A Game Based Analysis of the Client Puzzle Approach to Defend Against DoS Attacks

Boldizsár Bencsáth István Vajda Levente Buttyán
Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics, Hungary
{bencsath, vajda, buttyan}@crysys.hu

Abstract—**DoS attacks are aimed at the loss of or the reduction in availability, which is one of the most important general security requirements in computer networks. A promising approach proposed to alleviate the problem of DoS attacks is to use client puzzles. In this paper, we study this approach using the apparatus of game theory. In our analysis, we derive the optimal strategy for the attacked server (e.g., a web server on the Internet) in all conceivable cases. We also present two new client puzzles as examples.**

I. Introduction

Besides confidentiality and integrity, *availability* is one of the most important general security requirements in computer networks. Availability of a system means that it is accessible and usable upon demand by an authorized entity, according to performance specifications for the system [21]. In other words, a system is available if it provides services according to the system design whenever users request them.

If only accidental failures are considered, then replication methods can be used to ensure availability. Replication in itself, however, is not enough against malicious attacks that are specifically aimed at the loss of or reduction in availability. Such attacks are commonly called Denial of Service (DoS) attacks.

Roughly speaking, two types of DoS attacks against an online server can be distinguished: bandwidth consumption attacks and resource consumption attacks. In a bandwidth consumption attack, the attacker floods the server with requests so that the server becomes overloaded and cannot accept requests from legitimate clients anymore. In a resource consumption attack, the attacker sends some requests to the server such that the server uses up all of its resources to process the requests and that is why it can no longer accept requests from legitimate clients. Of course, flooding the server may also lead to the exhaustion of all server resources (e.g., two many processes are launched to serve the requests), but depending on the type of the service provided by the server, a few, carefully constructed requests may have the same effect too. In this paper, we are concerned with resource consumption attacks.

Interestingly enough, simply authenticating the clients may turn out to be not so useful against a resource consumption attack. This is because the authentication procedure may involve expensive operations (in terms of resource consumption) at the server side, and therefore, engaging into multiple parallel instances of the authentication protocol itself may consume all server resources.

An approach that alleviates the problem is to use *client puzzles* [9]. The idea is that before engaging into any resource consuming operations, the server sends a puzzle to the client, who has to solve the puzzle and send the result back to the server. The server continues with processing the client's request (which may involve resource consuming operations) only if it received a correct response to the puzzle. This approach does not make resource consumption attacks impossible, but it makes them more expensive for the attacker in the sense that successful attacks require considerably more resources from the attacker. Moreover, by varying the complexity of the puzzle, the cost of an attack can be varied too; this allows one to adjust the system according to the assumptions he has about the strength of the attacker.

In this paper, we analyze the client puzzle approach using game theory. We model the situation faced by the DoS attacker and the server as a two-player strategic game. In this game, the server's strategy is characterized by the complexity of the puzzles it generates, whereas the DoS attacker's strategy is characterized by the amount of effort he invests in solving the received puzzles. Our analysis of the game gives useful insights into the client puzzle approach; we show, for instance, that under certain conditions (which mainly depend on the computing cost of the different steps of the protocol) the optimal strategy for the server is a mixed strategy where it generates puzzles with various complexity levels according to some probability distribution. Our main result is the derivation of the optimal strategy for the server in all conceivable cases. To the best of our knowledge such a game based analysis of the client puzzle approach has not been published yet. This work is a follow up of the work presented in [3].

The organization of the paper is the following. In Section 2, we briefly overview some related work. In Section 3, we introduce our game model, and in Section 4, we analyze this model and determine the solution of the game. In Section 5, we propose a specific client puzzle. Finally, in Section 6, we conclude the paper and sketch some future research directions.

II. Related work

The latest DoS attacks on the Internet (Amazon, Ebay (2000), DNS root servers (2002)) are analyzed in [10]. Several methods have been proposed to alleviate the problem of DoS attacks in general (see e.g., [19], [7]), and to make cryptographic protocols resistant against DoS attacks in particular (see e.g., [13], [11], [6], [1]).

The concept of cryptographic puzzles were originally introduced by Merkle [15]. Later, Dwork and Naor used them to combat against junk mails [5]. Juels and Brainard introduced the idea of client puzzles to prevent TCP SYN flooding

[9], whereas Dean and Stubblefield applied the approach to the TLS protocol [4]. Time-lock puzzles that cannot be solved until a pre-determined amount of time has passed were introduced by Rivest, Shamir, and Wagner in [20].

A formal framework for the analysis of DoS attacks has been proposed by Meadows [14], but her approach does not based on game theory. Game theory has already been used to study network security problems [12], [8], but those papers do not address the problem of DoS attacks directly. Recently, Michiardi and Molva [16] used game theory to reason about a mechanism that would prevent a very special form of DoS attack (packet dropping) specific to wireless ad hoc networks. Our work is different as we focus on how to protect an on-line server from DoS attacks on the Internet.

III. Strategic game model

In this section, we construct a game based model of the client puzzle approach. For this, we first describe the client puzzle approach in an abstract way, and then we define a strategic game that is intended to model the situation faced by the DoS attacker and the attacked server when the client puzzle approach is used.

A. An abstract view of the client puzzle approach

Using the client puzzle approach means that before engaging in any resource consuming operations, the server first generates a puzzle and sends its description to the client that is requesting service from the server. The client has to solve the puzzle and send the result back to the server. The server continues with processing the request of the client, only if the client's response to the puzzle is correct. This is summarized in the following abstract protocol, where C and S denote the client and the server, respectively:

step 1	$C \rightarrow S$:	sending service request
step 2	S :	generation of a puzzle
step 3	$S \rightarrow C$:	sending description of the puzzle
step 4	C :	solving the puzzle
step 5	$C \rightarrow S$:	sending solution to the puzzle
step 6	S :	verification of the solution
if the solution is correct:		
step 7	S :	continue processing service request

One can view the first six steps of the protocol as a preamble preceding the provision of the service, which is subsumed in a single step (step 7) in the above abstract description. The preamble provides a sort of algorithmic protection against DoS attacks. The server can set the complexity level of the puzzle according to the estimated strength (computational resources) of the attacker. If the server manages to set an appropriate complexity level, then solving the puzzle slows down the DoS attacker who will eventually abandon his activity.

B. Defining the game: players, strategies, and payoffs

Given the above abstract protocol, the DoS attacker and the server are facing a situation in which both have several strategies to choose from. The strategies that they decide upon determine the outcome of their interaction, which is characterized by the amount of resources used by the server. The goal of the attacker is to maximize this amount, whereas the server's goal is to minimize its own resource consumption (in order to remain available for genuine clients). This situation can naturally be modelled using the apparatus of game theory [18].

Accordingly, we define a strategic game, where the players are the DoS attacker and the victim server. The attacker has three strategies:

- A_1: The attacker executes only step 1 and then quits the protocol. The attacker may choose this strategy when he does not want to waste resources to solve the puzzle or is not able to maintain two-way communication with the server (e.g., he is using a spoofed IP address).
- A_2: The attacker lets the protocol run until step 3, and then, instead of solving the puzzle, it generates some random sequence of bits and sends it to the server as the solution to the puzzle. By using this strategy, the attacker coerces the server to carry out the verification step (step 6) of the protocol without actually solving the puzzle. The probability that the garbage sent by the attacker is a correct solution to the puzzle is negligible, and therefore, the server will not process the service request any further.
- A_3: The attacker solves the puzzle and sends the solution to the server. Actually, this strategy corresponds to the correct behavior of a genuine client. As a result of following this strategy, the attacker coerces the server to process the service request, and thus, to consume considerable amount of resources. In fact, the role of the protocol preamble (steps 1 through 6) is to discourage the attacker from frequently selecting this strategy during the attack period.

The server has several strategies too, each of which corresponds to a particular complexity level of the puzzle. The selection of the complexity level can be based on attack indicators, which are continuously maintained by the server [2]. It is important to carefully select the complexity level of the puzzle: if the complexity level is too low, then the puzzle will be ineffective in deterring the attacker from mounting the attack; on the other hand, the complexity level should not be too high either, because that would result in an unnecessarily complex verification step at the server. In order to make the presentation easier, in the rest of the paper, we assume that the server can choose between two complexity levels only, which we call *low* and *high*. The corresponding strategies are denoted by S_1 and S_2, respectively. We note, however, that our results can easily be generalized for the case when the server can choose among more than two complexity levels.

We denote by $G(A_j, S_k)$ the outcome of the game when the attacker and the server choose strategy A_j ($j \in \{1, 2, 3\}$) and strategy S_k ($k \in \{1, 2\}$), respectively. In order to define the payoffs of the players in each of the possible outcomes of the game, we need to introduce some notation for the cost (in terms of resource consumption) of the various steps of the abstract protocol. This notation is summarized in Tables I and II, where

Step	Cost
service request (step 1)	ϱ
sending garbage in step 5	γ
solving the puzzle (steps 4 and 5)	$\sigma(k)$ ($k \in \{1,2\}$)
cost during service provision	ε

TABLE I
RESOURCE CONSUMPTION COSTS OF THE ATTACKER

Step	Cost
generating puzzle (steps 2 and 3)	$\pi(k)$ ($k \in \{1,2\}$)
verification of solution (step 6)	$\nu(k)$ ($k \in \{1,2\}$)
service provision	α

TABLE II
RESOURCE CONSUMPTION COSTS OF THE SERVER

k stands for the index of the strategy used by the server. Thus, $\sigma(1)$, for instance, denotes the resource consumption cost of solving a low complexity puzzle, and $\sigma(2)$ denotes the cost of solving a high complexity puzzle. Naturally, we assume that $\pi(1) \leq \pi(2)$, $\sigma(1) < \sigma(2)$, and $\nu(1) \leq \nu(2)$. Furthermore, $\alpha \gg \varepsilon$.

Using the notation given in Tables I and II, we can obtain the cost c_j of attacker strategy A_j for every $j \in \{1,2,3\}$ as follows:

$$c_1 = \varrho \quad (1)$$

$$c_2 = \varrho + \gamma \quad (2)$$

$$c_3(k) = \varrho + \sigma(k) + \varepsilon \quad (3)$$

In fact, c_j is the cost of strategy A_j when only one instance of the protocol is run. However, the attacker is assumed to have enough resources to run several instances of the same protocol with the server. More precisely, we assign a positive real number R to the attacker that represents the amount of his resources available for the attack. Then, we compute the number of protocol instances the attacker could run with the server simultaneously as R/c_j, assuming that the attacker follows strategy A_j in every instance.

Now, we are ready to determine the payoffs for the players in every possible outcome of the game. The payoff for the attacker in outcome $G(A_j, S_k)$ is equal to the resource consumption cost of the server when the attacker and the server follow strategies A_j and S_k, respectively, and the attacker uses all of his available resources R for the attack (i.e., he runs R/c_j instances of the protocol with the server). The payoff for the server in the same outcome is the (additive) inverse of the attacker's payoff. In other words, the game is a *zero sum* game. For this reason, it is enough to specify only the payoffs for the attacker in the matrix of the game, which is given in Table III. The matrix of the game is denoted by $\mathbf{M}$, and the element in the j-th row and k-th column of $\mathbf{M}$ is denoted by M_{jk}.

IV. SOLUTION OF THE GAME

In order to be general, we allow each player to select the strategy to be used according to a probability distribution over the set of strategies available for the player. This probability distribution is called *mixed strategy*, because it determines how the player mixes its pure strategies. We denote the mixed strategy of the server by $X = (x_1, x_2)$ and the mixed strategy of the attacker by $Y = (y_1, y_2, y_3)$. This means that the server plays according to strategy S_1 with probability x_1 and it follows strategy S_2 with probability x_2. Similarly, the attacker uses strategies A_1, A_2, and A_3 with probability y_1, y_2, and y_3, respectively.

Roughly speaking, the goal of the players is to maximize their payoffs. However, a more careful look at the problem reveals that there are several plausible ways to interpret this general goal (i.e., several goal functions could be considered). We assume that the server is cautious and it wants to minimize the maximum of its average resource consumption cost, where the maximum is taken over the attacker strategies. In other words, the server follows a *minimax* rule. Accordingly, the attacker wants to maximize the minimum of the server's average resource consumption cost.

Let us assume for a moment that the server uses the mixed strategy $X = (x_1, x_2)$ and the attacker follows his pure strategy A_j ($j \in \{1,2,3\}$) (i.e., he uses a mixed strategy where A_j is selected with probability 1). In this case, the average cost of the server is

$$M_{j1} \cdot x_1 + M_{j2} \cdot x_2$$

Similarly, if the attacker uses his mixed strategy $Y = (y_1, y_2, y_3)$ and the server applies its pure strategy S_k ($k \in \{1,2\}$), then the average cost of the server is

$$M_{1k} \cdot y_1 + M_{2k} \cdot y_2 + M_{3k} \cdot y_3$$

We can obtain the solution of the game by solving the following two systems of inequalities:

For the server:

$$M_{11} \cdot x_1 + M_{12} \cdot x_2 \leq v \quad (4)$$

$$M_{21} \cdot x_1 + M_{22} \cdot x_2 \leq v \quad (5)$$

$$M_{31} \cdot x_1 + M_{32} \cdot x_2 \leq v \quad (6)$$

$$x_1,\ x_2 \geq 0 \quad (7)$$

$$x_1 + x_2 = 1 \quad (8)$$

For the attacker:

$$M_{11} \cdot y_1 + M_{21} \cdot y_2 + M_{31} \cdot y_3 \geq v \quad (9)$$

$$M_{12} \cdot y_1 + M_{22} \cdot y_2 + M_{32} \cdot y_3 \geq v \quad (10)$$

$$y_1,\ y_2,\ y_3 \geq 0 \quad (11)$$

$$y_1 + y_2 + y_3 = 1 \quad (12)$$

A graphical representation of the system (4) – (8) is depicted in Figure 1, where the dashed lines correspond to the first three inequalities of the system. According to the minimax rule, the server selects the minimum average cost, which is the lowest point of the shaded region. The optimization step for the attacker is similar.

According to Neumann's classical theorem [17], there always exists a common optimum for the attacker and the server. The corresponding set of parameters $x_1^*, x_2^*, y_1^*, y_2^*, y_3^*, v^*$ are called the solution of the game.

	S_1	S_2
A_1	$M_{11} = \pi(1) \cdot \frac{R}{c_1}$	$M_{12} = \pi(2) \cdot \frac{R}{c_1}$
A_2	$M_{21} = (\pi(1) + \nu(1)) \cdot \frac{R}{c_2}$	$M_{22} = (\pi(2) + \nu(2)) \cdot \frac{R}{c_2}$
A_3	$M_{31} = (\pi(1) + \nu(1) + \alpha) \cdot \frac{R}{c_3(1)}$	$M_{32} = (\pi(2) + \nu(2) + \alpha) \cdot \frac{R}{c_3(2)}$

TABLE III

GAME MATRIX $\mathbf{M}$ (CONTAINS ONLY THE PAYOFFS OF THE ATTACKER)

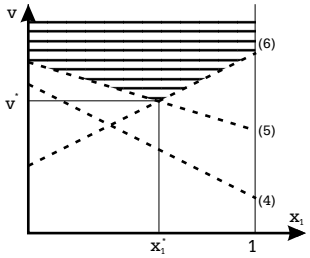

Fig. 1. Searching the optimum strategy of the server

Note that if $M_{21} > M_{11}$ and $M_{22} > M_{12}$, then the second row of the matrix of the game dominates the first one. This dominance exists if

$$\frac{\nu(k)}{\gamma} < \frac{\pi(k)}{\varrho} \tag{13}$$

The cost ϱ of sending the service request and the cost γ of sending garbage as a result to the puzzle can be considered as the cost of a communication step (i.e., sending a packet). Therefore, $\gamma = \varrho$ seems to be a reasonable assumption. From here, the mentioned dominance follows if the cost $\nu(k)$ of verification of the solution to the puzzle is smaller than the cost $\pi(k)$ of generating the puzzle. The existence of a dominated strategy simplifies the problem of searching for the solution of the game, because the dominated strategy can be ignored.

Now let us assume that $\nu(k) < \pi(k)$, and hence A_2 dominates A_1. This is illustrated in Figure 1 by the line corresponding to inequality (4) being entirely below the line corresponding to inequality (5). This means that only the latter needs to be considered when looking for the optimum. The lines corresponding to inequalities (5) and (6) intersect at $x_1 = 1/(1+w)$, where

$$w = \frac{M_{21} - M_{31}}{M_{32} - M_{22}} \tag{14}$$

Therefore, we get the following result:

Proposition 1: If $w > 0$, then the server's optimal strategy is a *mixed strategy* with probability distribution

$$X^* = \left(\frac{1}{1+w}, \frac{w}{1+w} \right) \tag{15}$$

If $w \leq 0$ then the server's optimal strategy is a *pure strategy* according to the following rules:

- if $-1 < w \leq 0$, then the server's optimal strategy is S_1,
- if $w \leq -1$, then the server's optimal strategy is S_2. □

The different cases listed in Proposition 1 are illustrated in Figure 2.

As an example, let us consider the following numerical values for the parameters: $\varrho = 1$, $\pi(1) = 14$, $\pi(2) = 16$, $\gamma = 1$, $\sigma(1) = 40$, $\sigma(2) = 400$, $\nu(1) = 10$, $\nu(2) = 12$, $\varepsilon = 10$, $\alpha = 3000$, and $R = 100000$. These values lead to a game the

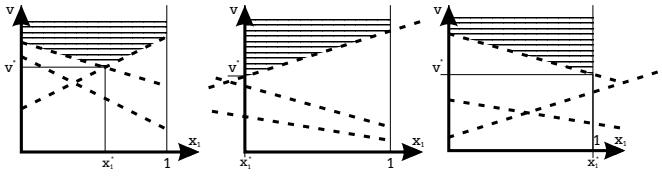

Fig. 2. Illustration of the different cases in Proposition 1

solution of which is the following: $v^* = 15679.8$, $x_1^* = 0.16$, $x_2^* = 0.84$, $y_1^* = 0.96$, $y_2^* = 0$, $y_3^* = 0.04$. Thus, the optimal strategies are mixed strategies.

If the server can choose from more than two strategies, then a more complex system of inequalities have to be solved in order to determine the solution of the game. Fortunately, the solution can always be obtained by using linear programming tools.

V. EXAMPLES FOR PUZZLES

In this section, we present two methods to generate puzzles for illustration purposes. An important characteristic of a client puzzle is that the amount of computation needed to resolve it can be estimated fairly well. Note that the puzzles used in the defense against DoS attacks need not require inherently sequential operations like the puzzles in [20], where it is important that an encrypted message cannot be decrypted by anyone until a pre-determined amount of time has passed. In case of DoS attacks, the amount of resources needed to solve the puzzle is more important than the time. Since, in general, parallelization does not reduce the amount of resources needed to perform a computation, the fact that a puzzle is parallelizable is not an issue here.

Let $T = \{p_1, p_2, \ldots, p_N\}$ be a publicly known, ordered set of different n bit prime numbers. A set S of k primes is selected randomly from T. The selection can be made with or without replacement. Below, we assume that it is done without replacement (i.e., S is a subset of T). The analysis can directly be extended to the case of selection with replacement. The elements of S are multiplied together, and the product is denoted by m:

$$m = p_{i_1} \cdot p_{i_2} \cdot \ldots \cdot p_{i_k}$$

Consider the following two puzzles:

- *Puzzle 1:* The puzzle is the product m.
- *Puzzle 2:* Let m' be a modification of m, such that ℓ consecutive bits, $m_r, m_{r+1}, \ldots, m_{r+\ell-1}$ are replaced with zeros in the binary representation of m. The puzzle is the resulting number m' together with position r.

For both puzzles, the task of the client is to find the prime factors of m (or m' in case of Puzzle 2), and respond with their corresponding indices in T (recall that T is ordered).

k	4	8	16	32	64
$D(100,k)$	80.8	89.8	95.1	97.9	99.4

TABLE IV
VALUES OF $D(N,k)$, $N = 100$

In case of Puzzle 1, the client calculates its response in the following natural way: Let μ be a variable that is updated in each step of the computation. Initially, $\mu = m$. In step i, the client checks if p_i is a factor of μ (and hence of m). If so, then μ is updated to take the value of $\frac{\mu}{p_i}$; otherwise μ does not change. This procedure is repeated until all factors of m is found (i.e., μ becomes 1).

During the above computation, at least $k-1$ divisions are made[1], where the divisors are n bit size primes, and the size of the dividend decreases gradually from kn to $2n$. The average number of divisions is given by the following formula:

$$D(N,k) = \sum_{i=k}^{N} q_i \cdot (i-1) \tag{16}$$

where q_i is the probability that the largest index in S is i, and it is computed as:

$$q_i = \frac{\left(\begin{array}{c} i-1 \\ k-1 \end{array} \right)}{\left(\begin{array}{c} N \\ k \end{array} \right)} \tag{17}$$

When k increases, $D(N,k)$ monotonically and quickly increases to N, and $D(N,k)$ is close to N even for relatively small values of k. In Table IV, numerical values of $D(N,k)$ are given for the instance of $N = 100$.

In case of Puzzle 2, the client is forced to do more calculations. The client may choose from the following two procedures:

- The client tries possible substitutions for the missing bits. If a substitution is incorrect, then the client will likely get prime factors that do not belong to set T. In this case, the client continues with choosing another substitution. The average number of divisions required is approximately $N2^{\ell-1}$, thus the complexity of solving the puzzle is increased with a factor of $2^{\ell-1}$ on average. For instance, in case of parameters $N = 100$ and $\ell = 6$, the average number of divisions required is at least 3200.
- The client directly calculates different products of k primes from set T until the tested product m' is obtained. The average number of multiplications required is

$$\frac{1}{2} \left(\begin{array}{c} N \\ k \end{array} \right) (k-1)$$

For instance, if $N = 100$ and $k = 8$, then approximately $6.51 \cdot 10^{11}$ multiplications are needed.

VI. CONCLUSION AND FUTURE WORK

In this paper, we studied the client puzzle approach to defend against resource consumption DoS attacks. We modelled the situation faced by the DoS attacker and the attacked server as a two-player strategic game. We analyzed this game and gave useful insights into the client puzzle approach. Our main contribution is the derivation of the optimal strategy for the server in all conceivable cases. To the best of our knowledge such a game based analysis of the client puzzle approach has not been published yet. We also presented and analyzed two new client puzzles with illustration purposes.

As future work, we intend to extend the Internet Key Exchange (IKE) protocol with the puzzles proposed in this paper, and to study how the complexity of the puzzles can be fine tuned in practice by experimenting with real systems.

[1] Note that the last prime factor need not be checked by division.

REFERENCES

[1] T. Aura and P. Nikander. Stateless protocols. In Proceedings of the ICICS'97 Conference, Springer-Verlag, LNCS volume 1334, 1997.

[2] B. Bencsath and I. Vajda. Protection against DoS attacks based on traffi c level measurments. Submitted for publication, April 2003.

[3] B. Bencsath and I. Vajda. A game theoretical approach to optimizing protection against DoS attacks. Rump session of the 2nd Central European Conference on Cryptography (HajduCrypt), no proceedings, July 2002.

[4] D. Dean and A. Stubblefi eld. Using client puzzles to protect TLS. Proceedings of the USENIX Security Symposium, August 2001.

[5] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology – Crypto '92*, Springer-Verlag, LNCS volume 740, pp. 139–147, August 1992.

[6] P. Eronen. Denial of service in public key protocols. In Proceedings of the Helsinki University of Technology Seminar on Network Security (Fall 2000), December 2000.

[7] T. M. Gil. MULTOPS: A data structure for denial-of-service attack detection. Technical Report, Vrije Universiteit, 2000.

[8] J.P. Hespanha and S. Bohacek. Preliminary results in routing games. In Proceedings of the American Control Conference, volume 3, pp. 1904–1909, 2001.

[9] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Proceedings of the IEEE Network and Distributed System Security Symposium (NDSS '99), pp. 151–165, February 1999.

[10] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajovic. Distributed denial of service attacks. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 2275–2280, October 2000.

[11] J. Leiwo, T. Aura, and P. Nikander. Towards network denial of service resistant protocols. In Proceedins of the IFIP SEC 2000 Conference, August 2000.

[12] K. Lye and J. M. Wing. Game strategies in network security. In Proceedings of the Workshop on Foundations of Computer Security, Copenhagen, Denmark, 2002.

[13] K. Matsuura and H. Imai. Protection of authenticated key-agreement protocol against a denial-of-service attack. In Proceedings of the International Symposium on Information Theory and Its Applications (ISITA'98), pp. 466–470, October 1998.

[14] C. Meadows. A formal framework and evaluation method for network denial of service. In Proceedings of the IEEE Computer Security Foundations Workshop, June 1999.

[15] R. C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21:294–299, April 1978.

[16] P. Michiardi and R. Molva. Core: A COllaborative REputation mechanism to enforce node cooperation in mobile ad hoc networks. In Proceedings of the 6th IFIP Communications and Multimedia Security Conference, September 2002.

[17] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. 1924.

[18] M. Osborne and A. Rubenstein. *A Course on Game Theory*. MIT Press, 1994.

[19] K. Park and H. Lee. On the effectiveness of probabilistic packet marking for IP Traceback under denial of service attack. Technical Report No. CSD-00-013, Department of Computer Sciences, Purdue University, June 2000.

[20] R. L. Rivest and A. Shamir and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report No. MIT/LCS/TR-684, 1996.

[21] R. Shirey. Internet security glossary. Internet RFC 2828, May 2000.