Defense Against Advanced Persistent Threats in Dynamic Cloud Storage: A Colonel Blotto Game Approach

Minghui Min*, Liang Xiao*, Caixia Xie*, Mohammad Hajimirsadeghi†, Narayan B. Mandayam†

*Department of Communication Engineering, Xiamen University, Xiamen, China. Email: lixiao@xmu.edu.cn
†WINLAB, Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ. Email: {mohammad, narayan}@winlab.rutgers.edu

Abstract—Advanced Persistent Threat (APT) attackers apply multiple sophisticated methods to continuously and stealthily steal information from the targeted cloud storage systems and can even induce the storage system to apply a specific defense strategy and attack it accordingly. In this paper, the interactions between an APT attacker and a defender allocating their Central Processing Units (CPUs) over multiple storage devices in a cloud storage system are formulated as a Colonel Blotto game. The Nash equilibria (NEs) of the CPU allocation game are derived for both symmetric and asymmetric CPUs between the APT attacker and the defender to evaluate how the limited CPU resources, the data storage size and the number of storage devices impact the expected data protection level and the utility of the cloud storage system. A CPU allocation scheme based on “hotbooting” policy hill-climbing (PHC) that exploits the experiences in similar scenarios to initialize the quality values to accelerate the learning speed is proposed for the defender to achieve the optimal APT defense performance in the dynamic game without being aware of the APT attack model and the data storage model. A hotbooting deep Q-network (DQN)-based CPU allocation scheme further improves the APT detection performance for the case with a large number of CPUs and storage devices. Simulation results show that our proposed reinforcement learning based CPU allocation can improve both the data protection level and the utility of the cloud storage system compared with the Q-learning based CPU allocation against APTs.

Index Terms—Colonel Blotto game, advanced persistent threats, cloud security, CPU allocation, reinforcement learning, data protection level.

I. INTRODUCTION

Cloud storage and cyber systems are vulnerable to Advanced Persistent Threats (APTs), in which an attacker applies multiple sophisticated methods such as the injection of multiple malwares to continuously and stealthily steal data from the targeted cloud storage system [1], [2] and [3]. APT attacks are difficult to detect and have caused privacy leakage and millions of dollars’ loss [4] and [5]. According to [6], more than 65% of the organizations in a survey in 2014 have experienced more APT attacks in their IT networks than last year.

The FlipIt game proposed in the seminal work [2] formulates the stealthy and continuous APT attacks and designs the scan interval to detect APTs on a given cyber system. The game theoretic study in [8] has provided insights to design the optimal scan intervals of a cyber system against APTs. Prospect theory has been applied in [9] to investigate the probability distortion of an APT attacker against cloud storage and cumulative prospect theory has been used in [10] to model the frame effect of an APT attacker to choose the attack interval. Most existing APT games ignore the strict resource constraints in the APT defense, such as the limited number of Central Processing Units (CPUs) of a storage defender and an APT attacker [7] and [11]. However, a cloud storage system with limited number of CPUs cannot scan all the data stored on the storage devices in a given time slot. To this end, encryptions and authentication techniques are applied to protect data privacy for cloud storage systems. On the other hand, an APT attacker with limited CPU resources cannot install malwares to steal all the data on the cloud storage system in a single time slot either [12].

It is challenging for a cloud storage system to optimize the CPU allocation to scan the storage devices under a large number of CPUs and storage devices without being aware of the APT attack strategy. Therefore, we use the Colonel Blotto game (CBG), a two-player zero-sum game with multiple battlefields to model the competition between an APT attacker and a storage defender, each with a limited total number of CPUs over a given number of storage devices. The player who applies more resources on a battlefield in a Colonel Blotto game wins it, and the overall payoff of a player in the game is proportional to the number of the winning battlefields [13]. The Colonel Blotto game has been recently applied to design the spectrum allocation of network service providers [14], the jamming resistance methods for Internet of Things [15] and [16].

Our previous work in [17] assumes that each storage device has the same amount of data and addresses APT attackers that does not change the attack policy. However, the storage devices usually have different amount of the data with different priority levels, and the data size and their priority level also change over time. By allocating more CPUs to scan the storage devices with more data, a storage defender can achieve a higher data protection level. Therefore, this work extends to a dynamic cloud storage system whose data size changes over time and addresses smart APTs, in which an attacker that can learn the defense strategy first chooses the attack strength to induce the storage system to apply a specific defense strategy and then attacks it accordingly.
By applying time sharing (or division), a defender can use a single CPU to scan multiple storage devices as battlefields to detect APTs in a time slot, and an attacker can use a single CPU to attack multiple devices with a single CPU yielding a roughly continuous CBG. According to [13], a pure-strategy Colonel Blotto game rarely achieves Nash equilibria (NEs). Therefore, we focus on the CBG with mixed strategies, in which both players choose their CPU allocation distribution and introduce randomness in their action selection to fool their opponent. The conditions under which the NEs exist in the CPU allocation game are provided to disclose how the number of storage devices, the size of the data stored in each storage device and the total number of CPUs in which the defender observes the impact on the data protection level and the utility of the cloud storage system against APTs.

The CBG-based CPU allocation game provides a framework to understand the strategic behavior of both sides, and the NE strategy relies on the detailed prior knowledge about the APT attack model. In particular, the cloud defender has to know the total number of the attack CPUs and the attack policy over the storage devices, which is challenging to accurately estimate in a dynamic storage system. On the other hand, the repeated interactions between the APT attacker and the defender over multiple time slots can be formulated as a dynamic CPU allocation game, and the defender can choose the security strategy according to the attack history. The APT defense decisions in the dynamic CPU allocation game can be approximately formulated as a Markov decision process (MDP) with finite states, in which the defender observes the state that consists of the previous attack CPU allocation and the current data storage distribution. Therefore, a defender can apply reinforcement learning (RL) techniques such as Q-learning to achieve the optimal CPU allocation over the storage devices to detect APTs in a dynamic game.

The policy hill-climbing (PHC) algorithm as an extension of Q-learning in the mixed-strategy game [18] enables an agent to achieve the optimal strategy without being aware of the underlying system model. For instance, the PHC-based CPU allocation scheme as proposed in our previous work in [17] enables the defender to protect the storage devices with limited number of CPUs without being aware of the APT attack model. In this work, a “hotbooting” technique as a combination of transfer learning [19] and RL exploits experiences in similar scenarios to accelerate the initial learning speed. We propose a “hotbooting” PHC-based CPU allocation scheme that chooses the number of the CPUs on each storage device based on the current state and the quality or Q-function that is initialized according to the APT detection experiences to reduce the exploration time at the initial learning stage.

We apply deep Q-network (DQN), a deep reinforcement learning technique recently developed by Google DeepMind in [20] and [21] to accelerate the learning speed of the defender for the case with a large number of storage devices and defense CPUs. More specifically, the DQN-based CPU allocation exploits the deep convolutional neural network (CNN) to determine the Q-value for each CPU allocation and thus suppress the state space observed by the cloud storage defender. Simulation results demonstrate that this scheme can improve the data protection level, increase the APT attack cost, and enhance the utility of the cloud storage system against APTs.

The main contributions of this paper are summarized as follows:

1. We formulate a CBG-based CPU allocation game against APTs with time-variant data size and attack policy. The NEs of the game are provided to disclose the impact of the number of storage devices, the amount of data stored in each device and the total number of CPUs on the data protection level of the cloud storage system.
2. A hotbooting PHC-based CPU allocation scheme is developed to achieve the optimal CPU allocation over the storage devices with low computational complexity and improve the data protection level compared with the PHC-based scheme as proposed in [17].
3. A hotbooting DQN-based CPU allocation scheme is proposed to further accelerate the learning speed and improve the resistance against APTs.

The rest of this paper is organized as follows: We review the related work in section II, and present the system model in section III. We formulate a CBG-based CPU allocation game and derive the NEs of the game in Section IV. A hotbooting PHC-based CPU allocation scheme and a hotbooting DQN-based scheme are developed in Sections V and VI, respectively. Simulation results are provided in Section VII and the conclusions of this work are drawn in Section VIII.

II. RELATED WORK

The seminal work in [7] formulates a stealthy takeover game between an APT attacker and a defender, who compete to control a targeted cloud storage system. The APT scan interval on a single device has been optimized in [22] based on the FlipIt model without considering the constraint of scanning CPUs. The game between an overt defender and a stealthy attacker as investigated in [23] provides the best response of the periodic detection strategy against a non-adaptive attacker. The online learning algorithm as developed in [24] achieves the optimal timing of the security updates in the FlipIt game, and reduces the regret of the upper confidence bound compared with the periodic defense strategy. The APT defense game formulated in [12] extends the FlipIt game in [7] to multi-node systems with limited resources. The game among an APT attacker, a cloud defender and a mobile device as formulated in [25] combines the APT defense game in [7] with the signaling game between the cloud and the mobile device. The evolutionary game can capture the long term continuous behavior of APTs on cloud storage [26]. The information-trading and APT defense game formulated in [27] analyzes the joint APT and insider attacks. The subjective view of APT attackers under uncertainty scanning duration was analyzed in [9] based on prospect theory.

Colonel Blotto game models the competition between two players each with resource constraints. For example, the Colonel Blotto game with mixed-strategy as formulated in [14] studies the spectrum allocation of network service providers, yielding a fictitious play based allocation approach to compute...
the equilibrium of the game with discrete spectrum resources. The anti-jamming communication game as developed in \cite{28} optimizes the transmit power over multiple channels in cognitive radio networks based on the NE of the CBG. The CBG-based jamming game as formulated in \cite{15} shows that neither the defender nor the attacker can dominate with limited computational resources. The CBG-based jamming game as formulated in \cite{16} shows how the number of subcarriers impacts the anti-jamming performance of Internet of Things with continuous and asymmetric radio power resources. The CBG-based phishing game as formulated in \cite{29} investigates the dynamics of the detect-and-takedown defense against phishing attacks.

Reinforcement learning techniques have been used to improve network security. For instance, the minimax-Q learning based spectrum allocation as developed in \cite{30} increases the spectrum efficiency in cognitive radio networks. The DQN-based anti-jamming communication scheme as designed in \cite{31} applies DQN to choose the transmit channel and node mobility and can increase the signal-to-interference-plus-noise ratio of secondary users against cooperative jamming in cognitive radio networks. The PHC-based CPU allocation scheme as proposed in \cite{17} applies PHC to improve the data protection level of the cloud storage system against APTs. Compared with our previous work in \cite{17}, this work improves the game model by incorporating the time-variant data storage model. We also apply both the hotbooting technique and DQN to accelerate the learning speed and thus improve the security performance for the case with a large number of storage devices and CPUs against the smart APT attacks in the dynamic cloud storage system.

III. SYSTEM MODEL

As illustrated in Fig. 1 the cloud storage system consists of $D$ storage devices, where device $i$ stores data of size $B_i^{(k)}$ at time $k$, with $1 \leq i \leq D$. Let $B^{(k)} = [B_i^{(k)}]_{1 \leq i \leq D}$ be the data size vector of the cloud storage system, and $\sum_{i=1}^{D} B_i^{(k)}$ denote the total amount of the data stored in the cloud storage system at time $k$.

In this work, we consider an APT attacker who combines multiple attack methods, tools, and techniques such as \cite{5} to steal data from the targeted cloud storage system over a long time. The attacker aims to steal more data from the $D$ storage devices with $S_N$ CPUs without being detected. At time $k$, $N_i^{(k)}$ out of the $S_N$ CPUs are used to attack storage device $i$, with $\sum_{i=1}^{D} N_i^{(k)} \leq S_N$. The attack CPU allocation at time $k$ is given by $N^{(k)} = [N_i^{(k)}]_{1 \leq i \leq D} \in \triangle_A$, where the attack action set $\triangle_A$ is given by

$$\triangle_A = \left\{ [N_i^{(k)}]_{1 \leq i \leq D} \mid 0 \leq N_i \leq S_N; \sum_{i=1}^{D} N_i \leq S_N \right\}. \quad (1)$$

A defender uses $S_M$ CPUs to scan the $D$ storage devices in the cloud storage system and aims to detect APTs as early as possible to minimize the total amount of the data stolen by the attacker. At time $k$, $M_i^{(k)}$ out of the $S_M$ CPUs are allocated to scan the device $i$ for APTs, with $\sum_{i=1}^{D} M_i^{(k)} \leq S_M$. As each time slot is quite short, the storage defender can not scan all the data stored in the $D$ storage devices in a single time slot. The defense CPU allocation vector denoted by $M^{(k)}$ is defined as $M^{(k)} = [M_i^{(k)}]_{1 \leq i \leq D} \in \triangle_D$, where the defense action set $\triangle_D$ is given by

$$\triangle_D = \left\{ [M_i^{(k)}]_{1 \leq i \leq D} \mid 0 \leq M_i \leq S_M; \sum_{i=1}^{D} M_i \leq S_M \right\}. \quad (2)$$

If the attacker uses more CPUs than the defender in the APT defense game, the data stored in the storage device are assumed to be at risk. More specifically, the data stored in storage device $i$ are assumed to be safe if the number of the defense CPUs is greater than the number of attack CPUs at that time, i.e., $M_i^{(k)} > N_i^{(k)}$, and the data are at risks if $M_i^{(k)} < N_i^{(k)}$. If $M_i^{(k)} = N_i^{(k)}$, both players have an equal opportunity to control the storage device. Let $\text{sgn}(x)$ denote a sign function, with $\text{sgn}(x) = 1$ if $x > 0$, $\text{sgn}(x) = -1$ if $x < 0$, and 0 otherwise. Therefore, the data protection level of the cloud storage system at time $k$ denoted by $R^{(k)}$ is defined as the normalized size of the “safe” data that are protected by the defender and is given by

$$R^{(k)} = \frac{1}{B^{(k)}} \sum_{i=1}^{D} B_i^{(k)} \text{sgn}(M_i - N_i). \quad (3)$$

For ease of reference, our commonly used notations are summarized in Table I. The time index $k$ in the superscript is omitted if no confusion occurs.

| TABLE I SUMMARY OF SYMBOLS AND NOTATIONS |
|------------------------------------------|
| $D$ | Number of storage devices |
| $S_{M/N}$ | Total number of defense/attack CPUs |
| $M^{(k)}/N^{(k)}$ | Defense/attack CPU allocation vector |
| $\triangle_{D/A}$ | Action set of the defender/attacker |
| $u^{(k)}_{D/A}$ | Utility of the defender/attacker at time $k$ |
| $B^{(k)}$ | Total size of the stored data at time $k$ |
| $B^{(k)}_i$ | Data size vector of $D$ devices at time $k$ |
| $R^{(k)}$ | Data protection level at time $k$ |
IV. CBG-BASED CPU ALLOCATION GAME

Colonel Blotto game is a powerful tool to study the strategic resource allocation of two agents each with limited resources in a competitive environment. Therefore, the interactions between the APT attacker and the defender of the cloud storage system regarding their CPU allocations can be formulated as a Colonel Blotto game with $D$ battlefields. By applying the time sharing (or division) technique, the defender (or attacker) can scan (or attack) multiple storage devices with a single CPU in a time slot, which can be approximately formulated as a continuous CBG. In this game, the defender chooses the defense CPU allocation vector $\mathbf{M}^{(k)} \in \Delta_D$ to scan the $D$ devices at time $k$, while the APT attacker chooses the attack CPU allocation $\mathbf{N}^{(k)} \in \Delta_A$. The utility of the defender (or the attacker) at time $k$ denoted by $u_D^{(k)}(\mathbf{M}, \mathbf{N})$ or $u_A^{(k)}(\mathbf{M}, \mathbf{N})$ depends on the size of the data stored in the $D$ devices, and the data protection level of each device at the time. In the zero-sum game, by (3) the utility of the defender is set as

$$u_D^{(k)}(\mathbf{M}, \mathbf{N}) = -u_A^{(k)}(\mathbf{M}, \mathbf{N}) = \sum_{i=1}^{D} B_i^{(k)} \text{sgn}(M_i - N_i). \tag{4}$$

The CBG-based CPU allocation game rarely has a pure-strategy NE, because the attack CPU allocation $\mathbf{N}^{(k)}$ can be chosen according to the defense CPU allocation $\mathbf{M}^{(k)}$ and to defeat it for a higher utility $u_A^{(k)}$. Therefore, we study the CPU allocation game with mixed strategies, in which each player randomizes the CPU allocation strategies to fool the opponent.

In the mixed-strategy CPU allocation game, the defense strategy at time $k$ denoted by $x_{i,j}^{(k)}$ is the probability that the defender allocates $j$ CPUs to scan device $i$, i.e., $x_{i,j}^{(k)} = \text{Pr}(M_i = j)$. Let $p_{m,j} \in [0, 1]$ be the $m$-th highest feasible value of $x_{i,j}^{(k)}$. The mixed-strategy defense action set denoted by $\mathbf{P}_M$ is given by

$$\mathbf{P}_M = \left\{ [p_{m,j}]_{1 \leq m \leq D, 0 \leq j \leq S_m} \mid p_{m,j} \geq 0, \forall j; \sum_{j=0}^{S_M} p_{m,j} = 1, \forall m \right\}. \tag{5}$$

The defense mixed strategy vector denoted by $\mathbf{x}^{(k)}$ is given by

$$\mathbf{x}^{(k)} = \left[ x_{i,j}^{(k)} \right]_{1 \leq i \leq D, 0 \leq j \leq S_M} \in \mathbf{P}_M. \tag{6}$$

Similarly, let $y_{i,j}^{(k)}$ denote the probability that $N_i^{(k)}$ CPUs are used to attack device $i$, i.e., $y_{i,j}^{(k)} = \text{Pr}(N_i^{(k)} = j)$, and $q_{m,j} \in [0, 1]$ be the $m$-th highest feasible value of $y_{i,j}^{(k)}$. The action set of the attacker in the mixed-strategy game denoted by $\mathbf{P}_N$ is given by

$$\mathbf{P}_N = \left\{ [q_{m,j}]_{1 \leq m \leq D, 0 \leq j \leq S_N} \mid q_{m,j} \geq 0, \forall j; \sum_{j=0}^{S_N} q_{m,j} = 1, \forall m \right\}. \tag{7}$$

The attacker chooses the CPU allocation strategy in this game denoted by $\mathbf{y}^{(k)}$ with

$$\mathbf{y}^{(k)} = \left[ y_{i,j}^{(k)} \right]_{1 \leq i \leq D, 0 \leq j \leq S_N} \in \mathbf{P}_N. \tag{8}$$

The expected utility of the defender (or the attacker) averaged over all the feasible defense (or attack) strategies is denoted by $U_D^{(k)}(\mathbf{x}, \mathbf{y})$ and given by (4) as

$$U_D^{(k)}(\mathbf{x}, \mathbf{y}) = -U_A^{(k)}(\mathbf{x}, \mathbf{y}) = E_{M \sim \mathbf{x}} \sum_{i=1}^{D} B_i^{(k)} \text{sgn}(M_i - N_i). \tag{9}$$

We first consider a CBG-based CPU allocation game $\mathcal{G}_1$ with symmetric CPU resources, $S_M = S_N$, i.e., the defender and the attacker have the same amount of computational resources. Let $1_{m \times n}$ (or $0_{m \times n}$) be an all-1 (or 0) $m \times n$ matrix, $\lfloor \cdot \rfloor$ be the lower floor function, and the normalized defense CPUs $\beta = 2S_M/B$.

**Theorem 1.** If $S_M = S_N$ and $B_i < \sum_{1 \leq h \neq i \leq D} B_h$, the CPU allocation game $\mathcal{G}_1$ has a NE $(\mathbf{x}^*, \mathbf{y}^*)$ given by

$$\mathbf{x}^* = \left[ \begin{array}{c} 1 \frac{B_M + 1}{B_D + 1} 1 \frac{B_M + 1}{B_D + 1} \\ 1 \frac{B_M + 1}{B_D + 1} 1 \frac{B_M + 1}{B_D + 1} \\ \vdots \ \vdots \\ 1 \frac{B_M + 1}{B_D + 1} 1 \frac{B_M + 1}{B_D + 1} \end{array} \right] \begin{array}{c} \frac{B_M + 1}{B_D + 1} \frac{0_{1 \times (S_M - |\beta B_M|)}}{1} \frac{0_{1 \times (S_M - |\beta B_M|)}}{1} \\ \frac{B_M + 1}{B_D + 1} \frac{0_{1 \times (S_M - |\beta B_M|)}}{1} \frac{0_{1 \times (S_M - |\beta B_M|)}}{1} \\ \vdots \ \vdots \\ \frac{B_M + 1}{B_D + 1} \frac{0_{1 \times (S_M - |\beta B_M|)}}{1} \frac{0_{1 \times (S_M - |\beta B_M|)}}{1} \end{array}. \tag{12}$$

**Proof.** The CPU allocation game $\mathcal{G}_1$ can be formulated as a CBG with symmetric players on $D$ battlefields. The resource budget of the defender is $S_M$, the value of the $i$-th battlefield is $B_i$, and the total value of $D$ battlefields is $B$. Let $U(m, n)$ denote the uniform distribution between $m$ and $n$. By Proposition 1 in (32), the mixed-strategy CBG game has an NE given by $(\mathbf{x}^*, \mathbf{y}^*)$, where $\mathbf{x}^*$ is the probability distribution of $\mathbf{M}$, and each vector coordinate $M_i$ is uniform distribution between 0 and $2S_M B_i/B$. Therefore, the CPU allocation of the $i$-th device $M_i^*$ is uniformly distributed on $[0, 2S_M B_i/B]$,

$$M_i^* \sim U \left( \left\{ 0, 1, 2, ..., \left\lfloor \frac{2S_M B_i}{B} \right\rfloor \right\} \right). \tag{13}$$
Thus this game has an NE given by \( (x^*, x^*) \), where \( \forall 0 \leq j \leq \lfloor 2S_MB_i/B \rfloor \), \( 0 \leq i \leq D \), each element of \( x \) is given by

\[
x^*_{i,j} = \Pr (M_i^* = j) = \frac{1}{(2S_MB_i/B) + 1},
\]

which results in (12).

**Corollary 1.** At the NE of the symmetric CPU allocation game \( G_1 \), the expected data protection level is zero and the utility of the defender is zero.

**Proof.** By (5) and (12), the data protection level over all the realizations of the mixed-strategy NE \((x^*, x^*)\) is given by

\[
E_x^s (R) = E_{x^s} \left( \frac{1}{B} \sum_{i=1}^D B_i \text{sgn} (M_i^* - N_i^*) \right)
\]

\[
= \frac{1}{B} \sum_{i=1}^D B_i \left( \Pr (N_i^* < M_i^*) - \Pr (N_i^* > M_i^*) \right) = 0.
\]

Similarly, by (4) and (2), we have \( U_D = U_A = 0 \).

**Remark:** If the APT attacker and the defender have the same number of CPUs and no storage device dominates in the game (i.e., \( B_i < \sum_{1 \leq h \neq i \leq D} B_i \), \( \forall i \)), both players choose a number from \( \{0, 1, \ldots, \lfloor 2S_MB_i/B \rfloor \} \) to attack or scan storage device \( i \) with probability \( 1/\left( \lfloor 2S_MB_i/B \rfloor + 1 \right) \) by (14). The data protection level \( R \) by definition ranges between \(-1 \) and \( 1 \). Therefore, the game makes a tie, yielding zero expected data protection level and zero utility of the defender.

We next consider a CBG-based CPU allocation game with asymmetric players denoted by \( G_2 \), in which the attacker and the defender have different number of CPUs and compete over \( D \) storage devices with an equal data size, i.e., \( B_i = B, \forall i \).

**Theorem 2.** If \( 2/D \leq S_N/S_M \leq 1 \), \( D \geq 3 \) and \( B_i = B, \forall 1 \leq i \leq D \), the NE of the CPU allocation game \( G_2 \) \((x^*, y^*)\) is given by

\[
x^* = \begin{bmatrix} 0_{D \times 1} & \frac{1}{(2S_M/B)} 1_{D \times \lfloor 2S_M/B \rfloor} & 0_{D \times (S_M - \lfloor 2S_M/B \rfloor)} \end{bmatrix}
\]

\[
y^* = \begin{bmatrix} \left(1 - S_N/S_M\right) 1_{D \times 1} & \left(S_N/S_M \frac{2S_M}{B} \right)^{1_{D \times \lfloor 2S_M/B \rfloor}} & 0_{D \times (S_M - \lfloor 2S_M/B \rfloor)} \end{bmatrix}
\]

By Theorem 2 in (13), the unique Nash equilibrium for the defender and the attacker with \( 2/D \leq S_N/S_M \leq 1 \) is given by

\[
x(M_i^*) \sim \mathcal{U} \left( \left[ 0, \frac{2S_M}{D} \right] \right)
\]

\[
y(N_i^*) \sim \left( 1 - \frac{S_N}{S_M} \right) \delta(N_i^*) + \frac{S_N}{S_M} \mathcal{U} \left( \left[ 0, \frac{2S_M}{D} \right] \right).
\]

Thus this game has an NE given by

\[
M_i^* \sim \mathcal{U} \left( \left[ 0, 1, 2, \ldots, \frac{2S_M}{D} \right] \right).
\]

**Corollary 2.** At the NE of the CPU allocation game \( G_2 \), the expected data protection level is \( 1 - S_N/S_M \) and

\[
U_D = -U_A = \left(1 - \frac{S_N}{S_M}\right) \hat{B}.
\]

**Proof.** According to (3), (15), (17) and (18), as \( B_i = B, \forall i \).

The CPU allocation game \( G_2 \) can be formulated as a CBG with asymmetric players on \( D \) battlefields, where the defender (or attacker) chooses the probability density functions \( x \) (or \( y \)) according to \( S_M \) (or \( S_N \)) resource budget, and the resources allocated to the \( i \)-th battlefield is \( M_i \) (or \( N_i \)).

By Theorem 2 in (13), the unique Nash equilibrium for the defender and the attacker with \( 2/D \leq S_N/S_M \leq 1 \) is given by

\[
x(M_i^*) \sim \mathcal{U} \left( \left[ 0, \frac{2S_M}{D} \right] \right)
\]

\[
y(N_i^*) \sim \left( 1 - \frac{S_N}{S_M} \right) \delta(N_i^*) + \frac{S_N}{S_M} \mathcal{U} \left( \left[ 0, \frac{2S_M}{D} \right] \right).
\]
The APT defense performance of the CPU allocation game $G_2$ at the NE with $D$ storage devices and $S_M$ defense CPUs against an APT attacker with 150 CPUs.

The APT defense performance of the CPU allocation game $G_2$ at the NE is presented in Fig. 2, in which the $D$ storage devices are threatened by an APT attacker with 150 CPUs. If the defender uses 1200 CPUs instead of 600 CPUs to scan the 20 devices, the data protection level increases about 10.5% to 93% and the utility of the defender increases by 18.75%. The data protection level of the cloud storage system protected by 1200 CPUs slightly decreases by 2.8%, if the number of the storage devices $D$ changes from 20 to 80. The APT defense performance of the CBG game at the NE provides the optimal defense performance with known APT attack model and defense model, and can be used as a guideline to design the CPU allocation scheme.

V. HOTBOOTING PHC-BASED CPU ALLOCATION

As a defender is usually unaware of the attack policy, we propose a hotbooting PHC-based CPU allocation scheme to scan $D$ storage devices in the dynamic APT detection game, as illustrated in Fig. 3. At each time slot, the defender of the cloud storage system observes the amount of the data stored in each storage device that is quantized into $L$ levels, $B_i^{(k)} \in \{l/L\}_{0 \leq l \leq L}$. In addition, the defender also evaluates the compromised storage devices that are found to be attacked by APTs in the last time slot, and uses them to estimate the last attack CPU allocation $N^{(k-1)}$. The defense CPU allocation is chosen according to the current state denoted by $s^{(k)}$, which consists of the current data sizes and the previous attack CPU allocation, i.e., $s^{(k)} = \{N^{(k-1)}, B^{(k)}\}$. The resulting defense strategy $M^{(k)} \in \triangle_D$, where $\triangle_D$ is the defense action set given by (2).

Algorithm 1 CPU allocation with hotbooting PHC

By setting the learning rate $\alpha \in (0,1]$ is the weight of the current experience, the discount factor $\gamma \in [0,1]$ indicates the uncertainty of the defender on the future reward, $s'$ is the next state if the defender uses $M^{(k)}$ at state $s^{(k)}$, and the value

![Graph](image-url)
function $V(\mathbf{s})$ maximizes $Q(\mathbf{s}, \mathbf{M})$ over the action set given by

$$V(\mathbf{s}(k)) = \max_{\mathbf{M} \in \Delta_D} Q(\mathbf{s}(k), \mathbf{M}).$$

(28)

The mixed-strategy table of the defender denoted by $\pi(\mathbf{s}, \mathbf{M})$ provides the distribution of the number of CPUs $\mathbf{M}$ over the $D$ storage devices under state $\mathbf{s}$ and is updated via

$$\pi(\mathbf{s}(k), \mathbf{M}) \leftarrow \pi(\mathbf{s}(k), \mathbf{M}) + \begin{cases} \delta, & \text{if } \mathbf{M} = \arg \max_{\mathbf{M} \in \Delta_D} Q(\mathbf{s}(k), \mathbf{M}') \\ 0, & \text{o.w.} \end{cases}$$

(29)

In this way, the probability of the action that maximizes the $Q$-function increases by $\delta$, with $0 < \delta \leq 1$, and the probability of other actions decrease by $\delta/(|\Delta_D| - 1)$. The defender then selects the number of CPUs $\mathbf{M}(k) \in \Delta_D$ according to the mixed strategy $\pi(\mathbf{s}(k), \mathbf{M})$, i.e.,

$$\Pr(\mathbf{M}(k) = \mathbf{M}) = \pi(\mathbf{s}(k), \mathbf{M}), \quad \forall \mathbf{M} \in \Delta_D.$$

(30)

We apply the hotbooting technique to initialize both the $Q$-values as the output of the hotbooting process based on the $\xi$ experiences denoted by $\overline{Q}$ is used to initialize the $Q$-values in Algorithm 1. Similarly, the mixed-strategy table as the output of Algorithm 2 based on the $\xi$ experiences denoted by $\overline{\pi}$ is used to initialize $\pi$ in Algorithm 1. The learning time of Algorithm 1 increases with the dimension of the action-state space $|\Delta_D| \times |\Delta_A| \times L^D$, which increases with the number of storage devices in the cloud storage system and the number of CPUs, yielding serious performance degradation.

VI. HOTBOOTING DQN-BASED CPU ALLOCATION

In this section, we propose a hotbooting DQN-based CPU allocation scheme to improve the APT defense performance of the cloud storage system. This scheme applies deep convolutional neural network, a deep reinforcement learning technique, to compress the action-state space and thus accelerate the learning process. As illustrated in Fig. 3 the deep convolutional neural network is a nonlinear approximator of the $Q$-value for each action. The CNN architecture allows a compact storage of the learned information between similar states.

The DQN-based CPU allocation as summarized in Algorithm 3 extends the state space $\mathbf{s}(k)$ as in Algorithm 1 to the experience sequence at time $k$ denoted by $\varphi(k)$ to accelerate the learning speed and improve the APT resistance. More specifically, the experience sequence consists of the current system state $\mathbf{s}(k)$ and the previous $W$ system state-action pairs, i.e., $\varphi(k) = (\mathbf{s}(k-W), \mathbf{M}(k-W), ..., \mathbf{s}(k-1), \mathbf{M}(k-1), \mathbf{s}(k))$. The experience sequence $\varphi(k)$ is reshaped into a $5 \times 5$ matrix and then input into the CNN, as shown in Fig. 3. The CNN consists of two convolutional (Conv) layers and two fully connected (FC) layers, with parameters chosen to achieve a good performance according to the experiment results as listed in Table II. The filter weights of the four layers in the CNN at time $k$ are denoted by $\theta(k)$ for simplicity. The first Conv layer includes 20 different filters. Each filter has size $2 \times 2$ and uses stride 1. The output of the first Conv is 20 different $4 \times 4$ feature maps that are then passed through a rectified linear function ($ReLU$) as an activation function. The second Conv layer includes 40 different filters. Each filter has size $2 \times 2$ and stride 1. The outputs of the 2nd Conv layer are 40 different $3 \times 3$ feature maps, which are flattened to a 360-dimension vector and then sent to the two FC layers. The first FC layer involves 180 rectified linear units, and the second FC layer provides the $Q$-value for each CPU allocation policy $\mathbf{M} \in \Delta_D$ at the current system sequence $\varphi(k)$.

| Layer          | Conv1       | Conv2       | FC1    | FC2    |
|---------------|-------------|-------------|--------|--------|
| Input         | $5 \times 5$| $4 \times 4 \times 20$ | $360$  | $180$  |
| Filter size   | $2 \times 2$| $2 \times 2$ | /      | /      |
| Stride        | $1$         | $1$         | /      | /      |
| # Filters     | $20$        | $40$        | $180$  | $|\Delta_D|$ |
| Activation    | $ReLU$      | $ReLU$      | $ReLU$ | $ReLU$ |
| Output        | $4 \times 4 \times 20$ | $3 \times 3 \times 40$ | $180$  | $|\Delta_D|$ |
The Q-function as the expected long-term reward for the state sequence $\varphi$ and the action $M$, is given by definition as
\[
Q(\varphi^{(k)}, M) = \mathbb{E}_{\varphi'} \left[ u_D^{(k)} + \gamma \max_{M'} Q(\varphi', M') | \varphi^{(k)}, M \right],
\] (31)
where $\varphi'$ is the next state sequence by choosing defense CPU allocation $M$ at state $\varphi^{(k)}$.

To make a tradeoff between exploitation and exploration, the defense CPU allocation is chosen according to the $\varepsilon$-greedy policy. More specifically, the CPU allocation $M^{(k)}$ that maximizes the Q-function is chosen with a high probability $1 - \varepsilon$, and other actions are selected with a low probability to avoid staying in the local maximum, i.e.,
\[
\Pr(M^{(k)} = \hat{M}) = \begin{cases} 1 - \varepsilon, & \hat{M} = \arg \max_{M} Q(\varphi^{(k)}, M) \\ \frac{\varepsilon}{|\Delta_d| - 1}, & \text{o.w.} \end{cases}
\] (32)

Based on the experience replay as shown in Fig. 4, the CPU allocation experience at time $k$ denoted by $e^{(k)}$ is given by $e^{(k)} = (\varphi^{(k)}, M^{(k)}, u_D^{(k)}, \varphi^{(k+1)})$, and saved in the replay memory pool denoted by $D$, with $D = \{e^{(1)}, \ldots, e^{(k)}\}$. An experience $e^{(d)}$ is chosen from the memory pool at random, with $1 \leq d \leq k$. The CNN parameters $\theta^{(k)}$ are updated by the stochastic gradient descent (SGD) algorithm in which the mean-squared error between the network’s output and the target optimal Q-value is minimized with the minibatch updates. The loss function denoted by $L$ in the stochastic gradient descent algorithm is chosen as
\[
L(\theta^{(k)}) = \mathbb{E}_{\varphi, M, u_D^{(k)}, \varphi'} \left[ \left( G - Q(\varphi, M; \theta^{(k)}) \right)^2 \right],
\] (33)
where the target value denoted by $G$ approximates the optimal value $u_D^{(k)} + \gamma Q^*(\varphi', M')$ based on the previous CNN parameters $\theta^{(k-1)}$, and is given by
\[
G = u_D^{(k)} + \gamma \max_{M} Q(\varphi', M'; \theta^{(k-1)}).
\] (34)

The gradient of the loss function with respect to the weights $\theta^{(k)}$ is given by
\[
\nabla_{\theta^{(k)}} L(\theta^{(k)}) = \mathbb{E}_{\varphi, M, u_D, \varphi'} \left[ \nabla_{\theta} Q(\varphi, M; \theta^{(k)}) \right] - \mathbb{E}_{\varphi, M} \left[ Q(\varphi, M; \theta^{(k)}) \nabla_{\theta} Q(\varphi, M; \theta^{(k)}) \right].
\] (35)

This process repeats $H$ times to update $\theta^{(k)}$ in Algorithm 3. Similar to Algorithm 1, we apply the hotbooting technique to initialize the CNN parameters in the DQN-based CPU allocation rather than initializing them randomly to accelerate the learning speed. As shown in Algorithm 4, the defender stores the emulational experience $(\varphi^{(k)}, M^{(k)}, u_D^{(k)}, \varphi^{(k+1)})$ in the database $E$ and the resulting $\hat{\theta}$ based on $\xi$ experiences are used to set $\theta$ as shown in Algorithm 3.

VII. SIMULATION RESULTS

Simulations have been performed to evaluate the APT defense performance of the CPU allocation schemes in a cloud storage system, with the CNN parameters as listed in Table III. In the simulations, some APT attackers applied the $\varepsilon$-greedy algorithm to choose the number of CPUs to attack each of the $D$ storage devices based on the defense history and some
We set an “optimal” defense strategy based on the estimated defense cost and smarter attackers first induced the defender to use a specific defense strategy. The defenders, according to the experiments not presented in this paper. The data protection level and the utility increase with the number of defense CPUs. For example, the PHC-based strategy takes less than 4% of the time to choose the CPU allocation in a time slot compared to the DQN-based scheme. Thus, the PHC-based scheme has a 14.92% higher utility than the PHC-based scheme. As a result, the hotbooting DQN-based scheme has a 30.69% higher utility than the PHC-based scheme. As the hotbooting DQN-based strategy at time slot 1000, which is 30.51% higher than the PHC-based scheme at time slot 1000, which is 30.51% higher than the Q-learning based strategy. As a result, the hotbooting DQN-based scheme has a 30.69% higher utility than the PHC-based scheme at time slot 1000, which is 97.87% higher than that of the Q-learning based scheme.

In the second simulation, the size of the data stored in each of the 3 storage devices of the cloud storage system changed every 1000 time slots. The total data size increases 1.167 times at the 1000-nd time slot and then increases 1.143 times at the 2000-nd time slot. The cloud storage system used 16 CPUs to scan the storage devices and the APT attacker used 4 CPUs to attack them. Besides, the attack policy changed every 1000 time slots. The APT attacker estimated the “optimal” defense strategy. The learning state space by using CNN to accelerate the learning process and enhance the security performance of the cloud storage system. If the interaction time is long enough, the hotbooting PHC and Q-learning scheme can also converge to the NE of the theoretical results in Theorem 2. The PHC-based scheme has less computation complexity than DQN. For example, the PHC-based strategy takes less than 4% of the time to choose the CPU allocation in a time slot compared with the DQN-based scheme.

Algorithm 3 Hotbooting DQN-based CPU allocation

1. Initialize $N^{(0)}$, $B^{(1)}$, $W$ and $H$
2. Set $\theta = \emptyset$, $D = \emptyset$
3. for $k = 1, 2, 3, \ldots$ do
4. Observe the current data size $B^{(k)}$
5. $s^{(k)} = \{N^{(k-1)}, B^{(k)}\}$
6. if $k \leq W$ then
7. Choose $M^{(k)} \in \Delta_D$ at random
8. else
9. $\varphi^{(k)} = (s^{(k-W)}, M^{(k-W)}, \ldots, s^{(k-1)}, M^{(k-1)}, s^{(k)})$
10. Set $\varphi^{(k)}$ as the input of the CNN
11. Observe the output of the CNN to obtain $Q(\varphi^{(k)}, M)$
12. Choose $M^{(k)} \in \Delta_D$ via $\mathbf{32}$
13. end if
14. for $i = 1, 2, \ldots, D$ do
15. Allocate $M_i^{(k)}$ CPUs to scan storage device $i$
16. end for
17. Observe the compromised storage devices and estimate $N^{(k)}$
18. Obtain $u_D^{(k)}$ via $\mathbf{4}$
19. Observe $\varphi^{(k+1)}$
20. $D \leftarrow D \cup \{(\varphi^{(k)}, M^{(k)}, u_D^{(k)}, \varphi^{(k+1)})\}$
21. for $d = 1, 2, 3, \ldots, H$ do
22. Select $(\varphi^{(d)}, M^{(d)}, u_D^{(d)}, \varphi^{(d+1)}) \in D$ at random
23. Calculate $G$ via $\mathbf{34}$
24. end for
25. Update $\theta^{(k)}$ via $\mathbf{35}$
26. end for

Algorithm 4 Hotbooting process for Algorithm 3

1. Initialize $N^{(0)}$, $B^{(1)}$, $\theta^{(0)}$, $\xi$, $K$ and $E = \emptyset$
2. for $i = 1, 2, 3, \ldots, \xi$ do
3. Emulate a similar CPU allocation scenario for the defender to scan storage devices
4. for $k = 1, 2, 3, \ldots, K$ do
5. Observe the output of the CNN to obtain $Q(\varphi^{(k)}, M)$
6. Choose $M^{(k)}$ via $\mathbf{32}$
7. for $i = 1, 2, \ldots, D$ do
8. Allocate $M_i^{(k)}$ CPUs to scan storage device $i$
9. end for
10. Observe the compromised storage devices and estimate $N^{(k)}$
11. Obtain $u_D^{(k)}$ via $\mathbf{4}$
12. Observe the resulting state sequence $\varphi^{(k+1)}$
13. $E \leftarrow E \cup \{(\varphi^{(k)}, M^{(k)}, u_D^{(k)}, \varphi^{(k+1)})\}$
14. Perform minibatch update as steps 19-23 in Algorithm 3 to update $\theta^{(k)}$
15. end for
16. end for
17. Output $\theta \leftarrow \theta^{(k)}$
defender with hotbooting DQN-based APT defense increase by 14.20% and 14.03%, respectively. In the dynamic game with $S_M = 16$, $D = 3$ and $S_N = 4$, the data protection level of the hotbooting DQN-based scheme is 15.85% higher than that of PHC, which is 21.62% higher than Q-learning, and the utility of the hotbooting DQN-based CPU allocation scheme is 15.04% higher than PHC, which is 21.64% higher than Q-learning.

As shown in Fig. 8, the APT defense performance slightly decreases with more number of storage devices $D$ in a cloud storage system. However, the hotbooting DQN-based scheme still maintains a high data protection level, i.e., $R = 92.50\%$, if 4 storage devices are protected by 21 CPUs and attacked by 4 CPUs. In another example, the hotbooting DQN-based scheme can protect up to 80.05% of the data stored in 6 storage devices in the cloud. The defender with less number of CPUs has to distribute its resources among all the storage devices to resist APTs, as at least one CPU has to scan each storage device. The performance gain of the hotbooting DQN-based CPU allocation scheme over the hotbooting PHC-based scheme increases with the number of storage devices in the cloud system.

VIII. CONCLUSION

In this paper, we have formulated a CBG-based CPU allocation game for the APT defense of cloud storage and cyber systems and provided the NEs of the game to show how the number of storage devices, the data sizes in the storage devices and the total number of CPUs impact on the data protection level of the cloud storage system and the defender’s utility. A hotbooting DQN-based CPU allocation strategy has been proposed for the defender to scan the storage devices without being aware of the attack model and the data storage model in the dynamic game. The proposed scheme can improve the data protection level with a faster learning speed and is more robust against smart APT attackers that choose the attack policy based on the estimated defense learning scheme. For instance, the data protection level of the cloud storage system and the utility of the defender increases by 22.29%
and 22.4%, respectively, compared with the Q-learning based scheme in the cloud storage system with 4 storage devices and 16 defense CPUs against an APT attacker with 4 CPUs. A hotbooting PHC-based CPU allocation scheme can reduce the computation complexity.

Fig. 7. APT defense performance of the cloud storage system with $S_M$ defense CPUs, and 3 storage devices that are attacked by 4 attack CPUs, averaged over 3000 time slots. The size of data stored in each storage device changes every 1000 time slots.

Fig. 8. APT defense performance of the cloud storage system with $D$ storage devices and 21 defense CPUs against an APT attacker with 4 attack CPUs, averaged over 3000 time slots. The size of data stored in each storage device changes every 1000 time slots.

REFERENCES

[1] P. Giura and W. Wang, “Using large scale distributed computing to unveil advanced persistent threats,” Science, vol. 1, no. 3, pp. 93–105, 2013.

[2] Y. Han, T. Alpcan, J. Chan, C. Leckie, and B. I. Rubinstein, “A game theoretical approach to defend against co-resident attacks in cloud computing: Preventing co-residence using semi-supervised learning,” IEEE Trans. Inf. Forensics Security, vol. 11, no. 3, pp. 556–570, Dec. 2016.

[3] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, “Enabling public auditability and data dynamics for storage security in cloud computing,” IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 5, pp. 847–859, May 2011.

[4] J. Vukalović and D. Delija, “Advanced persistent threats-detection and defense,” in Proc. IEEE Inf. and Commun. Technol., Electron. and Microelectronics, pp. 1324–1330, Opatija, Croatia, May 2015.

[5] C. Tankard, “Advanced persistent threats and how to monitor and deter them,” Network Security, vol. 8, no. 8, pp. 16–19, Aug. 2011.

[6] R. B. Sagi, “The Economic Impact of Advanced Persistent Threats,” IBM Research Intelligence, May 2014, [http://www-01.ibm.com].

[7] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest, “Flipit: The game of ‘stealthy takeover’,” J. Cryptol., vol. 26, no. 4, pp. 655–713, Oct. 2013.

[8] M. Zhang, Z. Zheng, and N. B. Shroff, “Stealthy attacks and observable defenses: A game theoretic model under strict resource constraints,” in Proc. IEEE Global Conf. Signal and Inf. Processing (GlobalSIP), pp. 813–817, Atlanta, GA, Dec. 2014.

[9] L. Xiao, D. Xu, Y. Li, N. B. Mandayam, and H. V. Poor, “Cloud storage defense against advanced persistent threats: A prospect theoretic study,” IEEE J. Sel. Areas Commun., vol. 35, no. 3, pp. 534–544, Mar. 2017.

[10] L. Xiao, D. Xu, N. B. Mandayam, and H. V. Poor, “Cumulative prospect theoretic study of a cloud storage defense game against advanced persistent threats,” in Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM), BigSecurity Workshop, pp. 1–6, Atlanta, GA, May 2017.

[11] M. Manshadi, Q. Zhu, T. Alpcan, T. Bacsar, and J.-P. Hubaux, “Game theory meets network security and privacy,” ACM Comput. Surv., vol. 45, no. 3, pp. 25:1–25:39, Jul. 2013.

[12] M. Zhang, Z. Zheng, and N. B. Shroff, “A game theoretic model for defending against stealthy attacks with limited resources,” in Proc. Int. Conf. Decision and Game Theory for Security, pp. 93–112, London, UK, Nov. 2015.

[13] B. Roberson, “The Colonel Blotto game,” Econ. Theory, vol. 29, no. 1, pp. 1–24, Sep. 2006.

[14] M. Hajimirsadeghi, G. Sridharan, W. Saad, and N. B. Mandayam, “Inter-
network dynamic spectrum allocation via a Colonel Blotto game,” in Proc. IEEE Annu. Conf. Inf. Sci. Syst. (CISS), pp. 252–257, Princeton, NJ, Mar. 2016.

[15] M. Labib, S. Ha, W. Saad, and J. H. Reed, “A Colonel Blotto game for anti-jamming in the Internet of Things,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), pp. 1–6, San Diego, CA, Dec. 2015.

[16] N. Namvar, W. Saad, N. Bahadori, and B. Kelley, “Jamming in the Internet of Things: A game-theoretic perspective,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), pp. 1–6, Washington, D.C., Dec. 2016.

[17] M. Min, L. Xiao, C. Xie, M. Hajimiradseghli, and N. B. Mandayam, “Defense against advanced persistent threats: A Colonel Blotto game approach,” in Proc. IEEE Int. Conf. Commun. (ICC), pp. 1–6, Paris, France, May 2017.

[18] M. Bowling and M. Veloso, “Rational and convergent learning in stochastic games,” in Proc. Int. Joint Conf. Artificial Intell., pp. 1021–1026, Seattle, D.C., Aug. 2001.

[19] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, Jan. 2015.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, et al., “Playing atari with deep reinforcement learning,” arXiv preprint arXiv:1312.5602, Dec. 2013.

[22] A. Laszka, B. Johnson, and J. Grossklags, “Mitigating covert compromises,” in Int. Conf. Web and Internet Econ., pp. 319–332, Cambridge, MA, Dec. 2013.

[23] Z. Zheng, N. B. Shroff, and P. Mohapatra, “When to reset your keys: Optimal timing of security updates via learning,” in Proc. AAAI Conf. Artificial Intelligence, pp. 1–7, San Francisco, CA, Feb. 2017.

[24] J. Pawlick, S. Farhang, and Q. Zhu, “Flip the cloud: Cyber-physical signaling games in the presence of advanced persistent threats,” in Proc. Int. Conf. Decision and Game Theory for Security, pp. 289–308, London, UK, Nov. 2015.

[25] A. Alabdel Abass, L. Xiao, N. B. Mandayam, and Z. Gajic, “Evolutionary game theoretic analysis of advanced persistent threats against cloud storage,” IEEE Access, vol. 5, pp. 8482–8491, Apr. 2017.

[26] P. Hu, H. Li, H. Fu, D. Cansever, and P. Mohapatra, “Dynamic defense strategy against advanced persistent threat with insiders,” in Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM), pp. 747–755, HK, China, May 2015.

[27] Y. Wu, B. Wang, K. R. Liu, and T. C. Clancy, “Anti-jamming games in multi-channel cognitive radio networks,” IEEE J. Sel. Areas Commun., vol. 30, no. 1, pp. 4–15, Jan. 2012.

[28] P. Chia and J. Chuang, “Colonel Blotto in the phishing war,” Decision and Game Theory for Security, pp. 201–218, 2011.

[29] B. Wang, Y. Wu, K. R. Liu, and T. C. Clancy, “An anti-jamming stochastic game for cognitive radio networks,” IEEE J. Sel. Areas Commun., vol. 29, no. 4, pp. 877–889, Apr. 2011.

[30] C. Thomas, “N-dimensional Blotto game with heterogeneous battlefield values,” Economic Theory, pp. 1–36, Jan. 2017.

[31] L. R. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” J. Artif. Intell. Res., vol. 4, pp. 237–285, 1996.

[32] E. Rodrigues Gomes and R. Kowalczyk, “Dynamic analysis of multiagent Q-learning with ξ-greedy exploration,” in Proc. ACM Annual Int’l. Conf. Mach. Learn. (ICML), pp. 369–376, Montreal, Canada, Jun. 2009.