Game Theoretic Resistance to Denial of Service Attacks Using Hidden Difficulty Puzzles

Harikrishna Narasimhan[1,⋆], Venkatanathan Varadarajan[1,⋆], and C. Pandu Rangan[2,⋆⋆]

[1] Department of Computer Science and Engineering, College of Engineering, Guindy, Anna University, Chennai, India
{nhari88,venk1989}@gmail.com
[2] Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India
prangan@iitm.ac.in

Abstract. Denial of Service (DoS) vulnerabilities are one of the major concerns in today's internet. Client-puzzles offer a good mechanism to defend servers against DoS attacks. In this paper, we introduce the notion of hidden puzzle difficulty, where the attacker cannot determine the difficulty of the puzzle without expending a minimal amount of computational resource. We propose three concrete puzzles that satisfy this requirement. Using game theory, we show that a defense mechanism is more effective when it uses a hidden difficulty puzzle. Based on the concept of Nash equilibrium, we develop suitable defense mechanisms that are better than the existing ones.

Keywords: Denial of Service (DoS) Attack, Proof of Work, Hidden Difficulty Puzzle, Infinitely Repeated Game, Nash Equilibrium.

1 Introduction

Denial of Service (DoS) vulnerabilities are one of the major concerns in today's internet. The aim of a DoS attack is to make a network service unavailable to its legitimate users. A denial of service attack may either be a brute force attack, where the attacker generates spurious network traffic to exhaust server resources or a semantic attack, where the attacker exploits the vulnerabilities of the protocol used [15].

Proofs of work or client-puzzles offer a good mechanism for a server to counterbalance computational expenditure when subjected to a denial of service attack. On receiving a request, the server generates a puzzle of appropriate difficulty and sends it to the client. When a response is received, the server verifies the solution and provides the requested service only if the solution is correct. This

⋆ Work supported by IITM Summer Fellowship Programme May-July 2009.
⋆⋆ This author was partially supported by Indo-Australian Project on Protecting Critical Information Infrastructure from DoS attacks, CSE/08-09/102/DSTX/SVRA-IAP.

J. Kwak et al. (Eds.): ISPEC 2010, LNCS 6047, pp. 359–376, 2010.

approach was first proposed by Dwork and Naor [4] to control junk mails. Over the years, lot of research has gone into this area and different client-puzzles have been proposed [13,8,2,18,19,6,16,20,17].

A challenge in the client-puzzle approach is deciding on the difficulty of the puzzle to be sent. One approach suggested by Feng et al. [6] is to adjust the puzzle difficulty proportional to the current load on the server. Juels and Brainard [8] suggested that the difficulty of the puzzle be scaled uniformly for all clients according to the severity of the attack on the server. In both these approaches, the quality of service to legitimate users is not considered. Alternatively, the server can generate puzzles of varying difficulties based on a probability distribution. Such an approach based on game theory can be seen in [3,5].

Though there have been several works that formally analyze denial of service attacks using game theory [3,11,5,9,10,14,1], only a few of them analyze the client-puzzle approach [3,5,10]. Bencsath et al. [3] modeled the client-puzzle approach as a single-shot strategic game and identified the server's optimal strategy. Fallah [5], on the other hand, used an infinitely repeated game to come up with puzzle-based defense mechanisms. He also proposed extensions to tackle distributed attacks. Recently, Jun-Jie [10] applied game theory to puzzle auctions.

Game theoretic defense mechanisms against DoS attacks focus on *fine tuning the parameters* of the system in such a way that the server is not overloaded by the attacker. Our work builds on the game theoretic model and defense mechanisms proposed by Fallah.

Our Contribution. In addition to the basic properties of a good puzzle [15], we introduce the following requirement: *the difficulty of the puzzle should not be determined by the attacker without expending a minimal amount of computational effort.* We propose three concrete puzzles that satisfy this requirement. Using game theory, we show that defense mechanisms are more effective when the puzzle difficulty is hidden from the attacker.

The rest of the paper is organized as follows: Section 2 contains an example of a hidden difficulty puzzle (HDP). In Section 3, we show using game theory that a defense mechanism is more effective when it uses HDPs. In Section 4, we develop defense mechanisms based on Nash equilibrium. New puzzles are described in Section 5 and we conclude the paper in Section 6.

2 Hidden Difficulty Puzzle (HDP)

The difficulty of a client puzzle is said to be hidden if it cannot be determined by an attacker without expending a minimal amount of computational effort. We first introduce a modified version of the hash-reversal puzzle [8], which satisfies this requirement. The puzzle generation and verification are detailed in Fig. 1.

A preimage X is generated by hashing a server secret S, a server nonce N_s and a session parameter M together. The server nonce is used to check whether the puzzle is recent and the session parameter allows the server to be stateless [15]. The preimage is again hashed to obtain Y and some of the first k bits of X are

Client		Defender
	$\xrightarrow{Request}$	$X = H(S, N_s, M)$ $Y = H(X)$
	$\xleftarrow{(X', Y), N_s}$	$X' = X \oplus (I_1, I_2, ..., I_{k-1}, 1, 0_{k+1}, ..., 0_n)$
Find rp such that $H(rp) = Y$	$\xrightarrow{rp, N_s}$	$X = H(S, N_s, M)$ $H(rp) \stackrel{?}{=} H(X)$

Fig. 1. Hidden Difficulty Puzzle 1. Here, H is a cryptographic hash function and I is a binary number chosen uniformly at random.

randomly inverted. Let X' be the resultant binary string. The puzzle consisting of X' and Y is sent to the client along with the server nonce.

Note that k determines the difficulty of the puzzle and is unknown to the client. In order to solve the puzzle he would have to carry out an exhaustive search and arrive at the solution after testing up to 2^k possible preimages. The solution to the puzzle along with the received nonce is sent back to the defender. The defender recomputes the preimage X and verifies the solution.

Here, puzzle generation takes 2 hash computations, while the verification takes 3 hash computations. Further, the client needs to compute an average of $\frac{(2^k+1)}{2}$ hashes to solve the puzzle.

Assume the defender uses two instances of the described hidden difficulty puzzle, P_1 and P_2 with difficulty levels k_1 and k_2 respectively. On receiving a puzzle, the attacker does not know whether it is P_1 or P_2. Any solution to P_1 would have the k_1^{th} bit inverted, while any solution to P_2 will have the k_2^{th} bit inverted. Clearly, the solution spaces of the two puzzles do not overlap. To solve the puzzle, the attacker could first test possible preimages for one of the puzzles and if it is not solved, test preimages for the other. He could also try out preimages for both puzzles simultaneously. In any case, the attacker would know the puzzle difficulty only after putting in the effort required to solve one of the puzzles. Clearly, the attacker cannot determine the puzzle difficulty without minimal resource expenditure.

3 Game Theoretic Analysis of HDP

We shall now see how a hidden difficulty puzzle can make a defense mechanism more effective. We assume the network consists of a server, a set of legitimate clients/users and an attacker. The attacker seeks to mount a denial of service attack on the client-server protocol by overloading the computational resources of the server. The client-puzzle approach is used as a defense mechanism against the attack. The interaction between the attacker and the defender during a denial of service attack is viewed as a two-player game. We use the same notations as in [5] to model the game.

Rational Attacker. Our primary assumption is that the attacker is rational. The objective of the attacker is to maximize the resource expenditure of the defender with minimum computational effort. This is reasonable from the point of view of the proof of work paradigm, where a rational attacker is the **strongest** attacker. On the other hand, if the attacker is not rational and takes non-optimal decisions, it would be in the interest of the defender.

3.1 Model

Consider a game between an attacker and a defender. We categorize the puzzles used by the defender as either easy or difficult. A puzzle is easy if the time taken to solve it is lesser than the time taken by the defender to provide the requested service and is difficult if the time taken to solve the puzzle is greater than the service time. Assume that the defender uses an easy puzzle P_1 and a difficult puzzle P_2 to defend himself. (We later show in Section 4.1 that two puzzles are sufficient for an effective defense mechanism.)

Let T be a reference time period. Let α_m be the fraction of the time T that the defender spends in providing the service, α_{PP} be the fraction of T he takes to produce a puzzle and α_{VP} be the fraction of T he takes to verify it. Let α_{SP_1} be the fraction of T that the attacker is expected to spend to solve P_1 and let α_{SP_2} be the fraction of T to solve P_2. As mentioned earlier, the defender chooses the puzzles P_1 and P_2 such that $\alpha_{SP_1} < \alpha_m < \alpha_{SP_2}$.

Attacker Actions. On receiving a puzzle, the attacker may choose from one among the following actions: (i) correctly answer the puzzle (CA), (ii) randomly answer the puzzle (RA) and (iii) try to answer the puzzle, but give up if it is too hard (TA). In the case of TA, the attacker gives a correct answer if the puzzle is solved and a random answer if he gives up. Note that TA is relevant only when the puzzle difficulty is hidden. If the attacker knows the difficulty of the puzzle on receiving it, he can immediately decide on whether to answer it correctly or randomly.

Attacker Payoff. Let u_2 denote the payoff of the attacker. Attacker's action is profitable if the defender expends computational resource, else it is a loss when he himself incurs an expenditure. Let P_i, $i = 1, 2$, be the puzzle received by the attacker. If he chooses CA, he incurs a cost α_{SP_i} in solving the puzzle, while the defender expends resources in generating and verifying the puzzle and providing the requested service. His payoff is therefore

$$u_2(P_i; CA) = \alpha_{PP} + \alpha_{VP} + \alpha_m - \alpha_{SP_i}.$$

If the attacker chooses RA, the attacker incurs no cost, while the defender incurs a cost in generating and verifying the puzzle.

$$u_2(P_i; RA) = \alpha_{PP} + \alpha_{VP}.$$

If the attacker's response is TA, his payoff depends on when he gives up.

Try and Answer. When the attacker receives puzzle P_1, he is better off answering it correctly, rather than answering it randomly. This is because $u_2(P_1; CA) > u_2(P_1; RA)$ (as $\alpha_{SP_1} < \alpha_m$). On the other hand, when he receives P_2, $u_2(P_2; CA) < u_2(P_2; RA)$ (as $\alpha_{SP_2} > \alpha_m$) and hence, RA would be a better choice than CA. A decision on RA and CA can be made only if the puzzle difficulty is known. In the case of HDPs, the attacker is sure that the puzzle is not P_1 only when he fails to solve it after expending α_{SP_1} amount of resource. Hence, when the attacker chooses TA, he puts in the (minimal) effort required to solve P_1 and gives up when he realizes the puzzle is P_2. If the puzzle sent is P_1, the attacker would solve it with the minimal effort and give the correct answer, while if it is P_2, he would give up and send a random answer. His payoff for the action TA is given by

$$u_2(P_1; TA) = \alpha_{PP} + \alpha_{VP} + \alpha_m - \alpha_{SP_1} \text{ and}$$

$$u_2(P_2; TA) = \alpha_{PP} + \alpha_{VP} - \alpha_{SP_1}.$$

3.2 Analysis of Attacker Payoff

Let $0 < p < 1$ be the probability with which the attacker receives puzzle P_1. ($1-p$ is the probability with which he receives P_2.) We denote the corresponding mixed strategy of the defender as α_1. If the difficulty of the puzzle is hidden, the expected payoff of the attacker for his actions is given by

$$U_2(\alpha_1; CA) = \alpha_{PP} + \alpha_{VP} + \alpha_m - p\alpha_{SP_1} - (1-p)\alpha_{SP_2}, \tag{1}$$

$$U_2(\alpha_1; RA) = \alpha_{PP} + \alpha_{VP} \text{ and} \tag{2}$$

$$U_2(\alpha_1; TA) = \alpha_{PP} + \alpha_{VP} + p\alpha_m - \alpha_{SP_1}. \tag{3}$$

The attacker's choice is influenced by the probability p and the values of α_{SP_1} and α_{SP_2}. The attacker would prefer RA over TA only if $p < \frac{\alpha_{SP_1}}{\alpha_m} = p_t$. He would prefer RA over CA when $p < \frac{\alpha_{SP_2} - \alpha_{SP_1}}{\alpha_{SP_2} - \alpha_m} = p_c$. From the payoffs, it is evident that the action CA is more beneficial than TA when $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$.

On the other hand, if the difficulty of the puzzle is known to the attacker, he would choose CA if the puzzle is P_1 and RA if it is P_2 [5]. We represent the corresponding strategy as (CA, RA) and his expected payoff is

$$U_2(\alpha_1; (CA, RA)) = \alpha_{PP} + \alpha_{VP} + p(\alpha_m - \alpha_{SP_1}). \tag{4}$$

We now show that **the attacker receives lower payoff when HDPs are used**. Consider each of the attacker's actions:
(i) ***RA***. The attacker chooses RA when (a) $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$ and $p < p_t$ or (b) $\alpha_{SP_2} - \alpha_{SP_1} > \alpha_m$ and $p < p_c$. From (2) and (4), for $0 < p < 1$,

$$U_2(\alpha_1; RA) < U_2(\alpha_1; (CA, RA)). \tag{5}$$

(ii) ***TA***. The attacker chooses TA when $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$ and $p > p_t$. From (3) and (4), for $0 < p < 1$,

$$U_2(\alpha_1; TA) < U_2(\alpha_1; (CA, RA)). \tag{6}$$

(iii) ***CA***. The attacker chooses CA when $\alpha_{SP_2} - \alpha_{SP_1} > \alpha_m$ and $p > p_c$. From (1) and (4), as $\alpha_{SP_2} < \alpha_m$ and $0 < p < 1$,

$$U_2(\alpha_1; CA) < U_2(\alpha_1; (CA, RA)). \tag{7}$$

In all three cases, the attacker is benefited less when the puzzle difficulty is hidden than when it is known to him.

Effectiveness. We define the effectiveness of a defense mechanism using proof of work as *the difference between the amount of work done by the attacker and the amount of work done by the defender*. Clearly, a defense mechanism would be more effective when it uses a HDP.

4 Defense Mechanisms

We propose two defense mechanisms against DoS attacks based on the concept of Nash equilibrium. Hidden difficulty puzzles are used in both the defense mechanisms. As in [5], the Nash equilibrium is used in a prescriptive way, where the defender selects and takes part in a specific equilibrium profile and the best thing for the attacker to do is to conform to his equilibrium strategy. Initially, we assume that the attack takes place from a single machine and later propose an extension to handle distributed attacks.

4.1 Strategic Game

The attacker is unaware of the difficulty of a puzzle when he receives it and the defender is unaware of the attacker's response when he sends the puzzle. We therefore model the interaction between an attacker and defender during a denial of service attack as a strategic game.

Defender's Actions. We assume the defender uses n puzzles P_1, P_2, ..., P_n such that $\alpha_{SP_1} < ... < \alpha_{SP_k} < \alpha_m < \alpha_{SP_{k+1}} < ... < \alpha_{SP_n}$, where α_{SP_i} is the cost incurred by an attacker in solving puzzle P_i. The generation and verification costs are same for all puzzles and equal to α_{PP} and α_{VP} respectively. (This assumption is reasonable as generation and verification time for a good client-puzzle is negligible [15].)

Defender's Payoff. The defender seeks to maximize the effectiveness of the defense mechanism and minimize the cost to a legitimate user. We introduce a balance factor $0 < \eta < 1$ that allows him to strike a balance between the two. His payoff is therefore given by $u_1 = (1-\eta)(\text{effectiveness}) + \eta(-\text{legitimate user cost})$.

Table 1. Cost incurred by the players and the legitimate user when action profile a is chosen. Here $1 \leq l \leq n$, $1 \leq i \leq k$ and $k+1 \leq j \leq n$.

a	$\psi_1(a)$	$\psi_2(a)$	$\psi_u(a)$
$(P_l; RA)$	$\alpha_{PP} + \alpha_{VP}$	0	α_{SP_l}
$(P_i; TA)$	$\alpha_{PP} + \alpha_{VP} + \alpha_m$	α_{SP_i}	α_{SP_i}
$(P_j; TA)$	$\alpha_{PP} + \alpha_{VP}$	α_{SP_k}	α_{SP_j}
$(P_l;\ CA)$	$\alpha_{PP} + \alpha_{VP} + \alpha_m$	α_{SP_l}	α_{SP_l}

Let $\psi_1(a)$ and $\psi_2(a)$ be the cost incurred by the defender and attacker, respectively, when the action profile a is chosen. Let $\psi_u(a)$ be the corresponding cost to a legitimate user. Hence,

$$u_1(a) = (1-\eta)(-\psi_1(a) + \psi_2(a)) + \eta(-\psi_u(a)).$$

The costs incurred by the players and the legitimate user for the various action profiles are tabulated in Table 1.

A legitimate user always solves the given puzzle and incurs a cost α_{SP_i} for a puzzle P_i. Here, it is assumed that the attacker and a legitimate user take equal time to solve a puzzle. The model can be easily extended to distributed attacks, where the computational power of the attacker is considered much higher than that of a legitimate user [12].

For the puzzles P_1, P_2, ... and P_k, the attacker is better off giving the correct answer, while for puzzles P_{k+1}, ... and P_n, the attacker is better off giving a random answer. When the puzzle difficulty is unknown, the attacker may choose to try and answer (TA), where the maximum effort he puts in is the effort required to solve P_k. If the puzzle is solved with a maximum resource expenditure of α_{SP_k}, he sends a correct answer. Otherwise, he gives up and sends a random answer.

Proposition 1. *In the strategic game of the client-puzzle approach, the best response of the defender to the attacker's action TA is the puzzle P_k or the puzzle P_{k+1} or a lottery over both.*

The proof for proposition 1 is available in the full version of this paper [12]. For all other propositions stated in this section, the proofs have been given in Appendix A.

Let P_1 and P_2 be the two puzzles corresponding to proposition 1.

Analysis of Defender Payoff. Let us consider the defender's mixed strategy α_1, where he chooses P_1 with probability $0 < p < 1$ and P_2 with probability $1-p$. A legitimate user would always incur a cost $\psi_u = p\alpha_{SP_1} + (1-p)\alpha_{SP_2}$. If the puzzle difficulty is hidden, the attacker would choose an action $a_2 \in \{RA, TA, CA\}$. The defender's payoff would then be

$$u_1(\alpha_1; a_2) = (1-\eta)(-u_2(\alpha_1; a_2)) + \eta(-\psi_u).$$

As discussed earlier, the attacker would choose $a_2 = (CA; RA)$ [5] if the puzzle difficulty is not hidden and the corresponding payoff to the defender would be

$$u_1(\alpha_1; (CA; RA)) = (1 - \eta)(-u_2(\alpha_1; (CA; RA))) + \eta(-\psi_u).$$

For the same value of η, it is seen from (5), (6) and (7) that $u_1(\alpha_1; a_2) > u_1(\alpha_1; (CA; RA))$ for all $a_2 \in \{RA, TA, CA\}$. Hence, **the defender receives higher payoff while using HDPs**.

Nash Equilibrium. We now analyze the existence of Nash equilibria in the game of the client-puzzle approach. One possible Nash equilibrium is where the attacker chooses the action TA. The conditions for such an equilibrium are given in the following proposition.

Proposition 2. *In the strategic game of the client-puzzle approach, for* $0 < \eta < \frac{1}{2}$*, a Nash equilibrium of the form* $(p \circ P_1 \oplus (1-p) \circ P_2; TA)$[1]*,* $0 < p < 1$*, exists if* $\eta = \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$*,* $\alpha_{SP_2} - \alpha_{SP_1} > \alpha_m$ *and* $p > \frac{\alpha_{SP_1}}{\alpha_m}$*.*

We now construct a defense mechanism against a DoS attack by prescribing the Nash equilibrium given in proposition 2. A Nash equilibrium allows us to predict the behavior of a rational attacker during a DoS attack, but does not prevent the flooding attack from being successful.

Mitigating DoS Attack. Let N be the maximum number of requests that an attacker can send in time T. It is assumed that the defender has a resource r_p for puzzle generation and verification and another resource r_m for providing the requested service [5]. As per the property of a good client puzzle, the generation and verification time must be negligible. In fact, the verification time can be minimized by using a table lookup [19]. Hence, it is reasonable to assume that r_p is not exhausted in an attack, i.e., $N(\alpha_{PP} + \alpha_{VP}) < 1$. On the other hand, the attack is successful when r_m is exhausted before all requests are serviced. If β is the probability with which the attacker solves a given puzzle, $N\beta$ is the expected number of attack requests for which the defender would provide service. When $N\beta\alpha_m > 1$, the defender is overwhelmed and the attack is successful. In order to mitigate an attack, we need to ensure that

$$N\beta\alpha_m \leq 1 \text{ or } \beta \leq \frac{1}{N\alpha_m}.$$

In the prescribed Nash equilibrium, $\beta = p$ and the following condition must hold for an attack to be unsuccessful: $\frac{\alpha_{SP_1}}{\alpha_m} < p < \frac{1}{N\alpha_m}$. Note that this is possible only if $\alpha_{SP_1} < \frac{1}{N}$.

The probability p is chosen such that the defender can provide service for all attack requests. It has to be remembered that even legitimate requests need to be serviced during an attack. Hence, out of the total number of requests that

[1] The notation $p_1 \circ a_1 \oplus p_2 \circ a_2 \oplus ... \oplus p_n \circ a_n$ denotes a lottery over the set of actions $\{a_1, a_2, ..., a_n\}$, where $p_1 + p_2 + ... + p_n = 1$.

the defender can service, we take $\frac{1}{\alpha_m}$ as the number of requests allocated to the defense mechanism, while the rest are for the legitimate users.

We propose a defense mechanism based on the prescribed Nash equilibrium. (We call it the **strategic game defense mechanism**.) The idea is to fine tune the various parameters such that the conditions for the equilibrium are satisfied.

1. For a desirable balance factor η, $0 < \eta < \frac{1}{2}$, choose two puzzles P_1 and P_2 such that

$$\alpha_{SP_1} < \frac{1}{N} < \alpha_m < \alpha_{SP_2},\ \alpha_{SP_2} - \alpha_{SP_1} > \alpha_m \text{ and}$$

$$\eta = \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}.$$

2. Choose a value for p such that

$$\frac{\alpha_{SP_1}}{\alpha_m} < p < \frac{1}{N\alpha_m}.$$

3. On receiving a request, generate a random variable $\mathbf{x}$ such that $Pr(\mathbf{x}{=}0) = p$ and $Pr(\mathbf{x}{=}1) = 1$ - p. If $\mathbf{x}$=0, send puzzle P_1. Otherwise, send puzzle P_2.

The other possible Nash equilibria in the strategic game have been discussed in the full version of this paper [12].

4.2 Infinitely Repeated Game

During a denial of service attack, the attacker repeatedly sends requests to the defender. Since the game is played repeatedly, this scenario can be modeled as a repeated game. Also, the probability of arrival of a request is non-zero at any point in time and hence, the game is infinitely repeated [5].

Threat of Punishment. In repeated games, the payoff of a player is not only influenced by his decision in the current game, but is also influenced by his decisions in all periods of the game. A player would therefore be willing to take sub-optimal decisions for the current game if it would give him a higher payoff in the long run. Deviation of a player from a desired strategy profile can be prevented if he is threatened with sufficient punishment in the future. Therefore, Nash equilibria with high payoffs can be achieved in an infinitely repeated game if a player is patient enough to see long term benefits over short term gains.

Minmax Payoff. The minmax payoff of a player is the minimum payoff that he can guarantee himself in a game, even when the opponents play in the most undesirable manner. The minmax payoff of player i in a strategic game is given by

$$v_i^* = \min_{\alpha_{-i} \in \Delta(A_{-i})} \max_{a_i \in A_i} u_i(a_i, \alpha_{-i}),$$

where $\Delta(X)$ is the set of probability distributions over X, A_i is the set of permitted actions for player i and u_i is his payoff function.

Nash Equilibrium. Consider a two-player infinitely repeated game. Let v_1^* be the minmax payoff of player 1 and v_2^* be the minmax payoff of player 2. Let the mixed strategy profile resulting in v_1^* and v_2^* be $M^1 = (M_1^1; M_2^1)$ and $M^2 = (M_1^2; M_2^2)$ respectively. Here, M_1^2 is player 1's minmax strategy against player 2 and M_2^1 is player 2's minmax strategy against player 1.

Let $(\alpha_1; \alpha_2)$ be a strategy profile such that $v_1 = u_1(\alpha_1; \alpha_2) > v_1^*$ and $v_2 = u_2(\alpha_1; \alpha_2) > v_2^*$. Fudenberg and Maskin [7] show that an equilibrium where each player i receives an average payoff of v_i is possible through threat of punishment. The following repeated game strategy for player i is a Nash equilibrium.

(A) Play α_i each period as long as $(\alpha_1; \alpha_2)$ was played last period. After any deviation from phase (A), switch to phase (B).

(B) Play M_i^j, $j \neq i$, τ times (say) and then start phase (A) again. If there are any deviations while in phase (B), restart phase (B).

A description of their theorem along with the calculation of τ has been detailed in the full version of the paper [12].

Interpretation. A possible equilibrium in the game of the client-puzzle approach consists of two phases:

Normal Phase (A). The defender and attacker choose a strategy profile, where each of them receive a payoff greater than the minmax payoff. Note that the strategy played may not be the optimal choice of the players in the given period. However, if either of them deviate, the game switches to the punishment phase (B).

Punishment Phase (B). In this phase, each player chooses a minmax strategy against the other player. This phase remains for τ periods, after which the game switches to the normal phase. Again, the minmax strategy may not be the optimal strategy of a player in the current period. But, any deviation from this strategy would restart the phase.

Any deviation in the normal phase is deterred by the threat of switching to the punishment phase. Similarly, a deviation from the punishment phase is deterred by the threat of prolonged punishment. Note that the **punishment period** τ must be sufficiently long for the equilibrium to exist.

The following propositions identify some minmax strategies in the game of the client-puzzle approach.

Proposition 3. *In the game of the client-puzzle approach, when* $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$*, one of the defender's minmax strategy against the attacker is*

$$p_1 \circ P_1 \oplus (1 - p_1) \circ P_2,$$

where $p_1 = \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}$.

Proposition 4. *In the game of the client-puzzle approach, when* $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$ *and* $0 < \eta < \frac{1}{2}$*, the attacker's minmax strategy against the defender is*

$$p_2 \circ CA \oplus (1 - p_2) \circ RA,$$

where $p_2 = \frac{\eta}{1-\eta}$.

Punishment Phase Strategies. During the punishment phase, the defender chooses the mixed strategy $p_1 \circ P_1 \oplus (1-p_1) \circ P_2$, while the attacker chooses the mixed strategy $p_2 \circ CA \oplus (1-p_2) \circ RA$. It can be shown that the corresponding strategy profile is a Nash equilibrium, where both the players receive their minmax payoff. This means that each player's strategy in the punishment phase is the best response to the opponent's strategy. Clearly, deviations in the punishment phase are not profitable.

Normal Phase Strategies. The following strategy profile is chosen during the normal phase:

$$(p \circ P_1 \oplus (1-p) \circ P_2; TA),$$

where $0 < p < 1$. For a Nash equilibrium, this profile must give each player a payoff greater than his minmax payoff. This is possible when

$$\frac{\alpha_{SP_1}}{\alpha_m} < p < \frac{\alpha_{SP_1} - \eta(\alpha_{SP_2} - \alpha_m + \alpha_{SP_1})}{\alpha_m - \eta(\alpha_{SP_2} + \alpha_m - \alpha_{SP_1})}.$$

It can shown that the defender receives higher payoff in the Nash equilibrium of the repeated game than in the Nash equilibrium of the single-shot strategic game described in Section 4.1.

Mitigating DoS Attack. As seen in the previous defense mechanism, the existence of a Nash equilibrium does not necessarily prevent flooding. In the **normal phase**, flooding is prevented if

$$N\alpha_m p < 1 \text{ or } p < \frac{1}{N\alpha_m}.$$

In the **punishment phase**, even if the attacker chooses to answer all the puzzles correctly, his average resource expenditure would be $p_1\alpha_{SP_1} + (1-p_1)\alpha_{SP_2} = \alpha_m$. Since he cannot overload the defender, flooding is not possible in this phase.

The defense mechanism based on the described Nash equilibrium is given below. (We call it the **repeated game defense mechanism**.)

1. For a desirable balance factor η, $0 < \eta < \frac{1}{2}$, choose two puzzles P_1 and P_2 such that $\alpha_{SP_1} < \frac{1}{N} < \alpha_m < \alpha_{SP_2}$ *and* $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$.
2. Choose a value for p such that

$$\frac{\alpha_{SP_1}}{\alpha_m} < p < min\left(\frac{1}{N\alpha_m}, \frac{\alpha_{SP_1} - \eta(\alpha_{SP_2} - \alpha_m + \alpha_{SP_1})}{\alpha_m - \eta(\alpha_{SP_2} + \alpha_m - \alpha_{SP_1})}\right)$$

   and determine the value of p_1 according to

$$p_1 = \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}.$$

3. Determine an appropriate value for τ.
4. Phase (A) and phase (B) of the defense mechanism have been described in Fig. 2, where $\mathbf{x}$ and $\mathbf{y}$ are random variables and $\phi(msg)$ is the phase corresponding to the puzzle whose solution has been received.

Here, we have considered only one possible minmax strategy profile in the defense mechanism. The other strategy profiles have been analyzed in [12].

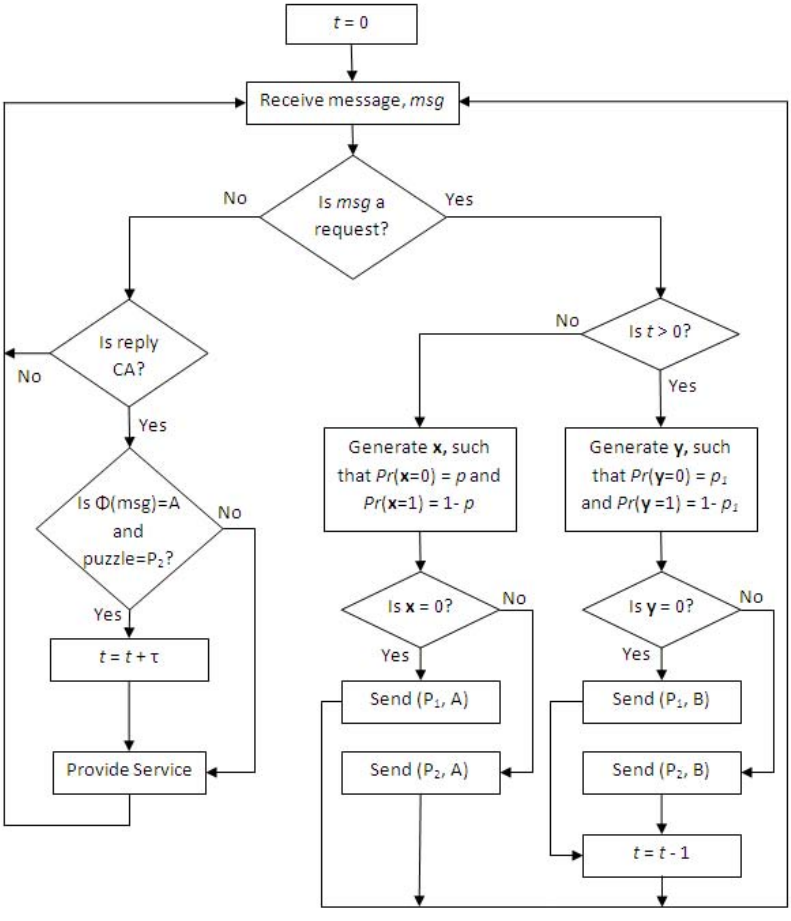

Fig. 2. Closed-Loop Defense Mechanism

4.3 Comparison with Previous Work

We now compare the proposed HDP-based defense mechanisms with the corresponding defense mechanisms proposed by Fallah [5], where puzzle difficulties are known to the attacker. Previously, while analyzing the payoffs of the attacker and defender, we did not consider the existence of a Nash equilibrium in the game. We now present an analysis of the attacker and defender payoffs under the equilibrium conditions given by the various defense mechanisms.

For the sake of convenience, the HDP-based strategic game defense mechanism shall be referred to as $HDM1$, while the HDP-based repeated game defense mechanism shall be referred to as $HDM2$. The corresponding puzzle-based defense mechanisms in [5] would be referred to as $PDM1$ and $PDM2$ respectively.

Proposition 5. *The expected equilibrium payoff of an attacker in HDM1 is lower than that in PDM1, while the expected equilibrium payoff of the defender is the same in both defense mechanisms.*

Note that the equilibrium conditions are different in the two defense mechanisms. As a consequence, the defender's expected payoff does not change when HDPs are used. However, the attacker is benefited less in $HDM1$ making it more effective than $PDM1$.

Proposition 6. *The minmax payoff of the defender in HDM2 is higher than that in PDM2, while the minmax payoff of the attacker is the same in both defense mechanisms.*

Since the minmax payoff is a lower bound on the defender's payoff, the defender is better off in HDM2. Moreover, in $PDM2$, only P_2 puzzles are used by the defender in the punishment phase. Whereas, in $HDM2$, the defender's minmax strategy against the attacker is a lottery over P_1 and P_2. Clearly, **a legitimate user is hurt less in the punishment phase of HDM2 as he has a chance of receiving** P_1.

5 Client-Puzzles

The requirements of a puzzle in our defense mechanisms are given below.

1. **Hidden Difficulty:** The difficulty of the puzzle should not be determined without a minimal number of computations.
2. **High Puzzle Resolution:** The granularity of puzzle difficulty must be high. This allows us to fine tune the parameters of the defense mechanisms.
3. **Partial Solution:** Submission of partial solutions should be possible without increasing the verification time. This requirement allows the defender to determine whether the attacker chose RA or TA.

We now introduce two new puzzles that conform to these requirements.

Puzzle 2: The puzzle is described in Fig. 3. Here, the client can submit a partial solution by giving the correct answer for the first part ($rp1$) of the puzzle and a random answer for the second part ($rp2$). It takes 4 hash computations for generating a puzzle and a maximum of 6 hash computations for verifying the puzzle solution. When $l = 1$, the average number of hash computations required to solve the puzzle is $\frac{(2^k+1) + (D+1)}{2}$. Thus, the difficulty of the puzzle can be varied exponential by adjusting k and linearly by adjusting D. Also, the client is not aware of the difficulty of the puzzle when he receives it.

Puzzle 3: Fig. 4 contains the puzzle description. When $l = 1$, the average number of hash computations required to solve this puzzle is $\frac{(D_a+1) + (D_b+1)}{2}$ and hence, the difficulty varies linearly with D_a and D_b. The production of the puzzle requires 4 hash computations and the verification requires a maximum of 6 hash computations. Even here, the puzzle difficulty is hidden from the client.

Client		Defender
	$\xrightarrow{Request}$	$X = H(S_1, N_s, M)$
		$Y = H(X)$
		$a = H(S_2, N_s, M) \bmod D + l$
		$X' = X - a$
		$Z = H(X')$
	$\xleftarrow{(X'', Y, Z), N_s}$	$X'' = X' \oplus (I_1, ..., I_{k-1}, 1, 0_{k+1}, ..., 0_n)$
Find $rp1$ such that $H(rp1) = Z$. Find a' such that $H(rp2) = Y$, where $rp2 = rp1 + a'$.	$\xrightarrow{rp1, rp2, N_s}$	$X = H(S_1, N_s, M)$
		$a = H(S_2, N_s, M) \bmod D + l$
		$H(rp1) \stackrel{?}{=} H(X - a)$
		$H(rp2) \stackrel{?}{=} H(X)$

Fig. 3. Hidden Difficulty Puzzle 2. Here, S_1 and S_2 are server secrets, N_s is a server nonce, M is a session parameter, D and k are difficulty parameters, l is a constant and I is a binary number chosen uniformly at random.

Client		Defender
	$\xrightarrow{Request}$	$X = H(S_1, N_s, M)$
		$Y = H(X)$
		$a = H(S_2, N_s, M) \bmod D_a + l$
		$X' = X - a$
		$Z = H(X')$
	$\xleftarrow{(X'', Y, Z), N_s}$	$X'' = X' - b$
Find b' such that $H(rp1) = Z$, where $rp1 = X'' + b'$. Find a' such that $H(rp2) = Y$, where $rp2 = rp1 + a'$.	$\xrightarrow{rp1, rp2, N_s}$	$X = H(S_1, N_s, M)$
		$a = H(S_2, N_s, M) \bmod D_a + l$
		$H(rp1) \stackrel{?}{=} H(X - a)$
		$H(rp2) \stackrel{?}{=} H(X)$

Fig. 4. Hidden Difficulty Puzzle 3. Here, b is a value chosen uniformly at random from $\{1, ..., D_b\}$.

6 Conclusions

In this paper, we have given emphasis on hiding the difficulty of client-puzzles from a denial of service attacker. We have constructed three concrete puzzles that satisfy this requirement. Using game theory, we have developed defense mechanisms that make use of such puzzles and have shown that they are more effective than the existing ones. Future direction of work would be to incorporate the proposed defense mechanisms in the Internet Key Exchange (IKE) protocol and to estimate its effectiveness in real-time.

References

1. Agah, A., Das, S.K.: Preventing dos attacks in wireless sensor networks: A repeated game theory approach. International Journal of Network Security 5(2), 145–153 (2007)
2. Aura, T., Nikander, P., Leiwo, J.: DOS-resistant authentication with client puzzles. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2000. LNCS, vol. 2133, pp. 170–177. Springer, Heidelberg (2001)
3. Bencsath, B., Vajda, I., Buttyan, L.: A game based analysis of the client puzzle approach to defend against dos attacks. In: Proceedings of the 2003 International Conference on Software, Telecommunications and Computer Networks, pp. 763–767 (2003)
4. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993)
5. Fallah, M.: A puzzle-based defense strategy against flooding attacks using game theory. IEEE Transactions on Dependable and Secure Computing 99(2), 5555
6. Feng, W., Kaiser, E., Luu, A.: Design and implementation of network puzzles. In: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, March 2005, vol. 4, pp. 2372–2382 (2005)
7. Fudenberg, D., Maskin, E.: The folk theorem in repeated games with discounting or with incomplete information. Econometrica 54(3), 533–554 (1986)
8. Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: Proceedings of NDSS 1999 (Networks and Distributed Security Systems), pp. 151–165 (1999)
9. Komathy, K., Narayanasamy, P.: Secure data forwarding against denial of service attack using trust based evolutionary game. In: Vehicular Technology Conference, VTC Spring 2008., pp. 31–35. IEEE, Los Alamitos (2008)
10. Lv, J.-J.: A game theoretic defending model with puzzle controller for distributed dos attack prevention. In: 2008 International Conference on Machine Learning and Cybernetics, July 2008, vol. 2, pp. 1064–1069 (2008)
11. Mahimkar, A., Shmatikov, V.: Game-based analysis of denial-of-service prevention protocols. In: CSFW 2005: Proceedings of the 18th IEEE workshop on Computer Security Foundations, pp. 287–301. IEEE Computer Society, Washington (2005)
12. Narasimhan, H., Varadarajan, V., Pandu Rangan, C.: Game theoretic resistance to denial of service attacks using hidden difficulty puzzles. Cryptology ePrint Archive, Report 2009/350 (2009), `http://eprint.iacr.org/`

13. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA (1996)
14. Sagduyu, Y.E., Ephremides, A.: A game-theoretic analysis of denial of service attacks in wireless random access. Wireless Networks 15(5), 651–666 (2009)
15. Smith, J.: Denial of Service: Prevention, Modelling and Detection. PhD thesis, Queensland University of Technology, Brisbane, QLD 4001 Australia (June 2007)
16. Tritilanunt, S., Boyd, C., Foo, E., Nieto, J.G.: Toward non-parallelizable cryptographic puzzles. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 247–264. Springer, Heidelberg (2007)
17. Tsang, P.P., Smith, S.W.: Combating spam and denial-of-service attacks with trusted puzzle solvers. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 188–202. Springer, Heidelberg (2008)
18. Wang, X.F., Reiter, M.K.: Defending against denial-of-service attacks with puzzle auctions. In: SP 2003: Proceedings of the 2003 IEEE Symposium on Security and Privacy, p. 78. IEEE Computer Society, Washington (2003)
19. Waters, B., Juels, A., Alex Halderman, J., Felten, E.W.: New client puzzle outsourcing techniques for dos resistance. In: CCS 2004: Proceedings of the 11th ACM conference on Computer and Communications Security, pp. 246–256. ACM, New York (2004)
20. Zhang, R., Hanaoka, G., Imai, H.: A generic construction of useful client puzzles. In: ASIACCS 2009: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, pp. 70–79. ACM, New York (2009)

A Proof of Propositions

A.1 Proposition 2

Proof. Let us prove the existence of a Nash equilibrium, where the defender uses a mixed strategy $\alpha_1 = p \circ P_1 \oplus (1-p) \circ P_2$, where $0 < p < 1$ and the attacker uses the pure strategy TA. The profile $(\alpha_1; TA)$ is a Nash equilibrium if

$$u_1(P_1; TA) = u_1(P_2; TA), \tag{8}$$

$$u_2(\alpha_1; TA) > u_2(\alpha_1; RA) \text{ and} \tag{9}$$

$$u_2(\alpha_1; TA) > u_2(\alpha_1; CA). \tag{10}$$

Equation (8) is satisfied when

$$\eta = \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}, \tag{11}$$

(9) is satisfied when $p > \frac{\alpha_{SP_1}}{\alpha_m}$ and (10) is satisfied when

$$\alpha_{SP_2} - \alpha_{SP_1} > \alpha_m. \tag{12}$$

From (11) and (12), it can be easily seen that the maximum value that η can take is less than $\frac{1}{2}$. Hence, $0 < \eta < \frac{1}{2}$.

A.2 Proposition 3

Proof. Let $\alpha_1 = p_1 \circ P_1 \oplus (1-p_1) \circ P_2$, $0 < p_1 < 1$, be the defender's minmax strategy against the attacker. By our assumption, $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$. Clearly, the attacker would prefer CA over TA. Therefore, the attacker's minmax payoff is $max(U_2(\alpha_1; CA), U_2(\alpha_1; RA))$, where U_i is the expected payoff of player i for $i = 1, 2$. Note that $U_2(\alpha_1; CA) > U_2(\alpha_1; RA)$ when $\alpha_{PP} + \alpha_{VP} + \alpha_m - p_1\alpha_{SP_1} - (1-p_1)\alpha_{SP_2} > \alpha_{PP} + \alpha_{VP}$ or $p_1 > \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}$. Higher the value of p_1 above $\frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}$, higher is the attacker's payoff. If $p_1 \leq \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}$, the attacker's payoff is minimum and equal to $\alpha_{PP} + \alpha_{VP}$. This is the attacker's minmax payoff and hence, the attacker is minmaxed when the defender chooses the mixed strategy $p_1 \circ P_1 \oplus (1-p_1) \circ P_2$, where $p_1 = \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}$.

A.3 Proposition 4

Proof. Let the attacker's minmax strategy against the defender be $\alpha_2 = q_1 \circ RA \oplus q_2 \circ CA \oplus q_3 \circ TA$, where $q_1 + q_2 + q_3 = 1$. By our assumption, $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$. When the attacker chooses CA, the defender would receive equal or lower payoff than when the attacker chooses TA. Hence, the attacker's minmax strategy against the defender should assign non-zero probabilities to CA and RA and zero probability to TA, i.e, $q_1 = p_2$, $q_2 = 1 - p_2$ and $q_3 = 0$, where $0 < p_2 < 1$. When $0 < \eta < \frac{1}{2}$, the defender's best response for the attacker's pure strategy RA is P_1 and that for CA is P_2. For the attacker's mixed strategy α_2, the defender's best response is P_1 only if $U_1(P_1; \alpha_2) > U_1(P_2; \alpha_2)$. This is possible when $p_2((1-\eta)\alpha_{SP_1}) - \eta\alpha_{SP_1} > p_2((1-\eta)\alpha_{SP_2}) - \eta\alpha_{SP_2}$ or $p_2 < \frac{\eta}{1-\eta}$. The lower the value of p_2 below $\frac{\eta}{1-\eta}$, higher is the defender's payoff. Similarly, if $p_2 > \frac{\eta}{1-\eta}$, the defender would prefer P_2 over P_1 and his payoff increases as p_2 increases. Clearly, the defender is minmaxed when $U_1(P_1; \alpha_2) = U_1(P_2; \alpha_2)$ or $p_2 = \frac{\eta}{1-\eta}$.

A.4 Proposition 5

Proof. Let α^1 be the equilibrium strategy profile used in HDM1. Under equilibrium conditions, the expected payoff of the defender is $U_1(\alpha^1) = (1-\eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_1}) - \eta\alpha_{SP_1}$. The attacker, on the other hand, receives an average payoff of

$$U_2(\alpha^1) = \alpha_{PP} + \alpha_{VP} + p\alpha_m - \alpha_{SP_1}. \tag{13}$$

In the case of PDM1 [5], the strategy profile $\alpha^2 = (p \circ P_1 \oplus (1 - p) \circ P_2; (CA,RA))$ corresponds to an equilibrium when $\alpha_{SP_1} < \alpha_m < \alpha_{SP_2}$, $\alpha_{VP} < \alpha_m - \alpha_{SP_1}$, $\alpha_{VP} < \alpha_{SP_2} - \alpha_m$ and $\eta = \frac{\alpha_m - \alpha_{SP_1}}{\alpha_m - 2\alpha_{SP_1} + \alpha_{SP_2}}$, where $0 < \eta < \frac{1}{2}$ and $0 < p < 1$. Under the equilibrium conditions, the expected payoff for the defender is $U_1(\alpha^2) = (1 - \eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_1}) - \eta\alpha_{SP_1}$ and that for the attacker is

$$U_2(\alpha^2) = \alpha_{PP} + \alpha_{VP} + p(\alpha_m - \alpha_{SP_1}). \tag{14}$$

Note that we are looking at two games with different equilibrium conditions. For a given value of $0 < \eta < \frac{1}{2}$, we choose the same values for α_{PP}, α_{VP}, α_{SP_1} and p and different values for α_{SP_2} in the two games such that equilibrium conditions are satisfied. Clearly, $U_1(\alpha^1) = U_1(\alpha^2)$, while from (13) and (14), for $0 < p < 1$,

$$U_2(\alpha^1) < U_2(\alpha^2).$$

Thus, the expected payoff for the defender is same in both the defense mechanisms, while the attacker's expected payoff is lower in HDM1.

A.5 Proposition 6

Proof. In PDM2, the minmax strategy profile in the infinitely repeated game is $(P_2; RA)$ [5]. The corresponding minmax payoff for the defender is

$$U_1(P_2; RA) = (1-\eta)(-\alpha_{PP} - \alpha_{VP}) - \eta\alpha_{SP_2} \tag{15}$$

and that for the attacker is $U_2(P_2; RA) = \alpha_{PP} + \alpha_{VP}$. In the case of HDM2, the minmax strategy profile is $(\alpha_1; \alpha_2)$, where $\alpha_1 = p_1 \circ P_1 \oplus (1-p_1) \circ P_2$ and $\alpha_2 = p_2 \circ CA \oplus (1-p_2) \circ RA$. The defender's minmax payoff in HDM2 is given by $U_1(\alpha_1; \alpha_2) = p_1(p_2 u_1(P_1; CA) + (1-p_2)u_1(P_1; RA)) + (1-p_1)(p_2 u_1(P_2; CA) + (1-p_2)u_1(P_2; RA))$, which reduces to

$$U_1(\alpha_1; \alpha_2) = (1-\eta)(-\alpha_{PP} - \alpha_{VP}) - \eta\alpha_m. \tag{16}$$

Similarly, the attacker's minmax payoff in HDM2 is given by $U_2(\alpha_1; \alpha_2) = p_1(p_2 u_2(P_1; CA) + (1-p_2)u_2(P_1; RA)) + (1-p_1)(p_2 u_2(P_2; CA) + (1-p_2)u_2(P_2; RA)) = \alpha_{PP} + \alpha_{VP}$. It is clear that the attacker's minmax payoff is same in both defense mechanisms. For a given value of $0 < \eta < \frac{1}{2}$, choosing same values for α_{PP}, α_{VP} and α_{SP_1} in equations (15) and (16) and different values for α_{SP_2} satisfying the corresponding equilibrium criteria, we have

$$U_1(\alpha_1; \alpha_2) > U_1(P_2; RA)$$

and $U_2(\alpha_1; \alpha_2) = U_2(P_2; RA)$. Thus, the defender's minmax payoff is higher in HDM2 when compared to PDM2, while the attacker's minmax payoff is same in both the defense mechanisms.