Hindawi
Security and Communication Networks
Volume 2022, Article ID 2500213, 17 pages
https://doi.org/10.1155/2022/2500213

Research Article

UIV-TSP: A Blockchain-Enabled Antileakage Sharing Protection Scheme for Undisclosed IIoT Vulnerabilities

Wenbo Zhang, Jing Zhang, Yifei Shi, and Jingyu Feng

National Engineering Laboratory for Wireless Security, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

Correspondence should be addressed to Jingyu Feng; fengjy@xupt.edu.cn

Received 14 May 2022; Accepted 14 September 2022; Published 10 October 2022

Academic Editor: Andrea Michienzi

With the large-scale deployment of industrial Internet of things (IIoT) devices in 5/6G environments, the number of vulnerabilities threatening IIoT security is growing dramatically, including a mass of undisclosed IIoT vulnerabilities that lack mitigation measures. Coordination vulnerability disclosure (CVD) is one of the most popular vulnerabilities sharing solutions, in which security workers (SWs) can develop undisclosed vulnerability patches together. However, CVD assumes that SWs are all honest and thus offering chances for dishonest SWs to internally leak undisclosed IIoT vulnerabilities. To combat such internal threats, we propose an undisclosed IIoT vulnerabilities sharing protection (UIV-TSP) scheme against internal leakage. In this paper, a dynamic token is an implicit access credential for an SW to acquire an undisclosed vulnerability message, which is only held by the system and constantly updated with the SW access. The latest updated token can be stealthily sneaked into the acquired information as the traceability token to prevent internal leakage. To quickly distinguish dishonest SWs, the feedforward neural network (FNN) is adopted to evaluate the trust value of SWs. Meanwhile, we design a blockchain-assisted continuous logs storage method to achieve the tamper-proofing of dynamic token and the transparency of undisclosed IIoT vulnerabilities sharing. The simulation results indicate that our proposed scheme is resilient to suppress dishonest SWs and protect the IIoT undisclosed vulnerabilities effectively.

1. Introduction

With the gradual deployments and applications of 5/6G, the Internet of things (IoT) technology is being applied to every part of our lives [1–4]. As a subset of IoT, industrial Internet of things (IIoT) has recently attracted attention [5]. By leveraging sensors, actuators, GPS devices, and mobile devices, the IIoT technology is being applied to advance the development of many industrial systems [6]. The industrial systems where this IIoT technology is integrated include energy [7], manufacturing [8], logistics [9], and transportation [10].

Currently, IIoT devices have been widely deployed with weak security features or a lack of security [11]. These features have made IIoT devices as a good target for attackers with malicious intentions, and in many cases, exploits using IIoT devices have been occurring [12]. There is an urgent need for a solution that provides a lightweight and low-cost mechanism for collaborative security response of IIoT devices against emerging vulnerabilities [13].

However, the IIoT vendors generally have weak security emergency response capabilities. It is better to invite some security workers (such as organizations, institutions, or white hats) to help them mitigate the new vulnerabilities of IIoT devices. In order to standardize the process of vulnerabilities patching and accelerate the development of mitigation measures, the vulnerability disclosure policy has been presented in [14], including vulnerabilities reporting, sharing, coordinating, and patching. An IIoT vulnerability can be officially disclosed after the patch is made; otherwise, it is called an undisclosed IIoT vulnerability (uiv). According to the vulnerability disclosure policy, the IIoT vendors can report a new uiv and share it with some security workers (SWs) who develop their patches together by means of the coordination vulnerability disclosure (CVD).

Unfortunately, CVD is a set of guidelines without mandatory measures [14, 15]. CVD assumes that SWs are all honest and thus offering chances for dishonest SWs to internally leak undisclosed IIoT vulnerabilities. Due to the widespread use of IIoT devices, the leakage of an uiv message could cause a large-scale damage. Therefore, it is necessary to prevent the internal leakage of the uiv. Although a lot of works have been done in the field of threat intelligence sharing [16, 17], they focus on the sharing protection of disclosed vulnerability information. Little attention has been paid to the sharing protection of undisclosed vulnerability information.

In this paper, we propose an undisclosed IIoT vulnerabilities trusted sharing protection (UIV-TSP) scheme against internal leakage. To enable endogenously secure IIoT, our final objective is to prevent the leakage of uiv until their patches are released. The main contributions of this paper are as follows:

(1) Introduce dynamic token as the implicit access credential and traceability clue for an SW. When uploading a new uiv, each internal SW is assigned a corresponding token called $\text{token}_{\text{access}}$, which is only held by the system and cannot be seen by anyone. Even if an SW is granted access through identify authentication, an uiv message cannot be acquired without $\text{token}_{\text{access}}$. To avoid malicious inference, $\text{token}_{\text{access}}$ should be updated dynamically. At the end of SW access, the current $\text{token}_{\text{access}}$ is revoked. A new random number is integrated into the hash generation of token to get a new $\text{token}_{\text{access}}$ as the next access credential. Meanwhile, the current $\text{token}_{\text{access}}$ and the MAC address of SWs are hashed to create the traceability token called $\text{token}_{\text{tracing}}$, which is embedded in the undisclosed IIoT acquired by an SW.

(2) Design a blockchain-assisted method to store the continuous logs of all SWs for the UIV-TSP scheme. To ensure the tamper-proofing of dynamic token, the original token and its all-subsequent updates should be stored on the blockchain. To achieve the transparency of uiv sharing, their metadata and the related SW access records are also stored on the blockchain.

(3) Present an internal leakage prevention method with one-step traceability. A benign logic bomb called $\text{code}_{\text{preleak}}$ is embedded into an uiv message, which checks that whether the current MAC address is the same as the preset destination address in $\text{token}_{\text{tracing}}$. Due to the confidentiality, an uiv message can only be reached by one step to the SW host that is licensed by the system. Once the uiv information leaves the SW host, $\text{code}_{\text{preleak}}$ will automatically trigger the self-destruct program to prevent leaks.

(4) Adopt the trust mechanism based on deep learning to evaluate the trust value of SWs according to their historical behaviors in an automatic and dynamic manner. With high trust value, honest SWs would be accepted to acquire uiv. With low trust value, dishonest SWs would be rejected. With medium trust value, it is difficult to determine the access authorities of semihonest SWs who may be suspiciously dishonest SWs. In this case, we can release a false uiv with $\text{token}_{\text{tracing}}$ to trap their external conspirators.

The architecture of this paper is as follows. In Section 2, we introduce the related works. Our UIV-TSP scheme is proposed in Section 3. In Section 4, we analyze the performance of UIV-TSP from the perspective of security and cost. We also discuss the application of UIV-TSP solution and future work in Section 5. Finally, we conclude the paper in Section 6.

2. Related Work

2.1. Vulnerability Disclosure Policy. ISO/IEC 29147 defines vulnerability disclosure as a process through which vendors and vulnerability finders may work cooperatively in finding solutions that reduce the risks associated with a vulnerability [14]. Currently, CVD [15] is a good choice to develop undisclosed vulnerability patches together among security workers. As shown in Figure 1, the CVD process is consisted of gathering, coordinating, disclosing, and patching, and more details are given in [15]. Furthermore, there are two extreme cases for vulnerability disclosure: (1) public disclosure: disclosure as soon as the vulnerability information is received. (2) private disclosure: keep the vulnerability information security. Both of them are regarded as irresponsible sharing.

These vulnerability disclosure schemes all rely on guidelines, but in practice, guidelines are not mandatory. Once a participant breaks the guidelines, the entire vulnerability disclosure process will be paralyzed, thus increasing the threat risks from the attackers.

2.2. Threat Intelligence Sharing. To prevent the leak of sensitive data in threat intelligence containing uiv, many studies have integrated cryptography primitives into their threat intelligence sharing scheme. Vakilinia et al. [16] designed a mechanism enables the organizations to share their cybersecurity information anonymously. Meanwhile, they proposed a new blind signature based on BBS+ to reward contributions anonymously. Badsha et al. [17] proposed a privacy preserving protocol where organizations can share their private information as an encrypted form with others and they can learn the information for future prediction without disclosing any private information. de Fuentes et al. [18] introduced PRACIS, a scheme for cybersecurity information sharing that guarantees private data forwarding and aggregation by combining STIX and homomorphic encryption primitives. Homan et al. [19] leveraged the security properties of blockchain and designed a more effective and efficient framework for cybersecurity information sharing network. Preuveneers et al. [20] employed blockchain and CP-ABE to offer fine-grained

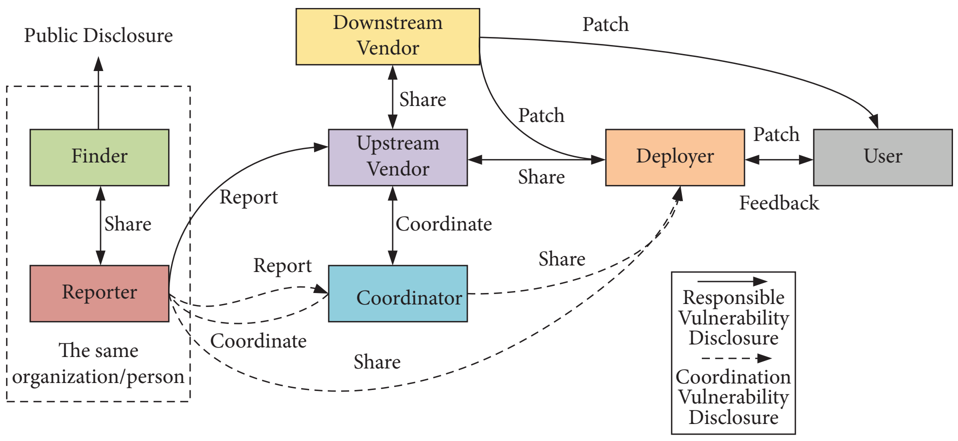

Figure 1: Flowchart of vulnerability disclosure policy [15].

protection and trustworthy threat intelligence sharing with the ability to audit the provenance of threat intelligence.

However, these schemes are helpful to protect the uiv information sharing, while the sharing protection of uiv information has not been involved. Without mitigation measures, the leakage of an undisclosed IIoT vulnerability could cause a large-scale damage [21–23]. Therefore, while protecting the sharing of uiv information, the responsibility of SWs should be traced back.

2.3. Data Leakage Prevention. Currently, some researchers focus on leveraging cryptography algorithms to implement an accountable and efficient data sharing in research of tracing data leakage. Mangipudi et al. [24] presented a committed receiver oblivious transfer (CROT) primitive to fairly track the traitor of leaked data by oblivious transfer (OT) protocol and zero knowledge (ZkPok). Huang et al. [25] designed an accountable and efficient data sharing scheme for industrial IoT (IIoT), named ADS/R-ADS/E-ADS, in which data receiver's private key (i.e., evidence) is embedded in sharing data, and data owner can pursue the responsibility of a public for profits while without permission. Zhang et al. [26] proposed a fair traitor tracing scheme to secure media sharing in the encrypted cloud media center by proxy re-encryption and fair watermarking. Ning et al. [27] presented a traitor tracing with CP-ABE scheme by two kinds of noninteractive commitments. Based on signatures, Imine et al. [28] proposed a novel accountable privacy-preserving solution for public information sharing allows to trace malicious users.

The above schemes all contribute to the sharing of threat intelligence. Nevertheless, these schemes cannot deal well with the trade-off between traceability time and robustness. A lightweight traceability scheme is required to uiv sharing.

3. Our Proposed UIV-TSP Scheme

With dynamic token, we propose an undisclosed IIoT vulnerabilities sharing protection scheme called UIV-TSP to prevent uiv leakage until their patches are released. Concretely, the UIV-TSP scheme consists of four collaborative modules: dynamic token management, blockchain-assisted continuous logs storage, internal leakage prevention with one-step traceability, and trust-based SWs distinction.

3.1. System Architecture. To prevent uiv leakage, we first present the system architecture of the UIV-TSP scheme. Figure 2 illustrates the overall system architecture of our scheme, which consists of the following entities:

(1) Trusted authority (TA): trusted authority is responsible for processing SW's access requests, generating and updating dynamic token, and evaluating trust value of SW_i.

(2) Security workers (SWs): there exist some workers who access uiv information in the sharing environment to develop their mitigation measures. We describe three types of workers: (1) honest SWs who do not engage in unauthorized access; (2) semi-honest SWs who have a chance of committing malicious behavior; (3) dishonest SWs who often leak uiv information. In this paper, SWs are defined as $\{SW_1, SW_2 \ldots SW_m\}$.

3.2. Dynamic Token Management. We introduce dynamic token as the implicit access credential and traceability clue for an SW. As shown in Figure 3, the lifecycle of dynamic token is consisted of generation and update.

3.2.1. Token Generation. When an SW submits a new undisclosed vulnerability vul_j, each internal SW is assigned a corresponding token called $\text{token}_{\text{access}}$, which is only held by the system and cannot be seen by anyone. TA generates the initial $\text{token}_{\text{access}}$ through a hash function. Meanwhile, we define vul_{meta} as the meta information of an uiv. The initial $\text{token}_{\text{access}}$ can be calculated as follows:

$$\text{token}_{\text{access}} = H\left(SW_i \| vul_{\text{meta}} \| tp \| \text{nonce}\right). \quad (1)$$

Even if an SW is granted access through identify authentication, the uiv cannot be acquired without $\text{token}_{\text{access}}$. Algorithm 1 is performed to generate and update $\text{token}_{\text{access}}$.

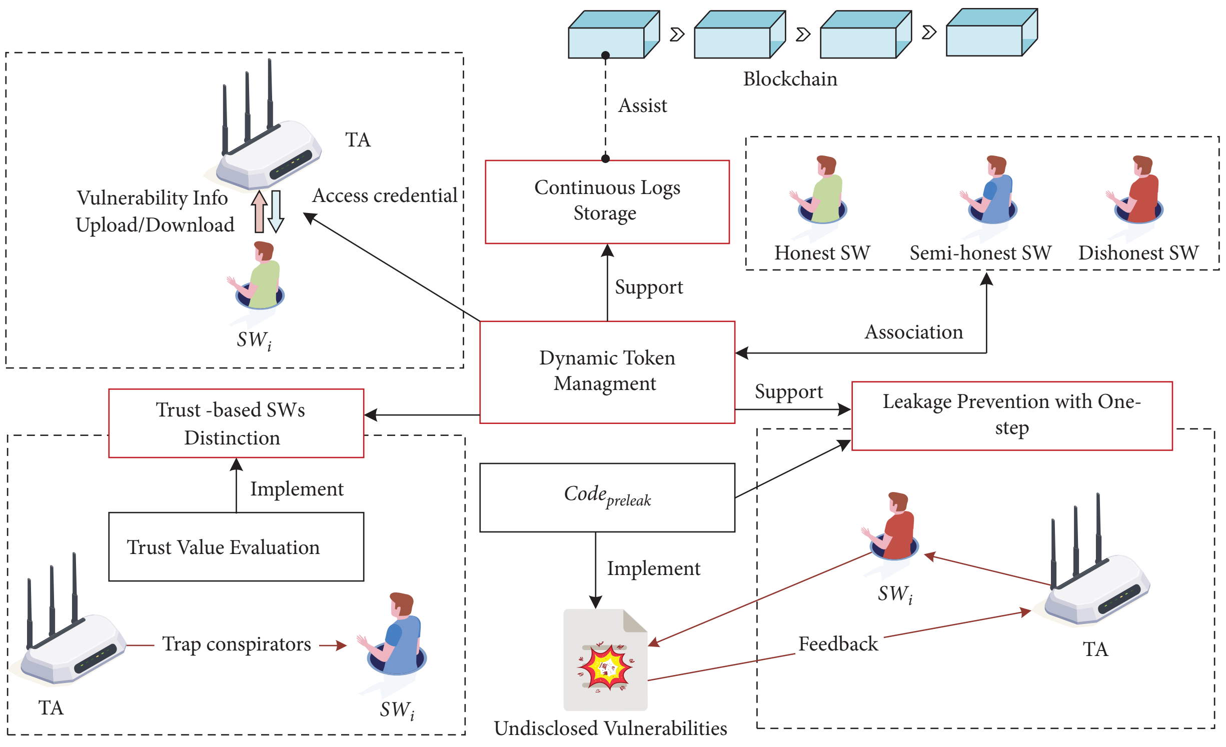

FIGURE 2: System architecture of the UIV-SP scheme.

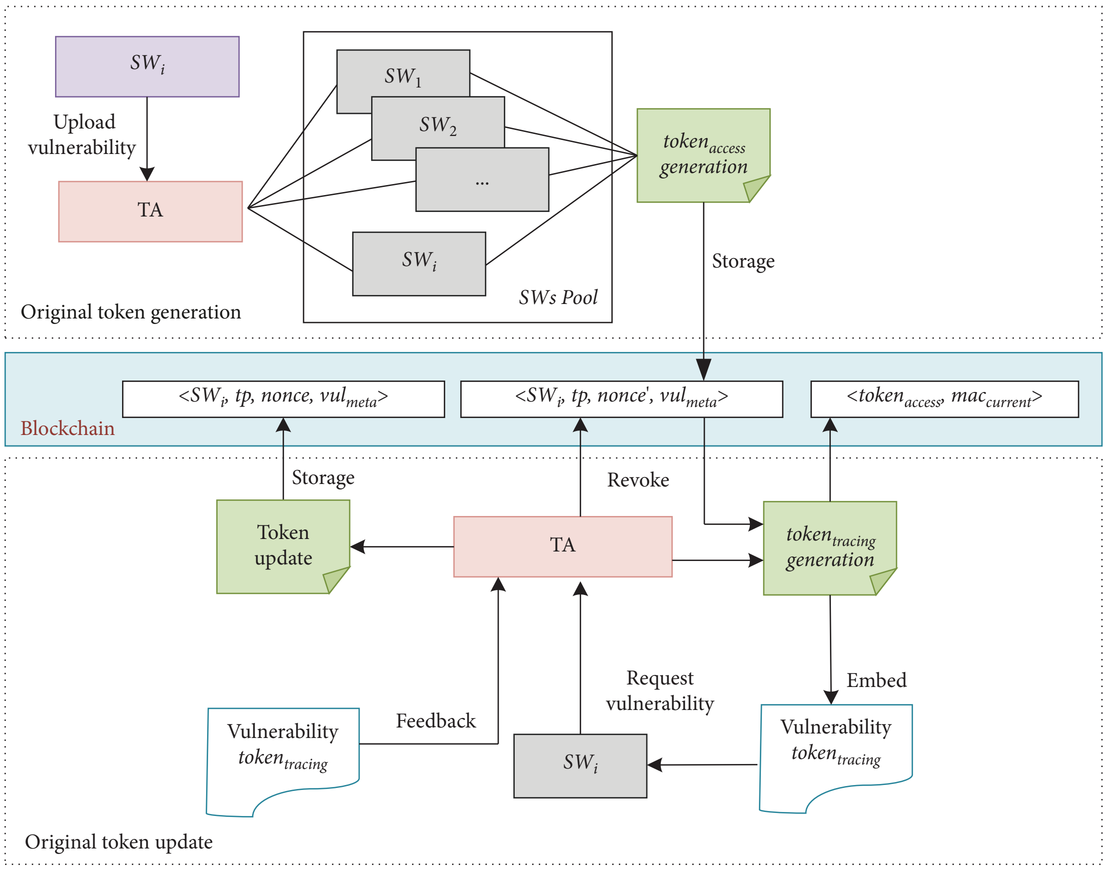
SW_i
Upload vulnerability
TA
SW_1
SW_2
...
SW_i
SWs Pool
$token_{access}$ generation
Storage
Original token generation
Blockchain
$<SW_i, tp, nonce, vul_{meta}>$
$<SW_i, tp, nonce', vul_{meta}>$
$<token_{access}, mac_{current}>$
Storage
Revoke
Token update
TA
$token_{tracing}$ generation
Embed
Request vulnerability
Feedback
Vulnerability $token_{tracing}$
SW_i
Vulnerability $token_{tracing}$
Original token update

FIGURE 3: Lifecycle of dynamic tokens.

```
(i) Input: $SW_i$
(ii) Output: $\text{token}_{\text{access}}$
(1) If $SW_i$ in $SW_{\text{pool}}$ then
(2)     $\text{token}_{\text{access}} = H(SW_i \| vul_{\text{meta}} \| tp \| \text{nonce})$.
(3)     If $\text{token}_{\text{access}}$ in blockchain then
(4)         Update $\text{token}_{\text{access}}$, $\text{token}_{\text{access}} \leftrightarrow SW_i$
(5)     else
(6)         Store $\text{token}_{\text{access}}$, $\text{token}_{\text{access}} \leftrightarrow SW_i$
(7)         If end of request access, then
(8)             Revoke $\text{token}_{\text{access}}$
(9)             $\text{tokent}_{\text{racing}} \longleftarrow H(\text{token}_{\text{access}} \| \text{mac}_{\text{current}})$
(10)            $\text{token}_{\text{access}} \longleftarrow H(SW_i \| vul_{\text{meta}} \| tp \| \text{nonce}')$
(11)                Store $\text{token}_{\text{access}}$, $SW_i \leftrightarrow \text{token}_{\text{tracing}}$
(12)                Embed $\text{token}_{\text{tracing}}$ to $vul_j$
(13)        End If
(14)    End If
(15) End If
```

Algorithm 1: Pseudocode of token generation and update.

3.2.2. Token Update. To avoid malicious inference, we integrate a one-time random number into the hash generation of token to update token dynamically. At the end of SW access, $\text{token}_{\text{access}}$ will be update as follows:

$$\text{token}_{\text{access}} \longleftarrow H\left(SW_i \| vul_{meta} \| tp \| \text{nonce}'\right). \quad (2)$$

Meanwhile, the current $\text{token}_{\text{access}}$ and the MAC address of the SW are hashed to create the traceability token called $\text{token}_{\text{tracing}}$. $\text{Token}_{\text{tracing}}$ can be defined as follows:

$$\text{token}_{\text{tracing}} \longleftarrow H(\text{token}_{\text{access}} \| \text{mac}_{\text{current}}). \quad (3)$$

In our scheme, $\text{token}_{\text{tracing}}$ can be stealthily sneaked into the acquired vul_j information as the traceability credential. The execution strategies of dynamic token management can be executed with four steps, the more details are given in [29].

3.3. Blockchain-Assisted Continuous Logs Storage. Due to the advantages including transparency, traceability, and tamper-proofing, many studies have integrated blockchain into the prior works to implement a reliable and efficient data storage [30]. In general, there are three types of blockchain data storage patterns [31]:

(1) Public blockchain: a public blockchain is the blockchain that can read by anyone in the world, anyone can send transactions to and expect to see them included, if they are valid, and anyone can participate in the consensus process, which determines what blocks get added to the chain and what the current state is.
(2) Private blockchain: a private blockchain is the blockchain where can write by only one organization.
(3) Consortium blockchain: a consortium blockchain is the blockchain where the consensus process is controlled by a preselected set of miners.

Since token can only be held by the system, the public blockchain does not meet the requirements. Hence, a private blockchain-assisted storage method is designed to centralized storage logs. To prevent attackers from tampering with data, the logs of SWs' activities in the shared process should also be continuously recorded on the blockchain. These continuous logs, including dynamic token, trust value of SW (Tr_i), and behaviors record ($R[i]$), are also only held by the system. It can be found that the private blockchain is suitable for our scheme.

In the blockchain, the block structure of continuous logs storage is shown in Figure 4.

3.3.1. Block Head. The block head is slightly different from the traditional structure. Except for previous hash, timestamp, Merkle root, and block ID, several new elements are integrated into the block head:

(i) SW_i: the ID of the i-th SW who requests the undisclosed vulnerability in the sharing environment.
(ii) Tr_i: the trust value of SW_i. In the block head, Tr_i can be quickly retrieved by TA.
(iii) vul_{meta}: the meta information of an undisclosed IIoT vulnerability.

3.3.2. Block Body. In the block body, the log data of an SW (such as SW_i) are hashed to build the Merkle tree. Except for $\text{token}_{\text{access}}$ and $\text{token}_{\text{tracing}}$, the log data of SW_i contain the following elements:

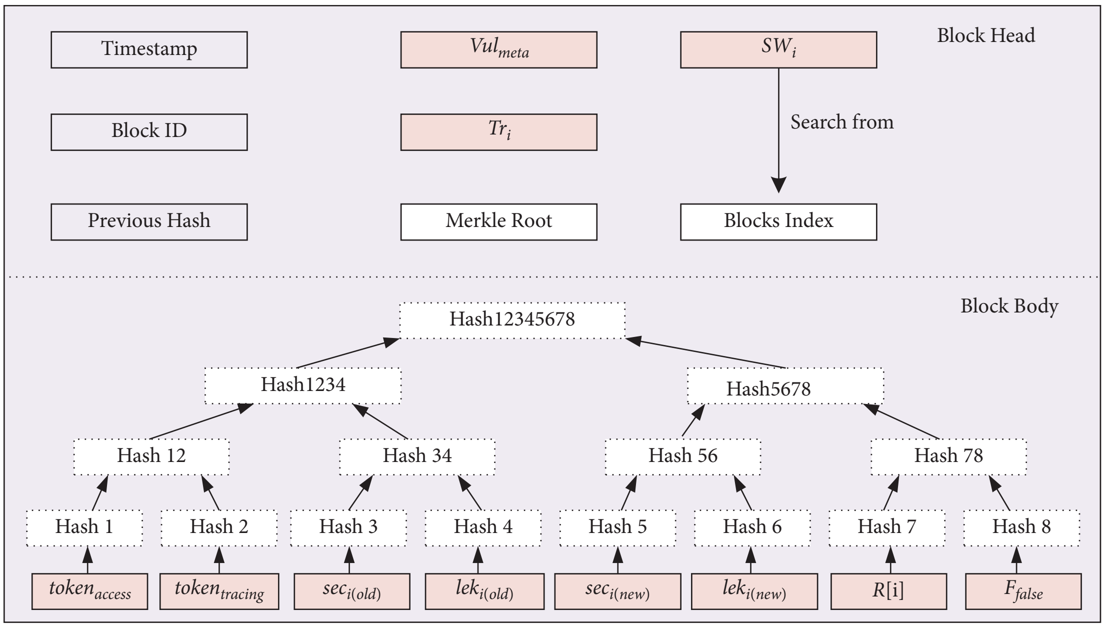

Figure 4: Block structure of continuous logs storage.

(i) $(\text{sec}_{i(old)}, \text{lek}_{i(old)})$: the historical trust data of SW_i, which can be used to evaluate the trust value of SW_i before the access.

(ii) $(\text{sec}_{i(\text{new})}, lek_{i(\text{new})})$: the current trust data of SW_i, which can be used to update the trust value of SW_i after the access.

(iii) $R[i]$: the access request record of SW_i to an *uiv* information.

(iv) F_{false}: the flag whether a false uiv information has been released.

3.4. Internal Leakage Prevention with One-Step Traceability.

3.4. Internal Leakage Prevention with One-Step Traceability. To prevent SWs leaking the acquired vul_j information, vul_j should be self-destruct when they leave the host of SWs one-step. Thus, we design a benign self-triggering logic bomb $\text{code}_{\text{preleak}}$. A logic bomb is a piece of code consisting of a trigger condition and a payload; when the trigger condition is met, the bomb is triggered (or activated) and the payload code gets executed [32].

$\text{code}_{\text{preleak}}$ is composed of trigger condition and response payloads. The trigger condition is designed to detect the access environment difference between honest SWs and dishonest SWs, so that the protection payload will be activated to destroy vul_j on the leakage side.

As shown in Figure 5, the functional structure of $\text{code}_{\text{preleak}}$ is consisted of self-checking and self-destruct.

3.4.1. Self-Check. Once the uiv information enters the SW host, $\text{code}_{\text{preleak}}$ will extract the current SW host Mac address and the revoked token to compute the verification value. Then, $\text{code}_{\text{preleak}}$ will match the verification value to $\text{token}_{\text{tracing}}$. If the result V_c is inconsistent, it will trigger self-destruct. V_c can be calculated as follows:

$$V_c \longleftarrow \text{token}_{\text{tracing}} == H(\text{token}_{\text{access}}, \text{mac}_{\text{current}})?. \quad (4)$$

Algorithm 2 is performed to match verification value.

3.4.2. Self-Destruct. If $V_c = 0$, the leakage has not happened. That is, the uiv information has not left the SW host. The protection payload continues to lurk.

If $V_c = 1$, it may leak vul_j. That is, the uiv information has left the SW host. In this case, the protection payload will be activated immediately to destroy vul_j.

At the same time as the self-destruct event, $\text{code}_{\text{preleak}}$ can automatically send encrypted feedback $ef = \{\text{token}_{\text{tracing}}, vul_j, SW_i, \text{mac}_{\text{current}}, t_{\text{feedback}}\}$ to TA.

(i) vul_j: the ID of undisclosed IIoT vulnerability.

(ii) t_{feedback}: the time to $\text{code}_{\text{preleak}}$ send the feedback message.

Algorithm 2 is performed to activate the protection payload.

3.5. Trust-Based SWs Distinction. Trust mechanism can be adopted to evaluate the trust value of SWs according to their historical behaviors. With trust value, we can quickly distinguish honest SWs and dishonest SWs. For semi-honest SWs, we can further validate their credibility by tracing whether they have external conspirators. Different SWs will gain different access authorities to acquire uiv information.

3.5.1. Trust Value Evaluation. In the process of vulnerabilities information sharing, the behaviors of an SW are generally dualistic: secret-keeping and leakage. With trust

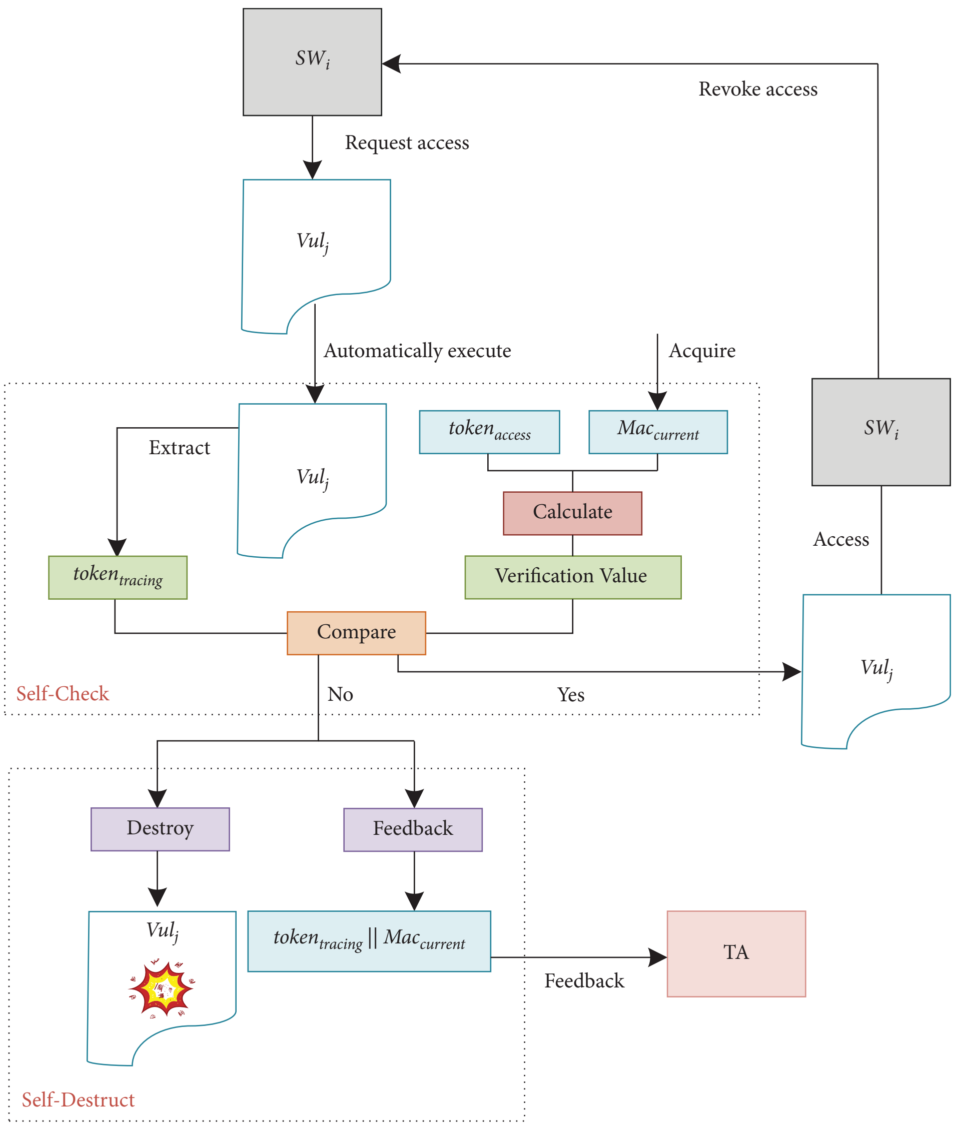

FIGURE 5: Functional structure of $\text{code}_{\text{preleak}}$.

mechanism, if an SW often leaks information, he will get a low trust value.

To quantify these duality behaviors, the beta function is one of the most popular trust value evaluation methods. It first counts the number of secret-keeping and leakage by an SW and then calculates the trust value with beta function denoted by $\text{Beta}(\alpha, \beta)$ [33].

$$\text{Beta}(\alpha, \beta) = \frac{\Gamma(\alpha, \beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1}, \quad (5)$$

where θ is the probability of duality behaviors, $0 \le \theta \le 1$, $\alpha > 0, \beta > 0$.

Take SW_i as an example, sec_i and lek_i denote the number of secret-keeping and leakage in the sharing environment, respectively. The expectation value of the beta function can be calculated as follows: $E[\text{Beta}(\alpha, \beta)] = \alpha/(\alpha + \beta)$. Considering that the trust value BT_i is limited in the interval [0, 1], BT_i can be described as follows:

$$BT_i = \frac{1 + \text{sec}_i}{1 + \text{sec}_i + \text{lek}_i}. \quad (6)$$

When $\text{sec}_i \ge 1$ and $\text{lek}_i = 0$, BT_i is always calculated as 1. The SW_i is completely trusted under this condition.

Obviously, the base trust value BT_i that decays too slowly will give dishonest SWs more opportunities to leak. So, it is very essential to introduce a penalty factor, which can be calculated as follows:

$$P_i = e^{-(\text{sec}_i + \text{lek}_i)/\text{sec}_i}. \quad (7)$$

With the punishment of P_i to BT_i, the trust value Tri of SW can be further evaluated as follows:

$$Tr_i = \begin{cases} \dfrac{1 + \text{sec}_i}{1 + \text{sec}_i + \text{lek}_i} \cdot e^{-(\text{sec}_i + \text{lek}i)/\text{sec}_i}, & \text{lek}_i < \text{sec}_i, \\ 0, & \text{lek}_i \ge \text{sec}_i. \end{cases} \quad (8)$$

In the (8), the Tri means the comprehensive trust value of SWs, which introduces a penalty factor to cause Tri to decrease faster when leakage occurs. Unfortunately, the trust mechanism utilizing the beta reputation engine is suitable for scenarios with a small amount of trust data. When the trustworthiness of SW_i is not represented by duality behaviors, as in the case of the collusion clique construction, it is possible for some malicious SWs to exhibit rational behaviors to avoid the detection. That is, they can keep high trust value in an alternant process by truly sharing their uiv information or leaking uiv information to their conspirators.

To suppress such behaviors, the special reputation engine is adopted to prevent the SW trust value growth of some rational conspirators at the uiv information sharing. Once the TA receives messages from an SW regarding some uiv information requests, the trust value of the SW will calculate by applying the feedforward neural network (FNN) algorithm [34], which is depicted in Figure 6.

The input layer of FNN consists of 11 input nodes, such as the collaborators of the target SW, the average trust value of SW collaborators, and the number of SW's conspirators. The complete input feature description is shown in Table 1. There are two hidden layers with totally 16 hidden neurons. The output layer produces the trust value of SW, so the result of SW trust evaluation depends on a set of individual trust parameters rather than duality parameters. Likewise, the output of the FNN algorithm will be stored on the blockchain.

As we know, it is essential to learn which appropriate weights and biases that make FNN have good performance. The feedforward neural network based on BP (backpropagation) algorithm constructs the identification of plant and inverse controller. Formula (9) describes how the cost in a neural network is computed.

$$\text{cost} = \sum_{j=0}^{n^{L-1}} \left(\alpha_j^L - y_j\right)^2, \tag{9}$$

where α_j^L represents the j-th activation of the last neuron, and L represents the current layer. Then, the j-th activation of the previous layer is $\alpha_j^{(L-1)}$. Assuming there are L hidden layers in total, then n^{L-1} represents the neuron in the last hidden layer before the output layer. In actual calculations, costs are always finding the differences of each neuron from the expected target output minus the current output.

By introducing the feedforward neural network algorithm, the trust value of SWs can be calculated accurately and learn the potential behavior patterns among the malicious SWs. In this case, we further design the distinction rules can successfully identify which SWs are malicious.

3.5.2. Distinction Rules. SWs can be split into honest, dishonest, and semihonest based on their trust value. $(\delta_h, \delta_l, \delta_m)$ are, respectively, set as the threshold of high, low, and medium trust value. The specific distinction rules are as follows:

R1: for $Tr_i > \delta_h$, the SW_i's access request for an uiv message will be accepted. In this situation, SW_i is classified as honest.

R2: for $Tr_i < \delta_l$, the SW_i's access request for an uiv message will be rejected. In this situation, SW_i is classified as dishonest.

R3: for $\delta_l < Tr_i < \delta_m$, it is difficult to determine the access authorities of SW_i. In this situation, SW_i is classified as semihonest who may be suspiciously dishonest SWs.

To validate the credibility of semihonest SWs, we can trace whether they have conspirators. Let $\mu_i(\cdot)$ denote number of conspirators of SW_i.

If $\mu_i = 0$, there are no conspirators. Hence, SW_i is temporarily considered as honest.

If $\mu_i \geq 1$, SW_i may have several external conspirators. The trust value of SW_i will be set to 0. As a result, SW_i will be removed from the CVD and barred from rejoining.

3.5.3. Trap External Conspirators. If SW_i is a semihonest SW, a false uiv message can be released to trap his external conspirators. This false uiv information is set to a valid time.

Within the valid time, $\text{code}_{\text{preleak}}$ will not trigger the protection payload and provide the feedback messages, which can be employed to build a set of leak path. Once an external conspirator is trapped, SW_i can be regarded as dishonest.

After the valid time, the protection payload is activated to destroy the false uiv message, so as not to spread too widely. Algorithm 3 is performed to trap external conspirators.

4. Performance Analysis

In this section, we conduct performance analysis on the UIV-TSP scheme. We analyze the security of the proposed scheme and then perform computer simulation to further analyze the cost of the proposed scheme.

4.1. Security Analysis. In this section, we analyze how UIV-TSP can achieve the following security requirements: uiv sharing against leakage, blockchain storage against continuous logs tampering. To analyze the security better, we also compare UIV-TSP with some typical data leakage prevention (DLP) schemes in Table 2.

4.1.1. UIV Sharing against Internal Leakage. Challenge 1: Dishonest SW may disclose uiv to cause a large-scale damage.

Lemma 1. *UIV-TSP is resistant internal leakage for uiv.*

Proof. In the UIV-TSP scheme, a benign logic bomb called $\text{code}_{\text{preleak}}$ is embedded into the uiv message, which checks that whether the current MAC address is the same as the present destination address in $\text{token}_{\text{tracing}}$. The MAC address of the device is always fixed, it is impossible to be modified. Furthermore, the $\text{token}_{\text{tracing}}$ is calculated by the hash operation $h(\cdot)$, and $\text{token}_{\text{access}}$ that is dynamically updated at the end of each uiv access. The dual dynamics ensures that

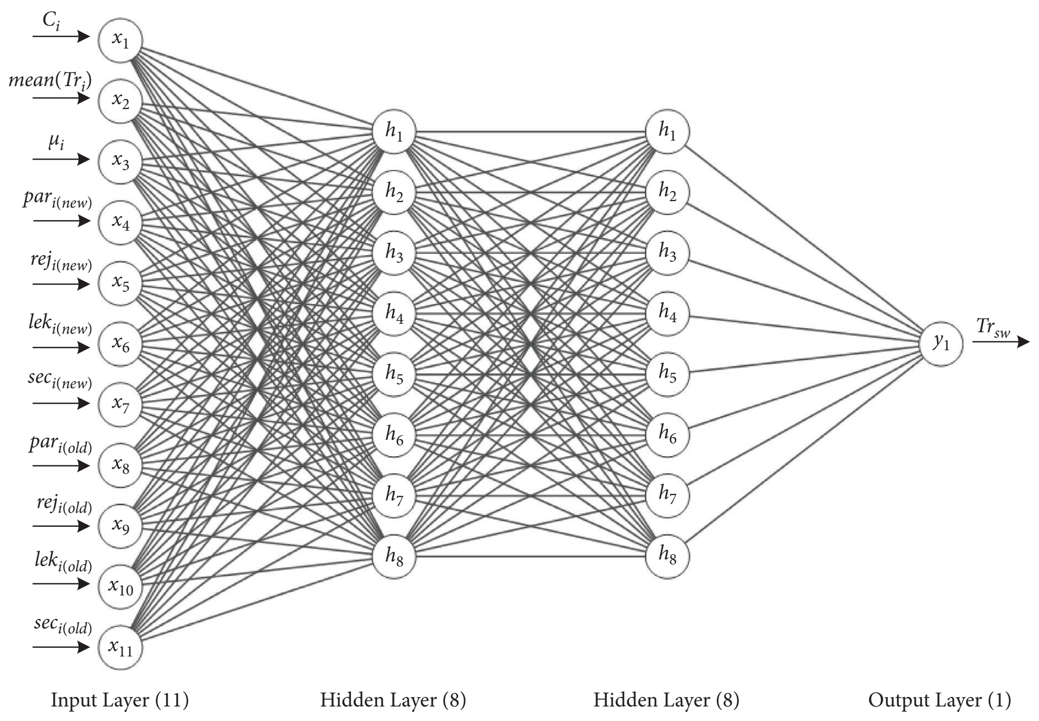

Figure 6: Structure of the feedforward neural network.

```
(i) Input: uiv
(ii) Output: access/deny
(1) $token_{tracing}$ = extract($vul_j$)
(2) **If** $token_{tracing}$ **not in** $vul_j$
(3)     Go to step 15.
(4) **Else**
(5)     Acquire $MAC_{current}$
(6)     $V_c \longleftarrow token_{tracing} == H(token_{access}, mac_{current})$?
(7)     **If** $V_c = 1$ **then**
(8)         Assign access authorities
(9)     **If** Extract($vul_j$) == $F_{false}$ **then**
(10)        Go to step 8.
(11)    **Else**
(12)        Send an encrypted feedback message $e_f$ to TA.
(13)        Destroy the target $vul_j$
(14)    End if
(15) End if
```

Algorithm 2: Pseudocode implementation of Match verification value and self-destruct.

$\text{token}_{\text{tracing}}$ cannot be inferred. According to $V_c \longleftarrow \text{token}_{\text{tracing}} == H(\text{token}_{\text{access}}, \text{mac}_{\text{current}})$?, $V_c = 1$ makes the protection payload be activated immediately to destroy vul_j, so it is believed that our UIV-TSP scheme can resist uiv leakage.

4.1.2. Blockchain Storage against Continuous Logs Tampering. Challenge 2: the trusted sharing environment with leakage-resilience construction relies on the trust value evaluation. Trust mechanisms evaluate the trust value of SWs according to their historical behaviors. Attackers can tamper with their historical logs to disturb trust mechanism and promote their trust quickly. Therefore, it is impossible to distinguish honest SWs.

Lemma 2. *UIV-TSP is resistant to continuous logs tampering.*

Proof. In our UIV-TSP scheme, a blockchain-assisted method to store the continuous logs of all SWs for the

Table 1: The input features of the FNN algorithm.

Input features	Description of the input features
Number of the target SW collaborators	A collection of collaborators for the target SW, including collaborator identities. An integer representing the number of SW's collaborators $c_i \in [0, 2000)$
Average trust value of SW collaborators	A real number between 0 and 1, where 0 and 1 represent the honest collaborator and dishonest collaborator, respectively.
Number of SW's conspirators	An integer representing the number of SW's conspirators $\mu_i \in [0, 2000)$.
Current number of SW participating behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $\text{par}_{i\text{new}} \in [0, 100)$
Current number of SW rejected behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $\text{rej}_{i(\text{new})} \in [0, 100)$
Current number of SW leak behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $\text{lek}_{i(\text{new})} \in [0, 50)$
Current number of SW keep-secret behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $\text{sec}_{i(\text{new})} \in [0, 50)$
Number of SW historical participating behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $\text{par}_{i(\text{old})} \in [0, 100)$
Number of SW historical rejected behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $\text{rej}_{i(\text{old})} \in [0, 100)$
Number of SW historical leak behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $\text{lek}_{i(\text{old})} \in [0, 50)$
Number of SW historical keep-secret behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $\text{sec}_{i(\text{old})} \in [0, 50)$

```
(i) Input: $SW_i$
(ii) Output: $\text{path}_i$
(1) Init trust value of SW, $\text{path}_i$, $\mu_i$ number of conspirators of SW
(2) $Tr_i$ evaluate based on the feedback $e_f$, retrieves SW historical trust values $Tr_i$
(3) If $\delta_l < Tr_i < \delta_m$ then
(4)     Release false $vul_j$ to SW to induce unauthorized behavior, $F_{\text{false}}$ = true
(5)     $SW_i \leftrightarrow \text{token}_{\text{tracing}}$ Extract the $\text{token}_{\text{tracing}}$ corresponding to the malicious SW
(6)     $\mu_i$ + + increased number of SW conspirators
(7)     If valid time expire then
(8)         Invoke Self- Destruct
(9)     Else:
(10)        $\text{mac}_{\text{current}}$ not in $path_i$
(11)        $\text{path}_i$ add ($\text{mac}_{\text{current}}$)
(12)    End if
(13) Else
(14)        Release normal $\text{vul}_j$ to SW.
(15) End if
```

Algorithm 3: Pseudocode implementation of trap external conspirators.

trusted sharing environment with leakage-resilience construction. The original token and its all-subsequent updates should be stored on the blockchain, which is a sequence of blocks, which holds a complete list of transaction records like conventional public ledger. Each block points to the immediately previous block via a reference that is essentially a hash value of the previous block called parent block [35]. Just like a linked list, each block depends on its previous block. Therefore, if the continuous logs maintained in a block are tampered, its latter blocks chained on the blockchain must be modified. Obviously, the longer the blockchain created, the higher cost it takes to tamper with a block. Assuming the number of latter blocks related on Block p is l_p under the condition that there are a number of sharing regions, the number of blocks that need to be tampered (t_p) is calculated as follows:

$$t_p = p_m + (p_m)^2 + \ldots + (p_m)^{l_p} = \frac{(p_m)^{l_p+1} - p_m}{p_m - 1}. \quad (10)$$

For instance, $t_p = 1,398,100$ when $n_r = 4$ and $l_p = 10$.

Therefore, it is nearly impossible to tamper with the continuous logs maintained in Block p due to the huge resource consumption.

4.2. Experimental Analysis. We perform simulations to validate the performance of the UIV-TSP scheme in Python 3.7.6, combined with NumPy, pandas, matplotlib, keras,

Table 2: Comparison of data leak prevention scheme.

Scheme	Data leakage preventing	Transmission confidentiality	Continuous logs tampering
UIV-TSP	Yes	Yes	Yes
CROT	No	No	No
ADS	No	Yes	No
R-ADS	No	Yes	No
E-ADS	No	Yes	No

Table 3: Description of simulation elements.

Parameters	Description	Default
SW_i	Number of SWs	2000
per	Percentage of dishonest SWs	10%–50%
δ	Threshold of trust value	(0.2, 0.5, 0.8)
Cycle	Number of cycle simulation	200
ε	Embedded times of access credential	(1, 2, 3, 4)
k	The length of access credential	(256, 512, 1024)

TensorFlow, and other python libraries. The default simulation elements are shown in Table 3.

The simulations are executed in cycle-based manner. At each cycle, a certain percentage of SWs are randomly selected as dishonest SWs. The behavioral pattern of honest SW is modeled to always keep secret, while dishonest SWs may leak an uiv message sometimes. Without punishment, dishonest SWs will hinder the establishment of a trusted vulnerability message sharing environment. In this section, the performance and overload of the proposed UIV-TSP scheme are experimentally evaluated under various settings.

The simulation uses 2,000 SWs to generate 399,284 access record samples for training. Meanwhile, all the data generated are outputted to a.csv file. The proportion of benign access record samples to malicious access record samples in the training set is almost 1 : 1. Then, the data used are split to 70% training, and 30% validation.

After dividing the dataset, this paper uses the Adam optimization algorithm to obtain the optimal parameter combination, such as learning rate, epochs, and batch size. Then, the parameters obtained are used for feedforward neural network training, and the performance is evaluated by cross validation. Figure 7 shows the variation of model scores under different learning rates. The x-coordinate represents the learning rate, while the y-coordinate represents the model score. The shaded graph indicates the fluctuation range of the model score for this training. It can be seen that the training score and testing score reach the optimal value when the learning rate is 0.1, epoch = 10, and batch_size = 16.

In order to evaluate the performance of the trust mechanism based on FNN algorithm, we analyze four evaluation metrics, namely, precision (P), recall (R), accuracy (A), and F1-score ($F1$). In this paper, the precision and recall are defined as follows:

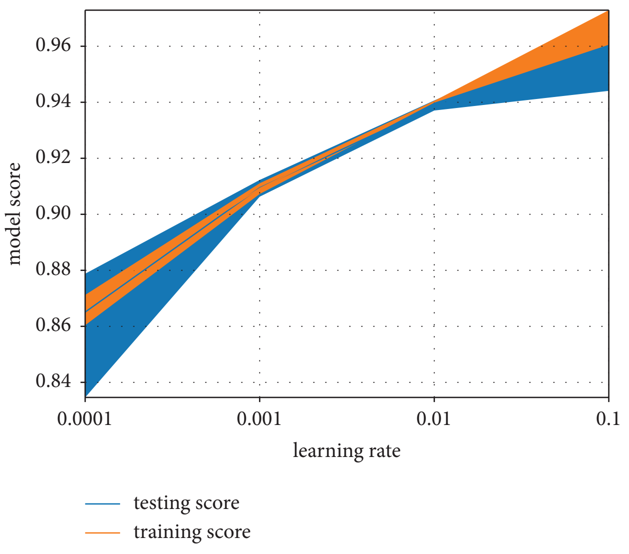

Figure 7: FNN learning rate score.

$$\begin{aligned} A &= \frac{\text{number of truly SWs detected}}{\text{total number of SW}s} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}, \\ R &= \frac{\text{number of truly malicious SWs detected}}{\text{total number of truly malicious SW}s} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \\ P &= \frac{\text{number of truly malicious SWs detected}}{\text{total number of malicious SW}s} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \\ F1 &= \frac{2*P*R}{P+R}. \end{aligned} \tag{11}$$

We compare the performance of feedforward neural network (FNN) for identifying malicious SWs with two other well-known deep learning models, namely, recurrent

Table 4: Comparison of the performance of different algorithms.

		Accuracy	Recall	Precision	F1-score	Time (s)
FNN	Train	0.9751	0.9518	0.9546	0.9532	228
	Test	0.9736	0.9050	0.9142	0.9096	
CNN	Train	0.7428	0.8370	0.6486	0.7306	337
	Test	0.7422	0.8366	0.6481	0.7305	
RNN	Train	0.8804	0.8455	0.8447	0.8452	410
	Test	0.8780	0.8454	0.8447	0.8449	

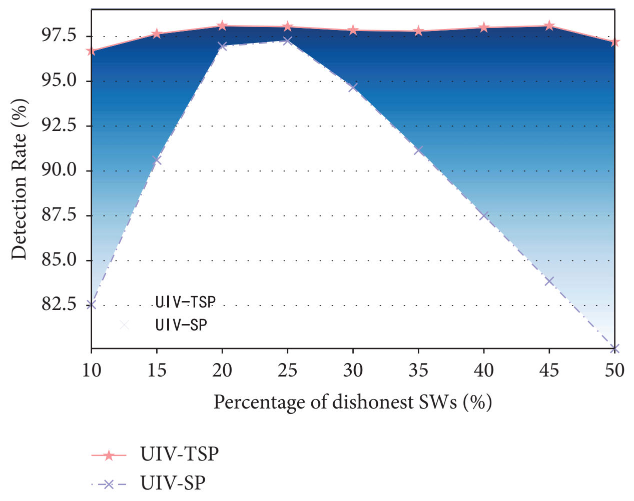

Figure 8: Detection rate under the percentage of dishonest SWs.

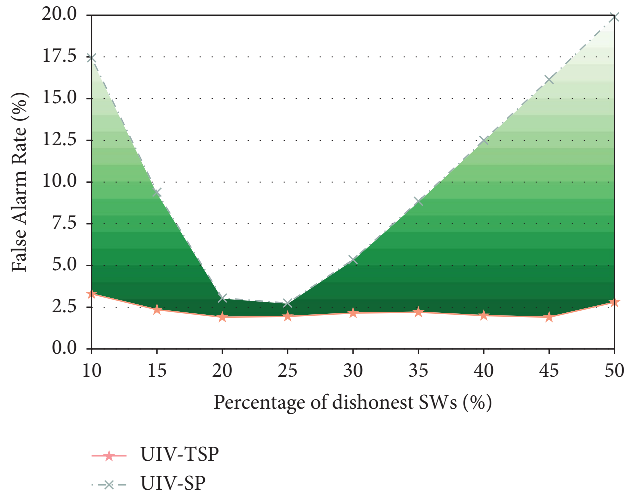

Figure 9: Alarm rate under the percentage of dishonest SWs.

neural network (RNN) and convolutional neural network (CNN). Table 4 shows that the FNN performs best in terms of the accuracy, recall, precision, and $F1$-score. In addition, FNN clearly lower than the other algorithms in terms of the training time costs. That is because the FNN algorithm has obvious advantages in network structure, and owns simpler node connections than RNN. Due to the convolution operation, CNN will require more computation. Therefore, we conclude that FNN is a suitable deep learning algorithm for SW trust evaluation.

Then, we analyze the detection and false alarm rate of our UIV-TSP scheme (i.e., probability of successful detection leaker) in Figures 8 and 9. To analyze the effectiveness better, we compare UIV-TSP with the undisclosed IIoT vulnerabilities sharing protection (UIV-SP) scheme without trust mechanism.

In this simulation, the detection rate of UIV-TSP is better than UIV-SP in Figure 8, while the false alarm rate of UIV-TSP is lower than UIV-SP in Figure 9. Therefore, our designed trust mechanism can improve the performance of

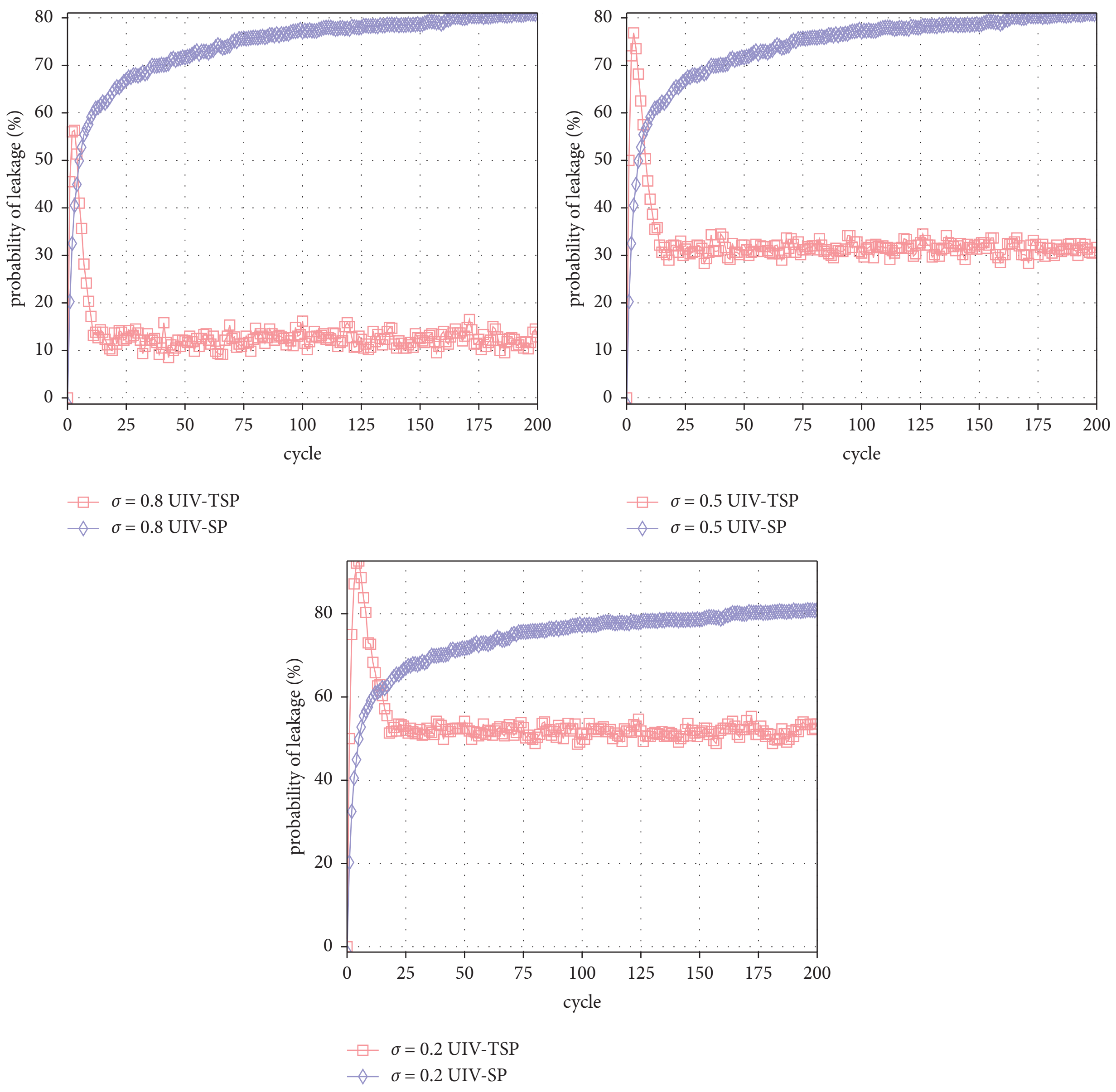

Figure 10: Suppressing leakage behaviors.

UIV-SP distinctly in the construction of trusted sharing environment.

Then, we validate the performance of our UIV-TSP scheme against dishonest SWs, in terms of suppressing leakage behaviors. Dishonest SWs will leak uiv in the sharing environment. Consequently, some leakage behaviors may generate in each cycle, which may cause unnecessary waste of network resources. So, the key performance indicator of UIV-TSP is to suppress these leakage behaviors. As shown in Figure 10, it is obvious that UIV-TSP is better than UIV-SP in suppressing leakage behaviors. This means that the trust mechanism plays a key role in the detection of dishonest SWs. In this simulation, δ is set as (0.2, 0.5, 0.8), respectively.

We also analyze the uiv leakage prevention performance of UIV-TSP in terms of leakage probability. In this simulation, the percentage of dishonest SWs is set as 30%, respectively. As shown in Figure 11, the uiv leakage probability of UIV-TSP is also lower than UIV-SP.

Finally, we evaluate the traceability performance of UIV-TSP in terms of computational complexity. We count the number of time-consuming operations such as the symmetric-key encryption/decryption (SKE), public-key encryption/decryption (PKE), cryptographic hash function (HASH), and exponential operation (EXP) in G_q multiplicative operation (MUL) in G_q. As shown in Table 5, we compare UIV-TSP with three types of traditional schemes.

It can be found that UIV-TSP cannot require any exponential operation or public-key encryption/decryption. Moreover, the requirements for hash and symmetric key operations are limited in UIV-TSP.

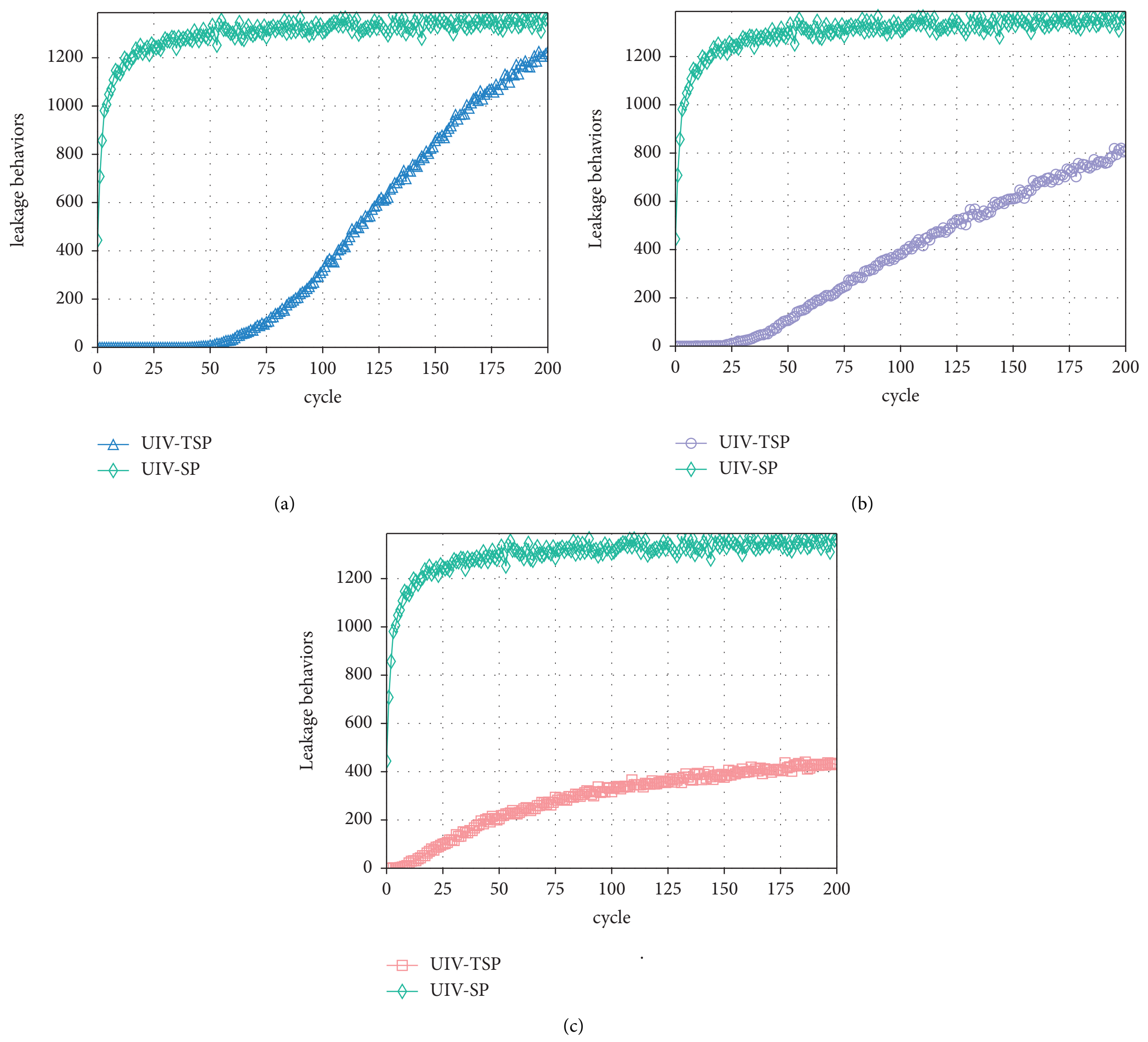

Figure 11: Probability of leakage at different δ. (a) $\delta = 0.8$. (b) $\delta = 0.5$. (c) $\delta = 0.2$.

Table 5: Computational complexity.

	SKE	HASH	PKE	EXP	MUL
CROT	$4t$	$8t$	$3t$	$6t+3$	$2t+1$
ADS	$2t$	$2t$	N/A	N/A	N/A
R-ADS	$2t$	$2t$	N/A	N/A	$4kt+2t$
Ours work	N/A	$3t$	N/A	N/A	N/A

To further evaluate the traceability performance of UIV-TSP in terms of computational complexity, we can observe the traceability delay of UIV-TSP and these traditional schemes.

We run 200 rounds of experiments and obtain their average traceability delay as the result. We define k as the length of access credential. In our UIV-TSP scheme, the access credential is a dynamic token. In the traditional schemes, the access credential is the private key of a user. A sufficient length of k can contribute to the collision resistance generated by hashing. As the length of k increases, the user capacity will be improved. Of course, with the increase of k and the embedded times of access credential, the traceability delay grows as well. As shown in Figure 12, UIV-TSP is more computationally efficient than ADS and CROT. The reason is that the sharing data in UIV-TSP need not to

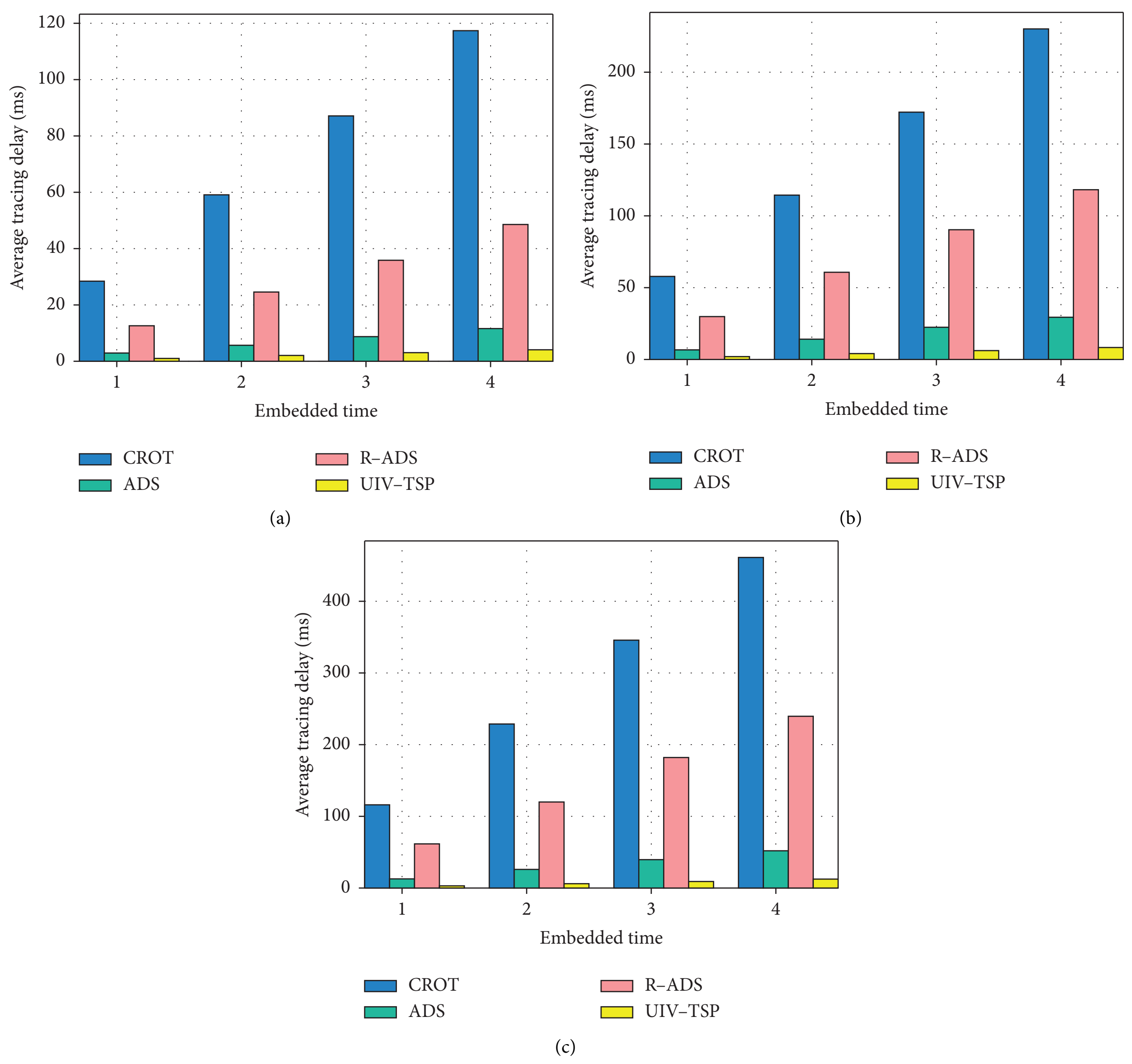

FIGURE 12: Average tracing delay. (a) $k = 256$. (b) $k = 512$. (c) $k = 1024$.

perform oblivious transfer (OT) and zero-knowledge proof. Once the embedded times vary from 1 to 4, the number of OT increases linearly.

In summary, our UIV-TSP scheme can prevent the uiv information leakage effectively, which merely requires limited traceability delay caused by multiple shared SWs.

5. Industrial Applications Discussion

Since the IIoT vendors generally have weak security emergency response capabilities, some SWs can be invited to help them path a new uiv by means of CVD. Our UIV-TSP scheme can prevent the uiv leakage effectively in CVD and trace the dishonest SWs with limited traceability delay. As shown in Figure 13, UIV-TSP can be applied to several IIoT scenarios, such as energy, logistics, manufacturing, and transportation.

Take the IIoT manufacturing as an example. Once an IIoT manufacturing vendor reports a new uiv in CVD, he can select several SWs. Then, TA will assign $\text{token}_{\text{access}}$ to each of them and store $\text{token}_{\text{access}}$ on the blockchain. Without the implicit access credential, the unselected SWs cannot acquire uiv. With the implicit access credential, a selected SW can only acquire uiv on the basis of his high trust value. In this way, the uiv sharing can be restricted within the scope of permission, and the leakage problem of dishonest SWs can be avoided in advance.

In our UIV-TSP scheme, the metadata of uiv and the related SW access records are also stored on the blockchain, which can make the access logs of the uiv sharing as tamper-resistant. After the access, $\text{token}_{\text{tracing}}$ and $\text{code}_{\text{preleak}}$ can be stealthily sneaked into the acquired uiv information. Once the uiv information is one step away from the selected SW host, $\text{code}_{\text{preleak}}$ will destroy the uiv information and sends

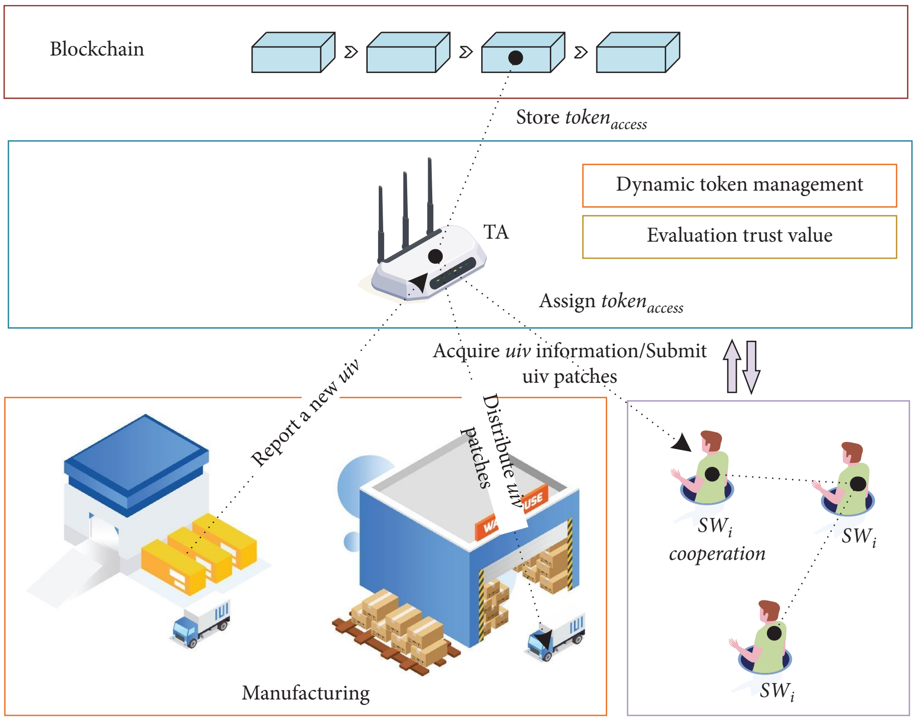

Figure 13: Industrial application case of UIV-TSP.

back feedback containing the token, thus avoiding a widespread damage to the users of the IIoT manufacturing devices after the uiv leakage. When the mitigation measures are developed, the uiv patch will be accurately distributed to the target IIoT manufacturing devices. Under the circumstances, uiv can be made public.

6. Conclusion and Future Works

In this paper, we propose an undisclosed IIoT vulnerabilities trusted sharing protection scheme against internal leakage. To facilitate the detection of leakage behaviors, we design a dynamic token as the implicit access credential and traceability clue. Assisted by blockchain, the continuous access logs of SWs can be securely stored. To prevent the leakage of a vulnerability, we present a benign logic bomb called $\text{code}_{\text{preleak}}$, which is embedded into the undisclosed IIoT vulnerability information. A trust management system based on deep learning is adopted to evaluate the trust value of SWs, which can quickly distinguish SWs. Simulation results indicate that our proposed scheme is resilient to suppress dishonest SWs, and merely require limited traceability delay.

For future works, we will investigate on the selfish SWs and motivate them to develop the mitigation measures of undisclosed IIoT vulnerabilities under the protection of UIV-TSP.

Data Availability

The data required for simulation are generated through experiments.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The study was supported by the National Natural Science Foundation of China (No. 61802302) and the Natural Science Basic Research Program of Shaanxi (No. 2019JM-442).

References

[1] J. Gui, L. Hui, N. N. Xiong, and J. Wu, “Improving spectrum efficiency of cell-edge devices by incentive architecture applications with dynamic charging,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 795–808, 2021.

[2] D. Cecchinato, T. Erseghe, and M. Rossi, “Elastic and predictive allocation of computing tasks in energy harvesting IoT edge networks,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1772–1788, 2021.

[3] F. Shamieh, X. Wang, and A. R. Hussein, “Transaction throughput provisioning technique for blockchain-based industrial IoT networks,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3122–3134, 2020.

[4] W. Lu, S. Hu, X. Liu, C. He, and Y. Gong, “Incentive mechanism based cooperative spectrum sharing for OFDM cognitive IoT network,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 662–672, 2020.

[5] R. Zhou, X. Wang, J. Wan, and N. Xiong, “EDM-fuzzy: an euclidean distance based multiscale fuzzy entropy technology for diagnosing faults of industrial systems,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4046–4054, 2021.

[6] L. D. Xu, W. He, and S. Li, “Internet of Things in industries: a survey,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov 2014.

[7] Z. Zhou, C. Zhang, C. Xu, F. Xiong, Y. Zhang, and T. Umer, “Energy-efficient industrial Internet of UAVs for power line inspection in smart grid,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2705–2714, 2018.

[8] C. Zhang, G. Zhou, H. Li, and Y. Cao, “Manufacturing blockchain of Things for the configuration of a data- and knowledge-driven digital twin manufacturing cell,” *IEEE Internet of Things Journal*, vol. 7, no. 12, pp. 11884–11894, 2020.

[9] Y. Zhang, Z. Guo, J. Lv, and Y. Liu, “A framework for smart production-logistics systems based on CPS and industrial IoT,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4019–4032, 2018.

[10] J. Feng, Y. Wang, J. Wang, and F. Ren, “Blockchain-based data management and edge-assisted trusted cloaking area construction for location privacy protection in vehicular networks,” *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2087–2101, 2021.

[11] J. M. Mcginthy and A. J. Michaels, “Secure industrial Internet of Things critical infrastructure node design,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8021–8037, Oct 2019.

[12] Y. Shah and S. Sengupta, “A survey on classification of cyber-attacks on IoT and IIoT devices,” in *Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0406–0413, New York, NY, USA, October 2020.

[13] W. Zhao and G. White, “A Collaborative Information Sharing Framework for Community Cyber Security,” in *Proceedings of the 2012 IEEE Conference on Technologies for Homeland Security (HST)*, pp. 457–462, Waltham, MA, USA, November 2012.

[14] “ISO/IEC 29147:2018 Information technology -Security techniques- Vulnerability disclosure,” 2018, https://www.iso.org/standard/72311.html.

[15] D. Allen, “The CERT guide to coordinated vulnerability disclosure,” 2019, https://vuls.cert.org/conf luence/display/CVD/The+CERT+Guide+to+Coordinated+Vulnerability+Disclosure.2019-12-12.

[16] I. Vakilinia, D. K. Tosh, and S. Sengupta, “Privacy-preserving cybersecurity information exchange mechanism,” in *Proceedings of the 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pp. 1–7, Seattle, WA, USA, July 2017.

[17] S. Badsha, I. Vakilinia, and S. Sengupta, “Privacy preserving cyber threat information sharing and learning for cyber defense,” in *Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0708–0714, Las Vegas, NV, USA, January 2019.

[18] J. M. de Fuentes, L. Gonzalez-Manzano, J. Tapiador, and P. Peris-Lopez, “PRACIS: privacy-preserving and aggregatable cybersecurity information sharing,” *Computers & Security*, vol. 69, pp. 127–141, 2017.

[19] D. Homan, I. Shiel, and C. Thorpe, “A new network model for cyber threat intelligence sharing using blockchain technology,” in *Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–6, Canary Islands, Spain, June 2019.

[20] D. Preuveneers, W. Joosen, J. Bernal Bernabe, and A. Skarmeta, “Distributed Security Framework for Reliable Threat Intelligence Sharing,” *Security and Communication Networks*, vol. 2020, pp. 1–15, Article ID 8833765, 2020.

[21] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, “Machine learning-based network vulnerability analysis of industrial Internet of Things,” *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, Aug 2019.

[22] V. Sharma, G. Choudhary, Y. Ko, and I. You, “Behavior and vulnerability assessment of drones-enabled industrial Internet of Things (IIoT),” *IEEE Access*, vol. 6, pp. 43368–43383, 2018.

[23] F. Xiao, L.-T. Sha, Z.-P. Yuan, and R.-C. Wang, “VulHunter: a discovery for unknown bugs based on analysis for known patches in industry Internet of Things,” *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 267–279, 2020.

[24] E. V. Mangipudi, K. Rao, J. Clark, and A. Kate, “Towards automatically penalizing multimedia breaches (extended abstract),” in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 340–346, Stockholm, Sweden, June 2019.

[25] C. Huang, D. Liu, J. Ni, R. Lu, and X. Shen, “Achieving accountable and efficient data sharing in industrial Internet of Things,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1416–1427, 2021.

[26] L. Y. Zhang, Y. Zheng, J. Weng, C. Wang, Z. Shan, and K. Ren, “You can access but you cannot leak: defending against illegal content redistribution in encrypted cloud media center,” *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1218–1231, 2020.

[27] J. Ning, Z. Cao, X. Dong, and L. Wei, “White-box traceable CP-abe for cloud storage service: how to catch people leaking their access credentials effectively,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 883–897, 2018.

[28] Y. Imine, A. Lounis, and A. Bouabdallah, “An accountable privacy-preserving scheme for public information sharing systems,” *Computers & Security*, vol. 93, Article ID 101786, 2020.

[29] W. Zhang, J. Zhang, Y. Shi, and J. Feng, “Blockchain-assisted undisclosed IIoT vulnerabilities trusted sharing protection with dynamic token,” 2021, https://arxiv.org/abs/2103.08908.

[30] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, “Blockchain and federated learning for privacy-preserved data sharing in industrial iot,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[31] V. Buterin, “On Public and Private Blockchains,” 2015, https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains.

[32] Q. Zeng, L. Luo, Z. Qian et al., “Resilient user-side android application repackaging and tampering detection using cryptographically obfuscated logic bombs,” *IEEE Transactions on Dependable and Secure Computing*, vol. 18, pp. 2582–2600, 2021.

[33] A. Jϕsang and R. Ismail, “The beta reputation system,” *Proc. The 15th Bled Electronic Commence Conference*, vol. 5, pp. 2502–2511, 2002.

[34] C. Zhang, W. Li, Y. Luo, and Y. Hu, “AIT: an AI-enabled trust management system for vehicular networks using blockchain technology,” *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3157–3169, 2021.

[35] H. Wang, Z. Zheng, S. Xie, H. N. Dai, and X. Chen, “Blockchain challenges and opportunities: a survey,” *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.