Defend Jamming Attacks: How to Make Enemies Become Friends

Dinh Thai Hoang[1], Mohammad Abu Alsheikh[2], Shimin Gong[3], Dusit Niyato[4], Zhu Han[5], and Ying-Chang Liang[6]
[1] School of Electrical and Data Engineering, University of Technology Sydney, Australia
[2] Faculty of Science & Technology, University of Canberra, Australia
[3] School of Intelligent Systems Engineering, Sun Yat-sen University, China
[4] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[5] Department of Electrical and Computer Engineering, University of Houston, USA
[6] Center for Intelligent Networking & Communications, University of Electronic Science and Technology, China

***Abstract*—In this paper, we consider a smart jammer that only attacks the channel if it detects activities of legitimate devices on that channel. To cope with such smart jamming attacks, we propose an intelligent deception strategy in which the legitimate transmitter sends fake transmissions to lure the jammer. Then, if the jammer launches attacks to the channel, the legitimate transmitter can either backscatter the jamming signals to transmit data or harvest energy from the jamming signals for future active transmissions. In this way, we can not only undermine the attack ability of the jammer, but also leverage jamming signals as means to enhance system performance. In addition, to find an optimal defense strategy for the legitimate device under uncertainty of wireless environment as well as incomplete information from the jammer, we develop $\mathbb{Q}$-learning and deep $\mathbb{Q}$-learning algorithms based on the Markov decision process. Through simulation results, we demonstrate that our proposed solution is able to not only deal with smart jamming attacks, but also successfully leverage jamming attacks to improve the system performance.**

Keywords- Ambient backscatter, smart jamming, deep $\mathbb{Q}$-learning, energy harvesting, MDP, deception, and $\mathbb{Q}$-learning.

I. Introduction

Over the last 5 years, we have been experiencing an explosion in IoT applications with great influences in many areas such as healthcare, transportation, industry, and agriculture [1]. However, due to broadcast characteristics of wireless communications and hardware constraints, IoT networks are still extremely vulnerable to jamming attacks. In particular, by generating strong signals in the target channel, a jammer can decrease signal-to-interference-plus-noise ratio (SINR) at the IoT gateway, i.e., the legitimate receiver. Consequently, the IoT gateway is unable to decode information from the legitimate transmitter, i.e., the IoT device. More importantly, wireless jamming attacks can be easily launched by using commercial off-the-shelf products [2], and thus they can cause serious consequences to human lives, especially in mission-critical sectors such as healthcare, military, and manufacturing. As a result, solutions to deal with jamming attacks are urgently needed for future development of IoT networks.

Anti-jamming methods can be classified into three main groups, i.e., power control, rate adaptation, and frequency hopping [3]. For power control-based anti-jamming methods, the legitimate transmitter tries to control its transmit power to deal with jamming attacks. Specifically, the transmitter can reduce its transmit power to a very low level so that the jammer cannot detect its transmission, or increase its transmit power to a very high level to significantly improve the SINR at the receiver over jamming signals. These methods are simple and easy to implement at the transmitter, but it is ineffective in the case that the jammer often attacks the channel at high powers. The second technique that is also well investigated in the literature is rate adaptation. In this method, the legitimate transmitter observes jamming attacks and then selects an appropriate transmission rate in which the receiver still can decode information under the jamming signals. However, this technique cannot deal with smart jamming, i.e., the jammer only attacks the channel after it detects activities of legitimate users. Finally, frequency hoping-based anti-jamming techniques have also received a lot of attention in the literature. Basically, these techniques allow legitimate devices to smartly switch to a predefined channel right after the current communication channel is being attacked. Nevertheless, these techniques require a set of multiple available channels as well as switching algorithms implemented on the legitimate devices in advance. Furthermore, these techniques might not be effective if jammers are able to attack multiple channels simultaneously.

Recently, some new techniques have been introduced to improve jamming defense efficiency for legitimate communication systems. In [4], the authors introduced an idea of harvesting energy from jamming signals. In particular, when a jammer performs jamming attack to the communication channel, the legitimate transmitter can harvest energy from jamming signals and use the energy to transmit data when the jammer does not attack the channel. In [5], the authors proposed a new solution adopting ambient backscatter technique [6], [7] to combat jamming attacks. The key idea is that when the jammer attacks the channel, the legitimate transmitter can backscatter modulated jamming signals to transmit data to the receiver. However, both methods proposed in [4] and [5] are applicable only for "static" jammers, i.e., the jammers perform attacks whatever the legitimate devices are active or not, and thus they cannot be widely implemented in practice.

In this paper, we introduce a novel method to deal with smart jamming attacks in which a jammer only attacks the channel if it detects activities of legitimate users on such channel. First, to deal with such smart jamming, we propose

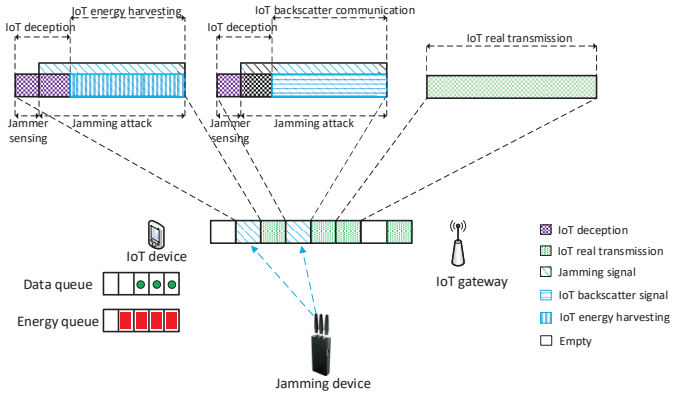


Fig. 1. System model.

a smart anti-jamming strategy developed based on deception tactic in military. The key idea of this strategy is to use fake transmissions to undermine the jammer ability. Furthermore, we introduce intelligent solutions which allow the legitimate device to effectively leverage jamming signals as means to improve performance for the legitimate system. In particular, when the jammer is tricked to perform jamming attacks, the transmitter can either harvest energy from jamming signals or backscatter jamming signals to transmit data using ambient backscatter techniques [6], [7]. In addition, to deal with uncertainty of jamming signals as well as incomplete information about the jammer, we adopt the Markov decision process and QLA to help the transmitter to find an optimal policy through real-time interactions with the jammer. Furthermore, we develop a deep reinforcement learning algorithm which utilizes advantages of neural networks to speed up the learning process, thereby obtaining the optimal policy for the transmitter much faster than that of conventional QLAs. Through simulation results, we show that our proposed solution is able to not only effectively defend smart jamming attacks, but also can make the jammer become a "friend" to improve system performance. Moreover, we show that our approach can achieve much better performance compared with those of other anti-jamming methods in the literature.

II. System Model

We consider a legitimate communication system, e.g., an IoT system, in which an IoT device, i.e., the transmitter, communicates with its gateway, i.e., the receiver, through a dedicated channel denoted by C. The IoT device is equipped with a data buffer and energy storage as illustrated in Fig. 1. We assume that time is slotted, and at each time slot, there are $n_{\rm a}$ packets coming to the IoT device with probability $p_{\rm d}$. When a new packet arrives at the IoT device, if the data queue is not full, the new packet will be stored in the data queue. Otherwise, the incoming data will be discarded. We denote D to be the maximum data queue size of the IoT device.

We assume that the IoT device is equipped with an energy harvesting circuit. This circuit will be used to harvest energy from surrounding environment, e.g., from surrounding RF signals. At each time slot, the IoT device can successfully harvest a unit of energy with probability $p_{\rm e}$. The harvested energy will be then stored in the energy queue, and the IoT device will use the energy to actively transmit packets stored in the data queue (i.e., active transmission mode) to the IoT gateway. We denote E to be the maximum capacity of the energy queue, and $e_{\rm d}$ to be the amount of energy which the IoT device needs to use to transmit a packet. Due to the hardware constraint, we assume that if the IoT selects the active transmission mode, it only can transmit maximum $d_{\rm a}$ packets per time slot.

In this paper, we consider a scenario in which there is a smart jammer [1] coexisting in the communication range of IoT system. The jammer aims to attack the IoT communication channel as much as it can. To do so, at each time slot, if the jammer decides to attack the channel, it will sense the channel to detect IoT communication activities. If the jammer detects that the IoT device is transmitting data to its gateway, the jammer will immediately attack the channel to interrupt the IoT's communication.

Alternatively, at each time slot, the jammer decides to sense the channel with probability p_s. We denote e_s to be the energy which the jammer needs to use to sense the channel and e_a to be the energy which the jammer needs to use to attack the channel. The average amount of energy which the jammer needs to use to sense and attack the channel must satisfy the following condition:

$$\frac{1}{T^{\rm J}}\sum_{t=1}^{T^{\rm J}} E_t^{\rm J} \leq \widehat{E_{\rm J}}, \tag{1}$$

where $T^{\rm J}$ is a predefined time period which the jammer targets to attack the channel, $E_t^{\rm J}$ is the total energy which the jammer uses at time slot t, and $\widehat{E_{\rm J}}$ is the average energy threshold which the jammer can use during period $T^{\rm J}$.

To defend such smart jamming attacks, we propose the novel strategies which can not only avoid jamming attacks, but also leverage jamming attacks as means to improve the IoT system performance. First, at the beginning of each time slot, the IoT device can choose either actively transmit real data to the IoT gateway or send a fake message to attract jamming attacks. If the IoT device chooses to perform the deception by sending the fake transmission, it can possibly deceive the jammer to attack and weaken the jammer by wasting its energy. Hence, when the IoT device transmits real data, the jammer may not be able to attack. We denote $e_{\rm f}$ and $e_{\rm r}$ to be the amount of energy that the IoT device needs to perform deception and real data transmission, respectively. In practice, $e_{\rm f} < e_{\rm r}$ because the IoT device only needs to use a fraction of a time slot to deceive the jammer.

After the IoT device performs deception, it will observe the channel for a short period of time. If the jammer attacks the channel, the IoT device can select either to harvest energy from jamming signals and store in the energy queue or to backscatter jamming signals to transmit data to the IoT gateway. If the IoT device selects to harvest energy from jamming signals, it can harvest $e_{\rm h}^{\rm J}$ units of energy. In contrast,

[1]We can extend this scenario by considering multiple smart jammers that are able to coordinate to attack the channel.

if the IoT device selects to backscatter jamming signals to transmit data, it can transmit $d_{\mathrm{b}}^{\mathrm{J}}$ packets. Note that during backscattering process, the IoT device still can harvest energy to server for its operation, so we do not need to consider energy consumption during backscattering communications at the IoT device. Furthermore, to improve the effectiveness of attacks, jammers are often located near the target system and use high transmit powers to perform attacks. Thus, the efficiency of harvesting energy as well as backscatter communications based on jamming signals is usually significantly greater than that of ambient RF signals, e.g., from a cellphone station or from a Wi-Fi access point. Alternatively, in the case that the IoT device performs deception, but the jammer does not attack the channel, the IoT device remains idle to save energy in the rest of time slot.

III. Problem Formulation

In this section, we adopt the MDP framework to formulate the energy and communication control problem for IoT device.

A. State Space

The state space of the IoT system is defined based on two main factors of the IoT device, i.e., energy and data, as follows:

$$\begin{aligned}\mathbb{S} \triangleq \Big\{ &(d,e) : d \in \mathscr{D} \triangleq \{0,\ldots,d,\ldots,D\}; \\ &\text{and} \quad e \in \mathcal{E} \triangleq \{0,\ldots,e,\ldots,E\}\Big\},\end{aligned} \tag{2}$$

where $d \in \mathscr{D}$ and $e \in \mathcal{E}$ represent the numbers of packets in the data queue and the energy units in the battery, respectively. D and E are the maximum size of the data queue and the energy storage, respectively. The system state is then defined as a composite variable $s \triangleq (d,e) \in \mathbb{S}$.

B. Action Space

The IoT device can perform one of the four actions, i.e., stay idle, transmit real data, perform deception and harvest energy if the jammer attacks the channel, and perform deception and backscatter jamming signals to transmit data to the gateway if the jammer attacks the channel. Then, the action space can be defined by $\mathbb{A} \triangleq \{a : a \in \{1,\ldots,4\}\}$, where

$$a = \begin{cases} 1, & \text{stays idle,} \\ 2, & \text{transmit real data,} \\ 3, & \text{perform deception and harvest energy,} \\ 4, & \text{perform deception and backscatter.} \end{cases} \tag{3}$$

C. Reward Function

The reward of the IoT device is defined as the number of packets that it can send to the IoT gateway. Thus, the immediate reward of the IoT device after an action a is executed at state s is defined as follows:

$$r = \begin{cases} d_{\mathrm{a}}, & a = 2 \text{ and there is no attack,} \\ d_{\mathrm{b}}^{\mathrm{J}}, & a = 4 \text{ and jammer attacks the channel,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

In this paper, we aim to find an optimal policy for the IoT device, denoted by Υ^*, to maximize its long-term average reward defined as follows:

$$\max_{\Upsilon} \qquad \mathcal{T}(\Upsilon) = \lim_{\mathsf{T}\to\infty} \frac{1}{\mathsf{T}} \sum_{t=1}^{\mathsf{T}} \mathbb{E}\left(r_t(\Upsilon)\right), \tag{5}$$

where $\mathcal{T}(\Upsilon)$ is the long-term average throughput of the IoT system under the policy Υ. $r_t(\Upsilon)$ is the immediate throughput under policy Υ at time step t. The optimal policy Υ^* will allow the IoT device to make its optimal decisions given its current states, i.e., the current energy level and the number of packets waiting in the data queue.

IV. Q Learning-based Deception Strategy

To deal with the uncertainty of jamming attacks, we develop a reinforcement learning algorithm based on the QLA (QLA) [10] to help the IoT device find the optimal policy without requiring any information about the jammer in advance. For the QLA, a $\mathbb{Q}$-table is made with the number of rows and columns corresponding to the number of actions and states, respectively. As a result, this $\mathbb{Q}$-table is able to store all values of state-action pairs. In order to maximize the objective function, i.e., the long-term average throughput of IoT system, we need to find a dynamic optimal policy which allows the IoT device to make decisions based on its current observed states. Specifically, given the current state, i.e., the number of packets waiting in the queue and the energy level, the IoT device will make a decision based on its current policy such that its objective function is maximized. Then, based on observations about the immediate throughput and next states, the IoT device will update the $\mathbb{Q}$-table. This process is repeated until the values of $\mathbb{Q}$-table converge. If we denote $\mathscr{V}^{\Upsilon}(s)$ to be the expected value function obtained by policy Υ at state $s \in \mathbb{S}$, then following Bellman equation, we can derive this function as follows:

$$\begin{aligned}\mathscr{V}^{\Upsilon}(s) &= \mathbb{E}_{\Upsilon}\Big[\sum_{t=0}^{\infty} \gamma^t r_t(s_t,a_t) | s_0 = s\Big] \\ &= \mathbb{E}_{\Upsilon}\Big[r_t(s_t,a_t) + \gamma \mathscr{V}^{\Upsilon}(s_{t+1}) | s_0 = s\Big],\end{aligned} \tag{6}$$

where $r_t(s_t,a_t)$ is the immediate reward obtained after action a_t is executed at state s_t, and γ is the discount factor that satisfies $\gamma \in [0,1]$ [10]. At each state s, an optimal action is made using the optimal value function as follows:

$$\mathscr{V}^*(s) = \max_{a_t} \Big\{\mathbb{E}_{\Upsilon}[r_t(s_t,a_t) + \gamma \mathscr{V}^{\Upsilon}(s_{t+1})]\Big\}, \quad \forall s \in \mathbb{S}. \tag{7}$$

Then, we can obtain the optimal $\mathbb{Q}$-functions by [10]:

$$\mathscr{Q}^*(s,a) \triangleq r_t(s_t,a_t) + \gamma \mathbb{E}_{\Upsilon}[\mathscr{V}^{\Upsilon}(s_{t+1})], \quad \forall s \in \mathbb{S}. \tag{8}$$

Here, we can express $\mathscr{V}^*(s)$ as follows:

$$\mathscr{V}^*(s) = \max_{a} \{\mathscr{Q}^*(s,a)\}. \tag{9}$$

As a result, we can update $\mathbb{Q}$-function by generating samples iteratively to find the temporal difference between the predicted Q-value and its current value as follows:

$$\mathscr{Q}_t(s_t,a_t) = \mathscr{Q}_t(s_t,a_t) + \alpha_t\Big[r_t(s_t,a_t) + \gamma \max_{a_{t+1}} \mathscr{Q}_t(s_{t+1},a_{t+1}) - \mathscr{Q}_t(s_t,a_{s_t})\Big], \tag{10}$$

where α_t is the learning rate which satisfies (11) to guarantee the convergence for the QLA [10].

$$\alpha_t \in [0,1), \sum_{t=1}^{\infty} \alpha_t = \infty, \text{ and } \sum_{t=1}^{\infty} (\alpha_t)^2 < \infty. \tag{11}$$

Under the update rule in (10) and learning rate condition given in (11), it is proved that the QLA will converge to the optimal solution with probability one [10]. Although the QLA can help the IoT device make optimal decisions, it usually experiences a long learning period to achieve the optimal defense policy. Thus, in the next section, we develop the deep QLA (DQLA) [12] which allows the IoT device to obtain the optimal quickly through utilizing advantages of the neural network architecture.

V. Defense Strategy using Deep $\mathbb{Q}$-learning

In this section, we first develop a deep neural network instead of using the $\mathbb{Q}$-table to approximate the values of $\mathscr{Q}^*(s,a)$. Algorithm 1 provides the details of DQLA with experience replay mechanism. In particular, the algorithm first initializes a replay memory pool $\mathbb{D}$ which contains transitions, i.e., experiences expressed by a set of (s_t,a_t,r_t,s_t). These experiences can be generated randomly using ϵ-greedy policy. Then, the DQLA selects transitions in the pool $\mathbb{D}$ randomly to train the deep neural network. After that, we can obtain $\mathbb{Q}$-values based on the trained neural network, and use these $\mathbb{Q}$-values to obtain new experiences.

Algorithm 1 The DQLA with Experience Replay and Fixed Target $\mathbb{Q}$-Network

1: Initialize replay memory $\mathbb{D}$ with capacity $\mathscr{D}$.
2: Initialize $\mathbb{Q}$-network $\mathbf{Q}$ with random weights $\mathbf{\Theta}$.
3: Initialize the target $\mathbb{Q}$-network $\hat{\mathbf{Q}}$ with random weights $\mathbf{\Theta'}$.
4: **for** t = 1 *to* T **do**
5: Select a random action a_t with probability ϵ, otherwise select $a_t = \arg\max \mathscr{Q}^*(s_t,a_t,\mathbf{\Theta})$.
6: Perform action a_t and observe immediate reward r_t and next state s_{t+1}.
7: Store transition (s_t,a_t,r_t,s_{t+1}) in $\mathbb{D}$.
8: Select randomly samples (s_i,a_i,r_i,s_j) from $\mathbb{D}$.
9: The weights of the neural network then are optimized by using stochastic gradient descent with respect to the network parameter $\mathbf{\Theta}$ to minimize the loss:

$$\Big[r_i + \gamma \max_{a_j} \hat{\mathscr{Q}}(s_j,a_j;\mathbf{\Theta'}) - \mathscr{Q}(s_i,a_i;\mathbf{\Theta})\Big]^2. \tag{12}$$

10: Reset $\hat{\mathbf{Q}} = \mathbf{Q}$ after every a fixed number of steps.
11: **end for**

In the proposed DQLA, we define two crucial features extracted from samples to use as inputs of the neural network. These features are data and energy which are two main dimensions of the state space. As a result, these features can cover all aspects of the state space, and thus the training process will be much more efficient. In this way, the loss function of neural network will be minimized as follows:

$$\mathcal{L}(\mathbf{\Theta}) = \mathbb{E}_{(s_i,a_i,r_i,s_j)\sim U(\mathbb{D})}\Bigg[\Bigg(r_i + \gamma \max_{a_j} \mathscr{Q}(s_j,a_j;\mathbf{\Theta'}) - \mathscr{Q}(s_i,a_i;\mathbf{\Theta})\Bigg)^2\Bigg], \tag{13}$$

where γ is the discount factor as defined in the previous section. $\mathbf{\Theta}$ and $\mathbf{\Theta'}$ express the parameters of $\mathbb{Q}$-network and target network $\hat{\mathscr{Q}}$, respectively. Then, the following results can be obtained based on the gradient technique:

$$\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}) = \mathbb{E}_{(s_i,a_i,r_i,s_j)}\Bigg[\Bigg(r_i + \gamma \max_{a_j} \mathscr{Q}(s_j,a_j;\mathbf{\Theta'}) - \mathscr{Q}(s_i,a_i;\mathbf{\Theta})\nabla_{\mathbf{\Theta}}\mathscr{Q}(s_i,a_i;\mathbf{\Theta})\Bigg)\Bigg]. \tag{14}$$

In order to minimize the loss function (13), the Stochastic Gradient Descent Algorithm (SGDA) is adopted because this is the most effective solution to calculate the gradient for deep learning algorithms [13]. Based on this algorithm, we can obtain the negative conditional log-likelihood based on the training dataset by:

$$\begin{aligned}\mathcal{J}(\mathbf{\Theta}) &= \mathbb{E}_{(s_i,a_i,r_i,s_j)\sim U(\mathbb{D})}\mathcal{L}\big((s_i,a_i,r_i,s_j);\mathbf{\Theta}\big),\\ &= \frac{1}{\mathscr{D}}\sum_{i=1}^{\mathscr{D}}\mathcal{L}\big((s_i,a_i,r_i,s_j);\mathbf{\Theta}\big),\end{aligned} \tag{15}$$

where $\mathscr{D}$ is the size of the memory pool. After that, we can obtain its gradient function as follows:

$$\nabla_{\mathbf{\Theta}}\mathcal{J}(\mathbf{\Theta}) = \frac{1}{\mathscr{D}}\sum_{i=1}^{\mathscr{D}}\nabla_{\Theta}L\big((s_i,a_i,r_i,s_j);\mathbf{\Theta}\big). \tag{16}$$

In Eq. (16), the computing complexity strongly depends on the variable $\mathscr{D}$. Thus, the higher the value of $\mathscr{D}$ is, the longer time the gradient process takes. Thus, by using the SGDA, we can mitigate this problem. In particular, the key idea of the SGDA is using an expectation for the gradient. As a result, even with a small number of samples, the algorithm still can estimate the expectation efficiently. Based on this idea, the algorithm first uniformly samples a small number of samples (i.e., mini-batch of experiences) from the memory pool $\mathbb{D}$. Then, these samples will be used to estimate the gradient. In this way, the training time of DQLA can be remarkably reduced. Then, the gradient estimation of SGDA can be formulated as follows:

$$\mathcal{G} = \frac{1}{\mathscr{B}}\nabla_{\mathbf{\Theta}}\sum_{i=1}^{\mathscr{B}}\mathcal{L}\big((s_i,a_i,r_i,s_j);\mathbf{\Theta}\big), \tag{17}$$

where $\mathscr{B}$ is the size of a mini-batch. Finally, values of $\mathbf{\Theta}$ can be updated in the following way to minimize the lost function:

$$\mathbf{\Theta} \leftarrow \mathbf{\Theta} - \nu\mathcal{G}, \tag{18}$$

where ν is the learning rate of the DQLA. Note that, after every C steps, Algorithm 1 updates the target network parameters $\boldsymbol{\Theta}'$ with $\mathbb{Q}$-network parameters $\boldsymbol{\Theta}$. However, the target network parameters remain unchanged between individual updates.

VI. Performance Evaluation

A. Experiment Setup

In the system under consideration, the maximum energy and data queues of the IoT device are set to be 10 units. In particular, the battery can store up to 10 units of energy with each energy unit set to be $60\mu J$ [8], and the data buffer can store up to 10 packets with each packet size set to be 300 bits [9]. Other parameters are provide in Table I. For Algorithm 1, we adopt parameters based on the common settings for designing neural networks [12], [13]. In the simulations, we vary two parameters, i.e., p_s and p_d, since they are key parameters to evaluate the influences of jammer and data on the system performance, respectively. Furthermore, to evaluate the efficiency of the proposed solution, i.e., Algorithm 1, we compare with three other strategies [4], [5]: (1) Deception and Backscatter, (2) Deception and Harvest Energy, and (3) Without Deception. To make fair comparisons, for first two strategies, their optimal policies are also obtained by Algorithm 1. For the last strategy, i.e., Without Deception, the transmitter will transmit data as long as it has data and sufficient energy.

TABLE I
Parameter Setting.

Symbol	d_{a}	n_{a}	e_{d}	E_t^{J}	e_{f}	e_{r}	$e_{\mathrm{h}}^{\mathrm{J}}$	$d_{\mathrm{b}}^{\mathrm{J}}$	T^{J}	p_{e}
Value	3	2	1	6	1	3	3	1	∞	0.3

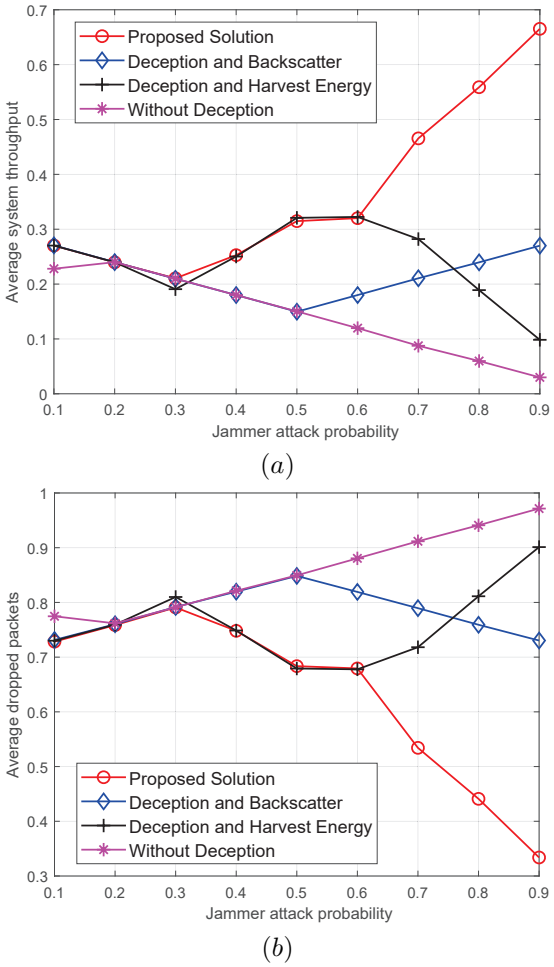
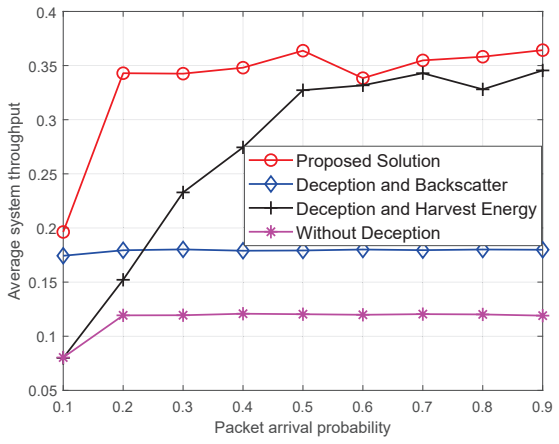


Fig. 2. Vary jamming attack probability.

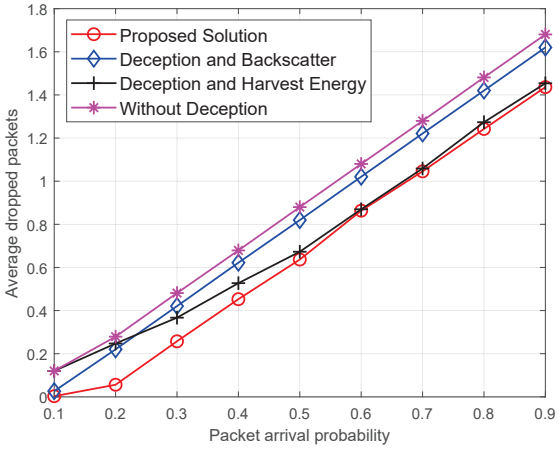
B. Simulation Results

In Fig. 2, we fix the packet arrival probability at 0.5 and vary the jamming capacity to evaluate the system performance in terms of average throughput. As shown in Fig. 2(a), as the jammer attack probability increases from 0.1 to 0.3, the average throughput obtained by the proposed solution slightly decreases from 0.28 to 0.21. However, interestingly, when the jammer attack probability increases from 0.3 to 0.9, the average throughput obtained by the proposed solution increases significantly from 0.21 to 0.67. The main reason is that when the jammer often attacks the channel, the IoT device can leverage jamming signals to improve the system performance through harvesting energy or backscattering signals for communications. However, when the jammer does not frequently attack the channel, the IoT device only can use its harvested energy from the environment for real data transmissions and deception activities. Consequently, when the jammer attacks the channel with a small probability, the average system throughput has a slight decrease. Compared with other strategies, the deception strategy together with energy harvesting from jamming signals achieves the results close to that of the proposed solution when the jammer attack probability is lower than 0.6. However, when the jammer attack probability is higher than 0.6, the average throughput obtained by the deception and energy harvesting is reduced. The reason is that in this case, the IoT device has a low chance to actively transmit data to the gateway. This result also clearly shows the advantage of our proposed solution, i.e., the IoT device is able to optimize energy harvesting and backscattering processes, thereby maximizing average throughput for the system. In Fig. 2(b), we show that not only average system throughput, but also the average dropped packets obtained by our proposed solution can be reduced significantly, and it is much lower than those of other strategies.

In Fig. 3, we fix the jammer attack probability at 0.6 and vary the packet arrival probability in order to evaluate the system performance in terms of average throughput and dropped packets. As observed in Fig. 3(a), when the packet arrival probability is small, i.e., less than 0.3, the performance of deception with backscatter communication strategy is greater than that of the deception with energy harvesting strategy. However, when the packet arrival probability increases, the average throughput obtained by the deception with energy harvesting is greatly improved and doubles that of the deception with backscatter when the packet arrival probability is higher than 0.5. The reason is that when the packet arrival probability is low and the jammer often attacks the channel, backscattering jamming signals to transmit data



(a)



(b)

Fig. 3. Vary packet arrival probability.

is more efficient than harvesting energy from jamming signals because the number of packets that can be simultaneously transmitted in one time slot is few and the chance for IoT device to actively transmit real data is low. Nevertheless, when the packet arrival probability is high, the IoT can actively transmit maximum d_{a} packets per time slot. As a result, harvesting energy from jamming signals and then using the energy for actively transmission is much more efficient than that using backscattering technique. Again, in all the results, our proposed solution always achieves the best performance because it can optimally control deception, energy harvesting and backscattering activities. It is also important to note that as shown in Fig. 2 and Fig. 3 without using the deception strategy, the IoT system is unable to resist smart jamming attacks, yielding the poorest performance compared with those using deception strategy.

VII. Summary

In this paper, we have introduced the intelligent method to defend smart jamming attacks. First, we have introduced the new idea of using deception strategy which not only undermines jammer's capacity, but also leverages jamming signals as effective means to improve system performance. Specifically, when the jammer is tricked to perform jamming attacks, the IoT device can either backscatter these signals to transmit data or harvest energy from these signals. In addition, to deal with uncertainly and incomplete information from the jammer, we have adopted the MDP framework and developed the deep reinforcement learning algorithm which can help the IoT quickly obtain the optimal policy. The simulation results then clearly show the effectiveness and outperformance of our proposed solution.

Acknowledgements

This work was supported in part by A*STAR-NTU-SUTD Joint Research Grant Call on Artificial Intelligence for the Future of Manufacturing RGANS1906, WASP/NTU M4082187 (4080), Singapore MOE Tier 1 under Grant 2017-T1-002-007 RG122/17, MOE Tier 2 under Grant MOE2014-T2-2-015 ARC4/15, Singapore NRF2015-NRF-ISF001-2277, and Singapore EMA Energy Resilience under Grant NRF2017EWT-EP003-041. This work is also partially supported by US MURI AFOSR MURI 18RT0073 , NSF CNS-1717454, CNS-1731424, CNS-1702850, CNS-1646607

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, Sept. 2013.

[2] M. K. Hanawal, M. J. A.-Rahman, and M. Krunz, "Joint adaptation of frequency hopping and transmission rate for anti-jamming wireless systems," *IEEE Trans. Mobile Computing*, vol. 15, no. 9, pp. 2247-2259, Sept. 2016,.

[3] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 42-56, Dec. 2009.

[4] J. Guo, N. Zhao, F. R. Yu, X. Liu, and V. C. M. Leung, "Exploiting adversarial jamming signals for energy harvesting in interference networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1267-1280, Feb. 2017.

[5] N. V. Huynh, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Jam me if you can: Defeating jammer with deep dueling neural network architecture and ambient backscattering augmented communications," arXiv preprint arXiv:1904.03897. 2019 Apr 8.

[6] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," in *ACM SIGCOMM*, pp. 39-50, Hong Kong, Aug. 2013.

[7] N. V. Huynh, D. T. Hoang, X. Lu, D. Niyato, P. Wang, and D. I. Kim, "Ambient backscatter communications: A contemporary survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2889-2922, Jan. 2018.

[8] G. Papotto et al., "A 90-nmCMOS 5-Mbps crystal-Less RF-powered transceiver for wireless sensor network nodes," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 2, pp. 335-346, Feb. 2014.

[9] P. Blasco, D. Gunduz, and M. Dohler, "A learning theoretic approach to energy harvesting communication system optimization," *IEEE Transactions Wireless Communications*, vol. 12, no. 4, pp. 1872-1882, Apr. 2013.

[10] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279-292, 1992.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.