A multi-factor integration-based semi-supervised learning for address resolution protocol attack detection in SDIIoT

Zhong Li\textsuperscript{1,2} and Huimin Zhuang\textsuperscript{1}

Abstract
Nowadays, in the industrial Internet of things, address resolution protocol attacks are still rampant. Recently, the idea of applying the software-defined networking paradigm to industrial Internet of things is proposed by many scholars since this paradigm has the advantages of flexible deployment of intelligent algorithms and global coordination capabilities. These advantages prompt us to propose a multi-factor integration-based semi-supervised learning address resolution protocol detection method deployed in software-defined networking, called MIS, to specially solve the problems of limited labeled training data and incomplete features extraction in the traditional address resolution protocol detection methods. In MIS method, we design a multi-factor integration-based feature extraction method and propose a semi-supervised learning framework with differential priority sampling. MIS considers the address resolution protocol attack features from different aspects to help the model make correct judgment. Meanwhile, the differential priority sampling enables the base learner in self-training to learn efficiently from the unlabeled samples with differences. We conduct experiments based on a real data set collected from a deepwater port and a simulated data set. The experiments show that MIS can achieve good performance in detecting address resolution protocol attacks with F1-measure, accuracy, and area under the curve of 97.28%, 99.41%, and 98.36% on average. Meanwhile, compared with fully supervised learning and other popular address resolution protocol detection methods, MIS also shows the best performance.

Keywords
Industrial Internet of things, software-defined networking, address resolution protocol attack, machine learning, multi-factor integration

Date received: 24 June 2021; accepted: 18 October 2021

Handling editor: Yanjiao Chen

Introduction
The industrial Internet of things (IIoT) is widely used in manufacturing, transportation, aerospace, and other industrial fields.\textsuperscript{1} It helps improve productivity and reduce production cost with connected sensors and controllers. But the security issues in IIoT cannot be ignored. Undoubtedly, IIoT security committed to evasion of risk plays an extremely important role during its development.

Due to the capability of directly influencing or controlling the physical world, IIoT is, and continues to be, the target of malicious attackers.\textsuperscript{2} A potential attack on a critical device in IIoT could lead to the
paralysis of the entire factory assembly line or the leakage of national key industrial information. Among various attacks, there is a type of attack that occurs very frequently, especially in the industrial ethernet, that is, the address resolution protocol (ARP) attack. In the normal communication, ARP is used to obtain a media access control (MAC) address according to an Internet protocol (IP) address. A source host broadcasts an ARP request packet containing the destination’s IP address to all hosts on the local area network and receives the reply packet to obtain the corresponding MAC address of the destination. In this process, an ARP attack happens when an attacker sends a false ARP reply packet to a victim by forging an IP-MAC mapping to pollute the victim’s ARP cache. According to the real data samples obtained in an industrial scenario, we do find that there are many ARP attacks in the industrial ethernet. (Here, we obtained a data set in 2020 from a large freight port in Shanghai, which is a deepwater port with container terminals. For security and confidentiality, we have to hide the enterprise’s name.) The statistical results are shown in Figure 1. This figure shows the number of ARP attacks observed from two switches’ ports.

Researchers have proposed some approaches to deal with ARP attacks. Previous studies3–5 have improved the ARP protocol through modification of the standard ARP protocol. However, these modifications are not executed as new standards. Many new application scenarios, such as IIoT, still use the traditional standard ARP. These improvements are difficult to promote to the real applications. Previous studies6–8 have used the cryptographic technology to achieve good solutions against ARP attacks. However, adding cryptography to the ARP itself would increase the network overheads and calculations when running ARP. Besides, previous studies9–11 have taken advantage of the Internet control messages protocol (ICMP) or dynamic host configuration protocol (DHCP) to assist ARP attack detection. However, this type of methods is easy to be defeated by an attacker through blocking the ICMP/DHCP packets. Furthermore, although cross-layer protocol design is feasible, it is still not supported very well in the current protocol stack.

We can see that the above three types of studies are to revise the protocol or protocol stack. Due to the strict limitation of industrial system standard and security, the current protocol stack of IIoT is protected and not allowed to be modified at will. Moreover, it is unrealistic to customize the protocols in the different scenarios of IIoT system. Therefore, in the above studies, the above three types of methods are not fit for the current IIoT environment.

Nowadays, many scholars tend to change the IIoT system structure and deploy the protecting measures for network security on the periphery of IIoT system architecture. They propose the idea of applying the software-defined networking (SDN) paradigm to IIoT and deploying intelligent algorithms on the SDN controller for security protection1,12–15. In this architecture, due to the global view on the network flows and strong computing capability of the controller, the machine learning algorithm can be deployed on the controller conveniently and flexibly. Recently, many studies16–18 have applied the machine learning to the ARP attack detection. Through learning the data samples, they can make the detection more intelligent to deal with complex situations.

However, the above machine learning–based methods have the following problems: (1) The features are extracted from only one aspect limitedly, without observing the ARP attacks from various perspectives in the network. This brings about the low detection accuracy. For example, the methods in the works by Ma et al.16 and Divya and Christopher17 extract ARP packet–based features without considering the global network traffics in the network. The method in the work by Gu et al.18 takes into account the network traffics but no ARP packet characteristics. In fact, features can be further extracted from a comprehensive perspective to help improve the detect performance of ARP attacks. (2) The methods based on supervised learning rely on large numbers of labeled data. But in fact, it is very difficult to obtain sufficient open-source labeled data (i.e. ground truth labels) coming from large companies at present. Since the labeled data are difficult to obtain, the supervised learning–based method trained on a small number of labeled samples only achieves poor detection result.

Based on the above analysis, our article uses the software-defined industrial Internet of Things (SDIIoT) as our basic system architecture. In our article, we design a multi-factor integration-based semi-supervised learning method, called MIS, to solve the ARP attack problem specially for SDIIoT. Meanwhile, this method can solve the above two issues in the traditional

![Figure 1. The number of ARP attacks observed through two switches’ ports.](image)
machine learning–based ARP attack detection studies. Our main contributions are listed as follows:

- For the problem of limited feature extraction, in MIS, we introduce a multi-factor integration idea for extraction of ARP attack features. We define and extract features from three aspects, that is, ARP packet–based features, entropy-based features, and vulnerability features. These features can help detect ARP attacks with reducing the ARP attack misjudgment.

- For the problem of limited labeled samples, in MIS, we propose a differential priority sampling–based semi-supervised learning framework for ARP attack detection. In this framework, the differential priority sampling in self-training can improve the training efficiency by getting more information from different samples. Meanwhile, the designed weighted ensemble learning mode in this framework reduces the mislabeling on the unlabeled training samples and further improves the detection performance.

- We conduct experiments based on a real data set collected from a deepwater port and a simulated data set. The experiments show that MIS can achieve good performance in detecting ARP attacks with F1-measure, accuracy, and AUC (area under the curve) of 97.28%, 99.41%, and 98.36% on average. Meanwhile, compared with fully supervised learning and other popular ARP detection methods, MIS also shows the best performance.

Related work

So far, researchers have put forward many approaches to solve the ARP attack problem. In this section, we introduce and classify them into four categories.

Modified ARP protocol–based solutions

Alharbi et al. designed a controller component to make ARP stateful by maintaining the list that records the hosts sending ARP request packets. This component stores the IP-MAC addresses of the source host, overwrites them with pre-set safety values, and checks the ARP reply’s validity based on whether there is a corresponding request packet. Kponyo et al.4 modified ARP by adding a padding layer with the pseudo-random binary sequence and the sequence number to the ARP request and reply packets. This method makes the request packet match its corresponding reply packet. Nam et al.5 proposed an enhanced version of ARP in the ethernet, called man-in-the-middle-resistant ARP (MR-ARP). It uses a long-term mapping table to maintain the IP-MAC mappings of active hosts. In order to update this table correctly, this method defines ARP voting packets to vote on the mapping relationship among the hosts. When observing a new IP-MAC mapping from an ARP reply packet, the host sends a voting request to its neighboring hosts and then verifies the mapping credibility through the voting packets.

Cryptography-based solutions

Bruschi et al.6 presented a solution in the local area networks, called secure ARP (S-ARP). It uses asymmetric encryption technology to encrypt and decrypt ARP reply packets. Each host maintains a public–private key pair. The sender uses the private key to digitally sign the reply message, and the receiver searches the public key corresponding to the IP address of the source host to verify the signature. The key is stored in a deployed authoritative key distributor server. Goyal and Tripathy7 proposed an enhanced S-ARP with the combination of digital signatures and passwords based on hash chains in the local area networks. In order to reduce the extra computation, the solution allows multiple uses of the same digital signature in multiple ARP replies within a predefined period of time. Lootah et al.8 designed a ticket-based ARP (TARP) within local area networks. This method uses a local ticket agent to distribute centrally issued secure IP-MAC address pairs included in the tickets, and meanwhile a key management server issues public keys to obtain the tickets.

ICMP and DHCP protocols–assisted solutions

Jinhua and Kejian9 proposed an efficient detection algorithm based on ICMP protocol to detect malicious hosts. This method sends an ICMP trap ping packet to the suspicious host that sent the ARP reply packet containing the invalid source IP-MAC mapping and analyzes ICMP reply packets to probe for the malicious host. Besides, Cox et al.10 designed an SDN security module in the local area networks, called network flow guard for ARP (NFGA). They hash each device’s MAC address with an appropriate IP-Port association by monitoring DHCP offers, requests, and acknowledgments from valid DHCP servers to prevent ARP spoofing. Jehan and Haneef11 proposed a scalable ethernet architecture using SDN. In this solution, the ARP module is written into the controller and maintains the IP-MAC address mapping table of hosts in the controller. The mapping information is obtained from the DHCP packets exchanged between hosts and the DHCP server. When receiving an ARP request or reply packet, the ARP module checks the existence of the packet’s IP-MAC pair in the table to detect the ARP attack.
Machine learning–based solutions

Ma et al.\textsuperscript{16} designed a Bayes-based ARP attack detection algorithm deployed in the cloud center. They calculate the probability that a host is an attacker, through the normal and abnormal ARP features. If the value of probability is above a certain threshold value, the host is declared as an attacker. Similarly, Divya and Christopher\textsuperscript{17} gave a Bayes support vector regression–based ARP (BSVR-ARP) spoofing protection mechanism. It can still detect the ARP attacks even if the attack frequency is low. Gu et al.\textsuperscript{18} proposed a reinforcement learning–based attack detection model that can automatically learn and recognize various attack patterns in the Internet of things. It uses entropy-based metrics to detect attacks in the network and uses Q-learning to continuously update the attack detection threshold.

To sum up, we can see these studies can deal with ARP attacks from different aspects. We provide a summary in Table 1. However, as stated in the introduction and Table 1, the limitations of these methods make them not suitable for the current IIoT. Therefore, we give the scenario and our proposed method in the following parts.

Preliminary and scenario of ARP attacks in SDIIoT

In network communications, hosts identify each other with IP addresses. When a source host has data to send to a destination, it must know the destination’s IP address provided by the network layer in advance. However, the data transmission through the physical network must depend on the MAC address rather than the IP address. The work of converting the known IP address to the MAC address is completed by the ARP protocol. The source host broadcasts the ARP request packet to obtain the ARP reply packet of the destination. The reply packet contains the destination’s IP address and MAC address. In this way, the source host can successfully transfer data and record the address mapping relationship in the ARP cache of the source to facilitate the subsequent data transmission between the two parties. The ARP protocol is stateless and has no authentication mechanism. Therefore, any malicious host can actively send false ARP packets to other hosts and can be trusted by the victim host unconditionally. Then the victim updates the local ARP cache table according to the content of the received packets. Attackers can use this method to collect the user’s private data and modify or even stop the data transmission.

Figure 2 shows a scenario of ARP attacks against the information transmitted in SDIIoT, where the procedures $\ddag$–$\ddag$ describe an ARP attack process. Among the three hosts in Figure 2, we assume that Host1 and Host2 are legitimate users, while Host3 is an attacker. Here, we assume that the ARP attacks occur due to external intrusion into the internal computer Host3. That is to say, Host3 is disguised as a legitimate user, but it is actually an illegal user who launches ARP attacks. The virus invading the host makes Host3 send false ARP packets and establish a connection with key devices. The purpose of this attack is to actively

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Method & Modified ARP protocol–based solutions & Cryptography-based solutions & ICMP and DHCP protocols–assisted solutions & Machine learning–based solutions \\
\hline
Work & 3–5 & 6–8 & 9–11 & 16–18 \\
Advantages & Simple principle of modifying protocol & Good security brought by encryption & Effective use of other protocols & Good flexibility in complex situations \\
Disadvantages & Conflict between modified and standard ARP protocol & Increase in computing costs and overhead & Poor security and the implementation of cross-layer protocols & Limited features and insufficient labeled data \\
\hline
\end{tabular}
\caption{The summary of four categories of the related work.}
\end{table}

ARP: address resolution protocol; ICMP: Internet control messages protocol; DHCP: dynamic host configuration protocol.
eavesdrop on the information exchanged between affected parties, such as authentication information, operation information. When Host1 and Host2 communicate for the first time, they need to obtain each other’s MAC address with ARP packets. Assume that Host1 sends an ARP request packet to the switch to query the MAC address corresponding to IP2. Due to lack of the matching flow rule, the switch encapsulates the request packet in the packet-in message and sends it to the controller. Then the controller sends a packet-out message to the switches to broadcast the request packet. After receiving the request packet, Host2 and Host3 both send reply packets instead of just one expected Host2. Because the mapping table is updated with the latest received packet, Host3 forges and sends the reply packets to submerge the legitimate reply packet sent by Host2. Then the switch encapsulates them in packet-in messages and sends them to the controller.

The forged addresses cause the controller to update the wrong address mapping. Therefore, the controller sends the wrong flow rules to the switches for execution. Finally, the normal communications between Host2 and other hosts cannot be established. The information originally sent to Host2 will be forwarded to Host3.

**Implementation and framework of MIS**

The method MIS can be deployed on the SDN controller. The controller establishes an ARP global cache at the phase of network initialization. Each entry of this ARP cache table stores configuration information of a host. The configuration information specifically includes the host’s MAC address, IP address, access port (i.e. the port number of the switch connected to the host). In MIS, when an ARP request or reply arrives at the controller, the global cache does not immediately and unconditionally trust the mapping information in the packet. The cache only establishes a temporary mapping relationship according to the source IP address, MAC address, and port number. The temporary mapping is eventually updated to or released from the global cache based on the results of ARP attack detection.

Note that, when the corresponding MAC addresses are changed with the change of the network topology and components, the corresponding switch port can monitor/observe the sensors’ adding or removing through servers in Figure 2. Next, the switch sends the Port-Status message to the controller since the switch port configuration state changes. After receiving the Port-Status message, the controller will send an ARP request packet to this switch port and then further deliver to the node (sensor). Then the corresponding ARP reply packet is expected to be sent to the controller for the verification of MIS. In addition, DHCP is applied to the IIoT network for the dynamic allocation of IP addresses. When the DHCP changes the IP address of a node, the DHCP server will send a DHCP acknowledgment message to this node to confirm the effective new IP address. When the DHCP message is received by the switch port, the controller will send an ARP request packet to the updated node, so as to receive an ARP reply packet to verify the mapping relationship with MIS.

The framework of MIS is shown in Figure 3. MIS includes two main components. One is the multi-factor feature extraction, in which ARP packet–based features, entropy-based features, and vulnerability features can be extracted and organized into a feature vector. The other is our designed ARP attack detection model, in which a differential priority sampling method is given and a specific ARP detection model combining differential priority sampling with multi-factor features is proposed.

**Multi-factor feature extraction**

In this section, we give the extraction of multi-factor features. Here, we consider three types of features, that is, the ARP packet–based features, entropy-based features, and vulnerability features.

First, the ARP packet–based features describe the unique characteristics of the ARP attacks. Considering that ARP attacks are mainly implemented on the address mappings, we focus on the mapping relationships contained in ARP packets, the number of ARP request, and reply packets as features (section “ARP packet–based features”).

Besides, if we only extract features from the ARP packet itself, it is easy to ignore the global characteristics of various protocol packets and the randomness of the network traffic distribution based on the IP addresses. Thus, from the global perspective of the network traffics, we introduce the entropy-based features, including the protocol entropy, the IP source entropy, and the IP destination entropy (section “Entropy-based features”).

Third, we further consider the vulnerability features. The extraction of ARP vulnerability features consists of
the historical ARP attacks launched by a host and the security configurations of a host. This type of feature extractions can help to judge whether a host is vulnerable to become an ARP attacker (section “Vulnerability features”).

Note that, in the following description, we define a unified time window \( T \), during which we observe and extract these features. The value of \( T \) is given in the experiment. Here, we list the symbols mainly used in this article in Table 2.

### ARP packet–based features

In this section, we analyze when an ARP attack occurs, what are the possible representations on ARP packets. We can extract these representations as features.

First, as described in section “Preliminary and Scenario of ARP Attacks in SDIIoT,” in a normal ARP communication, when a source host broadcasts the ARP request packet, only the expected destination will send the ARP reply packet to respond to the source request. However, making use of this, the attacker could send a large number of ARP reply packets after receiving the ARP request packet to cover the real content sent by the legitimate host. It makes the final content written by the address information forged by the attacker. Based on this, we extract the number of request packets received by a host and the number of reply packets sent during the time window \( T \), denoted by \( N_{req} \) and \( N_{rep} \), respectively. When an ARP attack launches, \( N_{rep} \) may show a significant increase, and be larger than \( N_{req} \). Therefore, \( N_{req} \) and \( N_{rep} \) can be extracted as features.

Second, according to the address mapping information from the ARP reply packets, ARP attacks can be divided into two categories:

- One is that the attacker sends the incorrect IP-MAC mapping relationship in the ARP reply packet.
- The other is that the attacker sends true IP-MAC information of the other host associated with another switch port. In this case, the attacker can modify the MAC address of the network interface through hardware or software tools, becoming the same copy as the imitated.

The two categories are also shown in Figure 2, with procedure ©. In one case, attacker Host3 can forge its IP address into that of Host2 and send the reply packet containing the incorrect IP address to the switch. As a result, the controller learns the wrong mapping information (i.e. IP2-MAC3-Port3). In another case, Host3 can send the true IP-MAC mapping information of the legitimate Host2 (i.e. IP2-MAC2), instead of its own mapping IP3-MAC3. In this case, the controller updates the wrong access port information (i.e. IP2-MAC2-Port3). In both cases, the entry with incorrect host configuration information can finally affect the normal communication between Host2 and other hosts.

As stated in section “Implementation and framework of MIS,” the SDN controller has a global cache with storing the IP-MAC and MAC-Port mapping relationships. Based on the above descriptions, we have the following two types of mapping schemas:

### Table 2. Mainly used symbols.

| Symbols     | Description                                                                 |
|-------------|------------------------------------------------------------------------------|
| \( T \)     | the observing time window                                                    |
| \( N_{req} \) | the number of requests received by a host                                    |
| \( N_{rep} \) | the number of replies sent by a host                                         |
| \( R_{IP-MAC} \) | the mapping schema of IP-MAC                                                |
| \( R_{MAC-Port} \) | the mapping schema of MAC-Port                                            |
| \( H_{prot} \) | the protocol entropy                                                        |
| \( H_{IP} \) | the IP source entropy                                                        |
| \( H_{MAC} \) | the IP destination entropy                                                   |
| \( Num_{prot}(x_i) \) | the number of packets transmitted by a host with protocol \( x_i \)          |
| \( N_{prot} \) | the total number of network protocols                                        |
| \( p_{prot}(x_i) \) | the appearance probability of protocol \( x_i \)                           |
| \( N_{port} \) | the total number of packets sent by a host to different destination hosts    |
| \( p_{port}(y_i) \) | the appearance probability of IP destination address \( y_i \)             |
| \( F_{att} \) | the number of attacks conducted by a host historically                      |
| \( F_{\Delta} \) | the time interval from the last attack conducted by a host                  |
| \( N_{v} \) | the number of vulnerabilities belonging to the level “Critical”             |
| \( N_{h} \) | the number of vulnerabilities belonging to the level “High”                  |
| \( N_{m} \) | the number of vulnerabilities belonging to the level “Medium”                |
| \( F_{\alpha} \) | the vulnerability factor of a host                                          |
| \( K \) | the number of clusters in K-means algorithm                                  |
| \( C_i \) | the \( i \)-th cluster obtained by K-means                                  |
| \( N_i \) | the number of samples selected from the unlabeled data set in one differential priority sampling |
| \( B \) | the base learner                                                            |
| \( L \) | the labeled data set                                                         |
| \( U \) | the unlabeled data set                                                       |
| \( M \) | the ensemble size                                                           |
| \( L_m \) | the \( m \)-th labeled training subset for \( B_m \)                         |
| \( A_m \) | the \( m \)-th synthetic evaluation score                                   |
| \( B_m \) | the \( m \)-th base learner                                                  |
| \( U_m \) | the \( m \)-th unlabeled training subset for \( B_m \)                        |
| \( OOB_m \) | the \( m \)-th out-of-bag set to evaluate \( B_m \)                           |
| \( \delta \) | the number of selected unlabeled subsets for each base learner              |
| \( A_{best} \) | the \( m \)-th best synthetic evaluation score                              |
| \( B_{best} \) | the \( m \)-th best learner                                                 |
| \( \alpha_m \) | the weight assigned to \( B_{best} \)                                      |

IP-MAC: Internet protocol-media access control; IP: Internet protocol.
• In the normal ARP communication, an IP address should correspond to only one MAC address, which is also associated with only one access port. That is to say, IP-MAC and MAC-Port are, respectively, in one-to-one mapping schema.

• In the ARP attack scenario, the attacker unexpectedly sends a fake ARP reply to the source host when the source broadcasts an ARP request to ask others for the MAC address matching the known IP address. In one case, the attacker forges the MAC address. Then the ARP reply packet with forged mapping information causes the known IP address to be mapped to the wrong MAC address, which results in one-to-many IP-MAC mapping schema for this IP. In another case, the attacker imitates another legitimate host’s IP-MAC pair. Then the ARP reply can result in wrong mapping information of MAC-Port, which leads to one-to-many MAC-Port mapping schema for the true MAC mapping to this IP. For convenience, here we call this type of one-to-many mapping schema as the non-one-to-one mapping schema.

Thus, for each host, we can define two features $R_{ip-mac}$ and $R_{mac-port}$, where $R_{ip-mac}$ denotes the mapping schema of IP-MAC and $R_{mac-port}$ denotes the mapping schema of MAC-Port. The value of these two features is 1 (i.e. one-to-one mapping schema) or 0 (i.e. non-one-to-one mapping schema). To be specific, from each ARP reply packet, we can extract the IP-MAC and MAC-Port mapping relationships which are ready to be written to the SDN global cache. Then, we compare the above newly extracted IP-MAC and MAC-Port mapping relationships with the original IP-MAC and MAC-Port mapping relationships stored in the global cache, respectively. If the corresponding comparison result is same, the value of $R_{ip-mac}$ or $R_{mac-port}$ is 1. Otherwise, the value is set to 0.

Note that the legitimate user may change the mapping schema through the normal virtual machine migration and the normal network configuration change. These behaviors cause the legitimate user to appear the non-one-to-one mapping schema. When the virtual machine migrates, it changes the actual access port, although the IP-MAC mapping remains unchanged. It makes the non-one-to-one mapping schema appear in the MAC-Port mapping. In another case, during the host configuration modification, the legitimate host can change its IP address and keep the access port unchanged, which results in the non-one-to-one IP-MAC mapping schema.

Therefore, based on many of the cases above, we can see that more features are required to help the detection of ARP attacks. To sum up, according to the SDN cache updating mechanism and address mapping relationships, here we extract four ARP packet–based features, that is, $N_{req}$, $N_{rep}$, $R_{ip-mac}$, and $R_{mac-port}$.

### Entropy-based features

Entropy is one of the indicators to measure the dispersion degree of sample sets in mathematics. It is used to measure the randomness of information variables in the information theory. When the information variable shows large randomness, the entropy value also tends to be large.

Generally speaking, the normal communication behavior of a host is relatively random. Therefore, the entropy value of the normal network traffic through a host tends to be stable in the long run. However, when an aggressive behavior conducted by a host, the traffic may concentrate on the attacker, which leads to the decrease of the randomness of the information variable and makes the value of entropy change.

In this section, according to the characteristic of ARP attacks, we extract three entropy-based features for each host, that is, $H_{e-prot}$, $H_{e-scrIP}$, $H_{e-dstIP}$, where $H_{e-prot}$ represents the protocol entropy based on network communications through a host, $H_{e-scrIP}$ and $H_{e-dstIP}$ represent IP source and IP destination entropy based on the IP address of a host, respectively.

First, there is no doubt that the ARP attack packet uses ARP protocol as its protocol field. Therefore, a large proportion of packets with ARP protocol can be obtained during an ARP attack launched by a host. This will lead to the decrease of protocol entropy.

We observe the number of packets transmitted by different protocols within the time window $T$. Suppose $Num_{prot}(x_i)$ denotes the number of packets transmitted by a host with the protocol $x_i$. For a host, we can calculate an appearance probability of protocol $x_i$

\[
P_{prot}(x_i) = \frac{Num_{prot}(x_i)}{\sum_{i=1}^{n} Num_{prot}(x_i)}
\]

where $N_{prot}$ represents the total number of active network protocols within $T$ on the host.

Furthermore, we can obtain the protocol entropy $H_{e-prot}$ for a host as follows

\[
H_{e-prot} = -\sum_{i=1}^{N_{prot}} P_{prot}(x_i) \log_2 P_{prot}(x_i)
\]

where the denominator is used as a normalization factor.

Second, when an ARP attack happens, the attacker tends to send many forged ARP reply packets to the victim. So, the attacker usually uses the victim host’s IP as the destination address of its attack packet. As a result, the number of packets received by the victim.
from the attacker increases. As the randomness of the source (attacker) address decreases, and the IP source entropy also change.

Thus, for a host, we can define an IP source entropy $H_{e-\text{scrIP}}$ to describe the access degree of this host to other IP addresses during the time window $T$. Like equation (2), we can obtain an appearance probability $P_{\text{scrIP}}(y_i)$ of IP address $y_i$ from the traffics sent by a certain host. We use $N_{\text{scrIP}}$ to represent the total number of packets sent by this host. Therefore, we can obtain the IP source entropy $H_{e-\text{scrIP}}$ for a host as follows:

$$H_{e-\text{scrIP}} = -\frac{\sum_{i=1}^{N_{\text{scrIP}}} P_{\text{scrIP}}(y_i) \log_2 P_{\text{scrIP}}(y_i)}{\log_2 N_{\text{scrIP}}}$$ (3)

The stronger the randomness of the IP address targeted by the current host is, the larger the entropy $H_{e-\text{scrIP}}$ is.

Similar to equation (3), we also can obtain an IP destination entropy $H_{e-\text{dstIP}}$ for a host. After establishing communications with an victim using the forged ARP reply packet, the attacker host may receive a lot of information from the victim. Therefore, the randomness of the destination (attacker) address decreases, which results in the decrease of IP destination entropy. Here, we do not repeat the calculation process.

To sum up, here we extract the above three entropy-based features, that is, $H_{e-\text{prot}}$, $H_{e-\text{scrIP}}$, and $H_{e-\text{dstIP}}$.

**Vulnerability features**

Here, we consider two types of vulnerability features. One is related to the historical attacks launched by a host. The other is related to the secure configuration environment of a host.

**The historical attacks launched by a host.** A host that has launched attacks may continue to attack the victim until its goal is achieved. The more attacks a host launches, the more threatening it is. In the process of ARP attack, the attacker can send ARP packets to the victim host at a fixed frequency, in order to increase the probability of the cache table being successfully cheated.

We define two variables $F_{\text{att}}$ and $F_{\Delta}$, to represent the number of attacks conducted by a host historically and the time interval from the last attack of it to now. Therefore, $F_{\text{att}}$ and $F_{\Delta}$ can be extracted as features.

**The security configuration environment of a host.** If a host is compromised to be an attacker continuously, it may be a critical device connected to other equipment which store important information related to business, and so on. Besides, it also may be a host with poor security configuration for many vulnerabilities and the intruder can use relatively low cost to invade it.

Vulnerability scanning is the most effective way to find vulnerabilities using tools to test the target hosts. Here, we use a popular vulnerability scanner called Nessus\textsuperscript{19} to scan each host and obtain its vulnerabilities. Nessus has its own vulnerability feature library, which can complete the vulnerability identification for a large number of hosts in the network. The scanning targets can be detected by uploading the list of IP addresses.

The vulnerabilities are divided into five levels in Nessus, that is, “Critical,” “High,” “Medium,” “Low,” and “None.” We focus on the first three levels and use $N_c$, $N_h$, $N_m$ to denote the number of vulnerabilities of one host belonging to those three levels, respectively. Meanwhile, we define a vulnerability factor $F_v$ for a host to describe its vulnerability to attacks, which is shown in equation (4)

$$F_v = 0.5N_c + 0.3N_h + 0.2N_m$$ (4)

Since we concern on the vulnerability here, the most serious level should be assigned with large coefficient, instead of equal coefficients. We assign values to them, respectively, according to the severity level of the vulnerabilities. Because $N_c$ denotes the number of the vulnerabilities with most serious level “Critical,” we assign 0.5 (one half) as the coefficient of it. The coefficient is relatively larger than the other two. In this way, the vulnerability factor can focus on the “Critical” vulnerabilities. The remaining 0.5 is allocated to $N_h$, and $N_m$ according to the ratio of three to two in that the level “High” is more serious than “Medium.” This coefficient assignment reflects the different security of diverse vulnerabilities in the face of attacks.

In the practical application, the value of coefficient can be slightly adjusted by the administrator according to specific conditions. The larger the $F_v$ value is, the worse the security configuration of the host is, which means the host is more difficult to resist ARP invasion.

To sum up, by considering historical attack behavior and vulnerability of a host, we can extract the above three vulnerability features, that is, $F_{\text{att}}$, $F_{\Delta}$, and $F_v$.

**The design of MIS method for ARP attack detection in SDIIoT**

As described in section “Introduction” in many practical applications, it is very difficult to obtain enough labeled data due to data confidentiality and privacy. However, obtaining unlabeled data is relatively easy. Thus, we have to rely on semi-supervised learning, which can deal with few labeled samples, to solve the ARP detection in our article. Different from the fully supervised learning, the semi-supervised learning uses
few labeled data and large number of unlabeled data to construct a detection model, making full use of the information contained in the labeled and unlabeled data.

In the semi-supervised learning, self-training and co-training are two types of training modes. Self-training is a semi-supervised learning mode in which a base learner keeps on labeling unlabeled examples and retraining itself on an enlarged labeled training set, while co-training is a multiview semi-supervised learning mode which uses predictors from different views to learn from each other. The former is simple and can be combined with any predictor; the latter requires full redundancy between different views of the samples, which is difficult to achieve in the practical application. In this article, we choose self-training mode to expand the labeled sample set in order to better detect ARP attacks.

In the following parts, we introduce the design of a multi-factor integration-based semi-supervised learning (MIS) for ARP attack detection in SDIIoT. First, in the self-training, we design a differential priority sampling method to optimize the sampling (section “Differential priority sampling”). Second, we give the detailed description about our ARP attack detection model with differential priority sampling and multi-factor features (section “ARP attack detection model with differential priority sampling and multi-factor features”).

Differential priority sampling

In the self-training, the base learner uses the limited labeled samples to train a model initially. Next, it will randomly select some unlabeled samples to pre-label them. Then, these new labeled samples will be added to the labeled training data set to retrain and optimize the learned model, as shown from step ① to step ④ in Figure 4. However, in this process, the random selecting usually leads to the imbalance of learning. If the learner selects some unlabeled samples with high similarity, the training efficiency will decline due to the low amount of information obtained from the similar samples.

Therefore, when selecting the unlabeled samples, we design a differential priority sampling method to solve this problem. The difference is reflected by dividing the unlabeled samples into several clusters with high intra-cluster similarity and low inter-cluster similarity through a clustering algorithm. The priority is reflected by giving priority to the samples close to the cluster center within one cluster. So we can obtain some unlabeled samples with differential priority. In this way, the various samples with differences are provided for model training so as to improve the training efficiency.

First, there are many clustering algorithms, such as K-means algorithm, hierarchical clustering algorithm, and density-based spatial clustering of applications with noise (DBSCAN). K-means algorithm is easily implemented when solving many practical problems, with low time complexity and better clustering effect on the large-scale data set according to the work by Xu and Wunsch. Therefore, considering the convenience and efficiency, we adopt the K-means algorithm here to cluster the unlabeled samples. In addition, the number of clusters K is also very important. The inappropriate choice of K brings about poor clustering results. In order to find the optimal value of K, Zhang et al. proposed an improved K-means algorithm based on density Canopy. This method has good anti-noise performance and can converge quickly. So we introduce this method to determine the K value. According to the maximum weight product method mentioned in this work, the K value and initial cluster centers can be obtained.

Next, we give the detailed procedures of the differential priority sampling method, which is shown below:

1. Based on the 10 multi-factors in section “Multi-factor feature extraction,” we implement the above-stated K-means algorithm on the unlabeled data set. Thus, we can obtain K clusters, denoted by \( \{C_i\}_{i=1}^{K} \). Meanwhile, there are \( |C_i| \) members in the i-th cluster \( C_i \).
2. Assume that now we need to select the number of \( N_i \) samples from the unlabeled samples in this iteration. The value of \( N_i \) depends on our differential priority sampling–based semi-supervised learning framework, which is given in the following section.
3. From each cluster, we select \( \frac{|C_i|}{\sum_{i=1}^{K} |C_i|} \times N_i \) samples with the nearest distance to the cluster center and remove them from cluster \( C_i \).

Based on the above steps, we can obtain \( N_i \) unlabeled samples in one differential priority sampling. This helps us to obtain unlabeled samples with large differences, which is good for further semi-supervised learning.
ARP attack detection model with differential priority sampling and multi-factor features

In this section, we give our core detection model-MIS, which combines the above extracted multi-factor features and a new designed differential priority sampling–based semi-supervised learning framework. As stated above, we choose self-training mode to design MIS. First, the self-training should choose a base learner to handle the unlabeled data and obtain the training model (section “The choice of a base learner”). Second, a differential priority sampling–based semi-supervised learning framework for ARP attack detection is given (section “Differential priority sampling–based semi-supervised learning framework for ARP attack detection”).

The choice of a base learner. Based on Figure 4, we can use the differential priority sampling in step ② to obtain a set of unlabeled samples which are ready to be fed into a base learner to get predicted labels. After that each unlabeled sample has different probability values belonging to different labels. Then, in self-training, the learner usually chooses several samples with highly ranked predictive probability values and adds them into the labeled data set (i.e. step ② in Figure 4). The new labeled data set would experience another training session in the base learner to continually expand the labeled samples in self-training.

In this process, we can see that the choice of a base learner can impact the ranking results of belonging probability of the predicted samples. It is very important since a suitable base learner could generate more precise pre-labeled samples to expand the labeled data set. Besides, our ARP attack detection is a classification problem. So, the selected base learner should produce a good ranking result of belonging probability of the pre-labeled samples and have a strong classification ability.29

Nowadays, there are many choices for the base learner, for example, decision tree, naive Bayes classifier, support vector machine (SVM), and K-nearest neighbor (KNN). KNN conducts predictions by searching the nearest neighbors in the entire training set for a testing sample. This makes its predictive phase relatively inefficient, which does not meet the need for quickly identifying attacks. SVM needs to solve the quadratic programming (QP) problem, whose size depends on the training set, to obtain support vectors. The large-scale training data would cause high time complexity and memory complexity.

Naive Bayes and decision tree are two algorithms widely used in classification problem for their good classification performance. Compared with KNN, they are more efficient in the testing phase because the testing results can be obtained by directly inputting the training model instead of calculating and searching the training samples. Compared with SVM, their time complexity growth is relatively more smooth when the training data increase. However, decision tree has difficulties in producing a good ranking because every sample from a particular leaf would receive the same probability estimate. That is to say, all data points on one side of the split are assigned the same probability, corresponding...
to the proportion of the class that falls on the corresponding side of the split. Compared with the decision tree, naive Bayes classifier is a good ranker. It can produce stable classification efficiency and process data quickly. Therefore, we choose naive Bayes classifier as the base learner in our self-training framework. In the experiment, we also do some related tests to verify the good performance of choosing naive Bayes classifier.

**Differential priority sampling–based semi-supervised learning framework for ARP attack detection.** After selecting an appropriate base learner, we should consider whether it can be optimized in the self-training process. An appropriate base learner cannot totally guarantee the improvement of learning performance in self-training. If there are wrong labels in a base learner, the propagation of wrong labels could lead to the poor prediction performance.

In order to avoid this problem, we introduce the paradigm of ensemble learning to make up for self-training. Ensemble learning is usually used to improve the generalization ability of learners and deal with the mislabeling. It constructs and combines several base learners to achieve better prediction performance. Zhou presented an argument that ensemble learning is indeed beneficial to semi-supervised learning.

Bagging, boosting, and stacking are three commonly used ensemble learning modes. Bagging and boosting combine base learners by following a deterministic strategy, while stacking combines base learners by training a meta-model. Thus, the training of the stacking model is commonly more time-consuming due to its further learning of the base learners’ predictions. Besides, bagging can process the computation of training different models in parallel. However, boosting can only train different models iteratively, which may lead to high computational overhead. Thus, from the above analysis, we choose bagging as our ensemble learning mode. Bagging has great randomness in the process of sampling for each base learner. The output of self-training is uncertain for the pre-labeled samples added to labeled data set. Thus, the base learners are much diverse, and their combined model can have strong generalization ability.

A differential priority sampling–based semi-supervised learning framework is given in Figure 5. In this figure, we highlight three main functional modules. The upper left of the figure is function, that is, weight assignment and the ensemble learner construction. It weights the optimal learners and integrates them into a final ensemble learner. Function provides highly different unlabeled data for the semi-supervised learning of the learner in function. Function provides function with the optimal learners obtained from the self-training process of the learners, and function integrates these learners to form a final high-performance learner for ARP attack detection.

In Figure 5, functional module, the differential priority sampling has been given in detail in section “Differential priority sampling.” Here, we do not repeat them. Next, combining with Figure 5, we give the pseudocodes of MIS in Algorithm 1 for an intuitive expression and describe its procedures in detail in the following parts.

In Algorithm 1, we take a labeled data set $L$ and an unlabeled data set $U$ as inputs, and take naive Bayes classifier as the base learner $B$. The multi-factor feature extraction in section “Multi-factor feature extraction” (including 10 features) is applied to the labeled data set $L$ and unlabeled data set $U$. The labeled data set contains the labels describing whether an ARP packet data item is aggressive or normal. In the input, we set $M$ as the ensemble size in bagging, that is, the ensemble learner contains $M$ base learners. Meanwhile, we set the sampling size of differential priority sampling $N_s$ as $\left\lfloor \frac{L}{M} \right\rfloor$. The output is an ensemble prediction model $B_{ens}$ for ARP attack detection.

First, execute the differential priority sampling on $U$ with the sampling size of $N_s$ and $M$ sampling times (Line 1). In each round, we choose $N_s = \left\lfloor \frac{L}{M} \right\rfloor$ unlabeled samples to put into a differential priority sampling subset, denoted by $\overline{U}_m$ ($m = 1, \ldots, M$). If the number of unlabeled samples is less than $\left\lfloor \frac{L}{M} \right\rfloor$ in the last $M$-th sampling, we directly put the remainders into the subset $\overline{U}_M$. We can obtain the unlabeled subset $\overline{U}_m$ ($m = 1, \ldots, M$) with descending priority. The smaller the value of $m$ is, the better the difference within the set is and the higher the priority is.

Then, for each learner $B_m$ which constitutes the ensemble model, optimize itself on the labeled and unlabeled training sets, and get the best $B_m$ with the evaluation score in the iterations of self-training (Lines 2–24).

To be specific, the training set acquisition process is shown from Line 4 to Line 6. Each learner $B_m$ randomly selects $\delta$ unlabeled subsets and stores them in the unlabeled training set $\overline{U}_m$ (Line 4). In the experiment, we test that the number of selected unlabeled subsets is not the more the better. When the number of selected unlabeled subsets $\delta$ is 3, the calculation time and the prediction performance are the best. Sample the labeled data set $L$ with replacement to generate a training subset $L_m$ with the size of $|L|$ for each learner $B_m$ ($m = 1, \ldots, M$) (Line 5)
and take the remaining unsampled data as the out-of-bag set $OOB_m$ (Line 6).

The process of learner optimization and selection is shown from Line 7 to Line 24. Train naive Bayes classifier $B_m$ on each labeled sample set $L_m$ (Line 7). When obtaining a naive Bayes classifier $B_m$, evaluate the base classifier $B_m$ with the set $OOB_m$ using a synthetic evaluation score with averaging on F1-measure (it comprehensively takes into account the true positives, the false positives and the false negatives), accuracy (it is the proportion of correct predictions (both true positives and true negatives) among the total number of samples examined), and AUC (AUC is defined as the area under the ROC (receiver operating characteristic curve). When comparing different learning models, the ROC curve of each model can be drawn. The area under ROC can be used as a metric to evaluate the performance of the learning model, which indicates the probability that the predicted positive cases are ranked before the negative ones.) to calculate a prediction score $A_m$ (Line 8). We store the initial prediction score $A_m$ into a variable $A_m^{best}$ and correspondingly store the initial learner into a variable $B_m^{best}$ (Lines 9–10).

We carry out the following steps in the subsequent iterations of self-training. Set an empty set $U_m^{conf}$ to store the selected samples with high prediction probabilities in Line 15. Use base learner $B_m$ to pre-label the unlabeled data in $U_m$ to generate labels and corresponding prediction probabilities (Line 13). Rank the pre-labeled data according to its probability (Line 14). Add top $10\% \cdot |L|$ pre-labeled probability value samples to $U_m^{conf}$ in proportion to the positive and negative class distribution of $L_m$ (Line 15). This makes the added data keep the class distribution unchanged. Move all samples from $U_m^{conf}$ to $L_m$ to expand the labeled set (Lines 16–17).

Next, train $B_m$ on the extended labeled set $L_m$ (Line 18). Evaluate $B_m$ with the set $OOB_m$ to obtain $A_m$ (Line 19). We compare $A_m$ with $A_m^{best}$ and take the higher score as the best score $A_m^{best}$ and its learner as the best learner $B_m^{best}$ (Lines 20–23). We terminate the iteration when the set $U_m$ is empty or the number and proportionality conditions of Line 15 cannot be met (Line 24).

We can obtain the optimal naive Bayes classifier $B_m^{best}$ and its optimal score in the above procedures. Next, we can assign a weight $\alpha_m$ to $B_m^{best}$ according to its score $A_m^{best}$ (Line 25). Combine the $M$ base learners to construct an ensemble learner. The weight of $B_m^{best}$ can be calculated by equation (5)

$$\alpha_m = \frac{A_m^{best}}{\sum_{m=1}^{M} A_m^{best}} \tag{5}$$

We can get the final ensemble predictor $B_{ens}$, given in equation (6) (Line 26)

$$B_{ens} = \sum_{m=1}^{M} \alpha_mB_m^{best} \tag{6}$$

Based on the above descriptions, we can see that $L_m$ is extended by adding the unlabeled data and their predicted labels continually and MIS completes the training processes.

Finally, for a testing sample, the ensemble learner can obtain the testing sample’s predictive label. The detection result is determined by the final predictive probability obtained from the output predictive probability of each base learner.

**Experiment and analysis**

In this section, we conduct some different experiments to verify the effectiveness of our proposed ARP attack detection method (MIS).

**Data set preprocessing and parameter settings**

In the following experiments, we evaluate the performance of MIS based on a real data set and a simulated data set. The real attack data set is obtained from a real IIoT environment with different types of attacks, including ARP attacks. The simulated data set is obtained from the simulated SDN network environment with ARP attacks. The real data set is a large data set with containing 1520 sensors, and the simulated data set is a relatively small data set with 20 hosts. Both are under control of the SDN controller. The two data sets are different scales. In the section “Performance evaluation of MIS,” all the experiments are done based on the two data sets and we can also observe the scalability performance of our proposed method MIS. More details of these two data sets are described as follows:

**Real data set:** The data are collected from a large freight port in Shanghai, which is a deepwater port with container terminals. The real data set contains the collected data from the IIoT with 1520 sensors. The collection devices access the switch using bypass mode. They copy and forward the network traffics of the switch ports to the designated ports, so as to obtain the mirror data. In total, there are 20,440,608 data items, mainly including 58,234 ARP attacks and 20,502 Denial of service (DoS) attacks. The data files are saved as PCAP (i.e. packet capture) format and can be opened using the Wireshark tool.

In the collected data set, a piece of data describes a complete packet. We can get the serial number, the source IP and MAC addresses, the destination IP and MAC addresses, the protocol type, and the attack type (label). In addition, we use Nessus tool to obtain the
vulnerability factors of the active hosts in the network and match them with the packets whose destination are the hosts the vulnerability factors belong to.

**Simulated data set:** We use the Mininet platform, the Open vSwitch, and the Ryu controller to simulate a SDN environment. First, install VMware virtual machine on the computer. Then, build a Ubuntu system with VMware and install Mininet platform on it. In Mininet, we set up a controller, a switch, and 20 host nodes. After the network topology is built successfully in Mininet, the Ryu controller can be started to realize communication between hosts.

We use Scapy to simulate normal traffics and ARP attack traffics. Scapy is a powerful and extensible program written by Python language to operate network packets. It can forge or decode diverse packets and send them on the wire, match requests, and replies, and conduct much more tasks. Scapy has functions to construct packets of most common network protocols. We can construct an ARP packet by combining the built-in functions of Ether() and ARP() in Scapy and forge an ARP attack packet by setting false IP or MAC address.

We set the attack rate to 300 packets/s and the number of attackers to 10 in Scapy. Meanwhile, we randomly assign the vulnerability factors to the 20 hosts. The generated communication packets are captured by Wireshark and saved in PCAP format. In total, 3,546,980 data items are generated, including 464,609 ARP attacks. Others are normal traffics.

The raw data in above two types of data sets mainly include timestamp, source IP, and MAC addresses as well as destination IP and MAC addresses. We further use “libpcap” and “dpkt” toolkits in Python library to analyze the packets in the two data sets, respectively. We identify a host by source MAC address in the ARP packet combining with its associated port. Then we take the host as the key to map the raw data set into a new processed data set with the 10 multi-factor features stated in section “Multi-factor feature extraction” within the time window. In MIS, we set the value of time window $T$ to 10 s and the ensemble size $M$ to 25. We choose Gaussian naive Bayes as the base classifier and use its default settings in Python.

**Performance evaluation of MIS**

In the experiments, we use 80% of the data set for training and 20% for testing. Among the training data, 15% is taken as labeled data, with the rest as unlabeled data. We mainly use three metrics F1-measure, accuracy, and AUC to measure the performance of MIS. The definitions of the three metrics have previously been given in the text. Here, we do not repeat them.

**Performance results on real and simulated data sets.** We evaluate the performance of MIS on the real and simulated data sets, respectively. The performance results are shown in Figure 6. From this figure, we can see that the F1-measure, accuracy, and AUC on the real data set can reach 97.01%, 99.38%, and 98.28%, respectively. And the F1-measure, accuracy, and AUC on the simulated data set can reach 97.54%, 99.44%, and 98.43%, respectively. This indicates that MIS can predict ARP attacks well on both real and simulated data sets. MIS has good performance against ARP attacks.

**Comparison between different base classifiers.** In section “ARP attack detection model with differential priority sampling and multi-factor features,” we choose naive Bayes as our base learner. Here, for naive Bayes (NBayes) and decision tree C4.5, we use F1-measure and accuracy to assess the prediction performance, and use AUC to evaluate the ranking ability. From Table 3, we can draw a conclusion that both of the base learners have good prediction performance on results. They are similar in F1-measure and accuracy. However, the ranking ability of naive Bayes classifier is better than that of decision tree with more than 20%. So it is reasonable to use naive Bayes classifier in our self-training.

**Comparison with several supervised learning methods.** We investigate whether MIS can successfully learn useful information for ARP attack detection from unlabeled data by comparing two supervised learning methods.

- Single naive Bayes classifier (S-NBayes): A single naive Bayes classifier trained on the labeled data.
- Bagging: A bagging model combining multiple naive Bayes classifiers trained on labeled data. Here, we set the number of base classifiers to 25.

The results are shown in Table 4. MIS outperforms the other methods significantly. Bagging provides diversity for the training of Bayes learners by sampling the
labeled data set with replacement randomly. So the results show that bagging has better performance than a single Bayes learner.

Comparing MIS with bagging, we can see that the performance of MIS has been further improved. In MIS, we design a differential priority sampling method and determine the final base classifiers based on the prediction score $A_m^\text{best}$ in self-training. This sampling method can help the classifier learn useful information from the unlabeled samples with high efficiency. Meanwhile, taking the base learner with the highest score $A_m^\text{best}$ in self-training can suppress the negative impact of wrong labels on the prediction performance. When combining the base learners, we also weight the basic classifiers based on prediction scores. Therefore, MIS obtains the best performance.

In addition, we observe the computational efficiency of these algorithms in terms of the running time. Obviously, compared with a single Bayes learner, the ensemble learner integrated by multiple Bayes learners needs to cost more running time. In MIS, the additional unlabeled data are used to iteratively optimize the single learner that constitutes the final learner. Since the other two are based on supervised learning, it is normal that MIS consumes more time than the other two. If the two also use the 15% labeled data as MIS, their detection performance is very poor. Therefore, MIS has the advantage of using a little bit longer training time to obtain a good performance on detection performance, especially with ability of handling the unlabeled data.

**Comparison with other ARP detection methods.** Here, we compare MIS with other three popular ARP attack detection methods. They are the threshold-based method,\textsuperscript{16} BSVR-ARP (i.e. Bayes support vector regression–based ARP)\textsuperscript{17} and the entropy-based method.\textsuperscript{18}

We compare MIS with the three methods on the real and simulated data sets. The results are shown in Table 5. The threshold-based method and BSVR-ARP both have relatively good performance. However, the features extracted from only one aspect limitedly makes these two methods perform worse than MIS. In contrast, the entropy-based method has the worst performance because it does not analyze the characteristic of the ARP packet elaborately. We can see that MIS has achieved the best performance by taking

### Algorithm 1 MIS

**Input:**
- $L$—the labeled data set
- $U$—the unlabeled data set
- $B$—the base learner
- $M$—the ensemble size

**Output:**
- $B_{\text{ens}}$—the final ensemble predictor

1. Conduct differential priority sampling on $U$ with the size of $N_s$ and the times of $M$.
2. For $m = 1$ to $M$ do
3. $B_m \leftarrow B$
4. $U_m \leftarrow$ Select $\delta$ subsets from $(U_1, U_2, \ldots, U_M)$ randomly
5. $L_m \leftarrow$ Sample $L$ with replacement
6. $OOB_m \leftarrow L \setminus L_m$
7. $B_m \leftarrow$ Train $B_m$ on $L_m$
8. $A_m \leftarrow$ Evaluate $B_m$ on $OOB_m$
9. $A_m^\text{best} \leftarrow B_m$
10. $A_m \leftarrow B_m$
11. Repeat
12. $U_{\text{conf}} \leftarrow ()$
13. Pre-label $U_m$ with $B_m$
14. Rank $U_m$ based on each data’s corresponding prediction probability
15. $U_{\text{conf}} \leftarrow$ Select top 10% of $|L_m|$ samples of $U_m$ in proportion to the positive and negative class distribution of $L_m$
16. $U_m \leftarrow U_m \setminus U_{\text{conf}}$
17. $L_m \leftarrow L_m \cup U_{\text{conf}}$
18. $B_m \leftarrow$ Train $B_m$ on $L_m$
19. $A_m \leftarrow$ Evaluate $B_m$ on $OOB_m$
20. if $A_m > A_m^\text{best}$ then
21. $A_m^\text{best} \leftarrow A_m$
22. $B_m^\text{best} \leftarrow B_m$
23. end if
24. Until $U_m$ is empty or does not meet the selection condition in Line 15
25. Calculate a weight $a_m$ for each $B_m^\text{best}$ according to equation (5)
26. Construct the ensemble learner $B_{\text{ens}}$ with $a_m$ and $B_m^\text{best}$ according to equation (6)
27. End for

Table 3. Comparison between different base classifiers.

| Methods | Data sets | F1-measure | Accuracy | AUC |
|---------|-----------|------------|----------|-----|
|         | Real      | Simulated  | Real     | Simulated | Real | Simulated |
| C4.5    | 0.8578    | 0.8695     | 0.9044   | 0.9111 | 0.5778 | 0.6426 |
| NBayes  | 0.8624    | 0.8782     | 0.9030   | 0.9124 | 0.8462 | 0.9168 |

AUC: area under the curve.
multidimensional features into account and making effective use of the unlabeled data.

In addition, we record the algorithm running time on the real and simulated data sets in Table 5 to evaluate on the computational cost of each method. Compared with the threshold-based method, BSVR-ARP is slower due to a additional unifying loss function designed for support vector regression to improve the Bayes performance. Besides, with the increase of algorithm complexity, the latter two have longer running time. The entropy-based method use the $E/C_0$ greedy strategy in Q-learning and has a slow convergence speed. MIS has a longest running time among four methods for the effective use of unlabeled data. However, the first three methods are based on supervised learning and have the input of all labeled data, while MIS is based on semi-supervised learning and its input contains many unlabeled data. When supervised learning–based methods are conducted on 15% of labeled data, they can only achieve poor results with accuracy below 0.6 and other performance indicators are also poor. The training on unlabeled data directly in MIS improves the detection performance and reduces the acquisition cost of the labeled data.

**Impacts of different parameters.** Here, we observe the impacts of different parameter settings on the prediction performance of MIS.

First, we study the impact of ensemble size $M$. Table 6 shows how F1-measure, accuracy, and AUC change with the increase of the ensemble size on the

---

### Table 4. Comparison with several supervised learning methods.

| Methods  | Data sets | F1-measure | Accuracy | AUC | Running time |
|----------|-----------|------------|----------|-----|--------------|
|          | Real      | Simulated  | Real     | Simulated  | Real      | Simulated  | Real      | Simulated  |
| S-NBayes | 0.8624    | 0.8782     | 0.9030   | 0.9124     | 0.8462    | 0.9168     | 5min4s    | 2min38s    |
| Bagging  | 0.9237    | 0.9285     | 0.9550   | 0.9563     | 0.9062    | 0.9459     | 7min19s   | 3min42s    |
| MIS      | 0.9701    | 0.9754     | 0.9938   | 0.9944     | 0.9828    | 0.9843     | 10min43s  | 5min02s    |

AUC: area under the curve.

### Table 5. Comparison with several ARP detection methods.

| Methods             | Data sets | F1-measure | Accuracy | AUC | Running time |
|---------------------|-----------|------------|----------|-----|--------------|
|                     | Real      | Simulated  | Real     | Simulated  | Real      | Simulated  | Real      | Simulated  |
| Threshold-based     | 0.8442    | 0.8574     | 0.9002   | 0.9123     | 0.8576    | 0.8425     | 8min19s   | 3min49s    |
| BSVR-ARP            | 0.9245    | 0.9261     | 0.9330   | 0.9426     | 0.9167    | 0.9185     | 9min23s   | 4min03s    |
| Entropy-based       | 0.8053    | 0.8135     | 0.8531   | 0.8672     | 0.8362    | 0.8418     | 10min10s  | 4min52s    |
| MIS                 | 0.9701    | 0.9754     | 0.9938   | 0.9944     | 0.9828    | 0.9843     | 10min43s  | 5min02s    |

AUC: area under the curve; BSVR-ARP: Bayes support vector regression-address resolution protocol.

### Table 6. The impact of ensemble size $M$.

| Data sets | $M$ | F1-measure results | Accuracy results | AUC results |
|-----------|-----|--------------------|------------------|------------|
|           |     | Real               | Simulated        | Real       | Simulated  |
|           |     | 1                  | 5                | 10         | 25         | 50         | 75         |
|           |     | 0.7560             | 0.7746           | 0.9521     | 0.9701     | 0.9629     | 0.9537     |
|           |     | 0.7571             | 0.7835           | 0.9618     | 0.9754     | 0.9753     | 0.9630     |
|           |     | 0.7122             | 0.8074           | 0.9461     | 0.9938     | 0.9624     | 0.9438     |
|           |     | 0.7552             | 0.8343           | 0.9687     | 0.9944     | 0.8442     | 0.9613     |
|           |     | 0.7013             | 0.7246           | 0.9326     | 0.9828     | 0.9422     | 0.9623     |
|           |     | 0.7235             | 0.7824           | 0.9728     | 0.9843     | 0.9481     | 0.9611     |
real data set and on the simulated data set. The overall performance shows a rising trend when the value of $M$ varies from 1 to 25, and it converges when $M = 25$ approximately. However, a too large $M$ value needs much more training time. Thus, in our article, we set the value of $M$ to 25.

Second, we study the impact of labeled sample ratio on the prediction performance of MIS. We vary the labeling ratio (i.e. the proportion of labeled samples in the training data) from 5% to 25%. Figure 7 shows that under different labeling ratios, although the performance does not necessarily improve with the increase of labeling ratio, the overall situation presents a rising trend in the beginning and remains stable after reaching a certain value. MIS maintains the high performance with F1-measure, accuracy, and AUC of more than 90% on the ARP attack prediction when the labeling ratio is greater than 11%. When the labeling ratio reaches to 15%, the three evaluation metrics are high and stable.

**Conclusion**

In this article, we propose a multi-factor integration-based semi-supervised learning method, named MIS, for ARP attack detection in SDIIoT. In MIS method, we design a multi-factor integration-based feature extraction method and propose a semi-supervised learning framework with differential priority sampling. We take into account the ARP attack features from different aspects to help the model make correct judgment. Meanwhile, the differential priority sampling enables the base learner in self-training to learn efficiently from the unlabeled samples with differences. The experimental results show that our proposed MIS method can achieve the best performance compared with fully supervised learning and other popular ARP detection methods on the real data set and the simulated data set.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the National Natural Science Foundation of China under grant 61972080, in part by the Shanghai Rising-Star Program under grant 19QA1400300.

**ORCID iD**

Zhong Li https://orcid.org/0000-0003-2304-923X

**References**

1. Li X, Li D, Wan J, et al. Adaptive transmission optimization in SDN-based industrial Internet of Things with edge computing. *IEEE Internet Things J* 2018; 5: 1351–1360.

2. Choo K-KR, Gritzalis S and Park JH. Cryptographic solutions for industrial Internet-of-Things: research challenges and opportunities. *IEEE Trans Ind Inform* 2018; 14: 3567–3569.

3. Alharbi T, Durando D, Pakzad F, et al. Securing ARP in software defined networks. In: *IEEE 41st conference on local computer networks (LCN)*, Dubai, United Arab Emirates, 7–10 November 2016, pp.523–526. New York: IEEE.

4. Kponyo J, Agyemang J and Klogo GS. Detecting End-Point (EP) Man-In-The-Middle (MITM) attack based on...
ARP analysis: a machine learning approach. *Int J Commun Netw Inform Secur* 2020; 12: 384–388.

5. Nam SY, Kim D and Kim J. Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks. *IEEE Commun Lett* 2014; 18: 187–189.

6. Bruschi D, Ornaghi A and Rosti E. S-ARP: a secure address resolution protocol. In: 19th annual computer security applications conference, Las Vegas, NV, 8–12 December 2003, pp.66–74. New York: IEEE.

7. Goyal V and Tripathy R. An efficient solution to the ARP cache poisoning problem. In: Australasian conference on information security and privacy, Brisbane, QLD, Australia, 4–6 July 2005, pp.40–51. Singapore: Springer.

8. Lootah W, Enck W and McDaniel P. TARP: Ticket-based address resolution protocol. *Comput Netw* 2007; 51: 4322–4337.

9. Jinhua G and Kejian X. ARP spoofing detection algorithm using ICMP protocol. In: *International conference on computer communication and informatics*, Coimbatore, India, 46 January 2013, pp.1–6. New York: IEEE.

10. Cox JH, Clark RJ and Owen HL. Leveraging SDN for ARP security. In: *Southeast conference*, Norfolk, VA, 30 March–3 April 2016, pp.1–8. New York: IEEE.

11. Jehan N and Haneef AM. Scalable ethernet architecture using SDN by suppressing broadcast traffic. In: *Fifth international conference on advances in computing and communications (ICACC)*, Kochi, India, 2–4 September 2015, pp.24–27. New York: IEEE.

12. Du M and Wang K. An SDN-enabled pseudo-honeypot strategy for distributed denial of service attacks in industrial Internet of Things. *IEEE Trans Ind Inform* 2019; 16: 648–657.

13. Qiu C, Yu FR, Yao H, et al. Blockchain-based software-defined industrial Internet of Things: a dualing deep Q-learning approach. *IEEE Internet Things J* 2018; 6: 4627–4639.

14. Chaudhary R, Aujla GS, Garg S, et al. SDN-enabled multi-attribute-based secure communication for smart grid in IoT environment. *IEEE Trans Ind Inform* 2018; 14: 2629–2640.

15. ORaw J, Laverty D and Morrow DJ. Securing the industrial Internet of Things for critical infrastructure (IIoT-CI). In: *IEEE 5th world forum on Internet of Things (WF-IoT)*, Limerick, Ireland, 15–18 April 2019, pp.70–75. New York: IEEE.

16. Ma H, Ding H, Yang Y, et al. Bayes-based ARP attack detection algorithm for cloud centers. *Tsinghua Sci Technol* 2016; 21: 17–28.

17. Divya C and Christopher X. Security against ARP spoofing attacks using Bayesian support vector regression. *Int J Innov Technol Explor Eng (IJITEE)* 2019; 8: 2278–3075.