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

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

[1]Department of Computer Science and Engineering,
College of Engineering Guindy,
Anna University, Chennai, India.
{nhari88,venk1989}@gmail.com

[2]Theoretical Computer Science Laboratory,
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. Game theory is used to develop defense mechanisms, which make use of such puzzles. New puzzles that satisfy the requirements of the defense mechanisms have been proposed. We also show that our defense mechanisms are more effective than the ones proposed in the earlier work by Fallah.

**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 [18].

Proof-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.

* Work supported by IITM Summer Fellowship Programme May-July 2009

Thus, the server is assured that the client is willing to commit his resources to the protocol. This approach was first proposed by Dwork and Naor [5] to control junk mails. Over the years, lot of research has gone into this area and different client-puzzles have been proposed [16, 11, 1, 9, 20, 21, 7, 19, 3, 23].

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. [7] is to adjust the puzzle difficulty proportional to the current load on the server. However, the attacker may herd simple puzzles from the server during low load, solve them and thus flood the server [18]. Juels and Brainard [11] 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 [2, 6].

Though there have been several works that formally analyze denial of service attacks using game theory [2, 22, 14, 17, 6, 12, 13], only few of them analyze the client-puzzle approach. Bencsath et al. [2] modeled the client-puzzle approach as a single-shot strategic-form game and identified the defender's equilibrium strategy. Fallah [6], 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 [13] applied game theory to puzzle auctions and came up with a defense mechanism against distributed DoS. Our work is based on the game theoretic model proposed by Fallah.

In addition to the basic properties of a good puzzle [18], 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 develop game theoretic defense mechanisms which make use of such puzzles and show that they are more effective than the existing ones [6]. In our defense mechanisms, we prescribe a Nash equilibrium, where the best thing for the attacker to do is to try to solve every puzzle and give a random answer if it is too hard. We also introduce new puzzles which are suited for our defense mechanisms. Previous game theoretic approaches to client-puzzles [2, 6] assume that the random answers given by the attacker are always wrong. But the attacker's random guess could end up being correct with a small probability. Taking this into account, we propose modifications to the game model and identify the conditions for equilibrium. Our basic contribution in this paper is the notion of hidden puzzle difficulty, which, to the best of our knowledge, has not been emphasized in previous works.

The rest of the paper is organized as follows: Section 2 describes the game of client-puzzle approach. In Section 3, defense mechanisms based on Nash equilibrium are proposed. Section 4 specifies modifications to existing client-puzzles to suit the proposed defense mechanisms and also introduces new puzzles. We conclude the paper in Section 5.

## 2 Game Model

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 by the server/defender. In addition to the assumptions in [11], we make the following assumptions:

1. **Puzzle generation and verification does not lead to denial of service.** The defender has enough resources to generate and verify puzzles for every request of the attacker without denying service to a legitimate user.
2. **The computational effort to solve a puzzle is the same for a legitimate user and an attacker.** When the attack is carried out from a single machine, the time taken to solve a puzzle is the same for a legitimate user and an attacker.
3. **The attacker is rational.** The attacker always chooses the best action given his preferences and beliefs.

The interaction between the attacker and the defender during a denial of service attack is viewed as a two-player infinitely repeated game with discounting. Both the players are rational. The objective of the attacker is to maximize the resource expenditure of the defender with minimum computational effort. On the other hand, the defender seeks to maximize the resource expended by the attacker and minimize not only his resource expenditure, but also that of a legitimate user. The legitimate user is not considered a player in this game.

### 2.1 Stage-game

Let $G$ be a strategic-form game between the defender and the attacker. The set of all action profiles is denoted by $A$ and the payoff function of each player $i$ is denoted by $u_i$ [15]. On receiving a request from the attacker, the defender generates one among $n$ puzzles of varying difficulties. We adopt the same notations as in [6] to represent the players' actions. The set of actions that the defender can take is given by $A_1 = \{P_1, P_2, ..., P_n\}$. The action $P_i$, $1 \leq i \leq n$ stands for generating a puzzle of difficulty level $i$. A puzzle of difficulty level $i$ is more difficult than a puzzle of difficulty level $j$ if $i > j$. In [6], three actions were permitted for the attacker, namely $QT$, $RA$ and $CA$, where $QT$ stands for quitting the protocol without giving a solution, $RA$ stands for randomly answering the puzzle and $CA$ stands for solving the puzzle and giving the correct answer. It has to be noted that in our game, the attacker will not know the difficulty level of a puzzle when he receives it. Hence, we introduce two more actions $TQ$ (Try and Quit) and $TA$ (Try and Answer), where the attacker expends a minimal amount of computational effort in an attempt to solve the puzzle. In the case of $TQ$, he gives a correct answer if he could solve the puzzle with minimal effort and quits otherwise. $TA$ is similar, but the attacker gives a random answer when the puzzle is not solved. The set of actions possible for the attacker is $A_2 = \{QT, RA, TQ, TA, CA\}$.

**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 $k{+}1 \leq j \leq n$.

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

Since the difficulty level of the puzzle sent is unknown to the attacker, the defender's action is not observed by the attacker. Hence, we model the game as a strategic-form game as against Fallah's extensive-form game [6].

In the client-puzzle approach, we assume that the defender uses two types of resources, one for producing and verifying the puzzle and the other for providing the service protected by the defense mechanism. 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. In our model, the attacker has the same payoff function as in [6], but the payoff function for the defender is slightly different. Since an attacker would prefer the action profile that maximizes the defender's cost and minimizes his cost, his payoff is given by $u_2(a) = \psi_1(a)$ - $\psi_2(a)$. The defender's payoff depends on the level of quality of service $\eta$, $0 \leq \eta \leq 1$, that he wishes to provide to a legitimate user. The defender seeks to maximize the effectiveness of the defense mechanism and minimize the cost to a legitimate user. The factor $\eta$ allows him to strike a balance between the two. Hence, his payoff is given by $u_1(a) = (1 - \eta)(-\psi_1(a) + \psi_2(a)) + \eta(-\psi_u(a))$.

We shall now quantify the costs incurred in the game as in [6]. Let $T$ be a reference time period. Let $\alpha_m$ be the fraction of the time $T$ that the defender spends in providing the service, $\alpha_{PP_i}$ be the fraction of $T$ he takes to produce a puzzle $P_i$ and $\alpha_{VP_i}$ be the fraction of $T$ he takes to verify it. Let $\alpha_{SP_i}$ be the fraction of $T$ that the attacker or legitimate user is expected to spend to solve $P_i$. Note that $\frac{1}{\alpha_m}$ is the number of requests that can be serviced by the defender in time $T$ and $\frac{1}{\alpha_{SP_i}}$ is the number of $P_i$ puzzles that the attacker can solve in the same time. It is assumed that $\alpha_{VP_i} = \alpha_{VP}$ and $\alpha_{PP_i} = \alpha_{PP}$ for $1 \leq i \leq n$ and $\alpha_{SP_1} < ... < \alpha_{SP_k} < \alpha_m < \alpha_{SP_{k+1}} < ... < \alpha_{SP_n}$. The costs incurred by the players and the legitimate user for the various action profiles are tabulated in Table 1. In the case of actions $TQ$ or $TA$, before deciding to quit or give a random answer respectively, the attacker expends minimal computational effort in an attempt to solve the puzzle. It is reasonable to assume that this minimal effort is the effort required to solve the puzzle $P_k$. Hence, if the defender's action is $P_i$, $1 \leq i \leq k$, the attacker would solve the puzzle when he chooses $TQ$ or $TA$. When the defender chooses $P_j$, $k{+}1 \leq j \leq n$, the attacker expends some effort ($\alpha_{SP_k}$) and either quits ($TQ$) or gives a random answer ($TA$). It is assumed that

**Table 2.** Stage-game payoffs for the defender and the attacker. The rows indicate the defender's actions where, $1 \leq i \leq k$ and $k+1 \leq j \leq n$ and the columns indicate the attacker's actions. Payoffs for the defender: $u_{i1} = (1 - \eta)(-\alpha_{PP}) + \eta(-\alpha_{SP_i})$, $u_{i2} = (1 - \eta)(-\alpha_{PP} - \alpha_{VP}) + \eta(-\alpha_{SP_i})$, $u_{i3} = u_{i4} = u_{i5} = (1 - \eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_i}) + \eta(-\alpha_{SP_i})$, $u_{j1} = (1 - \eta)(-\alpha_{PP}) + \eta(-\alpha_{SP_j})$, $u_{j2} = (1 - \eta)(-\alpha_{PP} - \alpha_{VP}) + \eta(-\alpha_{SP_j})$, $u_{j3} = (1 - \eta)(-\alpha_{PP} + \alpha_{SP_k}) + \eta(-\alpha_{SP_j})$, $u_{j4} = (1 - \eta)(-\alpha_{PP} - \alpha_{VP} + \alpha_{SP_k}) + \eta(-\alpha_{SP_j})$, $u_{j5} = (1 - \eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_j}) + \eta(-\alpha_{SP_j})$. Payoffs for the attacker: $v_{i1} = v_{j1} = \alpha_{PP}$, $v_{i2} = v_{j2} = \alpha_{PP} + \alpha_{VP}$, $v_{i3} = v_{i4} = v_{i5} = \alpha_{PP} + \alpha_{VP} + \alpha_m - \alpha_{SP_i}$, $v_{j3} = \alpha_{PP} - \alpha_{SP_k}$, $v_{j4} = \alpha_{PP} + \alpha_{VP} - \alpha_{SP_k}$, $v_{i5} = \alpha_{PP} + \alpha_{VP} + \alpha_m - \alpha_{SP_j}$.

| | $QT$ | $RA$ | $TQ$ | $TA$ | $CA$ |
|---|---|---|---|---|---|
| $P_i$ | $u_{i1}$, $v_{i1}$ | $u_{i2}$, $v_{i2}$ | $u_{i3}$, $v_{i3}$ | $u_{i4}$, $v_{i4}$ | $u_{i5}$, $v_{i5}$ |
| $P_j$ | $u_{j1}$, $v_{j1}$ | $u_{j2}$, $v_{j2}$ | $u_{j3}$, $v_{j3}$ | $u_{j4}$, $v_{j4}$ | $u_{j5}$, $v_{j5}$ |

the attacker never solves a puzzle by giving a random answer and hence, the defender does not provide service for such requests. The payoff matrix is given in Table 2.

### 2.2 Repeated Game

During a denial of service attack, the attacker repeatedly sends requests to the defender. The defender responds with a puzzle and the attacker may choose from one of his available actions. Clearly, 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.

Given the strategic-form game $G$, the infinitely repeated game of $G$ with the discount factor $\delta$ is an extensive-form game in which the set of terminal histories is the set of infinite sequences $(a^1, a^2, ...)$ of action profiles in $G$ and the preference of each player $i$ to the terminal history $(a^1, a^2, ...)$ is given by the discounted average $(1 - \delta)\sum_{t=1}^{\infty} \delta^{t-1} u_i(a^t)$ [15]. The infinitely repeated game has perfect information if each player knows the actions taken previously by his opponent before making his choice. But, in the game of the client-puzzle approach, the defender, on sending a puzzle to a client, may receive another request before receiving the solution from the client. Clearly, the defender will not know the action taken by the attacker before sending the next puzzle. On the other hand, the attacker will never know the difficulty level of the puzzle received when he gives a random answer or quits. The solutions proposed in the paper are based on the assumption that the game has perfect information. Hence, we have proposed suitable modifications to tackle the imperfect observability of the defender.

## 3 Defense Mechanisms

We propose a number of defense mechanisms against DoS attacks based on the concept of Nash equilibrium. As in [6], the concept of 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.

The Nash equilibrium of an infinitely repeated game can either be open-loop, where the stage-game equilibrium is played during every period irrespective of the corresponding history, or closed-loop, where the players' strategies are history-dependent. Defense mechanisms based on both these types of equilibria have been proposed.

### 3.1 Open-Loop Nash Equilibrium

In an open-loop Nash equilibrium, the defender plays a stage-game equilibrium in every period. Such a Nash equilibrium may not give the highest payoff to the players. Owing to the imperfect observability in the game of the client-puzzle approach, open-loop strategies are desirable as the players need not know the actions taken previously by their opponents. One desirable open-loop Nash equilibrium is where the attacker chooses the action *TA* in every period. The conditions for such an equilibrium are given in Theorem 1.

**Theorem 1** *In the game of the client-puzzle approach, assume that the defender uses n puzzles* $P_1$, $P_2$, ..., $P_n$ *such that* $\alpha_{SP_1} < ... < \alpha_{SP_k} < \alpha_m < \alpha_{SP_{k+1}} < ... < \alpha_{SP_n}$. *Also, assume* $\alpha_{VP_i} = \alpha_{VP}$ *and* $\alpha_{PP_i} = \alpha_{PP}$ *for* $1 \leq i \leq n$. *Then, for* $0 < \eta < \frac{1}{2}$, *a stage-game Nash equilibrium of the form* $(p \circ P_k \oplus (1 - p) \circ P_{k+1}; TA)^\star$, $0 < p < 1$, *exists if* $\eta = \frac{\alpha_m}{\alpha_m + \alpha_{SP_{k+1}} - \alpha_{SP_k}}$, $\alpha_{SP_{k+1}} - \alpha_{SP_k} > \alpha_m$ *and* $\frac{\alpha_{SP_k}}{\alpha_m} < p < 1$.

*Proof.* Let us prove the existence of a Nash equilibrium, where the defender uses a mixed strategy $\alpha_1 = p_1 \circ P_1 \oplus p_2 \circ P_2 \oplus .. \oplus p_n \circ P_n$, where $p_1 + p_2 + ... + p_{n=1}$ and the attacker uses the pure strategy $TA$. The profile $(\alpha_1; TA)$ is a Nash equilibrium if

$$u_1(P_i; TA) = u_1(P_j; TA) > u_1(P_l; TA), \tag{1}$$

where $i, j \in \{h | 1 \leq h \leq n, p_h \neq 0\}$, $l \in \{h | 1 \leq h \leq n, p_h = 0\}$ and

$$u_2(\alpha_1; TA) > u_2(\alpha_1; a_2), \tag{2}$$

where $a_2 \in \{QT, RA, TQ, CA\}$. Note that $u_1(P_i; TA) = (1 - \eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_i}) - \eta\alpha_{SP_i}$ if $1 \leq i \leq k$ and $u_1(P_i; TA) = (1 - \eta)(-\alpha_{PP} - \alpha_{VP} + \alpha_{SP_k}) - \eta\alpha_{SP_i}$ if $k + 1 \leq i \leq n$.

Let $I_L = \{h | 1 \leq h \leq k, p_h \neq 0\}$ and $I_H = \{h | k + 1 \leq h \leq n, p_h \neq 0\}$. When $\eta = 0$, (1) is satisfied if $|I_L| = 0$ and $|I_H| \geq 1$, where $|X|$ is the cardinality of set $X$. When $\eta = \frac{1}{2}$, (1) is satisfied if $|I_L| \geq 1$ and $|I_H| = 0$. In both cases, (2) is not satisfied. If $|I_L| = |I_H| = 1$, the corresponding Nash equilibrium is of the form

$^\star$ 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 $p_1 + p_2 + ... + p_n = 1$.

$(p \circ P_i \oplus (1-p) \circ P_j; TA)$, where $i \in I_L$ and $j \in I_H$. Here, (1) is satisfied when,

$$\eta = \frac{\alpha_m}{\alpha_m + \alpha_{SP_j} - \alpha_{SP_i}} \tag{3}$$

and (2) is satisfied when,

$$\alpha_{SP_j} - \alpha_{SP_i} > \alpha_m \tag{4}$$

and $p > \frac{\alpha_{SP_i}}{\alpha_m}$. From (3) and (4), it can be easily seen that the maximum value that $\eta$ can take is less than $\frac{1}{2}$. Hence, $0 < \eta < \frac{1}{2}$.

We claim that $i = k$ and $j = k+1$. On the contrary, if $i < k$, then $u_1(P_i; TA) > u_1(P_k; TA)$, which means $(1-\eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_i}) - \eta\alpha_{SP_i} > (1-\eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_k}) - \eta\alpha_{SP_k}$ or $(1-2\eta)\alpha_{SP_i} > (1-2\eta)\alpha_{SP_k}$. Since $\eta < \frac{1}{2}$, the inequality reduces to $\alpha_{SP_i} > \alpha_{SP_k}$, which is a contradiction. Hence, $i = k$. Similarly, it can be shown that $j = k+1$.

According to Theorem 1, two levels of puzzle difficulty is sufficient for the defense mechanism. Consider two puzzles $P_1$ and $P_2$ such that $\alpha_{SP_1} < \alpha_m < \alpha_{SP_2}$. A stage-game Nash equilibrium is then possible if the hypothesis in Theorem 1 is satisfied. However, the prescribed equilibrium strategy does not necessarily prevent a denial of service attack.

Let $N$ be the maximum number of requests that an attacker can send in time $T$. By our assumption, puzzle generation and verification does not lead to denial of service. Hence, $N(\alpha_{PP} + \alpha_{VP}) \leq 1$. Let $\beta$ be the probability with which the attacker solves a given puzzle. Clearly, $N\beta$ is the expected number of attack requests for which the defender provides service. An attacker cannot mount a successful denial of service attack if $N\beta\alpha_m \leq 1$ or $\beta \leq \frac{1}{N\alpha_m}$. In the prescribed Nash equilibrium $\beta = p$. Hence, the condition $\frac{\alpha_{SP_1}}{\alpha_m} < p < \frac{1}{N\alpha_m}$ must hold. This is possible only if $\alpha_{SP_1} < \frac{1}{N}$.

We now propose a defense mechanism based on the open-loop Nash equilibrium:

1. For a desirable level of quality of service $\eta$, $0 < \eta < \frac{1}{2}$, 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 **x** such that the probability $Pr(\mathbf{x}{=}0) = p$ and $Pr(\mathbf{x}{=}1) = 1 - p$. If **x**=0, send puzzle $P_1$. Otherwise, send puzzle $P_2$.

It has to be noted that out of the total number of requests that the defender can service, $\frac{1}{\alpha_m}$ is the number of requests allocated to the defense mechanism, while the rest are for the legitimate users.

The maximum average payoff that the defender can expect in the prescribed equilibrium is $(1 - \eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_1}) - \eta\alpha_{SP_1}$. The attacker, on the other hand, receives an average payoff of

$$y_1 = \alpha_{PP} + \alpha_{VP} + p\alpha_m - \alpha_{SP_1}. \tag{5}$$

Adopting our modeling of the players' payoffs to Fallah's extensive-form game [6], for $0 < \eta < \frac{1}{2}$ and $0 < p < 1$, a stage-game equilibrium of the form $(p \circ P_1 \oplus (1 - p) \circ P_2; (CA,RA))$ exists if $\alpha_{SP_1} < \alpha_m < \alpha_{SP_2}$, $\alpha_{VP} < \alpha_m - \alpha_{SP_1}$, $\alpha_{VP} < \alpha_{SP_2} - \alpha_m$ and $\eta = \frac{\alpha_m - \alpha_{SP_1}}{\alpha_m - 2\alpha_{SP_1} + \alpha_{SP_2}}$. In this equilibrium, the maximum average payoff for the defender is $(1 - \eta)(-\alpha_{PP} - \alpha_{VP} - \alpha_m + \alpha_{SP_1}) - \eta\alpha_{SP_1}$ and that for the attacker is

$$y_2 = \alpha_{PP} + \alpha_{VP} + p(\alpha_m - \alpha_{SP_1}). \tag{6}$$

We now compare the effectiveness of our defense mechanism with that of the open-loop solution given by Fallah [6]. For a given $\eta$, we choose the same values for $\alpha_{PP}$, $\alpha_{VP}$, $\alpha_{SP_1}$ and $p$ and different values for $\alpha_{SP_2}$ in the two games such that equilibrium conditions are satisfied. It is evident that the average payoff for the defender is the same in both the defense mechanisms, but the attacker's maximum average payoff is lower in our solution (from (5) and (6)). Hence, our open-loop solution is more effective than the one given in [6].

For other possible stage-game Nash equilibria in the game of the client-puzzle approach, refer Appendix B.

### 3.2 Closed-Loop Nash Equilibrium

We now investigate the existence of an equilibrium that gives the defender higher payoff than the open-loop solution. High payoffs are possible in an infinitely repeated game if the players are sufficiently patient and take decisions based on the corresponding history. We now define the minmax point $(v_1^*, v_2^*)$. For each player $i$,

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

where $\Delta(X)$ is the set of probability distributions over $X$. 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 $U = \{(v_1, v_2) | \exists a \in A$ with $v_1 = u_1(\mathrm{a})$ and $v_2 = u_2(\mathrm{a})\}$, $V$ be the convex hull of $U$ and $V^* = \{(v_1, v_2) \in V | v_1 > v_1^*$ and $v_2 > v_2^*\}$. Here, $V$ is the set of

feasible payoffs and $V^*$ is the set of strictly individual rational payoffs (SIRP). The existence of a perfect Nash equilibrium that supports a payoff vector in $V^*$ is given by Theorem 2.

**Theorem 2** ***(By Fudenberg and Maskin [8])*** *For any ($v_1$, $v_2$) $\in V^*$, there exists $\delta_0 \in$ (0, 1) such that for all $\delta \in (\delta_0, 1)$, there exists a subgame perfect equilibrium of the infinitely repeated game in which player $i$'s average payoff is $v_i$ when the players have the discount factor $\delta$.*

Let $\bar{v_i} = \max_{\alpha_1 \in \Delta(A_1), \alpha_2 \in \Delta(A_2)} u_i(\alpha_1; \alpha_2)$ and $\bar{u_i} = u_i(M_1^2, M_2^1)$. For ($v_1$, $v_2$) $\in V^*$, choose a value for $\delta_0 \in (0, 1)$ such that for each player $i$,

$$\delta_0 > \frac{\bar{v_i} - v_i}{\bar{v_i} - v_i^*} \tag{7}$$

and a value for $\tau_0$ such that

$$\frac{v_i^* - \bar{u_i}}{v_i - \bar{u_i}} < \delta_0^{\tau_0} < \frac{v_i - \bar{v_i} + \delta_0(\bar{v_i} - \bar{u_i})}{\delta_0(v_i - \bar{u_i})}, \tag{8}$$

subject to $\tau_0 > 0$. Clearly, for any $\delta > \delta_0$, there exists a corresponding $\tau(\delta)$ such that (7) and (8) hold for $(\delta, \tau(\delta))$ [6]. (Refer Appendix C.1.)

Let $(\alpha_1; \alpha_2)$ be a correlated one-shot strategy profile corresponding to ($v_1$, $v_2$), i.e, $u_i(\alpha_1; \alpha_2) = v_i$ for $i = 1, 2$. The following repeated game strategies for player $i$ is a perfect Nash equilibrium, where the desired payoff is achieved through the threat of punishment.

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

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

The following theorem identifies the minmax strategies in the game of the client-puzzle approach.

**Theorem 3** *In the game of the client-puzzle approach, assume the defender uses two puzzles $P_1$ and $P_2$ such that $\alpha_{SP_1} < \alpha_m < \alpha_{SP_2}$, $\alpha_{VP_1} = \alpha_{VP_2} = \alpha_{VP}$ and $\alpha_{PP_1} = \alpha_{PP_2} = \alpha_{PP}$. Then, for $0 < \eta < 1$, the defender's minmax strategy against the attacker is $p_1 \circ P_1 \oplus (1-p_1) \circ P_2$, where $0 \leq p_1 \leq \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}} < 1$ when $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$ and $0 \leq p_1 \leq \frac{\alpha_{SP_1}}{\alpha_m} < 1$ when $\alpha_{SP_2} - \alpha_{SP_1} \geq \alpha_m$ and for $0 < p_2 < 1$, the attacker's minmax strategy against the defender is (i) $p_2 \circ CA \oplus (1-p_2) \circ RA$ when $\alpha_{SP_2} - \alpha_{SP_1} \leq \alpha_m$, $\eta < \frac{1}{2}$ and $p_2 = \frac{\eta}{1-\eta}$, (ii) $p_2 \circ TA \oplus (1-p_2) \circ RA$ when $\alpha_{SP_2} - \alpha_{SP_1} \geq \alpha_m$, $\eta < \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$ and $p_2 = \left(\frac{\eta}{1-\eta}\right)\left(\frac{\alpha_{SP_2} - \alpha_{SP_1}}{\alpha_m}\right)$, (iii) $CA$ when $\eta \geq \frac{1}{2}$ and (iv) $TA$ when $\eta \geq \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$.*

*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. Assume $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$. Clearly, the attacker would prefer $CA$ over $TA$ and $TQ$ and $RA$ over $QT$. 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 can be enforced by the defender using the mixed strategy $p_1 \circ P_1 \oplus (1-p_1) \circ P_2$, where $0 \leq p_1 \leq \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}} < 1$. Similarly, when $\alpha_{SP_2} - \alpha_{SP_1} \geq \alpha_m$, it can be shown that the minmax strategy of the defender against the attacker is $p_1 \circ P_1 \oplus (1-p_1) \circ P_2$, where $0 \leq p_1 \leq \frac{\alpha_{SP_1}}{\alpha_m} < 1$.

On the other hand, let the attacker's minmax strategy against the defender be $\alpha_2 = q_1 \circ QT \oplus q_2 \circ RA \oplus q_3 \circ TA \oplus q_4 \circ TQ \oplus q_5 \circ CA$, where $q_1 + q_2 + q_3 + q_4 + q_5 = 1$.

Assume $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$. When the attacker chooses $CA$, the defender would receive equal or lower payoff than when the attacker chooses $TA$ or $TQ$. Also, $RA$ would give the defender lower payoff than $QT$. Hence, the attacker's minmax strategy against the defender should assign non-zero probabilities to $CA$ and $RA$ and zero probability to the rest, i.e, $q_1 = q_3 = q_4 = 0$, $q_2 = p_2$ and $q_5 = 1 - p_2$, $0 < p_2 < 1$. When $0 < \eta < \frac{1}{2}$, the defender's best response for the attacker's pure strategy $RA$ is $P_1$ and for $CA$ is $P_2$. For the attacker's mixed strategy $\alpha_2$, 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 $p_2 = \frac{\eta}{1-\eta}$. It can be shown that the attacker's minmax strategy against the defender is the same when $\alpha_{SP_2} - \alpha_{SP_1} = \alpha_m$. Hence the proof for case (i). A similar argument can be used to prove the case (ii).

We now give a proof for case (iii). Assume $\eta > \frac{1}{2}$. The defender would then prefer action $P_1$ over action $P_2$ when the attacker chooses $CA$. If $\alpha_{SP_2} - \alpha_{SP_1} > \alpha_m, \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}} < 1$ and hence, $\eta > \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$. The defender's best response to the attacker's action $TA$ would then be $P_1$. $P_1$ is also the defender's best response to the attacker's action $RA$. Note that the attacker's actions $RA$ and $TA$ would give the defender lower payoff than the actions $QT$ and $TQ$ respectively. Since $u_1(P_1; CA) = u_1(P_1; TA)$ and $u_1(P_1; CA) < u_1(P_1; RA)$, the defender is minmaxed when the attacker's action is $CA$. If $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$, the defender's best response to the attacker's action $TA$ would either be $P_1$ or $P_2$ depending on the chosen value of $\eta$. In either case, it can be seen that $CA$ would minmax the defender. Also, it can be shown that $CA$ is a minmax strategy of the attacker against the defender when $\eta = \frac{1}{2}$. Hence the proof for case (iii). Case (iv) can be proved in a similar manner.

The defender's minmax strategy against the attacker is either the pure strategy $P_2$ or a lottery over $P_1$ and $P_2$. The latter is more beneficial for a legitimate user as he has a chance of getting an easy puzzle. Under the condition $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$, the maximum probability with which the defender can send

$P_1$ and still minmax the attacker is $\frac{\alpha_{SP_2}-\alpha_m}{\alpha_{SP_2}-\alpha_{SP_1}}$. When the defender adopts such a strategy, the attacker receives the same payoff for both $CA$ and $RA$, but a lower payoff for other actions. The defender's payoff, on the other hand, is the same whether the attacker chooses $CA$ or $RA$.

Consider the attacker's minmax strategy against the defender $p_2 \circ CA \oplus (1-p_2) \circ RA$, where $p_2 = \frac{\eta}{1-\eta}$. Then, the strategy profile $\alpha^1 = (p \circ P_1 \oplus (1-p) \circ P_2; TA)$ corresponds to a strictly individual rational payoff if

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

It can be easily shown that a value of $0 < p < 1$ that satisfies the given inequality exists under the conditions stated in Theorem 3. Moreover, in order to prevent flooding, the condition $p < \frac{1}{N\alpha_m}$ must also hold.

As described earlier, through the threat of punishment, the desired Nash equilibrium can be achieved. In phase (A), the strategy profile $\alpha^1$ is played until a deviation occurs, after which the strategy profile $\alpha^2 = (p_1 \circ P_1 \oplus (1-p_1) \circ P_2; p_2 \circ CA \oplus (1-p_2) \circ RA)$ is played for a period $\tau(\delta)$ given by (8). It has to be noted that the only deviation that gives a momentary gain for the attacker in phase (A) is playing $CA$. In phase (B), the best response for the attacker is either $CA$ or $RA$ or a lottery on both. Hence, if the attacker deviates in this phase, his payoff either decreases or remains the same. On the other hand, his deviation in phase (B) would either increase the defender's payoff or keep it constant. Also, note that the strategy profile $\alpha^2$ is a mixed strategy Nash equilibrium in which both players receive their minmax payoff.

For the sake of simplicity, the only deviation that we consider in phase (A) is when a correct answer is received for puzzle $P_2$. Since an attacker will not profit by deviating in phase (B), deviations are not considered in this phase. On the other hand, if deviations were considered in phase (B), a legitimate user's action may unnecessarily extend this phase as he always solves the given puzzle.

An important assumption that Theorem 2 makes is that a player can observe his opponent's past mixed strategies. This is possible if the outcomes of the players' randomizing devices are jointly observable *ex-post* [8]. Moreover, submission of partial solutions to puzzles must be allowed, so that the defender observes the attacker's action $TA$. In section 4, we propose puzzles that satisfy this requirement.

The defense mechanism based on the closed-loop solution is given below.

1. For a given desirable level of quality of service $\eta$, $0 < \eta < \frac{1}{2}$, choose two puzzles $P_1$ and $P_2$ such that

$$\alpha_{SP_1} < \frac{1}{N} < \alpha_m < \alpha_{SP_2} \tag{9}$$

$$\text{and } \alpha_{SP_2} - \alpha_{SP_1} < \alpha_m. \tag{10}$$

2. Choose a value for $p$ such that

$$\frac{\alpha_{SP_1}}{\alpha_m} < p < min\left(\frac{1}{N\alpha_m}, \frac{\alpha_{SP_1} - \eta(\alpha_{SP_2} - \alpha_m + \alpha_{SP_1})}{\alpha_m - \eta(\alpha_{SP_2} + \alpha_m - \alpha_{SP_1})}\right) \tag{11}$$

and determine the value of $p_1$ according to

$$p_1 = \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}. \tag{12}$$

3. Determine the value of $\delta_0$ satisfying (7). Choose a value of $\delta > \delta_0$ and the corresponding value of $\tau(\delta)$ such that (8) is satisfied for $(\delta, \tau(\delta))$.
4. Phase (A) and phase (B) of the defense mechanism have been described in Fig. 1.

The defender may receive a new request before the attacker gives his response for the previous request. Hence, the defender will have to choose a puzzle without knowing the attacker's previous action. Fallah [6] says that the attacker's response to a simple puzzle is known to the defender very soon and this allows him to compensate for the wrong decisions.

Let $u_1'$ be the defender's payoff function in the stage-game described earlier and $u_1''$ be his payoff function in the repeated game. In order to compare the two payoffs, we keep $\alpha_{PP}$, $\alpha_{VP}$, $\alpha_{SP_1}$ and $p$ same in both games and use different values for $\alpha_{SP_2}$ such that the equilibrium criteria are satisfied. It can be easily seen that $u_1'(P_1; TA) = u_1'(P_2; TA) = u_1''(P_1; TA) < u_1''(P_2; TA)$. The maximum average payoff $p(u_1''(P_1; TA)) + (1-p)(u_1''(P_2; TA))$, $0 < p < 1$, that the defender receives in phase (A) is definitely greater than his payoff in the open-loop solution. Similarly, it can be shown that the defender's minmax payoff in our repeated game is greater than the minmax payoff in Fallah's repeated game, i.e.,

$$(1-\eta)(-\alpha_{PP} - \alpha_{VP}) - \eta\alpha_m > (1-\eta)(-\alpha_{PP} - \alpha_{VP}) - \eta\alpha_{SP_2}.$$

In both these games, the minmax payoff serves as a lower bound on the defender's payoff and hence, our closed-loop solution is better than the one in [6]. Moreover, in our defense mechanism, a legitimate user is hurt less in the punishment phase as he has a chance of receiving $P_1$.

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

In the repeated game, a strategy profile that corresponds to a high SIRP for the defender is $(P_1; p \circ CA \oplus (1-p) \circ RA)$, $0 < p < 1$, which has been proposed as a closed-loop solution in [6]. (Refer Appendix C.2.) We feel that this strategy profile is less intuitive than the one proposed as it requires the attacker to make decisions based on the outcome of a public randomizing device.

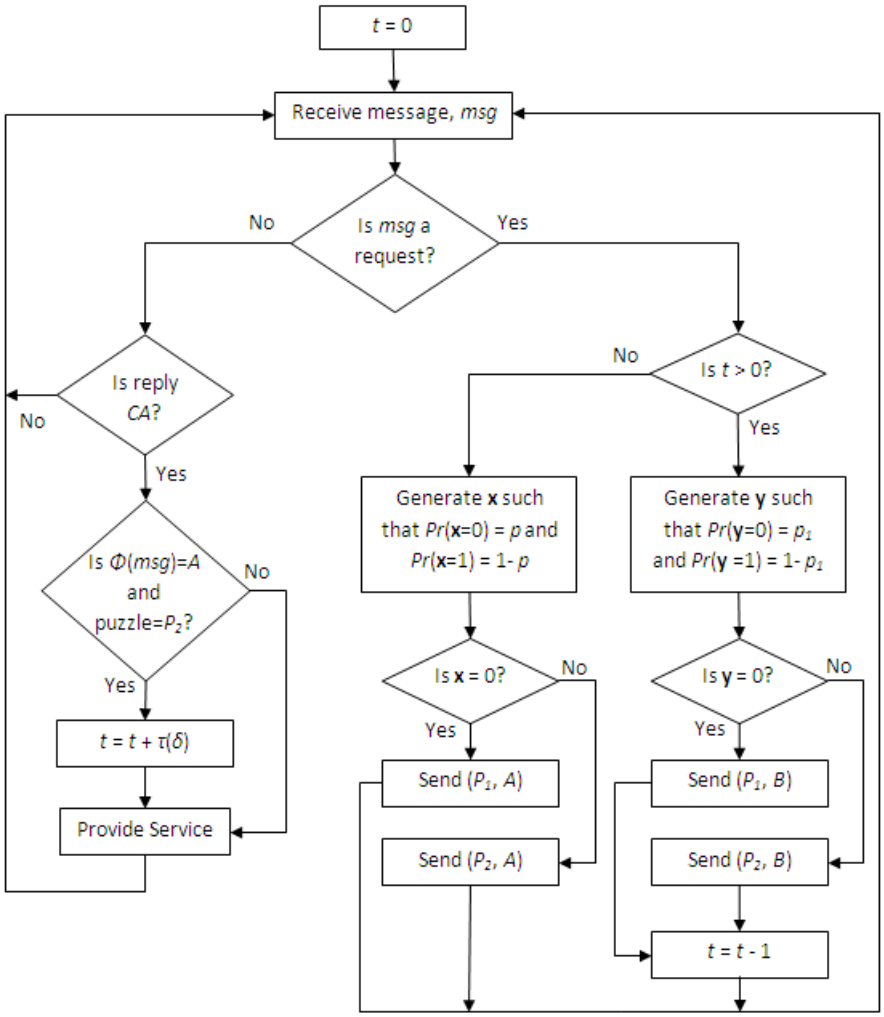


**Fig. 1. Closed-Loop Defense Mechanism.** Two random variables **x** and **y** are used, where **x** determines which puzzle has to be sent in phase (A) and **y** determines the puzzle to be sent in phase (B). $\phi(msg)$ is the phase corresponding to the puzzle, whose solution has been received.

### 3.3 Implementation Issues

In the defense mechanisms proposed, the defender must be capable of servicing at least $\frac{2}{\alpha_m}$ requests in time $T$. Out of this, $\frac{1}{\alpha_m}$ requests are allocated to the defense mechanism and the remaining are for the legitimate user. Even if all the $P_1$ puzzles go to the attacker, the defender can service up to $\frac{1}{\alpha_m}$ legitimate requests.

Consider the closed-loop defense mechanism described earlier. If an attacker chooses to correctly answer a $P_2$ puzzle in phase (A), the corresponding deviation is discerned by the defender only when he receives the solution after a time period $T\alpha_{SP_2}$. In the mean time, the attacker can correctly answer as many puzzles as possible without immediate punishment. Eventually, the attacker will be subjected to $\tau(\delta)$ periods of the punishment phase (B) for each deviation. The maximum number of $P_2$ puzzles that the attacker can solve before the transition to phase (B) is $\frac{1-p}{\alpha_{SP_2}}$. Here, $(1-p)$ is the probability with which defender sends puzzle $P_2$ in phase (A). To ensure that the attacker does not overwhelm the defender through his deviations in phase (A), the defender must be able to provide service for an additional $\frac{1-p}{\alpha_{SP_2}}$ attack requests in time $T$. On the whole, in the closed-loop defense mechanism, the defender must be capable of servicing up to $\frac{2}{\alpha_m} + \frac{1-p}{\alpha_{SP_2}}$ requests in time $T$.

A legitimate user always solves the given puzzle. When he solves $P_2$ in phase (A), the defender would consider it a deviation and switch to phase (B). Hence, a legitimate user is unnecessarily hurt in the absence of an attack. This problem can be avoided by making use of the defense mechanism only when the time elapsed since the last request is lesser than a certain value. Let $t_1$ be the time interval between the current request and the previous request. Consider the following implementation.

1. If $t_1 \geq T\alpha_m$, send $P_1$.
2. Otherwise, send a puzzle according to the defense mechanism.

We now show that a denial of service attack will not be successful in the proposed implementation. Assume that in time $T$, the attacker sends $n_1$ requests separated by a time interval greater than $T\alpha_m$. Let $N_1$ be the remaining number of requests that he can send in $T$. Clearly,

$$N_1 \leq N\left(\frac{T - n_1 T\alpha_m}{T}\right) = N(1 - n_1\alpha_m). \tag{13}$$

On the whole, the attacker is expected to receive $P_1$ puzzles for $n_1 + pN_1$ requests, where the maximum value of $p$ is $\frac{1}{N\alpha_m}$. The denial of service attack is successful only if $(n_1 + pN_1)\alpha_m > 1$, which from (13) is not possible for $0 \leq n_1 \leq \frac{1}{\alpha_m}$.

Assume in the absence of an attack, requests arrive according to a Poisson process with parameter $\lambda$. Since the service rate must be greater than the rate at which the requests arrive, $\lambda \leq \frac{1}{T\alpha_m}$. Then,

$$\tilde{p} = Pr(t_1 \geq T\alpha_m) = e^{-\lambda T\alpha_m} \geq \frac{1}{e} = 0.3679.$$

The probability with which a client gets puzzle $P_1$ in the absence of an attack is given by $\tilde{p} + (1 - \tilde{p})p \geq 0.3679 + 0.6321p \geq 0.3679$ for $0 \leq p \leq 1$. This shows that at least 36.79% of the clients' requests receive $P_1$ if there is no attack.

The game theoretic defense mechanisms proposed are based on the assumption that both players are rational. However, from the implementation point of view, the defender is a piece of software that runs on the server and always follows the prescribed strategy. Therefore, the attacker will never have to punish the defender as the defender would never deviate. Hence, the outcomes of the defender's randomizing device need not be revealed to the attacker. Also, it is sufficient for the punishment period to be long enough to make the attacker's deviation from phase (A) unprofitable. In other words, while choosing $\delta_0$, $\delta$ and $\tau(\delta)$, it is sufficient to satisfy (7) and (8) for the attacker ($i = 2$) alone.

### 3.4 Extension to Distributed Attacks

When the attacker carries out the attack from more than one machine, his computational power proportionally increases. Let $s$ be the number of machines in the attack coalition. The attacker can then send $N$ requests from a single machine and $sN$ request from the attack coalition in time $T$. The number of $P_1$ puzzles that he can solve in time $T$ is $\frac{s}{\alpha_{SP_1}}$ and the number of $P_2$ puzzles that he can solve in the same time is $\frac{s}{\alpha_{SP_2}}$.

We now modify the conditions for the open-loop defense mechanism to handle a distributed attack.

$$\frac{\alpha_{SP_1}}{s} < \frac{1}{N} < \alpha_m < \frac{\alpha_{SP_2}}{s},$$

$$\alpha_{SP_2} - \alpha_{SP_1} > s\alpha_m,$$

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

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

In a distributed attack, the defender receives a large number of requests before knowing the attacker's decision. This delays the transition from phase (A) to phase (B) in the closed-loop defense mechanism. To prevent a successful DoS attack, the defender must be capable of providing service for up to $\frac{2}{\alpha_m} + \frac{s(1-p)}{\alpha_{SP_2}}$ requests in time $T$.

The conditions for the closed-loop defense mechanism to handle distributed attacks are:

$$\frac{\alpha_{SP_1}}{s} < \frac{1}{N} < \alpha_m < \frac{\alpha_{SP_2}}{s},$$

$$\alpha_{SP_2} - \alpha_{SP_1} < s\alpha_m \text{ and}$$

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

Note that the size of the attack coalition is unknown to the defender. In [6], a fair learning process is used to estimate the size of the attack coalition. The same approach can be adopted in our defense mechanisms.

### 3.5 Considerations for Random Guesses

The defense mechanisms that we have proposed till now and the results in previous works [2, 6] are based on the assumption that the attacker cannot solve a puzzle by giving a random answer. However, the attacker's random guess may sometimes end up as the correct answer. Taking this into account, we make some changes in the game model. We make use of hint-based hash reversal puzzles [7] in the analysis.

In a hint-based hash reversal puzzle, the defender generates a preimage $X$ and computes $Y = H(X)$, where $H$ is a cryptographic hash function. He also computes the hint, $a = X - b$, where $b$ is chosen uniformly at random from $\{1, ..., D\}$. $Y$ and $a$ are then sent to the client. To solve the puzzle, the client searches for a value $1 \leq c \leq D$ such that $H(a + c) = H(X)$. The difficulty level of the puzzle is given by parameter $D$. Clearly, the value of $Y$ and $a$ do not reveal the puzzle difficulty to the attacker.

Assume the defender uses two hint-based hash reversal puzzles $P_1$ and $P_2$ with difficulty parameters $D_1$ and $D_2$ respectively, where $D_2 > D_1$. The value of $b$ for $P_1$ is chosen uniformly at random from $\{1, ..., D_1\}$ and that for $P_2$ is chosen from $\{1, ..., D_2\}$. The actions that the attacker can take are $\{S_0, S_1, ..., S_{D_1}, ..., S_{D_2-1}\}$, where $S_k$, $0 \leq k < D_2$, indicates that the attacker gives an answer after at most $k$ hash computations. Note that the attacker requires at most $D_2 - 1$ hash computations to solve $P_2$ as there will be only one possible solution left after failure in $D_2 - 1$ attempts.

In order to solve a puzzle of difficulty $D$, the attacker will have to search among $D$ possible solutions. Each such solution in the search space has a probability $\frac{1}{D}$ of being correct. Note that the attacker is unaware whether the puzzle difficulty is $D_1$ or $D_2$. When the attacker takes action $S_k$, $0 \leq k < D_2$, he tries out at most $k$ possible solutions and if the puzzle is still not solved, he chooses an answer uniformly at random from the remaining $D_2 - k$ possible solutions. If the puzzle sent was $P_1$, the attacker would have solved the puzzle with a maximum of $D_1$ hash computations. The probability that the attacker solves $P_1$ with $0 \leq k \leq D_1$ hash computations is $\frac{k}{D_1}$ and his random guess from the remaining possible solutions would be correct with a probability $\frac{1}{D_1}\frac{D_1-k}{D_2-k}$. On the other hand, the probability that the attacker solves $P_2$ with $0 \leq k < D_2$ hash computations is $\frac{k}{D_2}$ and his random guess from the remaining possible solutions would be correct with a probability $\frac{1}{D_2}$. Let $p_k(i)$ be the probability that the attacker gives a correct answer for the puzzle $P_i$, $i = 1, 2$, by choosing action $S_k$. Then, the value of $p_k(1)$ is $\frac{1}{D_1}\frac{D_1-k}{D_2-k} + \frac{k}{D_1}$ if $0 \leq k \leq D_1$ and 1 otherwise and $p_k(2) = \frac{k+1}{D_2}$ for $0 \leq k < D_2$.

Let $N_h$ be the total number of hashes that the attacker can compute in time $T$. Let $\alpha_{S_k}(i)$ be the average fraction of time $T$ that the attacker spends in

attempting to solve the puzzle when his action is $S_k$ and the puzzle received is $P_i$. Note that $\alpha_{S_0}(i) = 0$. Assume the defender chooses puzzle $P_1$. If the attacker chooses $S_k$, $1 \leq k \leq D_1$, he would compute $1 \leq j < k$ hashes with a probability $\frac{1}{D_1}$ each and $k$ hashes with a probability $1 - \frac{k-1}{D_1}$. For the action $S_k$, $D_1 < k < D_2$, the attacker would compute $1 \leq j \leq D_1$ hashes, each with a probability $\frac{1}{D_1}$. The average number of hash computations for the action $S_k$ is

$$\sum_{j=1}^{k-1} j\left(\frac{1}{D_1}\right) + k\left(1 - \frac{k-1}{D_1}\right) = \frac{k(2D_1 - k + 1)}{2D_1}$$

when $1 \leq k \leq D_1$ and

$$\sum_{j=1}^{D_1} j\left(\frac{1}{D_1}\right) = \frac{D_1 + 1}{2}$$

when $D_1 < k < D_2$. Hence, for $1 \leq k \leq D_1$, $\alpha_{S_k}(1) = \frac{k(2D_1 - k + 1)}{2N_h D_1}$ and for $D_1 < k < D_2$, $\alpha_{S_k}(1) = \frac{D_1 + 1}{2N_h}$. Similarly, for $1 \leq k < D_2$, the average number of hash computations for the action $S_k$ is

$$\sum_{j=1}^{k-1} j\left(\frac{1}{D_2}\right) + k\left(1 - \frac{k-1}{D_2}\right) = \frac{k(2D_2 - k + 1)}{2D_2}.$$

Hence, $\alpha_{S_k}(2) = \frac{k(2D_2 - k + 1)}{2N_h D_2}$ for $1 \leq k < D_2$. Note that a legitimate user always solves the puzzle and hence, the cost incurred by him for $P_1$ is $\alpha_{S_{D_1}}(1)$ and that for $P_2$ is $\alpha_{S_{D_2-1}}(2)$. In a game where the defender chooses $P_i$ and the attacker chooses $S_k$, the payoff for the attacker is $\alpha_{PP} + \alpha_{VP} + p_k(i)\alpha_m - \alpha_{S_k}(i)$ and that for the defender is $(1-\eta)(-\alpha_{PP} - \alpha_{VP} - p_k(i)\alpha_m + \alpha_{S_k}(i)) + \eta(-\alpha_u(i))$, where $\alpha_u(1) = \alpha_{S_{D_1}}(1)$ and $\alpha_u(2) = \alpha_{S_{D_2-1}}(2)$.

The following theorem states the conditions for a Nash equilibrium in such a stage-game.

**Theorem 4** *In the game of the client-puzzle approach, assume the defender uses two hint-based hash reversal puzzles $P_1$ and $P_2$ of difficulty levels $D_1$ and $D_2$ respectively such that $\alpha_{PP_1} = \alpha_{PP_2} = \alpha_{PP}$, $\alpha_{VP_1} = \alpha_{VP_2} = \alpha_{VP}$, $4 < D_2 < N_h$, $0 < D_1 < \frac{D_2}{2}$, $\frac{D_1+1}{2N_h} < \alpha_m < \frac{D_2+1}{2N_h}$ and $D_2 - D_1 > 2N_h\alpha_m$ and the attacker uses $n$ actions $S_1$, $S_2$, ..., $S_{D_1}$, ..., $S_{D_2-1}$. Then, for $0 < \eta < 1$, a stage-game Nash equilibrium of the form $(p \circ P_1 \oplus (1-p) \circ P_2; S_{D_1})$, $0 < p < 1$, exists if*

$$p > \frac{D_1(2D_2 - D_1 + 1) - 2N_h D_1 \alpha_m}{2(D_2 - D_1 - 1)N_h\alpha_m + (D_2 - D_1)(D_1 - 1)} \text{ and}$$

$$\eta = \frac{2N_h(D_2 - D_1 - 1)\alpha_m + (D_2 - D_1)(D_1 - 1)}{2N_h(D_2 - D_1 - 1)\alpha_m + (D_2 - D_1)(D_2 + D_1 - 1) - 2}.$$

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

pure strategy $S_k$, $0 \leq k < D_2$. The expected payoff of the attacker $U_2(\alpha_1; S_k)$ is $\alpha_{PP}+\alpha_{VP}+\big(\frac{1}{D_2}\big)\alpha_m$ if $k = 0$, $\alpha_{PP}+\alpha_{VP}+p\big[\big(\frac{1}{D_1}\frac{D_1-k}{D_2-k}+\frac{k}{D_1}\big)\alpha_m-\frac{k(2D_1-k+1)}{2N_hD_1}\big]+(1-p)\big[\frac{k+1}{D_2}\alpha_m-\frac{k(2D_2-k+1)}{2N_hD_2}\big]$ if $1 \leq k \leq D_1$ and $\alpha_{PP}+\alpha_{VP}+p\big(\alpha_m-\frac{D_1+1}{2N_h}\big)+(1-p)\big[\frac{k+1}{D_2}\alpha_m-\frac{k(2D_2-k+1)}{2N_hD_2}\big]$ if $D_1 < k < D_2$.

The attacker would prefer $S_{D_1}$ over $S_0$ when $U_2(\alpha_1; S_{D_1}) > U_2(\alpha_1; S_0)$, which is possible if

$$p > \frac{D_1(2D_2 - D_1 + 1) - 2N_hD_1\alpha_m}{2(D_2 - D_1 - 1)N_h\alpha_m + (D_2 - D_1)(D_1 - 1)}. \tag{14}$$

Consider the action set $S^1 = \{S_1, S_2, ..., S_{D_1}\}$. We claim that under the assumptions made, the second derivative $U_2^{''}(\alpha_1; S_k) = p\big(\frac{1}{D_1N_h}-\frac{2(D_2-D_1)\alpha_m}{D_1(D_2-k)^3}\big)+(1-p)\big(\frac{1}{D_2N_h}\big)$ is positive for $1 \leq k \leq D_1$ and $0 < p < 1$. On the contrary, if $U_2^{''}(\alpha_1; S_k) \leq 0$, then $\frac{1}{D_1N_h}-\frac{2(D_2-D_1)\alpha_m}{D_1(D_2-k)^3} < 0$. As $D_1 > 0$, $\frac{1}{N_h} < \frac{2(D_2-D_1)\alpha_m}{(D_2-k)^3}$. Considering the assumption, $\alpha_m < \frac{D_2+1}{2N_h}$, we have $\frac{(D_2-D_1)(D_2+1)\alpha_m}{(D_2-k)^3} > \alpha_m$. Since $\alpha_m > 0$, $(D_2 - D_1)(D_2 + 1) > (D_2 - k)^3$. When $k$ takes its highest value $D_1$, this inequality reduces to $D_2 + 1 > (D_2 - D_1)^2$, which cannot be satisfied as $D_1 < \frac{D_2}{2}$ and $D_2 > 4$. Clearly, for no value of $1 \leq k \leq D_1$, the inequality is satisfied. This is a contradiction and hence, $U_2^{''}(\alpha_1; S_k) > 0$. By the second derivative test, $U_2(\alpha_1; S_k)$ does not have a local maximum for $1 < k < D_1$. It can be shown that when (14) holds, $U_2(\alpha_1; S_{D_1}) > U_2(\alpha_1; S_1)$. Therefore, no action in $S^1$ gives a higher payoff for the attacker than $S_{D_1}$ (provided (14) holds). Similarly, under the assumption $D_2 - D_1 > 2N_h\alpha_m$, it can be shown that $S_{D_1+1}$ gives the highest payoff for the attacker in the action set $S^2 = \{S_{D_1+1}, ..., S_{D_2-1}\}$. Under the same assumption, it is evident that the attacker would prefer $S_{D_1}$ over $S_{D_1+1}$. We conclude that the attacker's best response to the defender's mixed strategy $\alpha_1$ is $S_{D_1}$.

On the other hand the defender does not gain by deviating from his mixed strategy $\alpha_1$ when the expected payoff $U_1(P_1; S_{D_1}) = U_1(P_2; S_{D_1})$, which implies

$$\eta = \frac{2N_h(D_2 - D_1 - 1)\alpha_m + (D_2 - D_1)(D_1 - 1)}{2N_h(D_2 - D_1 - 1)\alpha_m + (D_2 - D_1)(D_2 + D_1 - 1) - 2}. \tag{15}$$

Under the assumptions made, it can be seen that $\eta < 1$.

Using theorem 4, for a desirable level of quality of service $\eta$, an effective defense mechanism can be constructed by choosing appropriate values for $D_1$, $D_2$ and $p$.

## 4 Client-Puzzles

### 4.1 Puzzle Requirements

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.
3. **Partial Solution:** Submission of partial solutions should be possible without increasing the verification time.

The third requirement allows the defender to determine whether the attacker chose $RA$ or $TA$.

### 4.2 Analysis of Existing Client-Puzzles

We examine whether the existing puzzles satisfy the requirements mentioned above.

- **Hash-Reversal Puzzle:** Given a random number with first $n$ bits erased and its hash value, the client needs to reverse the hash [11]. This puzzle is not suitable for our defense mechanisms as the number of preceding 0 bits will indicate the puzzle difficulty. Further, the resolution of this puzzle is low as increasing $n$ by 1 increases the difficulty two fold. Though puzzle resolution can be increased by using multiple hash-reversal puzzles, the other requirements are not satisfied.
- **Time-Lock Puzzle:** In a time-lock puzzle [16], given $n$, $a$ and $t$, the client needs to compute $b = a^{(2^t)} \bmod n$. The puzzle guarantees that the client expends a fixed amount of computational resource. Here, the parameter $t$ reveals the difficulty of the puzzle. Also, puzzle generation is time consuming [7], leaving this puzzle unsuitable for the defense mechanism.
- **Hint-Based Hash-Reversal Puzzle:** Hint-based puzzles, described in Section 3.5, are the closest bid as they satisfy all the requirements except partial solution.

### 4.3 Design of New Client-Puzzles

We introduce three new puzzles that are designed to work with the proposed defense mechanisms.

**Puzzle 1:** We propose a modification to the hash-reversal puzzle [11] in order to hide the puzzle difficulty. We achieve this by providing a preimage with some of the first $k$ bits inverted. The puzzle generation and verification is detailed in Fig. 2.

Note that the difficulty parameter $k$ is unknown to the client. Hence, he would carry out a brute-force search and arrive at the solution after testing up to $2^k$ possible preimages. On an average the client computes $\frac{(2^k+1)}{2}$ hash computations to solve the puzzle. Puzzle generation takes 2 hash computations and the verification takes 3 hash computations. This puzzle does not allow partial solutions to be submitted.

**Fig. 2. Puzzle 1 Description**. Here, $H$ is a cryptographic hash function, $S$ is a server secret, $N_s$ and $N_c$ are nonces generated by the server and client respectively, $k$ is the difficulty parameter, $I$ is a binary number that is used to randomly invert some of the first $k$ bits of $X$ and $par_1$, $par_2$, etc. are parameters that are specific to the session or protocol. The defender uses the server nonce to check whether the solution received is for a recently sent puzzle.

| **Client** | | **Defender** |
|---|---|---|
| | $\xrightarrow{Request, N_c}$ | $X = H(S, N_s, N_c, par_1, par_2, ...)$<br>$Y = H(X)$ |
| | $\xleftarrow{(X', Y), N_s, par_1, par_2, ...}$ | $X' = X \oplus (I_1, I_2, ..., I_{k-1}, 1, 0_{k+1}, ..., 0_n)$ |
| Find $p$ such that<br>$H(p) = Y$ | $\xrightarrow{p, N_s, N_c, par_1, par_2, ...}$ | $X = H(S, N_s, N_c, par_1, par_2, ...)$<br>$H(p) \stackrel{?}{=} H(X)$ |

**Puzzle 2:** The second puzzle proposed uses a combination of the modified hash-reversal puzzle and hint-based puzzle. It is described in Fig. 3.

Here, the client can submit a partial solution by giving the correct answer for the first part ($p$) of the puzzle and a random answer for the second part ($q$). It takes 4 hash computations for puzzle generation 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 when he receives the puzzle. It has to be noted that the occurrence of a collision, while solving the first part of the puzzle, may lead to an unprecedented change in the puzzle difficulty. However, if the hash function used is collision resistant, the probability of such a collision is negligible.

**Puzzle 3:** We propose a puzzle, where the effect of hash collision on puzzle difficulty is negligible. Refer Fig. 4 for 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 $D_b$. Also, production of the puzzle requires 4 hash computations and the verification requires a maximum of 6 hash computations.

If a collision occurs when solving the first part of the puzzle, the client would have found $X_2 < X'$ such that $H(X_2) = Z$. Following this, the client will solve the second part with $X - X_2$ hash computations. Further, while solving the second part, the occurrence of collision does not increase the puzzle difficulty. Thus, hash collisions have a marginal effect on the puzzle difficulty.

**Fig. 3. Puzzle 2 Description**. Here, $S_1$ and $S_2$ are server secrets, $D$ and $k$ are difficulty parameters and $l$ is a constant.

| Client | | Defender |
|---|---|---|
| | $\xrightarrow{Request, N_c}$ | $X = H(S_1, N_s, N_c, par_1, par_2, ...)$<br>$Y = H(X)$<br>$a = H(S_2, N_s, N_c) \bmod D + l$<br>$X' = X - a$<br>$Z = H(X')$ |
| | $\xleftarrow{(X'', Y, Z), N_s, par_1, par_2, ...}$ | $X'' = X' \oplus (I_1, ..., I_{k-1}, 1, 0_{k+1}, ..., 0_n)$ |
| Find $p$ such that $H(p) = Z$.<br>Find $a'$ such that $H(q) = Y$,<br>where $q = p + a'$. | $\xrightarrow{p, q, N_s, N_c, par_1, par_2, ...}$ | $X = H(S_1, N_s, N_c, par_1, par_2, ...)$<br>$a = H(S_2, N_s, N_c) \bmod D + l$<br>$H(p) \stackrel{?}{=} H(X - a)$<br>$H(q) \stackrel{?}{=} H(X)$ |

### 4.4 Protocol Description

We now discuss how puzzle 3 can be used in the closed-loop solution. The difficulty parameter $D_b$ is kept the same for both $P_1$ and $P_2$. The value of $a$ is set to a constant $c$ for $P_1$. For $P_2$, $D_a$ is chosen according to the equilibrium conditions and $l$ is set to $c + 1$. Clearly, the attacker cannot determine puzzle difficulty without putting in the effort required to solve $P_1$.

When the defender receives a correct solution for a puzzle, he needs to know whether the puzzle is $P_1$ or $P_2$ and the phase in which the puzzle was sent. Since maintaining state information at the server is a source of vulnerability, the phase and the puzzle type are included as parameters in the puzzle construction. While $\phi$ is sent along with the puzzle, the value of $n$ is determined by the client based on the number of hashes he computes while solving the puzzle. Both these values are sent back to the defender along with the puzzle solution. If the attacker tries to forge the puzzle type or phase to avoid punishment, verification will fail. Note that the defender does not need to know the phase or puzzle type for a random answer.

The protocol is described in Fig. 5.

### 4.5 An Example

On an AMD Opteron 8354 2.2 GHz processor, the computation of the SHA-512 hash function for an input size of 64 bytes takes 0.396 $\mu$s and the 1024-bit DH key agreement takes 0.56 ms [4]. Assume the closed-loop solution is used to

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

| Client | | Defender |
|---|---|---|
| | $\xrightarrow{Request, N_c}$ | $X = H(S_1, N_s, N_c, par_1, par_2, ...$<br>$Y = H(X)$<br>$a = H(S_2, N_s, N_c) \bmod D_a + l$<br>$X' = X - a$<br>$Z = H(X')$ |
| | $\xleftarrow{(X'', Y, Z), N_s, par_1, par_2, ...}$ | $X'' = X' - b$ |
| Find $b'$ such that $H(p) = Z$, where $p = X'' + b'$. Find $a'$ such that $H(q) = Y$, where $q = p + a'$. | $\xrightarrow{p, q, N_s, N_c, par_1, par_2, ...}$ | $X = H(S_1, N_s, N_c, par_1, par_2, ...)$<br>$a = H(S_2, N_s, N_c) \bmod D_a + l$<br>$H(p) \stackrel{?}{=} H(X - a)$<br>$H(q) \stackrel{?}{=} H(X)$ |

defend a DH key agreement protocol against DoS attacks. Let the reference time $T$ be 2 ms. Then, $\alpha_m = 0.28$. The total number of hashes computable in time $T$, $N_h$ is approximately 5050. If puzzle 3 is used, the maximum number of hash computations for puzzle production is 4 and that for verification is 6. We take $\alpha_{PP} = 0.0008$ and $\alpha_{VP} = 0.0012$. It is reasonable to assume that the maximum number of requests $N$ that the attacker can send within 2 ms is 10. If the difficulty parameter $D_b$ is set to 500 and $c = 1$, $\alpha_{SP_1} = \frac{1}{N_h}\left(\frac{D_b+1}{2} + 1\right) = 0.05 < \frac{1}{N}$. For a desirable quality of service $\eta = 0.4$, we choose $\alpha_{SP_2} = 0.29$, satisfying (9) and (10). From $\alpha_{SP_2} = \frac{1}{N_h}\left(\frac{D_a+1}{2} + 1 + \frac{D_b+1}{2}\right)$, we get $D_a = 2423$.

We choose the probabilities $p = 0.357$ and $p_1 = 0.042$ based on (11) and (12) respectively. Note that 35.7% of the requests in phase (A) and 4.2% of the requests in phase (B) receive $P_1$. The corresponding SIRP vector is (-0.1129, 0.0520). As mentioned in Section 3.3, while determining the punishment period, it is sufficient to satisfy (7) and (8) for $i = 2$. Choosing $\delta_0 = 0.9$ and $\delta = 0.99 > \delta_0$, we obtain $\tau(\delta) = 4$.

Assume in the absence of an attack, requests arrive according to a Poisson process at a rate of 750 requests/s. Puzzle $P_1$ is sent if a time period $T\alpha_m = 1.12$ ms has elapsed since the last request. Otherwise, a puzzle is sent according to the defense mechanism. Clearly, $\tilde{p} = Pr(t_1 \geq 1.12 \text{ ms}) = 0.432$. In the absence of an attack, a client receives $P_1$ with probability $0.432 + 0.568p = 0.635$ and will have to compute around 695 hashes on an average. During an attack, a client will have to compute on an average around 1032 hashes in phase (A) and around 1414 hashes in phase (B).

**Fig. 5. Protocol Description**. Here, $H$ is a collision resistant cryptographic hash function, $S_1$ and $S_2$ are server secrets, $N_s$ and $N_c$ are nonces generated by the server and client respectively, $M$ is a session parameter, $\phi$ is the phase of the defense mechanism, $D_a$ and $D_b$ are the difficulty parameters and $b$ is a value chosen uniformly at random from $\{1, ..., D_b\}$. Also, $n$ is the puzzle type, where $n = 1$ for $P_1$ and $n = 2$ for $P_2$.

| Client | | Defender |
|---|---|---|
| | $\xrightarrow{Request, N_c}$ | Choose puzzle type $n$. |
| | | $X = H(S_1, N_s, N_c, M, n, \phi)$ |
| | | $Y = H(X)$ |
| | | $a = \begin{cases} 1 & \text{if } n = 1 \\ H(S_2, N_s, N_c) \bmod D_a + 2 & \text{if } n = 2 \end{cases}$ |
| | | $X' = X - a$ |
| | | $Z = H(X')$ |
| | $\xleftarrow{(X'', Y, Z), N_s, \phi}$ | $X'' = X' - b$ |
| Find $b'$ such that $H(p) = Z$, where $p = X'' + b'$. Find $a'$ such that $H(q) = Y$, where $q = p + a'$. Find puzzle type $n$. | $\xrightarrow{p, q, N_s, N_c, n, \phi}$ | Check that $N_s$ is recent. |
| | | $X = H(S_1, N_s, N_c, M, n, \phi)$ |
| | | $a = \begin{cases} 1 & \text{if } n = 1 \\ H(S_2, N_s, N_c) \bmod D_a + 2 & \text{if } n = 2 \end{cases}$ |
| | | $H(p) \stackrel{?}{=} H(X - a)$ |
| | | $H(q) \stackrel{?}{=} H(X)$ |

## 5 Conclusions

In this paper, we have given emphasis on hiding the difficulty of client-puzzles from a denial of service attacker. Using game theory, we have developed appropriate defense mechanisms and have shown that they are more effective than the ones previously proposed. The concept of Nash equilibrium in infinitely repeated games has been used to come up with suitable defense mechanisms.

Unlike previous works [2, 6], we have taken into account the fact that an attacker's random guess could sometimes be correct and have suitably modified the game model. We have also discussed how the proposed defense mechanisms can be extended to handle distributed attacks. New puzzles that meet the requirements of our defense mechanisms have been described and a specific instance of the protocol has been given.

Future direction of work would be to incorporate the proposed defense mechanisms in the Internet Key Exchange (IKE) protocol [10] and to estimate its effectiveness in real-time.

## References


1. Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. Dos-resistant authentication with client puzzles. In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 170–177, London, UK, 2001. Springer-Verlag.
2. Boldizsar Bencsath, Istvan Vajda, and Levente Buttyan. A game based analysis of the client puzzle approach to defend against dos attacks. In *http://www.hit.bme.hu/ buttyan/publications/BencsathVB03softcom.pdf. Proceedings of the 2003 International Conference on Software, Telecommunications and Computer Networks*, pages 763–767, 2003.
3. Ellick M. Chan, Carl A. Gunter, Sonia Jahid, Evgeni Peryshkin, and Daniel Rebolledo. Using rhythmic nonces for puzzle-based dos resistance. In *CSAW '08: Proceedings of the 2nd ACM workshop on Computer security architectures*, pages 51–58, New York, NY, USA, 2008. ACM.
4. Wei Dai. Crypto++ 5.6.0 benchmarks. Website at http://www.cryptopp.com/benchmarks.html.
5. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *the 12th Annual International Cryptology Conference on Advances in Cryptology*, volume 740 of *Lecture Notes In Computer Science*, pages 139–147. Springer-Verlag.
6. Mehran Fallah. A puzzle-based defense strategy against flooding attacks using game theory. *IEEE Transactions on Dependable and Secure Computing*, 99(2), 5555.
7. W. Feng, E. Kaiser, and A. Luu. Design and implementation of network puzzles. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2372–2382, March 2005.
8. Drew Fudenberg and Eric Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica*, 54(3):533–54, May 1986.
9. Chun-Kan Fung and M. C. Lee. A denial-of-service resistant public-key authentication and key establishment protocol. In *PCC '02: Proceedings of the Performance, Computing, and Communications Conference, 2002. on 21st IEEE International*, pages 171–178, Washington, DC, USA, 2002. IEEE Computer Society.
10. D. Harkins and D. Carrel. *The Internet Key Exchange (IKE), RFC 2409*. Internet Engineering Task Force, 1998.
11. A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pages 151–165, 1999.
12. K. Komathy and P. Narayanasamy. Secure data forwarding against denial of service attack using trust based evolutionary game. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 31–35, May 2008.
13. Jun-Jie Lv. A game theoretic defending model with puzzle controller for distributed dos attack prevention. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 2, pages 1064–1069, July 2008.
14. Ajay Mahimkar and Vitaly Shmatikov. Game-based analysis of denial-of-service prevention protocols. In *CSFW '05: Proceedings of the 18th IEEE workshop on Computer Security Foundations*, pages 287–301, Washington, DC, USA, 2005. IEEE Computer Society.
15. M.J. Osborne. *An Introduction to Game Theory*. Oxford University Press, USA, 2007.

16. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996.
17. Y.E. Sagduyu and A. Ephremides. A game-theoretic analysis of denial of service attacks in wireless random access. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, 2007. WiOpt 2007. 5th International Symposium on*, pages 1–10, April 2007.
18. Jason Smith. *Denial of Service: Prevention, Modelling and Detection*. PhD thesis, Queensland University of Technology, Brisbane, QLD 4001 Australia, June 2007.
19. Suratose Tritilanunt, Colin Boyd, Ernest Foo, and Juan Gonzalez Nieto. Toward non-parallelizable cryptographic puzzles. In *Cryptology and Network Security (CANS 2007)*, volume 4856/2007 of *Lecture Notes in Computer Science*, pages 247–264. Springer Berlin / Heidelberg, December 2007.
20. XiaoFeng Wang and Michael K. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 78, Washington, DC, USA, 2003. IEEE Computer Society.
21. Brent Waters, Ari Juels, J. Alex Halderman, and Edward W. Felten. New client puzzle outsourcing techniques for dos resistance. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 246–256, New York, NY, USA, 2004. ACM.
22. Jun Xu and Wooyong Lee. Sustaining availability of web services under distributed denial of service attacks. *IEEE Trans. Comput.*, 52(2):195–208, 2003.
23. Rui Zhang, Goichiro Hanaoka, and Hideki Imai. A generic construction of useful client puzzles. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 70–79, New York, NY, USA, 2009. ACM.

## A Notations

**Mixed Strategy:** Let $A_i = \{a_1, a_2, ..., a_n\}$ be a set of actions permitted for player $i$ and $\Delta(A_i)$ denote a probability distribution over $A_i$. The mixed strategy $\alpha_i \in \Delta(A_i)$ is denoted as $p_1 \circ a_1 \oplus p_2 \circ a_2 \oplus ... \oplus p_n \circ a_n$, where $p_1 + p_2 + ... + p_n = 1$.

**Strategy Profile:** In a two player game with players 1 and 2, we represent a pure strategy profile as $(a_1; a_2)$, where $a_1 \in A_1$ and $a_2 \in A_2$ and a mixed strategy profile as $(\alpha_1; \alpha_2)$, where $\alpha_1 \in \Delta(A_1)$ and $\alpha_2 \in \Delta(A_2)$. Note that $A$ is the set of all pure strategy profiles in the game.

## B Open-Loop Nash Equilibria

In the game of the client-puzzle approach, the set of all possible stage-game Nash equilibria and the corresponding conditions have been listed in Table 3. In the equilibrium profiles 1, 2 and 3, the defender sends only puzzle $P_1$ and the attacker always solves the puzzle. This may lead to a successful DoS attack. In the case of profile 4, the quality of service cannot be adjusted. Among the remaining profiles, $P_1$ is sent with a higher probability in 5 and 6. The equilibrium profile $(p_1 \circ P_1 \oplus (1-p_1) \circ P_2; TA)$ has been chosen for the open-loop defense mechanism as it is more suitable in preventing a DoS attack from being successful.

**Table 3. Stage-game Equilibrium Profiles.** For a desirable of quality of service $0 < \eta < 1$ and probabilities $0 < p_1 < 1$ and $0 < p_2 < 1$, the various stage-game Nash equilibria are given.

| Equilibrium Profile | Equilibrium Conditions |
|---|---|
| 1. $(P_1; CA)$ | $\eta > \frac{1}{2}$ |
| 2. $(P_1; TA)$ | $\eta > \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$ |
| 3. $(P_1; p_2 \circ CA \oplus (1-p_2) \circ TA)$ | $\eta > max\left(\frac{1}{2}, \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}\right)$ |
| 4. $(p_1 \circ P_1 \oplus (1-p_1) \circ P_2; CA)$ | $\eta = \frac{1}{2}$<br>$\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$<br>$p_1 > \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}$ |
| 5. $(p_1 \circ P_1 \oplus (1-p_1) \circ P_2; TA)$ | $\eta = \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$<br>$\alpha_{SP_2} - \alpha_{SP_1} > \alpha_m$<br>$p_1 > \frac{\alpha_{SP_1}}{\alpha_m}$ |
| 6. $(p_1 \circ P_1 \oplus (1-p_1) \circ P_2; p_2 \circ CA \oplus (1-p_2) \circ TA)$ | $\eta = \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$<br>$\alpha_{SP_2} - \alpha_{SP_1} = \alpha_m$<br>$p_1 > \frac{\alpha_{SP_1}}{\alpha_m}$ |
| 7. $(p_1 \circ P_1 \oplus (1-p_1) \circ P_2; p_2 \circ CA \oplus (1-p_2) \circ RA)$ | $\alpha_{SP_2} - \alpha_{SP_1} < \alpha_m$<br>$p_1 = \frac{\alpha_{SP_2} - \alpha_m}{\alpha_{SP_2} - \alpha_{SP_1}}$<br>$p_2 = \frac{\eta}{1-\eta}$ |
| 8. $(p_1 \circ P_1 \oplus (1-p_1) \circ P_2; p_2 \circ TA \oplus (1-p_2) \circ RA)$ | $\alpha_{SP_2} - \alpha_{SP_1} > \alpha_m$<br>$p_1 = \frac{\alpha_{SP_1}}{\alpha_m}$<br>$p_2 = \left(\frac{\eta}{1-\eta}\right)\left(\frac{\alpha_{SP_2} - \alpha_{SP_1}}{\alpha_m}\right)$ |

# C Closed-Loop Nash Equilibria

## C.1 Calculation of $\tau(\delta)$ [8, 6]

Let $\bar{u_i} = u_i(M_1^2; M_2^1)$ and take

$$\bar{v_i} = \max_{\alpha_1 \in \Delta(A_1), \alpha_2 \in \Delta(A_2)} u_i(\alpha_1; \alpha_2),$$

where $\Delta(X)$ is the set of probability distributions over $X$. For $(v_1, v_2) \in V^*$, choose a value for $\delta_0 \in (0, 1)$ and $\tau_0$ such that for each player $i = 1, 2$

$$v_i > \bar{v_i}(1 - \delta_0) + \delta_0 v_i^{**}, \tag{16}$$

where

$$v_i^{**} = (1 - \delta_0^{\tau_0})\bar{u_i} + \delta_0^{\tau_0} v_i, \tag{17}$$

with

$$v_i^{**} > v_i^*. \tag{18}$$

From (16), (17) and (18),

$$v_i^* < (1 - \delta_0^{\tau_0})\bar{u_i} + \delta_0^{\tau_0} v_i < \frac{v_i - \bar{v_i}(1 - \delta_0)}{\delta_0}. \tag{19}$$

This necessitates

$$\delta_0 > \frac{\bar{v_i} - v_i}{\bar{v_i} - v_i^*} \tag{20}$$

and for $\delta_0$ satisfying (20), $\tau_0$ is obtained from

$$\frac{v_i^* - \bar{u_i}}{v_i - \bar{u_i}} < \delta_0^{\tau_0} < \frac{v_i - \bar{v_i} + \delta_0(\bar{v_i} - \bar{u_i})}{\delta_0(v_i - \bar{u_i})}, \tag{21}$$

subject to $\tau_0 > 0$. Clearly, for any $\delta > \delta_0$, there exists a corresponding $\tau(\delta)$, such that (20) and (21) hold for $(\delta, \tau(\delta))$.

### C.2 Strictly Individual Rational Payoffs

Consider the game of the client-puzzle approach with two puzzles $P_1$ and $P_2$ such that $\alpha_{SP_1} < \alpha_m < \alpha_{SP_2}$. From theorem 3, it is seen that the attacker's minmax payoff is always $u_2(P_1; RA) = u_2(P_2; RA)$. On the other hand, the attacker has four possible minmax strategies against the defender.

**Case (i).** When $\alpha_{SP_2} - \alpha_{SP_1} \leq \alpha_m$ and $\eta < \frac{1}{2}$, the attacker's minmax strategy against the defender is $p_2 \circ CA \oplus (1-p_2) \circ RA$, where $p_2 = \frac{\eta}{1-\eta}$. The convex hull of the payoff vectors, the minmax point and the set of SIRP for the game are shown in Fig. 6.

**Case (ii).** When $\alpha_{SP_2} - \alpha_{SP_1} \geq \alpha_m$ and $\eta < \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$, the attacker's minmax strategy against the defender is $p_2 \circ TA \oplus (1-p_2) \circ RA$, where $p_2 = \left(\frac{\eta}{1-\eta}\right)\left(\frac{\alpha_{SP_2} - \alpha_{SP_1}}{\alpha_m}\right)$. The corresponding convex hull of the payoff vectors, minmax point and set of SIRP are shown in Fig. 7.

**Case (iii).** When $\eta \geq \frac{1}{2}$, the attacker's minmax strategy against the defender is $CA$. Fig. 8 contains the convex hull of the payoff vectors, the minmax point and the set of SIRP.

**Case (iv).** When $\eta \geq \frac{\alpha_m}{\alpha_m + \alpha_{SP_2} - \alpha_{SP_1}}$, the attacker's minmax strategy against the defender is $TA$. The convex hull of the payoff vectors, the minmax point and the set of SIRP are shown in Fig. 9.

In all four cases, the strategy profile $(P_1; p \circ CA \oplus (1-p) \circ RA)$, $0 < p < 1$, corresponds to a high SIRP for the defender. This strategy profile is not very intuitive as it requires the attacker to make decisions based on the outcome of a public randomizing device. Strategy profiles involving the attacker's actions $TQ$ and $QT$ are also not considered as they require the defender to maintain a timer for the puzzles sent and check for timeouts. Two strategy profiles corresponding

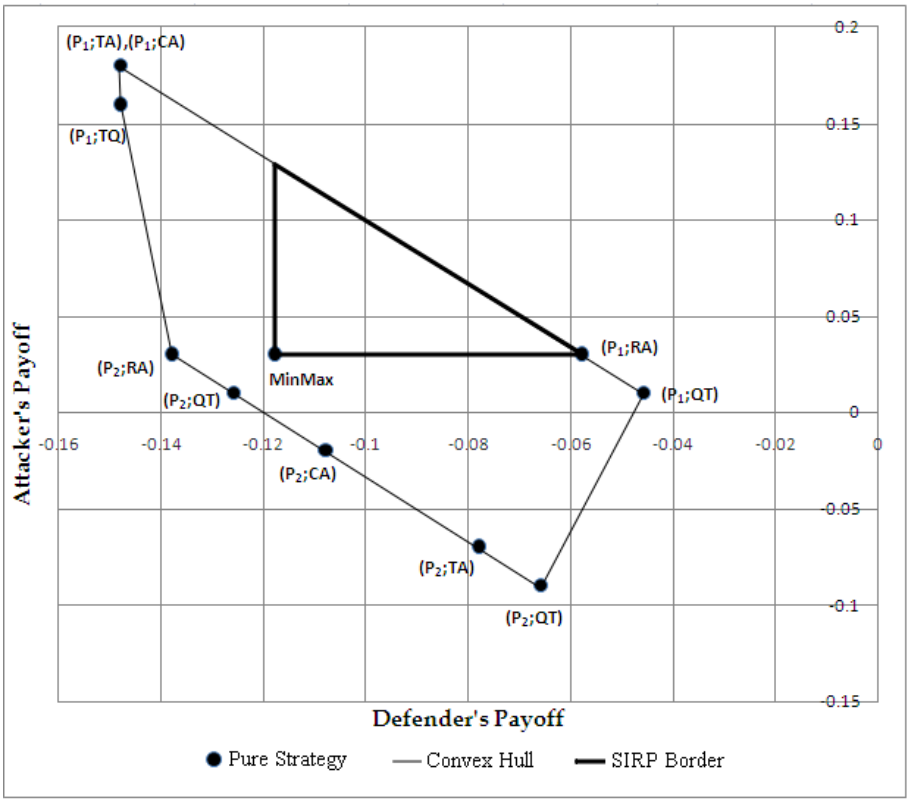


**Fig. 6. Convex Hull of Payoff Vectors and the Set of SIRP for Case (i).** Here, $\alpha_{PP} = 0.01$, $\alpha_{VP} = 0.02$, $\alpha_m = 0.25$, $\alpha_{SP_1} = 0.1$, $\alpha_{SP_2} = 0.3$ and $\eta = 0.4$. The value of $p_2$ is 0.667 and the corresponding minmax point is $(-0.118, 0.03)$.

to reasonably high SIRP for the defender are $(p \circ P_1 \oplus (1-p) \circ P_2; TA)$, $0 < p < 1$ (case (i)) and $(p \circ P_1 \oplus (1-p) \circ P_2; CA)$, $0 < p < 1$ (case (ii)). In the case of the second profile, a successful DoS attack cannot be prevented. Hence, the first profile has been chosen for the closed-loop defense mechanism.

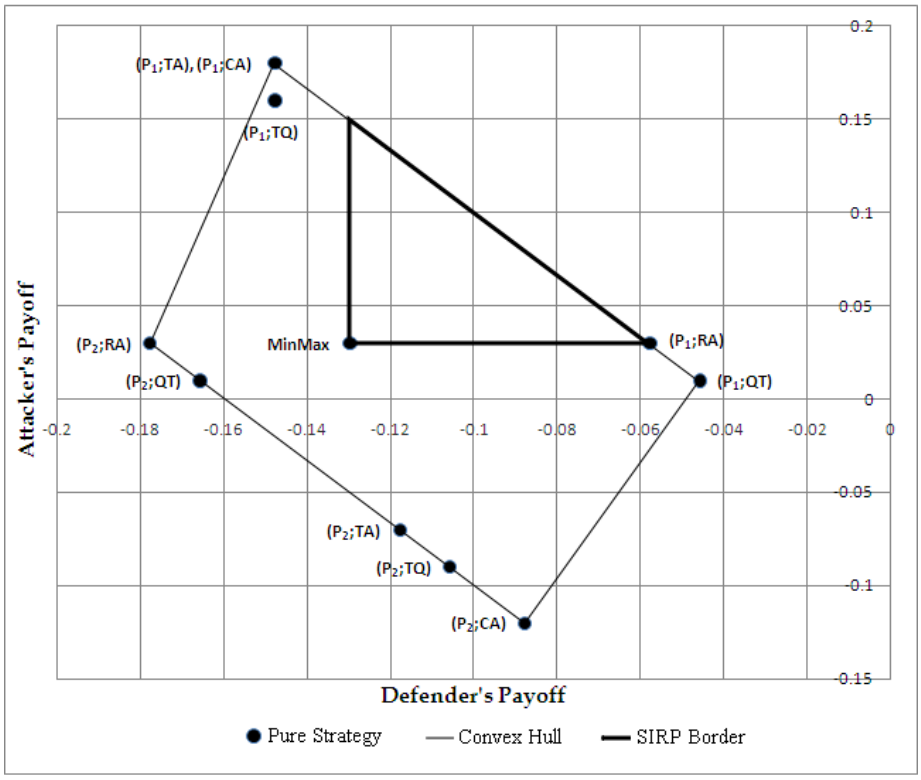


**Fig. 7. Convex Hull of Payoff Vectors and the Set of SIRP for Case (ii).** Here, $\alpha_{PP} = 0.01$, $\alpha_{VP} = 0.02$, $\alpha_m = 0.25$, $\alpha_{SP_1} = 0.1$, $\alpha_{SP_2} = 0.4$ and $\eta = 0.4$. The value of $p_2$ is 0.8 and the corresponding minmax point is $(-0.13, 0.03)$.

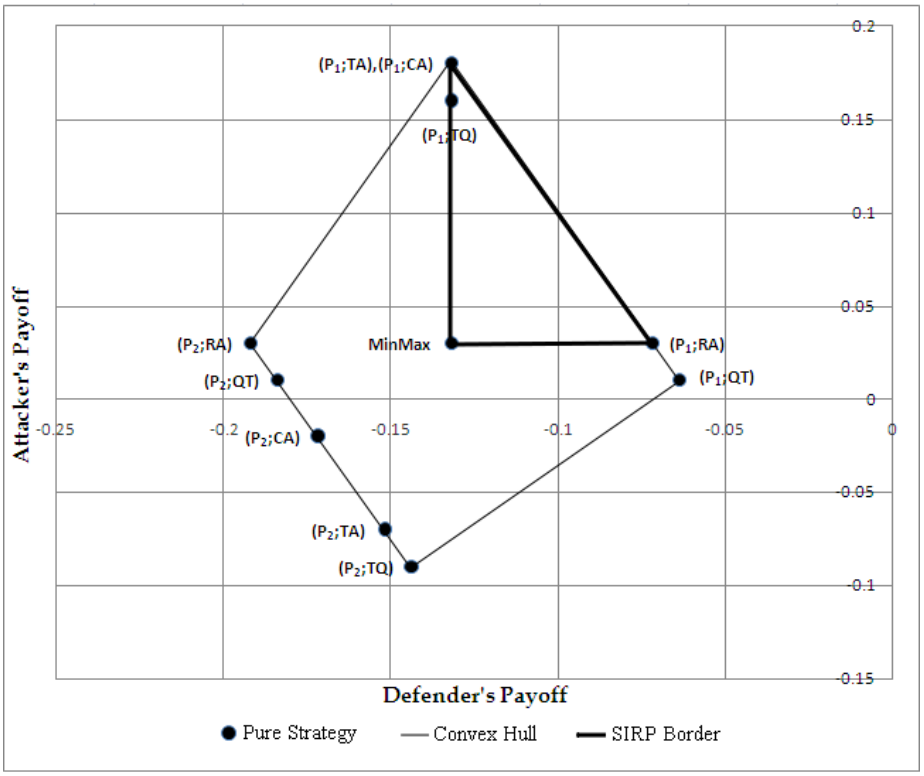


**Fig. 8. Convex Hull of Payoff Vectors and the Set of SIRP for Case (iii).** Here, $\alpha_{PP} = 0.01$, $\alpha_{VP} = 0.02$, $\alpha_m = 0.25$, $\alpha_{SP_1} = 0.1$, $\alpha_{SP_2} = 0.3$ and $\eta = 0.6$. The minmax point is $(-0.132, 0.03)$.

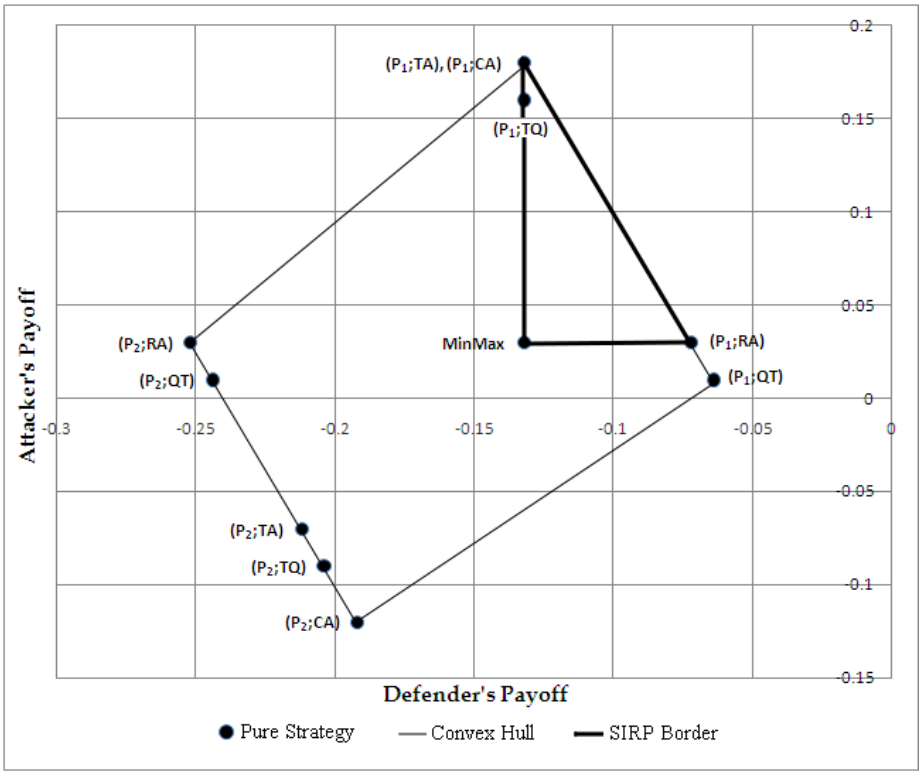


**Fig. 9. Convex Hull of Payoff Vectors and the Set of SIRP for Case (iv).** Here, $\alpha_{PP} = 0.01$, $\alpha_{VP} = 0.02$, $\alpha_m = 0.25$, $\alpha_{SP_1} = 0.1$, $\alpha_{SP_2} = 0.4$ and $\eta = 0.6$. The minmax point is $(-0.132, 0.03)$.