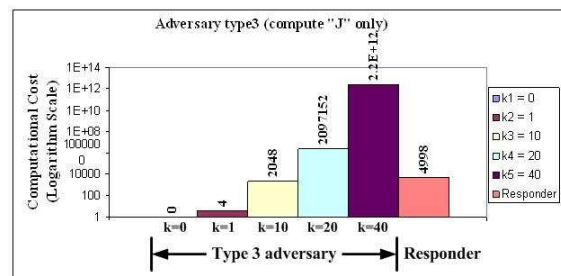

 (a) Computational Cost between *hc* and a Responder

 (b) Computational Cost between *ad2* and a Responder

 (c) Computational Cost between *ad3* and a Responder

 Figure 3: Comparison of Computational Cost on HIP with different ranges of k

Comparison between Figures 3(a) and 3(b) shows that *hc* and *ad2* incur similar computational costs for the same value of k chosen. This illustrates well the effectiveness of HIP in achieving its aims in resisting DoS attacks, at least against this type of adversary. On the other hand, *ad3* and *ad4* spend very small computational resources compared with the responder because both adversaries use some random message elements. This situation would bring the responder to the risk of DoS attacks if the value of k is not chosen properly. Figure 3(c) indicates that a value

of k a little above 10 would be appropriate to ensure that **ad3** uses as much computation as the responder.

Experiment 2: The purpose of the second experiment is to measure a tolerance of the responder and to identify the most effective strategy to deny service. The result can be used to compare with the first experiment for confirming whether Meadows’ cost-based framework is effective for evaluating the DoS-resistant protocols or not. In order to examine the protocol, we select two possible values from the experiment 1, i.e. $k = 1$ and $k = 10$ for achieving this simulation. We choose those two values because we want to investigate the different behaviour of **ad3** and **ad4** (if we choose $k = 0$ no client puzzle is required, the task of both adversaries will be identical). Also, both of those values do not put excessive computational effort to the honest client and the total task is still in the acceptable threshold comparing to tasks on the responder (see Figure 3(a)).

In order to allow the responder to flexibly adjust puzzle difficulty between those two values more efficiently, we simply insert a counter into the model for measuring the condition of a responder’s workload. Once the workload has reached the maximum tolerance, the responder will increase the puzzle difficulty to the higher level for delaying the incoming rate of requested messages.

Additional to the first experiment, we allow a responder to participate with a pair of initiators (**hc** and a single type of adversary). We assume that a responder has to deal with different strategies of adversary and different amount of packets which consist of both legitimate and bogus messages. With regard to the number of packets, we allow three different scenarios for both honest client and adversary;

- *Honest Client:* initiates requests, C , at 80%, 100%, and 150% of the responder’s capacity (R).
- *Adversary:* floods bogus requests, Z , at 100%, 200%, and 1000% of the responder’s capacity (R).

Considering the initiator’s packet, there are different actions from initiators and the responder during the protocol run. Honest clients initiate a request only once and keep waiting to process next steps. This delay is described by means of Timed CPNs, i.e. every transition which relates to cryptographic operations is defined as a timed process. This amount of time is calculated and assigned by using data from cryptographic benchmarks shown in Table 1. During the simulation, if honest client’s requests have been rejected under DoS circumstances, **hc** gives up to open another session. In term of adversaries, there are two different situations when their packets are rejected by the responder; 1) the responder detects the bogus messages during the verification steps, and 2) the responder does not have enough resources for serving any requests. Once the responder detects the attack and rejects those packets, adversaries will lose those packets from the system.

Finally, in order to examine the tolerance of HIP protocol under different attack strategies, each individual adversary has been made a pair with an honest client during the protocol execution. To make an evaluation, the number of successful legitimate requests that the responder can serve under different adversary’s abilities has been measured as the percentage of successful packets. We achieve this task by inserting places for displaying the number of completed and rejected messages at the responder’s network.

Experimental Result: In the second experiment when we prohibit the responder’s ability to adjust k ,

we have seen from Figure 4 that when adversaries increase the number of bogus messages in the system, the percentage of successful messages from honest clients to obtain a service will drop drastically. Comparing different types of adversary, the most effective is **ad4** who sends bogus messages to the responder by crafting messages randomly. This is because **ad4** can flood a large number of messages to overwhelm the responder’s resource quicker than the others, which causes the responder to reject the next incoming messages from honest clients. Although adversaries’ packets will be detected and discarded by the responder, all adversaries are able to flood new bogus messages as soon as they receive returned packets from the responder at phase two.

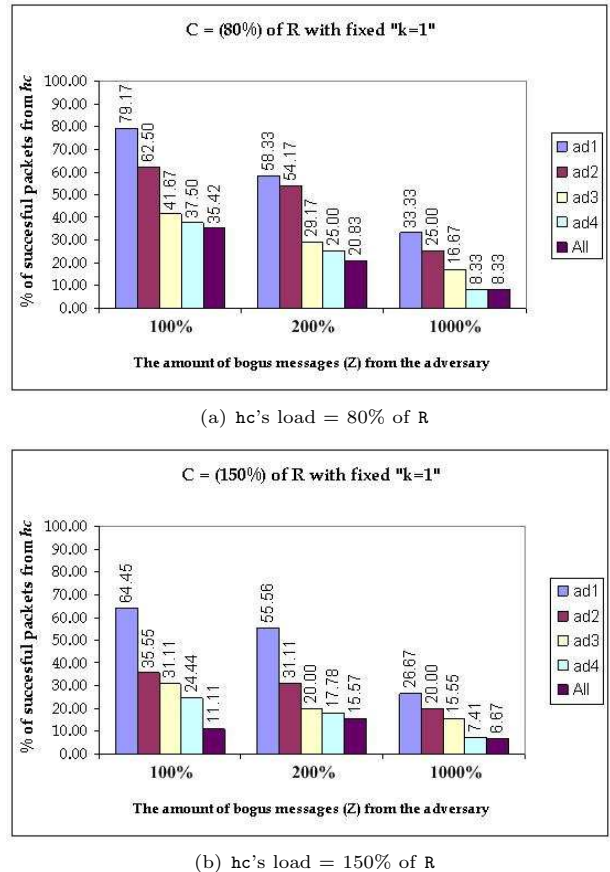


Figure 4: Percentage of throughput from hc with $k=1$

Comparing **ad1** and **ad4**, even though both of them craft random messages, **ad4** achieves its goal at a higher rate than **ad1**. That is because the responder can process the incoming request at step 1 and clear a connection queue faster than step 3. (In step 1, the responder only participates in the protocol by choosing the puzzle difficulty (k) and pre-computed information, and then returns it to **ad1**, which involves an expensive verification in step 3.) 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 the problem of flooding messages to overwhelm a resource at step 1 by adopting a stateless-connection. On the other hand, the task of **ad4** to fill up the responder’s queue at step 3 by flooding random messages can be achieved more easily than **ad1**. Even though HIP integrates a gradual authentication in this step, the process of checking a puzzle solution and a digital signature is longer than the whole process at step 1, therefore, the responder’s queue would be more easily overwhelmed in step 3 than in step 1.

Considering **ad2** and **ad3** who attempt to deny services at phase 3 by computing the puzzle solution, the result shows that **ad3** succeeds at a higher proportion than **ad2**. The reason is that **ad3** floods attack messages at higher speed than **ad2** who uses more effort in the generation of message two than **ad3**. Nonetheless, both of them can force the responder to engage in signature verification. (Although **ad4** floods a 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 forgery at the cheap phase; the puzzle verification.)

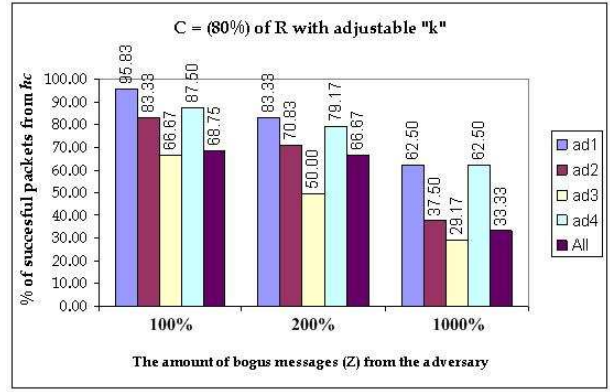
Without the ability to adjust puzzle difficulty, the percentage of successful messages in the system with **hc** and **ad4** is lower than the others because **ad4** floods message three at a higher rate than the other types. As a result, the most effective adversary to deny service to the responder would be **ad4** that attacks the verification phase. Particularly, most key agreement protocols incorporate verification tasks that would be susceptible to resource exhaustion attacks.

Finally, the result of the combination attack, illustrated as **A11** in the graph of figure 5, 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 increment of bogus messages.

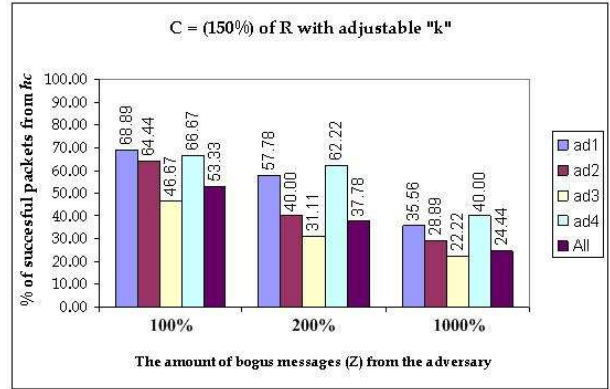
Having analysed the system with non-adjustable client puzzle, we would like to compare such results to the system integrating with adjustable client puzzle for investigating the usefulness of puzzle difficulty under equivalent magnitude of resource exhaustion attacks. In order to adjust the puzzle difficulty, we allocate two possible values for the responder to determine. Under normal circumstances, the responder selects $k=1$, which means an 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 to a next level that we set to $k=10$. This parameter would help the responder to slow down the incoming rate by requiring work of the initiator to solve puzzles at the factor of 2^{10} because this puzzle technique is Aura’s hash-based puzzle (Aura, Nikander & Leiwo 2000).

From the experimental result in figure 5, the number of attacking machines that the responder can tolerate is increased, i.e. honest clients have succeeded to establish a connection at higher proportion compared to the result of an experiment which includes attack techniques from four types of adversary.

Another interesting result is that the successful rate of message from **hc** in the case of **ad4**’s strategy is higher than the non-adjustable technique. The reason is that **ad4** does not compute the puzzle solution, so, no matter what the puzzle difficulty is, **ad4** can flood bogus messages at the equivalent speed as the simulation of a non-adjustable puzzle. However, at that amount of bogus messages, there are only messages from **ad4** (no legitimate traffic because honest clients have to spend some amount of time to solve the puzzle solution), or 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 have arrived. Undoubtedly, these bogus messages from **ad4** will be rejected at the first step of verification which requires only a short period and removes such attack from the connection queue. However, this situation does not occur in the case of **ad3** because **ad3** has to spend time to solve the puzzle as well as **hc**.



(a) hc's load = 80% of R



(b) hc's load = 150% of R

Figure 5: Percentage of throughput from **hc** with k is chosen between 1 and 10

5 Discussion

Comparing the system when the responder is unable to adjust puzzle difficulty, the experimental result from Meadows’ cost-based framework identifies that **ad3** is the most effective strategy (Table 2 in the column $k=1$) because **ad3** spends tiny computational cost to force the responder to compute up to 4998 computational units. (Although **ad4** does not spend any computation, the responder requires only a cheap computation for detecting and discarding those bogus messages.) In contrast, the Timed CPNs simulation indicates that **ad4** is the most effective strategy shown in Figure 4.

When we allow adjustable puzzle difficulty (Figure 5), **ad3** becomes the most effective technique in the Timed CPNs simulation-based analysis, while **ad4** is identified as the most effective technique in Meadows’ cost-based framework. (Once the responder slightly increases puzzle difficulty, the different gap of computational cost between **ad3** and the responder becomes narrow, but remains the same between **ad4** and the responder. The cause of this result from **ad4** is already explained in Section 4.3.)

The reason why both simulations provide us a different result is because we allow the adversaries’ capability to constantly flood new bogus messages when they receive return messages at the second phase in the time-based analysis. As a result, **ad4** should be the most effective adversary who depletes the responder’s connection queue and blocks legitimate users from acquiring services faster than others. Comparing to **ad3**, even though those bogus messages from **ad4** will be removed from the connection queue quicker, a new stream of **ad4**’s bogus messages still keeps the responder busy to verify them.

Contrasting the experimental result of Meadows’

cost-based framework and the Timed CPNs simulation, the cost-based analysis considers only the cost to an individual adversary to perform a set of identified actions to a certain point of the protocol execution compared to the cost of protocol engagement by the responder's machine. The simulation-based analysis is able to define more sophisticated techniques of the adversary as well as to involve a large amount of packets launched by multiple participants. Moreover, some hidden parameters, such as *time* as examined by Chan et al. (2006), can be included and evaluated in the analysis. As a result, lack of ability to model realistic events related with time factors and to handle large amount of messages is a limitation for analysing DoS-resistant protocols in Meadows' cost-based framework.

Finally, the most obvious benefit from the simulation-based analysis is that we can observe not only the behaviour of adversaries, but also the consequence of attacks to the system during the protocol execution. In addition, the analyst is able to more easily understand and evaluate the final outcome contrasting with cost-based analysis. In the cost-based evaluation, the analyst has to take extra care in order to consider and identify adversary capability as well as the effect of such attacks because sometimes the cost-based framework might generate ambiguous output; for example, considering the result of `ad3` comparing with `ad4` under cost-based analysis from Table 2, we conclude that the most effective strategy is performed by `ad3` because the total cost difference between the responder and such adversaries is the most. However, some may argue that `ad4` should be the most destructive scenario because the ratio of computational effort between the responder and such adversaries is infinity no matter what the chosen value of puzzle difficulty is.

6 Conclusion

This work has achieved the aims of extending the Meadows' cost-based framework to provide more accurate representation of computational cost and shown the potential of automated analysis. Moreover, we have explored unbalanced computational vulnerabilities of HIP by using a simulation approach provided in CPN Tools. By comparing experimental results from Meadows' cost-based and simulation-based analysis, we have found a limitation of Meadows' framework to define the ability of advanced adversaries and address DoS threats.

Because the nature of DoS attacks is quite subtle, the protocol developer should take extra care in the design and evaluation phase. Simulation is one promising technique that provides state exploration for searching all possible vulnerabilities in DoS-resistant protocols. In future work, we plan to extend this research by using the model checking capabilities of CPN tools to automatically verify the system by traversing the model and checking whether the cost tolerance between initiator and responder exceeds some reasonable threshold. Moreover, the power of adversaries can be extended in different ways in order to model more powerful attacks. For example, the advanced adversary, who attempts to attack the protocol at the third message, can be extended to flood reused packets from previous connections, eavesdrop messages from a valid communication, or craft bogus messages using existing messages including valid or invalid puzzle solutions as well as digital signatures. When inserting such advanced abilities into the model, we also require a technique to measure and identify cost of those operations in order to achieve a formal analysis.

References

- Aiello, W., Bellovin, S. M., Blaze, M., Ioannidis, J., Reingold, O., Canetti, R. & Keromytis, A. D. (2002), Efficient, DoS-resistant, secure key exchange for internet protocols, *in* 'The 9th ACM Conference on Computer and Communications Security', ACM Press, Washington, DC, USA, pp. 48–58.
- Al-azzoni, I. (2004), The Verification of Cryptographic Protocols using Coloured Petri Nets, Master of Applied Sciences Thesis, Department of Software Engineering, McMaster University, Ontario, Canada.
- Aura, T., Nagarajan, A. & Gurtov, A. (2005), Analysis of the HIP Base Exchange Protocol, *in* 'the 10th Australasian Conference on Information Security and Privacy (ACISP 2005)', Vol. 3574 of *Lecture Notes in Computer Science*, Springer-Verlag, Brisbane, Australia, pp. 481 – 493.
- Aura, T. & Nikander, P. (1997), Stateless Connections, *in* 'International Conference on Information and Communications Security', Springer-Verlag, Beijing, China, pp. 87–97.
- Aura, T., Nikander, P. & Leiwo, J. (2000), DoS-resistant authentication with client puzzles, *in* 'Security Protocols Workshop 2000', Cambridge, pp. 170–181.
- CERT (1996a), Denial-of-Service Attack via ping. [Online]. Available: <http://www.cert.org/advisories/CA-1996-26.html>.
- CERT (1996b), TCP SYN Flooding and IP Spoofing Attacks. [Online]. Available: <http://www.cert.org/advisories/CA-1996-21.html>.
- Chan, M. C., Chang, E., Lu, L. & Ngiam, P. S. (2006), Effect of Malicious Synchronization, *in* '4th International Conference on Applied Cryptography and Network Security (ACNS'06)', Singapore, pp. 114–129.
- Dai, W. (2004), Crypto++ 5.2.1 Benchmarks. [Online]. Available: <http://www.eskimo.com/~weidai/benchmarks.html>.
- Doyle, E. M. (1996), 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.
- Gupta, V., Gupta, S., Chang, S. & Stebila, D. (2002), Performance Analysis of Elliptic Curve Cryptography for SSL, *in* 'WISE'02: Proceedings of the 3rd ACM Workshop on Wireless Security', ACM Press, Atlanta, GA, USA, pp. 87–94.
- Han, Y. (1996), 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.
- Jakobsson, M. & Juels, A. (1999), Proofs of work and bread pudding protocols, *in* 'the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS 99)'.
- Jensen, K. (1997), *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Vol. 2nd edition, Vol. 1-3, Springer-Verlag.

Jensen, K. (1998a), A brief introduction to Colored Petri Nets, in ‘Workshop on the Applicability of Formal Models’, Aarhus, Denmark, pp. 55–58.

Jensen, K. (1998b), An introduction to the Theoretical Aspects of Colored Petri Nets, in ‘Workshop on the Applicability of Formal Models’, Aarhus, Denmark.

Juels, A. & Brainard, J. (1999), Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks, in ‘the 1999 Network and Distributed System Security Symposium (NDSS ’99)’, Internet Society Press, Reston, San Diego, California, USA, pp. 151–165.

Meadows, C. (2001), ‘A Cost-Based Framework for Analysis of DoS in Networks’, *Journal of Computer Security* **9**(1/2), 143–164.

Meadows, C. (2003), ‘Formal methods for cryptographic protocol analysis: emerging issues and trends’, *IEEE Journal on Selected Areas in Communications* **21**(1), 44–54.

Moskowitz, R. (2006), ‘The Host Identity Protocol (HIP)’, Internet Draft, Internet Engineering Task Force. <http://www.ietf.org/internet-drafts/draft-ietf-hip-base-06.txt>.

Neih, B. B. & Tavares, S. E. (1993), Modelling and Analysis of Cryptographic Protocols using Petri Nets, in ‘Advances in Cryptology’, Berlin, Germany, pp. 275–295.

Petri, C. A. (1962), Kommunikation mit Automaten, PhD Thesis, Institut für Instrumentelle Mathematik, Schriften des IIM.

Ramachandran, V. (2002), Analyzing DoS-Resistance of Protocols Using a Cost-Based Framework, Technical Report YALEU/DCS/TR-1239, Department of Computer Science, Yale University.

Smith, J., Nieto, J. M. G. & Boyd, C. (2006), Modelling Denial of Service Attacks on JFK with Meadows’s Cost-Based Framework, in ‘Fourth Australasian Information Security Workshop (AISW-NetSec 2006)’, Vol. 54, CRPIT series, pp. 125–134.

Tan, Z., Lin, C., Lin, H. & Li, B. (2004), ‘Optimization and Benchmark of Cryptographic Algorithms on Network Processors’, *IEEE Micro* **24**(5), 55–69.

The Department of Computer Science, University of Aarhus, Denmark (2004), CPN Tools: Computer Tool for Coloured Petri Nets. [Online]. Available: <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>.

Appendix

A Cost-Based Definition

To analyse the protocol specification, we begin with the notation introduced by Meadows (2001).

Definition 1: *The sequence of messages sent from the principal A to the principal B in the protocol is written in the form*

$$A \rightarrow B : T_1, \dots, T_k \parallel M \parallel O_1, \dots, O_n$$

- T_i are the operations performed by the principal A for preparing the message M.

- M is the sent message from the principal A and subsequently received by the intended recipient B.

- O_j are the operations performed by the principal B for processing the message M.

Definition 2: *An event in the operation $A \rightarrow B : T_1, \dots, T_k \parallel M \parallel O_1, \dots, O_n$ is one of:*

- an operation in T_i or an operation O_j .
- A sends M to B, or B receives M from A.

There are three types of events:

1. **Normal events** appear at both principals and always succeed and are followed by the next events.
2. **Verification events** appear only at the recipient side and can evaluate to either success or failure.
3. **Accept events** appear at the end of the operations O_j indicating the completion of the process.

Definition 3: *A cost set C is a set with operation + with partial order \geq such that*

$$x + y \geq x \text{ and } x + y \geq y \quad \text{for all } x, y \in C.$$

Definition 4: *Event cost function (δ) and protocol engagement cost function (Δ)*

- δ is an event cost function iff it transfers sets of operations in the protocol into the cost set C which consists of four values: expensive > medium > cheap > 0.

- δ' is a message processing cost function related to δ defined on verification events $\{V_i\} \subset \{O_j\}$ such that

$$\text{for } A \rightarrow B : T_1, \dots, T_k \parallel M \parallel O_1, \dots, O_n \\ \text{if } \{V_i\} = \{O_j\}, \text{ then} \\ \delta'(V_i) = \delta(O_1) + \dots + \delta(O_j)$$

- $\Delta(O_n)$ is the sum of all operational costs appearing at the responder B up to the accept event O_n , plus operational costs of processing the response message related to such accept event O_n .

Definition 5: *Attacker event cost function ϕ , attacker cost function Φ , and attack cost function Θ are defined as follows.*

- $\Phi(\{x_1, \dots, x_n\}) = \phi(x_1) + \dots + \phi(x_n)$ where x_i are the set of available attacker’s capabilities and ϕ is a function for interpreting attacker’s capabilities to cost set.

- Θ is a function from the set of events defined by the protocol to a cost set.

Definition 6: *Defender cost set C, attacker cost sets G, tolerance relation τ are defined as follows.*

- $\tau \subset C \times G$; consists of all pairs (c, g) such that the protocol designer is willing to tolerate the responder expending resources up to cost c, as long as the attacker has to extend resources of cost g or greater. A tuple (c', g') is said to be within the tolerance relation if there exists (c, g) $\in \tau$, such that $c' \leq c$ and $g' \geq g$.

Once the actions of each protocol principal are classified into the computational costs cheap, medium, or expensive, all actions of the protocol run can be compared between the initiator and the responder. The protocol is secure against DoS attacks, if the final cost is great enough from the point of view of the attacker in comparison with the cost of engaging in the events up to an accepted action from the point of view of the defender. Otherwise, we conclude that the protocol is insecure against DoS attacks.