

# Multi-agent deep reinforcement learning-based key generation for graph layer security

LIANG WANG, School of Cybersecurity, Northwestern Polytechnical University, China

ZHUANGKUN WEI, Imperial College London, United Kingdom

WEISI GUO, Cranfield University, United Kingdom

Recently, the emergence of Internet of Things (IoT) devices has posed a challenge for securing information and avoiding attacks. Most of the cryptography solutions are based on physical layer security (PLS), whose idea is to fully exploit the properties of wireless channel state information (CSI) for generating symmetric keys between two communication nodes. However, accurate channel estimation is vulnerable for attackers and relies on powerful signal processing capability, which is not suitable for low-power IoT devices. In this paper, we expect to apply graph layer security (GLS) to exploit the common features of physical dynamics detected by IoT sensors placed in networked systems to generate keys for data encryption and decryption, which we believe is a new frontier to security for both industry and academic research. We propose a distributed key generation algorithm based on multi-agent deep reinforcement learning (MADRL) approach, which enables communication nodes to cooperatively generate symmetric keys based on their locally detected physical dynamics (e.g., water/gas/oil/electrical pressure/flow/voltage) with low computational complexity and without information exchange. In order to demonstrate the feasibility, we conduct and evaluate our key generation algorithm in both a simulated and real water distribution network. The experimental results show that the proposed algorithm has considerable performance in terms of randomness, bit agreement rate (BAR), etc.

CCS Concepts: • **Security and privacy** → **Key management**.

Additional Key Words and Phrases: Multi-agent deep reinforcement learning, Physical layer security, Graph layer security, IoT devices, and Physical dynamics.

## 1 INTRODUCTION

With recent advances in embedded systems and wireless sensor networks, Internet of Things (IoT) devices are becoming vital in most of the networked infrastructures, which serve our cities, agriculture, industries, and homes [46]. Specifically, through deploying wireless IoT devices that are buried underground or embedded in infrastructures, the monitored physical information (e.g., oil/water pressure) can be collected to maintain the infrastructures and ensure the safety. However, encrypting these kinds of data is important and challenging for national security, commerce, etc. Current wireless transmissions are based on Long Range Wide-Area Network

Authors' addresses: Liang Wang, School of Cybersecurity, Northwestern Polytechnical University, Xian, Shaanxi Province, China, liang.wang.133@nwpu.edu.cn; Zhuangkun Wei, Imperial College London, London, United Kingdom, zhuangkun.wei@imperial.ac.uk; Weisi Guo, Cranfield University, Milton Keynes, United Kingdom, weisi.guo@cranfield.ac.uk.

(LoRaWAN) [3], Bluetooth and ZigBee, which are vulnerable to eavesdroppers/adversaries. Thus, the challenge lies in the encryption mechanism in order to protect the data confidentiality [27]. The fundamental of the traditional solutions, such as the advanced symmetric standard (AES), is to generate a pair of secret keys shared between two communication nodes in advance. It is worth noting that public key cryptography (PKC) is needed, which employs public keys for encryption and secret keys for decryption. The traditional PKC-based solutions have considerable performance in protecting modern communication networks. However, applying PKC in IoT still has challenges. On one hand, traditional PKC-based solutions and protocols are normally complicated and require powerful computational capabilities, which are not suitable for lightweight IoT devices, due to the constraints of their computational resources for noise generator, public key distribution, and nonlinear optimization-based problem solver for decryption [30]. On the other hand, as most PKC-based solutions are based on Internet Protocol (IP) and Media Access Control (MAC) addresses for authentications, attackers can easily tamper the addresses by employing malware [44].

### 1.1 Physical Layer Security

Physical layer security (PLS) has been proposed in recent years as a way to overcome many of the aforementioned challenges, which can be categorized as key-less PLS [9, 10] and physical layer secret key generation (PL-SKG) [13–15, 24, 42]. Key-less PLS leverages the superiority of legitimate channels over wiretap ones and tries to maximize the secrecy rate or signal to interference plus noise ratio (SINR), by steering the variables, e.g., beamforming vectors, trajectory of autonomous system, and even the phases of a reconfigurable intelligent surface (RIS). Additionally, Artificial Noise (AN)-based transmission is viewed as a promising technology to realize secure communication, which is independent of the eavesdropper's channel status. In [39], a secure precoding optimization problem is studied for the AN scheme in Multiple-Input Single-Output (MISO) communication system, which is solved by a novel Deep Neural Network (DNN)-based secure precoding scheme. In [6], Gao et al. propose a DNN based self-interference cancellation (SIC)-free receiver AN assisted mechanism in Terahertz (THz) communication to address the in-beam security challenge. However, one challenge lies in that the superiority of the legitimate channel may not be always satisfied. For example, an Eve can make the channel being adversary to legitimate nodes by introducing an Eve-controlled RIS. In such cases, key-less PLS cannot guarantee the existence of feasible solutions for secrecy rate and SINR maximization.

Another family in PLS is PL-SKG, which exploits the common channel randomness to generate the shared secret key between legitimate nodes, without the need for a PKC distribution protocol [27, 33]. PL-SKG negates the risk of key intercept and high computational requirement of PKC schemes, which makes it very suitable for IoT devices. PL-SKG does however require that the wireless channel between nodes to be reciprocal (true for most propagation environments), random (small-scale fading), and unique [5, 40]. This is to ensure robust symmetric key generation and avoid brute-force attempts. Although PL-SKG has been applied to a variety of embedded wireless and wired communication systems [2, 8, 12, 29, 47], the shortfalls are always overlooked. First, the common features for data encryption are not detached from the communications, e.g., using channel phases or received signal strength (RSS), which may provide the chances of sophisticated eavesdroppers to decode the secret key [34, 35]. Second, for both key-less and PL-SKG, they require the IoT devices to make accurate estimations of the wireless channel statistics [44]. Accurate estimation requires reasonably powerful signal processing units and also requires a reasonably high communication signal to noise ratio (SNR). Many embedded or underground IoT devices operate in low communication SNR regimes, and therefore PLS is not suitable.

### 1.2 Graph Layer Security

In order to tackle the challenges that PKC requires extensive computational resources for noise generator, public key distribution and PLS requires high SINR for wireless channel estimation, we expect to apply graph layer

security (GLS) [36]. Motivated by the fundamentals that applying common heartbeat in different IoT devices across a body for key generation [45], the novel idea of GLS is to fully exploit the common features of physical dynamics detected by IoT sensors embedded in networked infrastructures, such as oil/gas/water networks. Precisely, it is widely acknowledged that most of physical dynamics in networked infrastructures can be described by common equations (e.g., a Navier-stokes equation for fluid mechanics, nonlinear schrödinger equation for optical fibres, drag equation for aerodynamics), which further motivate us to employ local physical signals detected by IoT devices to encrypt the data of communication nodes. Thus, compared with the PKC-based and PLS-based solutions, the GLS-based solution has the following advantages:

- Most IoT devices are cost-friendly but accurate enough to measure physical signals.
- The communication nodes do not require wireless channel estimation or information exchange for encryption/decryption.
- The generated keys are based on the physical signals monitored by the dedicated IoT devices, which enable the prevention of attacks from adversaries.

### 1.3 Multi-Agent Deep Reinforcement Learning based Systems

Recently, deep reinforcement learning (DRL), the combination of DNNs and reinforcement learning (RL) has become a popular topic. In [23], Mnih et al. succeed to implement the DNNs in RL and their proposed deep Q network (DQN) algorithm has remarkable performance in playing Atari 2600 games. Then, based on the work that has done in DQN, Lillicrap et al. [17] proposed a deep deterministic policy gradient (DDPG) algorithm for handling continuous control problems, which involves actor-critic approach.

However, the traditional DRL approaches are normally implemented in single-agent systems. As each of agents' policy keeps changing for exploration and exploitation during the training process, the environment is non-stationary from any of agents' view. Hence, the research of applying DRL in multi-agent systems is still challenging. Fortunately, multi-agent deep reinforcement learning (MADRL) is viewed as a promising direction and many contributions of MADRL have been widely acknowledged in wireless communication networks. In [26], Peng et al. investigated the resource management problem in mobile edge computing (MEC)- and unmanned aerial vehicle (UAV)-assisted vehicular networks, where they proposed a multi-agent deep deterministic policy gradient (MADDPG)-based algorithm to tackle the vehicle association and resource allocation decision problem. In [16], Liang et al. studied the spectrum sharing problem in vehicular networks, where a spectrum reuse mechanism was proposed among vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) links. In [25], the authors investigated the potential of applying MADRL in solving the transmit power control problem in wireless networks and they proposed a distributed MADRL-based power allocation scheme.

### 1.4 Contribution

We summarize the main contributions of this paper as follows:

- Different from the traditional PLS-based key generation approaches that rely on CSI, GLS is applied to utilize the randomness of physical dynamics across the networked infrastructure for key generation.
- We propose a key generation algorithm based on MADRL approach, which enables communication nodes to learn cooperatively to exploit common features of their detected sensor readings between each other. When the training process is completed, the communication nodes are controlled by their dedicated agents and can directly generate keys in a distributed manner without relying on frequent information exchange.
- We evaluate the performance of our proposed approach in both a simulated and real water distribution network. The experimental results prove that our proposed scheme performs well in terms of performance metrics of key generation, such as randomness, and bit agreement rate (BAR) after adequate training.

Table 1. Comparison between our work and the existing literature

| Reference | PLS/GLS | Secret key | No channel estimation | Physical dynamics                                 | Types of information exchange |
|-----------|---------|------------|-----------------------|---------------------------------------------------|-------------------------------|
| [37]      | PLS     | ✓          |                       | frequency selectivity of multipath fading channel | channel state                 |
| [19]      | PLS     | ✓          |                       | channel response                                  | CSI                           |
| [20]      | PLS     | ✓          |                       | RSS                                               | RSS                           |
| [45]      | PLS     | ✓          | ✓                     | heartbeat signal                                  | syndrome, CRC chechsum        |
| [41]      | PLS     | ✓          |                       | randomness of wireless channel                    | RTS/CTS                       |
| [43]      | PLS     | ✓          |                       | RSSI                                              | RSSI                          |
| [38]      | PLS     | ✓          | ✓                     | acceleration in BAN                               | —                             |
| [34]      | PLS     | ✓          |                       | random Gaussian matrices                          | —                             |
| [36]      | GLS     |            | ✓                     | speed deviation in IEEE 39-Bus system             | —                             |
| Our work  | GLS     | ✓          | ✓                     | water pressure                                    | —                             |

We further summarize the difference between our work and the existing literature in Table 1. Notably, the work in [45] is based on wireless body area network (WBAN), which utilizes the interpulse intervals (IPIs) between two adjacent peaks of heartbeat signals to provide randomness. However, the IPIs of heartbeat signals always remain the same in any places of body, which means the digital attacks could easily be conducted from any places of WBAN. In addition, the key generation scheme proposed in [45] still requires information exchanges for IPI extraction and key establishment. In [38], the authors utilize accelerometers of movement in WBAN as the physical dynamics and propose an artificial intelligence (AI)-based key generation scheme, which eliminates the information exchange. Essentially, their work is still a ‘manual’ key generation approach, which deploys a well-trained neural network at one communication node to predict the physical dynamics at the opposite node. As a result, only randomness of physical dynamics from one node is applied. Then, we introduce the system model in Section II. The rest of this paper is organized as follows. we demonstrate the proposed algorithm in Section III. The experimental results and analyses are given in Section IV. Finally, we give the conclusion and future work in Section V.

## 2 SYSTEM MODEL

In this section, we briefly introduce the system model. The networked system (infrastructure) could be gas/oil/water networks. In this paper, we consider a water distribution network as the networked system, whose overall structure can be seen in Fig. 1. Specifically, there are a variety of pipes and node junctions deployed, each of which equips an IoT sensor for monitoring the physical signals (e.g., flow rate, pressure) with a fixed time interval  $t^i$ . The node junctions are viewed as communication nodes, which expect to communicate with others over the wireless channels by applying encryption and decryption techniques for securing data. For simplicity, we choose  $N$  communication nodes and each of nodes will send/receive data to/from the rest of nodes in  $T$  time steps. Precisely, in time step  $t$ , node  $n$  generates  $N - 1$  pairs of keys for the rest nodes simultaneously according to the detected physical signals  $X_n(t) \in \mathbb{R}^M$ , which is described as

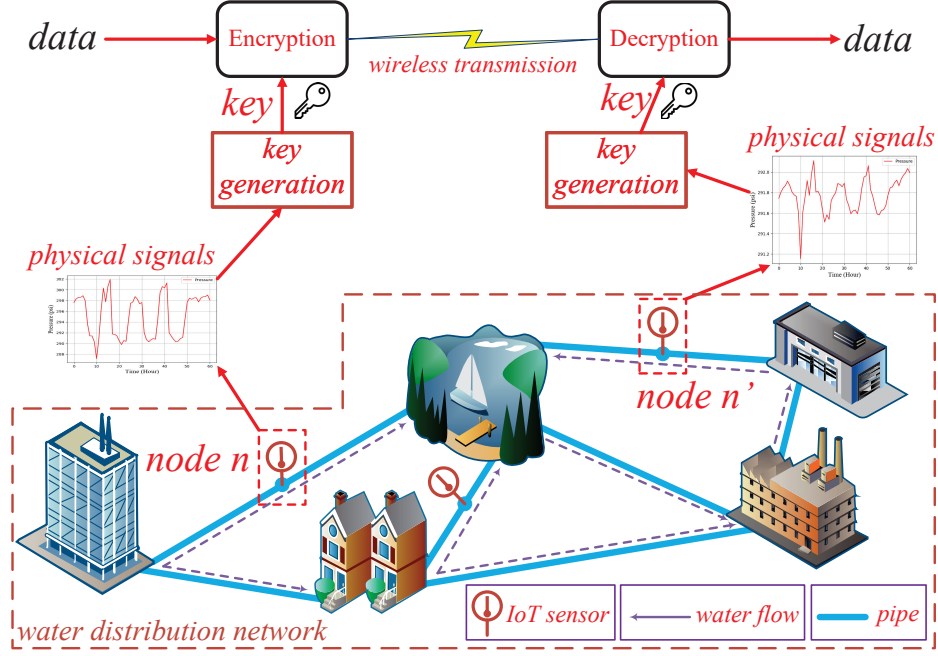


Fig. 1. System structure.

$$X_n(t) = [x_n^m(t), \forall m \in \mathcal{M}]^T, \forall t \in \mathcal{T}, \forall n \in \mathcal{N}, \quad (1)$$

where  $x_n^m(t)$  is the  $m$ -th signal value detected by node  $n$ ,  $\mathcal{M} \triangleq \{1, 2, \dots, M\}$ ,  $\mathcal{T} \triangleq \{1, 2, \dots, T\}$ , and  $\mathcal{N} \triangleq \{1, 2, \dots, N\}$ . In this paper, we assume the key generation rate (KGR) is 1, which means the generated keys between each pair of nodes have the same length as  $X_n(t)$ . Thus, the keys generated by node  $n$  between itself and node  $n'$  in time step  $t$  is expressed as

$$\mathbf{k}_{n,n'}(t) = [k_{n,n'}^1(t), \dots, k_{n,n'}^m(t) \dots k_{n,n'}^M(t)]^T, \quad (2)$$

where  $k_{n,n'}^m(t) \in \{0, 1\}$  is the  $m$ -th key bit. Then, the overall generated keys  $\mathbf{K}_n(t) \in \mathbb{R}^{(N-1) \times M}$  of node  $n$  can be defined as

$$\mathbf{K}_n(t) = [\mathbf{k}_{n,1}(t), \dots, \mathbf{k}_{n,n'}(t), \dots, \mathbf{k}_{n,N}(t)]. \quad (3)$$

In order to secure the data for avoiding digital attacks, the generated keys must be robust. The traditional PLS-based approaches rely on the common randomness of CSI among itself and others to generate keys, but they may suffer from extremely high complexity for accurate channel estimation and frequent information exchange, which is easily exposed to eavesdroppers. Considering the fact that most of IoT sensors detached in networked system are normally buried underground, which is hard to be attacked and the physical signals across the overall network normally have common features, the idea of this paper is to utilize the common features of physical signals for key generation. Therefore, the process of information exchange for key generation can be eliminated and the overall system can be fully distributed. In the next section, we will give the overall key generation scheme.

### 3 PROPOSED ALGORITHM

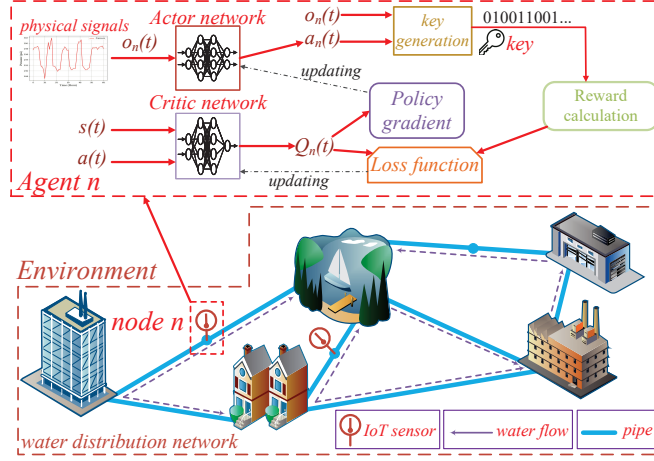


Fig. 2. Structure of agent  $n$  (training).

In this section, we describe the system model within the framework of deep reinforcement learning, which is based on the multi-agent version of Partially Observable Markov Decision Processes (POMDP) [18]. More specifically, we denote the overall key generation system as an environment, where  $N$  agents are deployed to interact with the environment at a sequence of discrete  $T$  time steps. In order to describe the configurations of all agents, the state space  $\mathcal{S}$ , the action space  $\mathcal{A}$ , and the sets of observations  $O_n, \forall n \in \mathcal{N}$  are further defined. Each agent  $n$  equips a policy function  $\pi_n(o_n(t), a_n(t))$  to map the relations between its local action  $a_n(t)$  and local observation  $o_n(t)$  in time step  $t$ . The target of each agent is to choose the optimal policies that can maximize the accumulated reward  $R = \sum_{t=0}^T \gamma^t r(t)$ , where  $\gamma \in [0, 1]$  denotes a discount factor,  $r(t)$  is the reward function in time step  $t$ .

Our proposed algorithm is motivated by the popular multi-agent deep deterministic policy gradient (MADDPG) [21], which involves the concept of centralized training with decentralized execution. The structure of agent  $n$  in the training process is shown in Fig. 2. Specifically, we assume that the node  $n$  is controlled by its dedicated agent  $n$ , which equips a DNN named actor network [17] with parameter  $\theta_n^\pi$ . The usage of actor network is to approximately estimate the policy function  $\pi_n(o_n(t), a_n(t) | \theta_n^\pi)$ , where  $o_n(t)$  is the local observation obtained by the IoT device, and  $a_n(t)$  is the local action generated by the actor network. Note that  $a_n(t)$  is utilized to extract common features of physical signals for key generation, which will be introduced later in Algorithm 2. During the training process, each of agents interacts with the environment by sending its local observation  $o_n(t)$  and local action  $a_n(t)$ . Then, the combinations of local observations and local actions of all agents will be denoted as state  $s(t)$  and  $a(t)$  respectively. The environment will obtain the reward  $r(t)$  and transfer to the next state  $s(t+1)$ . In addition, each of agents also equips a DNN named critic network for evaluating the Q-value function  $Q_n(s(t), a(t) | \theta_n^Q)$ , where  $\theta_n^Q$  is the parameter. Besides, both actor and critic networks have their target networks with the same structures and they will be updated with a fixed rate. We give the definition of local observation, local action, state and action as follows.

### 3.1 Definition of local observation, state space

For each agent  $n$ , the local observation in time step  $t$  is defined as the physical signals  $X_n(t)$ , e.g.,  $o_n(t) = X_n(t)$ .  $s(t)$  is the combination of observations of all agents, which is  $s(t) \triangleq \{o_n(t), \forall n \in \mathcal{N}\}$ . The set of local observations of agent  $n$  is  $O_n \triangleq \{o_n(t), \forall t \in \mathcal{T}\}$ . The state space  $\mathcal{S}$  is composed of the states from the initial time step up to the final time step, e.g.,  $\mathcal{S} \triangleq \{s(t), \forall t \in \mathcal{T}\}$ .

### 3.2 Definition of local action, action space

For each agent  $n$  in time step  $t$ , the local action  $a_n(t)$  generated by its actor network is defined as  $a_n(t) \triangleq \{a_{n,n'}^m(t), \forall m \in \mathcal{M}, \forall n' \in \mathcal{N}, n' \neq n\}$  with size  $(N-1) \times M$ , where  $a_{n,n'}^m(t) \in \{0, 1\}$ . Specifically, if  $a_{n,n'}^m(t) = 1$ , it indicates the  $m$ -th signal values of node  $n$  and node  $n'$  have common features and can be applied for key generation. Otherwise,  $a_{n,n'}^m(t) = 0$ . Therefore, the action space is defined as  $\mathcal{A} \triangleq \{a_n(t), \forall n \in \mathcal{N}, \forall t \in \mathcal{T}\}$ .

### 3.3 Definition of reward function

As the reward function is related to the performance metrics of generated keys, we define the reward function as follows:

$$r(t) = \begin{cases} s(\bar{t}), & \text{if } s(\bar{t}) < 1, \\ s(\bar{t}) + a(\bar{t}), & \text{otherwise,} \end{cases} \quad (4)$$

where  $s(\bar{t}) = 1 - \frac{\sum_{n=1}^N \sum_{n'=1, n' \neq n}^N \sum_{M=1}^M |k_{n,n'}^m(t) - k_{n',n}^m(t)|}{(N-1)NM} \in [0, 1]$  is the bit agreement rate (BAR) and it is one of the performance metrics of generated keys.  $a(\bar{t}) = \frac{\sum_{n=1}^N \sum_{n'=1, n' \neq n}^N \sum_{M=1}^M a_{n,n'}^m(t)}{(N-1)NM} \in [0, 1]$  is the average ratio of common features applied for key generation. Overall, the reward design is to guarantee that agents can obtain the same key with each other and maximally utilizing the common features among physical signals.

Furthermore, in our proposed algorithm, we apply the experience replay scheme [22] to stabilize the training process. Precisely, the transition, which consists of  $[s(t), a(t), r(t), s(t+1)]$  is stored in a memory with size  $K$ . When the training process starts,  $k$  transitions will be randomly sampled from the buffer. This scheme breaks the correlation between different transitions and therefore avoids the divergence.

The policy gradient approach [31] is applied to update the parameter of actor network of agent  $n$  according to the following equation:

$$\nabla_{\theta_n^\pi} J(\theta_n^\pi) \approx \mathbb{E}[\nabla_{\theta_n^\pi} \pi_n(o_n(t), a_n(t) | \theta_n^\pi) \nabla_{a_n(t)} Q_n(s(t), a(t) | \theta_n^Q) |_{a_n(t) = \pi_n(o_n(t) | \theta_n^\pi)}]. \quad (5)$$

The critic network is updated by minimizing the loss function, which is defined as

$$L(\theta_n^Q) = \mathbb{E}[(\hat{y} - Q_n(s(t), a(t) | \theta_n^Q))^2], \quad (6)$$

where  $\hat{y}$  is calculated by

$$\hat{y} = r(t) + \gamma Q'_n(s(t+1), a(t+1) | \theta_n^{Q'}) |_{a_n(t+1) = \pi'_n(o_n(t+1) | \theta_n^{\pi'})}, \quad (7)$$

where  $\pi'_n(\cdot | \theta_n^{\pi'})$ ,  $Q'_n(\cdot | \theta_n^{Q'})$  denote the corresponding policy and Q-value functions of the target networks.

We also provide the overall algorithm design in Algorithm 1. From Line 1 to 4, the actor  $\pi_n(\cdot | \theta_n^\pi)$ , critic  $Q_n(\cdot | \theta_n^Q)$  and two target networks  $\pi'_n(\cdot | \theta_n^{\pi'})$ ,  $Q'_n(\cdot | \theta_n^{Q'})$  of each of agents are initialized. Then, in Line 5, the experience replay memory with size  $K$  is initialized as well. The training process lasts  $E^{max}$  episodes, each of which has  $T$  time steps. In the initial time step, each of agents  $n$  initializes its local observation  $o_n(t)$  and state  $s(t)$  is the combination of observations obtained by all agents. From Line 12, each of agents  $n$  selects the local action  $a_n(t)$  generated by its actor network. Note that  $N_\epsilon$  denotes the action noise for a better exploration and it decays with a

---

**Algorithm 1** Proposed algorithm
 

---

```

1: for agent  $n = 1, \dots, N$  do
2:   Initialize actor  $\pi_n(\cdot|\theta_n^\pi)$ , critic  $Q(\cdot|\theta_n^Q)$  networks with parameter  $\theta_n^\pi$  and  $\theta_n^Q$  respectively. ;
3:   Initialize two target networks  $\pi'_n(\cdot|\theta_n^{\pi'}), Q(\cdot|\theta_n^{Q'})$  with parameters  $\theta_n^{\pi'}$  and  $\theta_n^{Q'}$  respectively.;
4: end for
5: Initialize experience replay memory with size  $K$ ;
6: for episode =  $1, 2, \dots, E^{max}$  do
7:   for agent  $n = 1, \dots, N$  do
8:     Initialize observation  $o_n(t)$ ;
9:   end for
10:  for time step  $t$  in  $\mathcal{T}$  do
11:    Obtain  $s(t) = \{o_1(t), \dots, o_N(t)\}$ ;
12:    for agent  $n = 1, \dots, N$  do
13:      Select action  $a_n(t) = \pi_n(o_n(t)|\theta_n^\pi) + N_\epsilon$ ;
14:      Execute action  $a_n(t)$ ;
15:      Obtain generated keys  $K_n(t)$  according to Algorithm 2;
16:    end for
17:    Obtain action  $a(t) = \{a_1(t), \dots, a_N(t)\}$ ;
18:    Calculate reward  $r(t)$  according to Eq. (4);
19:    Obtain next state  $s(t+1)$ ;
20:    Store transition  $[s(t), a(t), r(t), s(t+1)]$  into experience replay;
21:    if the training process starts then
22:      Randomly sample  $k$  transitions from experience replay memory;
23:      for agent  $n = 1, \dots, N$  do
24:        Update actor network  $\pi_n(\cdot|\theta_n^\pi)$  according to Eq. 5;
25:        Update critic network  $Q_n(\cdot|\theta_n^Q)$  according to Eq. 6;
26:        Update two target networks  $\pi'_n(\cdot|\theta_n^{\pi'}), Q'_n(\cdot|\theta_n^{Q'})$  with rate  $\tau$ :
27:         $\theta_n^{\pi'} \leftarrow \tau\theta_n^\pi + (1 - \tau)\theta_n^{\pi'}$ ;
28:         $\theta_n^{Q'} \leftarrow \tau\theta_n^Q + (1 - \tau)\theta_n^{Q'}$ ;
29:      end for
30:    end if
31:  end for
32: end for

```

---

fixed rate. After that, the generated keys  $K_n(t)$  are obtained by Algorithm 2. Starting from Line 17, we obtain the action  $a(t) = \{a_1(t), \dots, a_N(t)\}$  and calculate the reward  $r(t)$  using Eq. (4). The environment will transfer to the next state  $s(t+1)$ . In Line 20, the transition, which consists of  $[s(t), a(t), r(t), s(t+1)]$  is stored into experience replay memory. From Line 21, when the experience replay has enough transitions, the training process starts. Precisely, we randomly sample  $k$  transitions to train each of actor and critic networks according to Eq. (5) and Eq. (6). Finally, two target networks of each of agents are also updated with the rate of  $\tau$ .

We further introduce the proposed feature extraction scheme for key generation in Algorithm 2. Specifically, after obtaining action  $a_n(t)$ , if  $a_{n,n'}^m(t) = 1$  and in the meantime the current  $m$ -th signal value is larger than the previous  $(m-1)$ -th value, i.e.,  $x_n^m(t) \geq x_n^{m-1}(t)$ , we have  $k_{n,n'}^m(t) = 1$ . Otherwise, we have  $k_{n,n'}^m(t) = 0$ . Finally, we return  $K_n(t)$ .

After the training process, the network models of each agent are saved for validation. Note that our proposed algorithm is based on the framework of centralized training with decentralized execution, which means only actor networks of each agent are utilized for implementation. Thus, the overall system is fully distributed and its structure is shown in Fig. 3. Furthermore, we analyze the computational complexity of the saved actor network as follows. As the fully connected layers are applied in this work, the complexity for actor network can be calculated as  $\mathcal{O}(\sum_{z=1}^Z n_z \cdot n_{z-1})$ , where  $Z$  is the number of network layers,  $n_z$  denotes the number of neurons of  $z$ -th layer [7]. Therefore, it is clear to see that embedded IoT devices in the networked system will benefit from deploying the light-weighted actor networks.

---

**Algorithm 2** Proposed feature extraction scheme
 

---

```

1: Obtain action  $a_n(t) \triangleq \{a_{n,n'}^1(t), \dots, a_{n,n'}^M(t), \forall n' \in N, n' \neq n\}$  with size  $(N-1) \times M$ ;
2: Obtain  $X_n(t) = [x_n^m(t), \forall m \in M]^T$ ;
3: Initialize  $K_n(t) = [k_{n,1}(t), \dots, k_{n,n'}(t), \dots, k_{n,N}(t)]$  with size  $(N-1) \times M$ ;
4: for  $n' = 1, \dots, N, n' \neq n$  do
5:   for  $m = 1, \dots, M$  do
6:     if  $a_{n,n'}^m(t) = 1$  and  $x_n^m(t) \geq x_n^{m-1}(t)$  then
7:        $k_{n,n'}^m(t) = 1$ , e.g.,  $m$ -th element of  $k_{n,n'}(t)$  is 1;
8:     else
9:        $k_{n,n'}^m(t) = 0$ , e.g.,  $m$ -th element of  $k_{n,n'}(t)$  is 0;
10:    end if
11:  end for
12: end for
13: Return  $K_n(t)$ ;
  
```

---

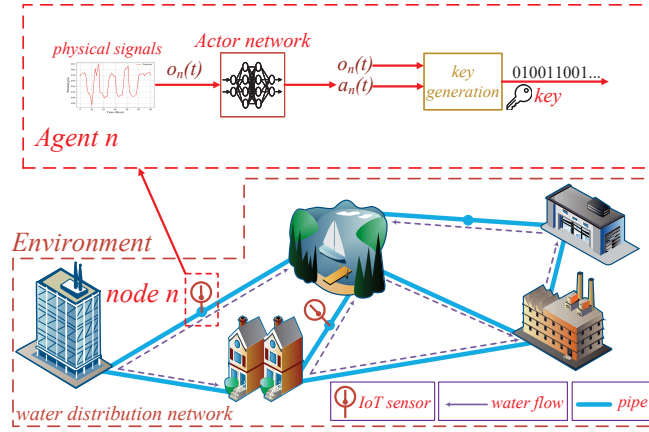


Fig. 3. Structure of agent  $n$  (validation).

## 4 EXPERIMENTAL RESULTS

### 4.1 Experimental Platform

In order to demonstrate the reliability and performance of our proposed algorithm, we implement a testbed based on a simulation environment. Specifically, we utilize a tiny water distribution network built by EPANET, which is a public domain software [28]. As shown in Fig. 4, the water distribution network is based on the North Marin Water District in Novato, CA, which has 1 reservoir, 3 tanks, 2 pumps, and 6.8 miles of pipes. For simplicity, we choose 3 node junctions (node 121, 141, 189) as the communication nodes in the experiment. The water pressures of the communication nodes are defined as the physical signals. We denote the previous water pressures for 160 minutes as the training date, whose snapshot and correlations between different communication nodes are shown in Fig. 5a and Fig. 5b respectively.

The training process of our proposed algorithm is performed with Python 3.7 [32], Tensorflow 2.0 [1]. The actor network of each agent has 1 input layer (100 neurons), 3 hidden layers (100 neurons), and 1 output layer ( $M(N-1)$  neurons).  $\text{elu}(\cdot)$  is used as the activation function for both input and hidden layers while  $\text{tanh}(\cdot)$  is used for the output layer. The critic network also has 1 input layer, 3 hidden layers, and 1 output layer. The architecture of actor and critic networks can be seen in Fig. 6a and 6b. In our experiment, each communication

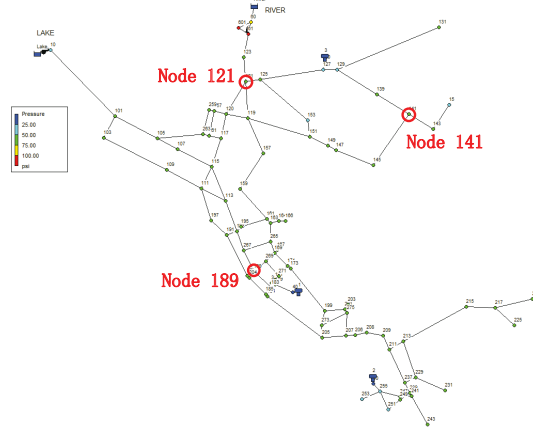


Fig. 4. Network topology (simulation).

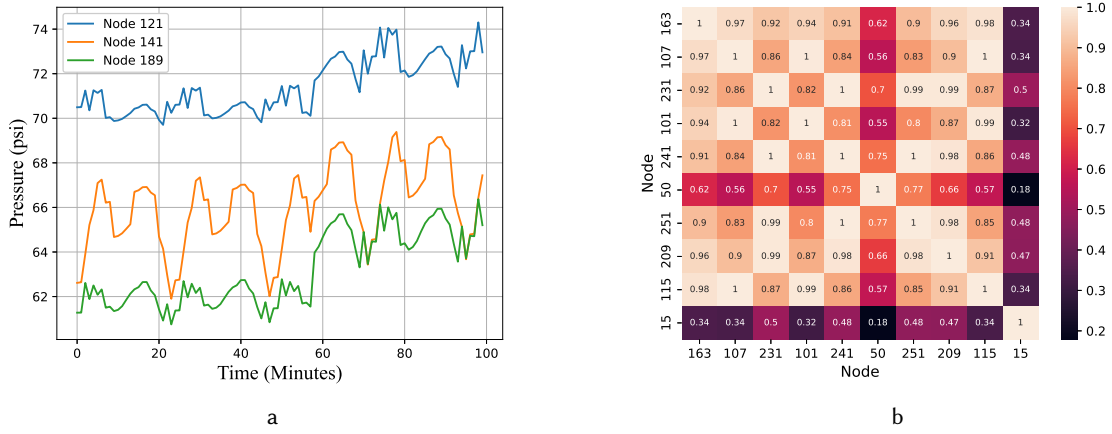


Fig. 5. (a) Snapshots of physical signals. (b) Correlations between different communication nodes.

node detects the physical signal with a particular time interval ( $t^i = 1$  minute). We define the previously detected 20 physical signal values of each communication node as  $o_n(t)$ , which means  $M = 20$ . The training process has overall  $E^{max} = 25000$  episodes, each of which has  $T = 20$  time steps. The size of the experience replay is  $K = 10^5$ . AdamOptimizer [11] is applied to update actor and critic networks with the rate of 0.0001 and 0.001 respectively, while two target networks are updated with the rate of  $\tau = 0.01$ . More parameters about system model and experiment can be found in Table 2.

## 4.2 Performance metrics of key generation

In this subsection, we introduce the performance metrics of key generation.

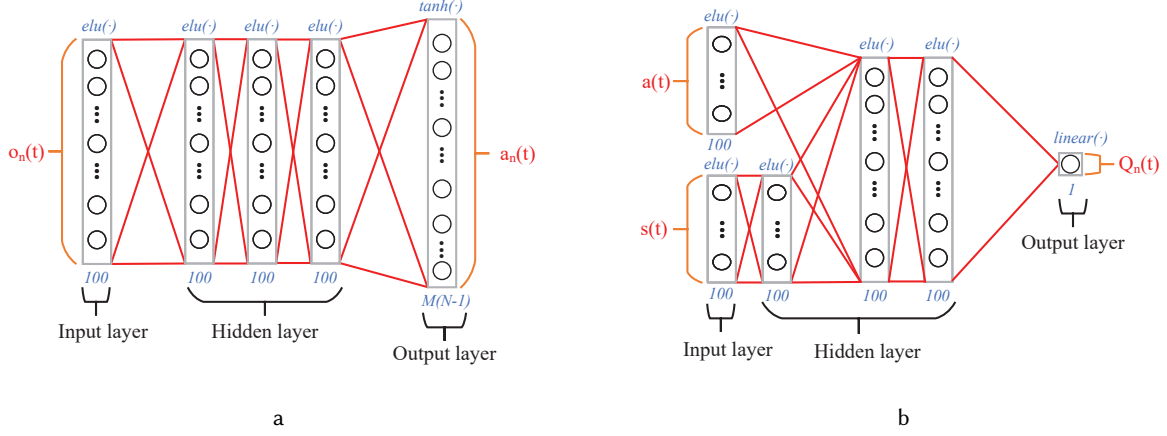


Fig. 6. Architectures of (a) actor network and (b) critic network.

Table 2. simulation parameters

| Parameters   | Setting  |
|--------------|----------|
| $t^i$        | 1 minute |
| $N$          | 3        |
| $T$          | 20       |
| $M$          | 20       |
| $\gamma$     | 0.99     |
| $K$          | $10^5$   |
| $k$          | 256      |
| $E^{max}$    | 25000    |
| $N_\epsilon$ | 1        |
| $\tau$       | 0.01     |

- **Bit agreement rate (BAR):** BAR represents the percentage of the same key bits between different communication nodes, which is described in Eq. (4).
- **Randomness:** In order to avoid brute force from attackers, the generated keys should be random. One widely used method is to apply a statistical randomness test suite designed by National Institute of Standards and Technology (NIST) [4], which evaluates the randomness of generated keys by returning a p-value within  $[0, 1]$ . If the returned p-value is larger than a threshold, i.e., 0.01, we can conclude the generated keys pass the randomness test.
- **Uniqueness:** It means whether the physical signals can represent different patterns and keys. In this paper, as the IoT devices are detached within the same water distribution network, we can conclude that uniqueness is always guaranteed.

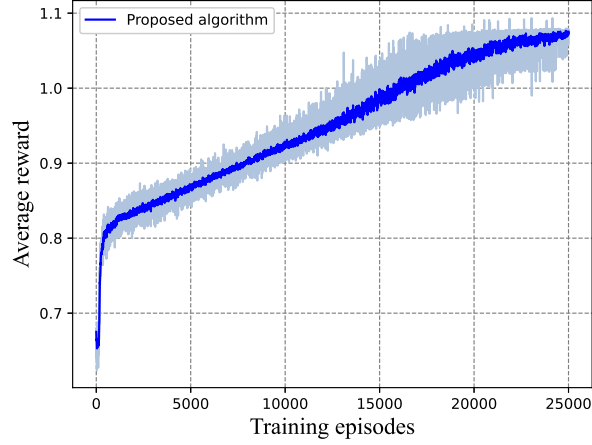


Fig. 7. Accumulated reward of proposed algorithm versus training episodes.

### 4.3 Experimental analysis

In this subsection, we give the experimental analysis of our proposed algorithm. First of all, we analyze the convergence performance through depicting the accumulated reward obtained in each training episode in Fig. 7. Specifically, the training curve of the proposed algorithm remains below 0.7 in the beginning. One plausible explanation is that the training process is not started and the agents are taking random exploration. When the experience replay memory has enough transitions, the learning process starts. The agents keep learning from the interactions with environment through exploration and exploitation, which results in the increase of accumulated reward. After that, it is observed that the training curve remains close to 1.1 eventually and we conclude the convergence is achieved.

After the training process, we further analyze the performance of the proposed algorithm in terms of BAR. To be more specific, as the number of communication nodes is 3, there are 3 pairs of generated keys between different nodes, which are node pair 121 - 141, node pair 121 - 189, and node pair 141 - 189. We analyze the average BAR of generated keys with different numbers of time steps in Table 3. It is easy to see that the average BARs of generated keys of node pair 121 - 141, node pair 121 - 189, and node pair 141 - 189 always remain above 0.90, which is acceptable and the mismatched key bits can be easily corrected by a standard information conciliation process, such as utilizing error correction code (ECC) or Bose-Chaudhuri-Hocquenghem (BCH) code.

Then, we analyze the performance of the proposed algorithm in terms of randomness in Table 4. It is obvious to see from the test results that all p-values of generated keys of node pair 121 - 141, node pair 121 - 189, and node pair 141 - 189 are larger than 0.01. Therefore, we can conclude our proposed algorithm can meet the requirement in terms of randomness.

Furthermore, we evaluate the performance of our proposed algorithm in a real water distribution network, whose topology is seen in Fig. 8. More precisely, the network has 1 water tank, plastic pipes, interconnectors, a manual valve, a pump, and water pressure sensors (node 1 and node 2). In order to realize the continuous changes in water pressure, we adjust the rotational speed of the pump and the manual valve. The water pressures of node 1 and node 2 can be seen in Fig. 9.

In Table. 5, we analyze the average BARs and p-values of generated keys of node 1 and node 2. It is observed that the proposed algorithm still has the outstanding performance of key generation in the real environment.

Finally, we consider the existence of eavesdroppers (Eve) and conduct the potential attacks for benchmark comparisons. The attacking schemes are described as follows.

- **Scheme 1:** Eve only knows the structure of neural network of node 1 or node 2 and passively generates keys using the water pressures captured from another water distribution network.
- **Scheme 2:** The structure of neural network is unknown but Eve can obtain the raw water pressures of node 1 or node 2. Then, Eve applies a random feature extraction scheme for key generation.
- **Scheme 3:** Eve knows the structure of neural network of node 1 or node 2 and also obtains the water pressures of nearby communication nodes.
- **Scheme 4:** In order to avoid the information exchange, which is vulnerable by Eve, node 1 and node 2 only apply a simple threshold for quantization and key generation, according to their dedicated water pressures.

Table 3. Average BARs of generated keys of different nodes (simulation)

| Time steps | Node pair<br>121 - 141 | Node pair<br>121 - 189 | Node pair<br>141 - 189 |
|------------|------------------------|------------------------|------------------------|
| 2          | 1                      | 1                      | 1                      |
| 4          | 0.98                   | 0.96                   | 1.0                    |
| 6          | 1.0                    | 1.0                    | 1.0                    |
| 8          | 0.96                   | 0.96                   | 1.0                    |
| 10         | 0.96                   | 0.96                   | 1.0                    |
| 12         | 1.0                    | 0.92                   | 1.0                    |
| 14         | 0.96                   | 1.0                    | 1.0                    |
| 16         | 1.0                    | 0.96                   | 0.96                   |
| 18         | 0.92                   | 0.96                   | 1.0                    |
| 20         | 1.0                    | 0.96                   | 0.96                   |

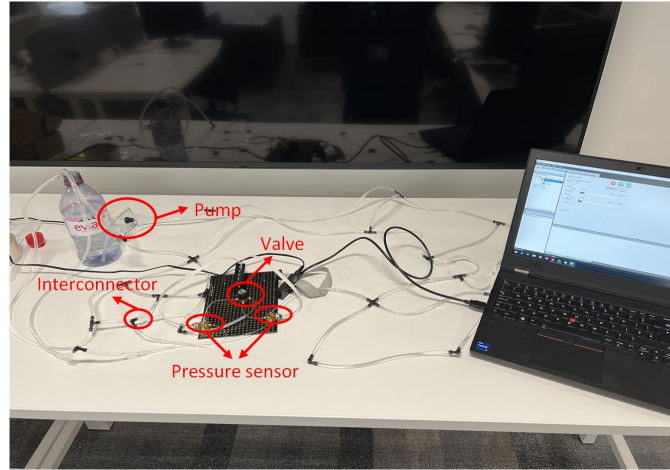


Fig. 8. Network topology (real environment).

Table 4. Average p-values of generated keys of different nodes (simulation)

| Time steps | Node pair<br>121 - 141 | Node pair<br>121 - 189 | Node pair<br>141 - 189 |
|------------|------------------------|------------------------|------------------------|
| 2          | 0.339                  | 0.339                  | 0.225                  |
| 4          | 0.261                  | 0.242                  | 0.256                  |
| 6          | 0.260                  | 0.245                  | 0.270                  |
| 8          | 0.255                  | 0.255                  | 0.276                  |
| 10         | 0.256                  | 0.252                  | 0.271                  |
| 12         | 0.261                  | 0.248                  | 0.272                  |
| 14         | 0.258                  | 0.255                  | 0.266                  |
| 16         | 0.264                  | 0.258                  | 0.267                  |
| 18         | 0.257                  | 0.259                  | 0.266                  |
| 20         | 0.257                  | 0.264                  | 0.266                  |

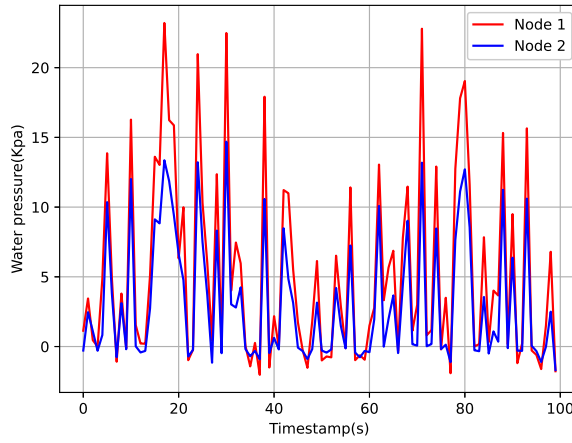


Fig. 9. A snapshot of water pressures (real environment).

Then, we compare the BARs of generated keys obtained by our proposed algorithm and the above attacking schemes in Fig. 10, where we observe that our proposed algorithm consistently achieves the reliable performance against different kinds of digital attacks. Overall, the BARs of all attacking schemes remain between 0.35 and 0.8. One plausible explanation is that for achieving the perfect attacking performance, Eve has to know the raw water pressures and structure of neural network of the attacked communication node simultaneously. However, neither inserting illegal IoT sensors in networked system or obtaining the legitimate embedded neural network is extremely difficult, which could be easily exposed.

## 5 CONCLUSION AND FUTURE WORK

In this paper, we investigate the key generation problem in networked systems. Motivated by the fundamentals of GLS, we propose a distributed key generation algorithm based on MADRL approach, in which the communication

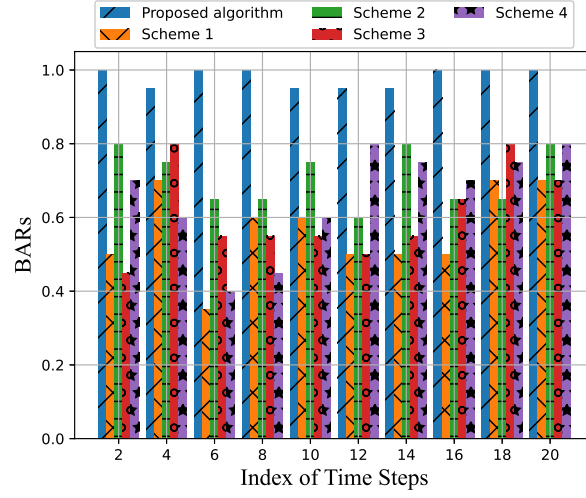


Fig. 10. BAR comparisons of the proposed algorithm and attacking schemes.

nodes of the networked system are enabled to generate keys based on their locally detected physical signals without information exchange and wireless channel estimation. We evaluate the performance metrics of key generation both in the simulated and real networked systems. The experimental results prove that the proposed algorithm achieves considerable performance in terms of randomness, and BAR after training.

As future work, we plan to extend our proposed algorithm into an entire key generation protocol, which contains information reconciliation and privacy amplification. In addition, we further expect to deploy our work into a real hardware testbed to verify the robustness performance and seek for potential prevention from digital attacks.

Table 5. Average BARs and p-values of generated keys between node 1 and node 2

| Time steps | BARs | p-values |
|------------|------|----------|
| 2          | 1    | 0.15     |
| 4          | 0.95 | 0.18     |
| 6          | 1    | 0.14     |
| 8          | 1    | 0.25     |
| 10         | 0.95 | 0.16     |
| 12         | 0.95 | 0.16     |
| 14         | 0.95 | 0.11     |
| 16         | 1    | 0.12     |
| 18         | 1    | 0.11     |
| 20         | 1    | 0.12     |

## 6 ACKNOWLEDGMENT

This work has been supported by the PETRAS National Centre of Excellence for IoT Systems Cybersecurity, which has been funded by the UK EPSRC under grant number EP/S035362/1. All research work was conducted whilst all authors were at Cranfield University. No data or knowledge transfer was conducted.

## REFERENCES

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. <http://tensorflow.org/> Software available from tensorflow.org.
- [2] Yun Ai, Michael Cheffena, Aashish Mathur, and Hongjiang Lei. 2018. On Physical Layer Security of Double Rayleigh Fading Channels for Vehicular Communications. *IEEE Wireless Communications Letters* 7, 6 (2018), 1038–1041. <https://doi.org/10.1109/LWC.2018.2852765>
- [3] Wael Ayoub, Abed Ellatif Samhat, Fabienne Nouvel, Mohamad Mroue, and Jean-Christophe Prévotet. 2019. Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs Standards and Supported Mobility. *IEEE Communications Surveys Tutorials* 21, 2 (2019), 1561–1581. <https://doi.org/10.1109/COMST.2018.2877382>
- [4] Lawrence Bassham, Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Stefan Leigh, M Levenson, M Vangel, Nathanael Heckert, and D Banks. 2010. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=906762](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762)
- [5] Xiaojie Fang, Ning Zhang, Shan Zhang, Dajiang Chen, Xuejun Sha, and Xuemin Shen. 2017. On Physical Layer Security: Weighted Fractional Fourier Transform Based User Cooperation. *IEEE Transactions on Wireless Communications* 16, 8 (2017), 5498–5510. <https://doi.org/10.1109/TWC.2017.2712158>
- [6] Weijun Gao, Chong Han, and Zhi Chen. 2022. DNN-Powered SIC-Free Receiver Artificial Noise Aided Terahertz Secure Communications With Randomly Distributed Eavesdroppers. *IEEE Transactions on Wireless Communications* 21, 1 (2022), 563–576. <https://doi.org/10.1109/TWC.2021.3098334>
- [7] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA, USA. <http://www.deeplearningbook.org>.
- [8] Sheng Hong, Cunhua Pan, Hong Ren, Kezhi Wang, and Arumugam Nallanathan. 2020. Artificial-Noise-Aided Secure MIMO Wireless Communications via Intelligent Reflecting Surface. *IEEE Transactions on Communications* 68, 12 (2020), 7851–7866. <https://doi.org/10.1109/TCOMM.2020.3024621>
- [9] Chong Huang, Gaojie Chen, and Kai-Kit Wong. 2021. Multi-Agent Reinforcement Learning-Based Buffer-Aided Relay Selection in IRS-Assisted Secure Cooperative Networks. *IEEE Transactions on Information Forensics and Security* 16 (2021), 4101–4112. <https://doi.org/10.1109/TIFS.2021.3103062>
- [10] Weiheng Jiang, Bolin Chen, Jun Zhao, Zehui Xiong, and Zhiguo Ding. 2021. Joint Active and Passive Beamforming Design for the IRS-Assisted MIMOME-OFDM Secure Communications. *IEEE Transactions on Vehicular Technology* 70, 10 (2021), 10369–10381. <https://doi.org/10.1109/TVT.2021.3106351>
- [11] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. <https://doi.org/10.48550/ARXIV.1412.6980>
- [12] Bin Li, Zesong Fei, Yan Zhang, and Mohsen Guizani. 2019. Secure UAV Communication Networks over 5G. *IEEE Wireless Communications* 26, 5 (2019), 114–120. <https://doi.org/10.1109/MWC.2019.1800458>
- [13] Guyue Li, Lei Hu, Paul Staat, Harald Elders-Boll, Christian Zenger, Christof Paar, and Aiqun Hu. 2022. Reconfigurable Intelligent Surface for Physical Layer Key Generation: Constructive or Destructive? *IEEE Wireless Communications* 29, 4 (2022), 146–153. <https://doi.org/10.1109/MWC.007.2100545>
- [14] Guyue Li, Chen Sun, Eduard A. Jorswieck, Junqing Zhang, Aiqun Hu, and You Chen. 2021. Sum Secret Key Rate Maximization for TDD Multi-User Massive MIMO Wireless Networks. *IEEE Transactions on Information Forensics and Security* 16 (2021), 968–982. <https://doi.org/10.1109/TIFS.2020.3026466>
- [15] Guyue Li, Chen Sun, Wei Xu, Marco Di Renzo, and Aiqun Hu. 2022. On Maximizing the Sum Secret Key Rate for Reconfigurable Intelligent Surface-Assisted Multiuser Systems. *IEEE Transactions on Information Forensics and Security* 17 (2022), 211–225. <https://doi.org/10.1109/TIFS.2021.3138612>
- [16] Le Liang, Hao Ye, and Geoffrey Ye Li. 2019. Spectrum Sharing in Vehicular Networks Based on Multi-Agent Reinforcement Learning. *IEEE Journal on Selected Areas in Communications* 37, 10 (2019), 2282–2292. <https://doi.org/10.1109/JSAC.2019.2933962>
- [17] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. <https://doi.org/10.48550/ARXIV.1509.02971>

- [18] Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*. Elsevier, 157–163.
- [19] Hongbo Liu, Yang Wang, Jie Yang, and Yingying Chen. 2013. Fast and practical secret key extraction by exploiting channel response. In *2013 Proceedings IEEE INFOCOM*. IEEE, 3048–3056.
- [20] Hongbo Liu, Jie Yang, Yan Wang, Yingying Chen, and Can Emre Koksul. 2014. Group secret key generation via received signal strength: Protocols, achievable rates, and implementation. *IEEE Transactions on Mobile Computing* 13, 12 (2014), 2820–2835.
- [21] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. <https://doi.org/10.48550/ARXIV.1706.02275>
- [22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. <https://doi.org/10.48550/ARXIV.1312.5602>
- [23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
- [24] Amitav Mukherjee, S. Ali A. Fakoorian, Jing Huang, and A. Lee Swindlehurst. 2014. Principles of Physical Layer Security in Multiuser Wireless Networks: A Survey. *IEEE Communications Surveys Tutorials* 16, 3 (2014), 1550–1573. <https://doi.org/10.1109/SURV.2014.012314.00178>
- [25] Yasar Sinan Nasir and Dongning Guo. 2019. Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks. *IEEE Journal on Selected Areas in Communications* 37, 10 (2019), 2239–2250. <https://doi.org/10.1109/JSAC.2019.2933973>
- [26] Haixia Peng and Xuemin Shen. 2021. Multi-Agent Reinforcement Learning Based Resource Management in MEC- and UAV-Assisted Vehicular Networks. *IEEE Journal on Selected Areas in Communications* 39, 1 (2021), 131–141. <https://doi.org/10.1109/JSAC.2020.3036962>
- [27] H Vincent Poor and Rafael F Schaefer. 2017. Wireless physical layer security. *Proceedings of the National Academy of Sciences* 114, 1 (2017), 19–26.
- [28] Lewis A Rossman et al. 2000. EPANET 2: users manual. (2000).
- [29] S. Rothe, N. Koukourakis, H. Radner, A. Lonnstrom, E. Jorswieck, and J. Czariske. 2020. Physical Layer Security in Multimode Fiber Optical Networks. *Scientific Reports* 10 (2020), 2740.
- [30] Hossein Shafagh and Anwar Hithnawi. 2014. Security comes first, a public-key cryptography framework for the internet of things. In *2014 IEEE International Conference on Distributed Computing in Sensor Systems*. IEEE, 135–136.
- [31] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).
- [32] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.
- [33] Hui-Ming Wang, Xu Zhang, and Jia-Cheng Jiang. 2019. UAV-Involved Wireless Physical-Layer Secure Communications: Overview and Research Directions. *IEEE Wireless Communications* 26, 5 (2019), 32–39. <https://doi.org/10.1109/MWC.001.1900045>
- [34] Zhuangkun Wei and Weisi Guo. 2021. Random Matrix based Physical Layer Secret Key Generation in Static Channels. *arXiv preprint arXiv:2110.12785* (2021).
- [35] Zhuangkun Wei, Weisi Guo, and Bin Li. 2022. A Multi-Eavesdropper Scheme against RIS Secured LoS-dominated Channel. *IEEE Communications Letters* (2022), 1–1.
- [36] Zhuangkun Wei, Liang Wang, Schyler Chengyao Sun, Bin Li, and Weisi Guo. 2022. Graph Layer Security: Encrypting Information via Common Networked Physics. *Sensors* 22, 10 (2022). <https://doi.org/10.3390/s22103951>
- [37] Matthias Wilhelm, Ivan Martinovic, and Jens B Schmitt. 2013. Secure key generation in sensor networks based on frequency-selective channels. *IEEE Journal on Selected Areas in Communications* 31, 9 (2013), 1779–1790.
- [38] Yuezhong Wu, Qi Lin, Hong Jia, Mahbub Hassan, and Wen Hu. 2020. Auto-Key: Using autoencoder to speed up gait-based key generation in body area networks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (2020), 1–23.
- [39] Sangseok Yun, Jae-Mo Kang, Il-Min Kim, and Jeongseok Ha. 2020. Deep Artificial Noise: Deep Learning-Based Precoding Optimization for Artificial Noise Scheme. *IEEE Transactions on Vehicular Technology* 69, 3 (2020), 3465–3469. <https://doi.org/10.1109/TVT.2020.2965959>
- [40] Guangchi Zhang, Qingqing Wu, Miao Cui, and Rui Zhang. 2019. Securing UAV Communications via Joint Trajectory and Power Control. *IEEE Transactions on Wireless Communications* 18, 2 (2019), 1376–1389. <https://doi.org/10.1109/TWC.2019.2892461>
- [41] Junqing Zhang, Ming Ding, David López-Pérez, Alan Marshall, and Lajos Hanzo. 2019. Design of an Efficient OFDMA-Based Multi-User Key Generation Protocol. *IEEE Transactions on Vehicular Technology* 68, 9 (2019), 8842–8852. <https://doi.org/10.1109/TVT.2019.2929362>
- [42] Junqing Zhang, Trung Q. Duong, Alan Marshall, and Roger Woods. 2016. Key Generation From Wireless Channels: A Review. *IEEE Access* 4 (2016), 614–626. <https://doi.org/10.1109/ACCESS.2016.2521718>
- [43] Junqing Zhang, Alan Marshall, and Lajos Hanzo. 2018. Channel-Envelope Differencing Eliminates Secret Key Correlation: LoRa-Based Key Generation in Low Power Wide Area Networks. *IEEE Transactions on Vehicular Technology* 67, 12 (2018), 12462–12466. <https://doi.org/10.1109/TVT.2018.2877201>
- [44] Junqing Zhang, Sekhar Rajendran, Zhi Sun, Roger Woods, and Lajos Hanzo. 2019. Physical layer security for the Internet of Things: Authentication and key generation. *IEEE Wireless Communications* 26, 5 (2019), 92–98.

- [45] Junqing Zhang, Yushi Zheng, Weitao Xu, and Yingying Chen. 2021. H2K: A Heartbeat-based Key Generation Framework for ECG and PPG Signals. *IEEE Transactions on Mobile Computing* (2021), 1–1. <https://doi.org/10.1109/TMC.2021.3096384>
- [46] Kuan Zhang, Jianbing Ni, Kan Yang, Xiaohui Liang, Ju Ren, and Xuemin Sherman Shen. 2017. Security and privacy in smart city applications: Challenges and solutions. *IEEE Communications Magazine* 55, 1 (2017), 122–129.
- [47] Y. Zhu, G. Zheng, and M. Fitch. 2018. Secrecy Rate Analysis of UAV-Enabled mmWave Networks Using Matérn Hardcore Point Processes. *IEEE Journal on Selected Areas in Communications* 36, 7 (Jul. 2018), 1397–1409.



**DSPACE**

<https://dspace.org/>

## **Multi-agent deep reinforcement learning-based key generation for graph layer security**

**Wang, Liang; Wei, Zhuangkun; Guo, Weisi**

**2025-05**

Association for Computing Machinery (ACM)

<https://doi.org/10.1145/3711900>

Wang L, Wei Z, Guo W. (2025) Multi-agent deep reinforcement learning-based key generation for graph layer security. ACM Transactions on Privacy and Security, Volume 28, Issue 2, May 2025, Article number 18