electronics

MDPI

Article

HyperLedger Fabric-Based Proactive Defense against Inside Attackers in the WSN With Trust Mechanism

Kyeongsun Cho and Youngho Cho *

Department of Defense Science (Computer Engineering), Graduate School of Defense Management, Korean National Defense University, Nonsan 33021, Korea; cksun89@gmail.com

* Correspondence: youngho@kndu.ac.kr

Received: 18 September 2020; Accepted: 9 October 2020; Published: 12 October 2020

Abstract: In Wireless Sensor Networks (WSNs), the Trust Mechanism (TM) is used to defend against insider attacks by measuring the trustworthiness of all inside sensor nodes in the network. Thus, each sensor node with TM observes its neighbor nodes' behaviors, evaluates their trustworthiness as numeric trust values, and captures untrustworthy nodes as inside attackers. Although the defense performance of trust mechanisms can be further improved by sharing the information about inside attackers detected by TM with all sensor nodes, the detected inside attacker list must be securely shared with and stored in all sensor nodes in the WSN. However, according to our survey, we observed that most existing studies simply assume that the communication channel for sharing the attacker detection list is reliable and trusted even in the presence of inside attackers in the WSN. In this paper, we propose and implement a proactive defense model that integrates the HyperLedger Fabric and trust mechanism to defend against inside attackers by securely sharing the detected inside attacker list with all sensor nodes in the WSN. In addition, we conduct comparative experiments to show that our proposed model can better defend against inside attackers than an existing trust mechanism. According to our experimental results, our proposed model could lower the attack damage (the number of packet drops) caused by an inside packet drop attacker by 59 to 67% compared to an existing trust mechanism.

Keywords: wireless sensor network; trust mechanism; blockchain; HyperLedger Fabric; insider attack defense

1. Introduction

Wireless Sensor Networks (WSNs) have been widely used in various areas such as fire detection, battlefield surveillance, smart grid, and so on [1–5]. With the rapid advancement of the Internet-of-Things (IoT) technologies, the computing and hardware performance of wireless sensor nodes has grown fast, and thus the usage of WSNs will be further accelerated in our lives [6,7].

In WSNs, every sensor node not only acts as a source node collecting data around it but also plays an important role as a relay node between sensor nodes in the network because of the limitations of a sensor's transmission range and energy conservation issue [8–10]. This unique feature of the WSN makes it more vulnerable to inside threats than other types of networks; the tiny sensor nodes without advanced, heavy security mechanisms can be easily captured or hacked by adversaries and then become traitors (i.e., inside attackers) [11,12]. To disrupt or break a WSN, inside attackers with authorized memberships to the network can avoid traditional security mechanisms such as cryptographic and access control mechanisms and perform various types of attacks such as packet drop attacks, packet modification attacks, misrouting attacks, and so on [4,13,14].

Trust Mechanisms (TMs) have been proposed as a promising defense method to counter such insider attacks [15–21]. In WSNs, TM measures the trustworthiness of all inside sensor nodes in the

network. Specifically, each sensor node with TM observes its neighbor nodes' behaviors, evaluates their trustworthiness as numeric trust values, and captures untrustworthy nodes as inside attackers. The defense performance of an existing TM can be further improved by sharing inside attackers already identified by the TM with all sensor nodes in the network since all sensor nodes in the network can remove the detected inside attackers in advance before they cooperate with the attackers [5,22,23]. Meanwhile, the detected inside attacker list must be securely, reliably shared with and stored in all sensor nodes in the WSN. Otherwise, inside attackers can recognize that they were detected, and then they try to hamper or break the sharing process. However, according to our survey, we observed that most existing studies simply assume the communication channel for sharing the attacker detection list is reliable and trusted even in the presence of inside attackers in the WSN.

The blockchain technology has been actively used for securely sharing data among peer nodes in distributed networks such as WSNs [24–27]. The blockchain is a secure distributed ledger that is a synchronized, distributed storage shared among participating nodes in the network, and it is known that the blockchain technology can not only tolerate the Single Point of Failure (SPoF) but also defend against malicious behaviors of inside attackers. Especially, the HyperLedger Fabric is one of the famous distributed ledger frameworks and provides many security mechanisms such as permissioned mode of operation, fine-grained access control based on membership service and certificates, and Public Key Infrastructure (PKI)-based encryption [28–31].

Consequently, in this paper, we propose the HyperLedger Fabric-based proactive defense model that securely shares the detected inside attacker list and then uses the detected attacker list to better defend against inside attackers by removing them in advance in the WSN.

The contributions of this study can be summarized as follows.

- We propose and implement a proactive defense model that integrates the HyperLedger Fabric and an existing trust mechanism with the beta trust model.
- We conduct experiments that show our proposed defense model can better defend against inside attackers compared with an existing trust mechanism; according to our experimental results, our proposed model could lower the attack damage (in terms of the number of packet drops) caused by an inside packet drop attacker by 59–67% compared with an existing trust mechanism.

The rest of this paper is organized as follows. In Section 2, we briefly overview WSNs, Trust Mechanism, and Blockchain Techniques including HyperLedger Fabric. In Section 3, we describe our research problem and the limitations of existing studies. In Section 4, we propose our method which is a proactive defense model against inside attackers in WSNs by integrating existing trust mechanisms and blockchain techniques. To validate our model, we implement and construct our proposed model based on HyperLedger Fabric platform in Section 5. In Section 6, we conduct experiments that show our proposed system can better defend against inside attacks than an existing trust mechanism. Finally, we make our conclusions with our future research directions in Section 7.

2. Background and Related Works

2.1. Inside Threats and Trust Mechanism in WSNs

In general, a WSN consists of sensor nodes, one or more sink nodes, and a base station as shown in Figure 1 and each element has the following roles:

- *Sensor Node*: A sensor node with various sensing equipment periodically collects data from the WSN area around its position, and sends the data to the Sink Node (SN). A sensor node consists of a processor, wireless communication module, power supply device, GPS, and so on.
- *Sink Node* (SN): A sink node collects data from sensors located in a sensor cluster (cluster A, cluster B, and cluster C) and delivers them to the Base Station (BS).
- *Base Station* (BS): A base station acts as a gateway linking to the network (i.e., the Internet) outside the WSN or a command and control center in the WSN.

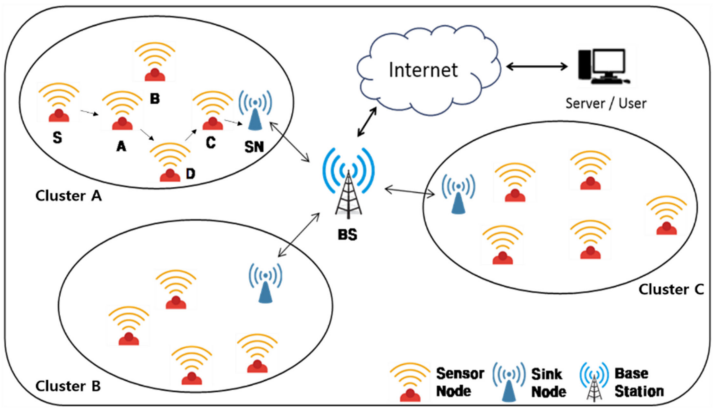


Figure 1. The general architecture of Wireless Sensor Networks (WSNs) (Sensor Nodes, Sink Nodes, a Base Station).

When a sensor delivers its data to the SN, it does not transmit the data packet directly to the SN. However, a sensor uses intermediate nodes' packet forwarding due to its limited communication range or the energy conservation issue [8–10]. For example, when Greedy Perimeter Stateless Routing (GPSR) is used in the WSN [32], a sending node chooses the next-hop node as the closest node to the SN among its neighbor nodes within the communication range. As a result, as shown in Figure 1, the routing path from node S to the SN by GPSR is S → A → D → C → SN. However, when any of the intermediate nodes (node A, D, or C) is an inside attacker, the packet delivery to the SN cannot be guaranteed since the attacker can maliciously modify or drop packets [13,20].

As a promising defense mechanism against inside attackers in WSNs, the Trust Mechanism (TM) using the notion of trust in human society has been actively proposed [5,22,33,34]. In general, TM works in the following three phases as follows [11,20,33,35–37]:

- *Phase 1 (Monitoring)*: Each node in the WSN observes neighbor nodes' behaviors and records them; for example, an observing node records a success for a neighbor node's successful cooperation; otherwise, it will record a failure [38].
- *Phase 2 (Trust Measurement)*: Based on records in Phase 1, each node calculates the trustworthiness of its neighbor nodes. Various trust models such as Bayesian, Entropy, and Game theory are used to evaluate the trustworthiness quantitively in this phase [11,35,39]. For example, when the beta trust model [33] is used to evaluate the trust value (T) based on packet forwarding behaviors, T can be measured as

$$T = \frac{s+1}{s+f+2} \tag{1}$$

  where s is the number of packet delivery successes ($s \geq 0$), f is the number of packet delivery failures ($f \geq 0$), and $0 < T < 1$. In addition, this phase can be improved by considering indirect trust values (or indirect observations) that can be provided by its neighbor nodes when they can also observe the evaluated node's packet forwarding behaviors [5,39]; the measured trust value can be thought as the reputation value T_R. In this case, T_R can be calculated by

$$T_R = w_D T_D + \sum_{k=1}^{n} w_{I,k} T_{I,k} \tag{2}$$

  where w_D is a weighting factor for the direct trust value T_D, $w_{I,k}$ is a weighting factor for indirect trust value $T_{I,K}$ provided by a neighbor node k, n is the number of neighbor nodes, and $0 \leq w_D$, $w_{I,K} \leq 1$.
- *Phase 3 (Attacker Detection)*: According to the calculated trust value in phase 2, the evaluating node can determine whether the evaluated node is an inside attacker if T is lower than a predetermined detection threshold (θ_T).

2.2. Blockchain Technology and HyperLedger Fabric

Blockchain is proposed as a securely distributed ledger that is a synchronized, distributed storage shared among participating nodes in the network [40,41]. Unlike a single ledger (e.g., a single server or database), it is known that the blockchain technology can not only tolerate the Single Point of Failure (SPoF) but also solve the Byzantine general problem in a network where faulty nodes or inside attackers reside [42].

The structure of the blockchain is shown in Figure 2 where blocks are connected like a chain by block hash values [40]. Each block consists of two main parts as a block header and block body as shown in Figure 2. The block header includes the previous block's hash value, block version, Merkle root, timestamp, and so on, and a block hash value is calculated based on the header information. In addition, the block body has a list of transactions containing data that need to be shared with nodes in the network. Thanks to this special chaining feature based on cryptographic hash values of blocks, the blockchain can resist various attacks breaking the integrity of it such as tampering attacks on existing blocks or maliciously adding a new block to the blockchain [28,30]. For example, when the attacker changes the data stored in the body of block #1, the hash value of block #1 is changed accordingly. However, since the hash value of block #2 does not match the hash value of the previous block stored in block #2, such malicious tampering attack will be detected immediately.

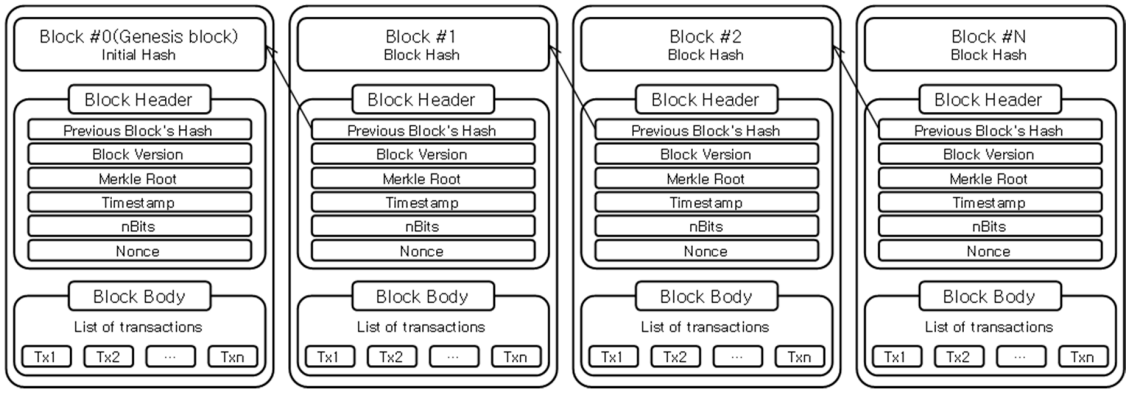


Figure 2. The structure of blockchain.

There are three types of blockchain: public, private, and consortium blockchain. While the public blockchain is an open blockchain network in which anyone can participate, the private and consortium blockchains are the permissioned blockchains in which only authorized nodes can participate. In addition, there are three representative types of consensus algorithms such as Proof of Work (PoW), Proof of Stake (PoS), and Delegated Proof of Stake (DPoS) [41]. A consensus algorithm in the blockchain refers to a procedure through which all the peers in the network can reach a common agreement on the states of certain data among distributed nodes [37,41]. There are many studies on advancing existing consensus algorithms in terms of security, fault tolerance, or efficiency [43,44]; in [43], authors proposed a PoS-based consensus protocol in which a meta node can maintain a consistent sub-chain in the Bitcoin blockchain. In [44], the authors proposed an extended version of the consensus protocol for consistently extending sub-chains embedded in the Bitcoin blockchain.

HyperLedger Fabric is one of the famous distributed ledger frameworks (Fabric, Iroha, Sawtooth, etc.), which is developed by the HyperLedger blockchain open source project [29] hosted by the Linux Foundation. HyperLedger Fabric is designed to develop applications or solutions with a modular architecture that can plug and play elements such as consensus and membership services. HyperLedger Fabric is a type of the permissioned blockchain, and Table 1 shows major components of the HyperLedger Fabric [28,29].

Table 1. Major components of HyperLedger Fabric.

Component	Description
MSP	Manages User Identifications (User ID) and authenticates clients; MSP stands for Membership Service Provider.
Client	An application that proposes transactions.
Peer	Commits proposed transactions and maintains its ledger; a peer endorses the transaction request which comes from the client, commits the block received from the Orderer, and updates its ledger.
Orderer	Sorts the endorsed transactions received from peers in the chronological order, creates a block containing the endorsed transactions, and distributes the created blocks to all peers in the network.
Chaincode	Executes transactions when per-defined conditions are satisfied; Chaincodes are deployed to all peers at the initialization stage of the fabric network.

Figure 3 shows how a block is created and updated in the HyperLedger Fabric. As we explained above, the HyperLedger Fabric is a permissioned blockchain, and thus only authorized nodes can participate in the blockchain network. Participating nodes in the fabric network can be identified because they must be registered in advance to the Membership Service Provider (MSP). In addition, all transactions are encrypted based on the Public Key Infrastructure (PKI).

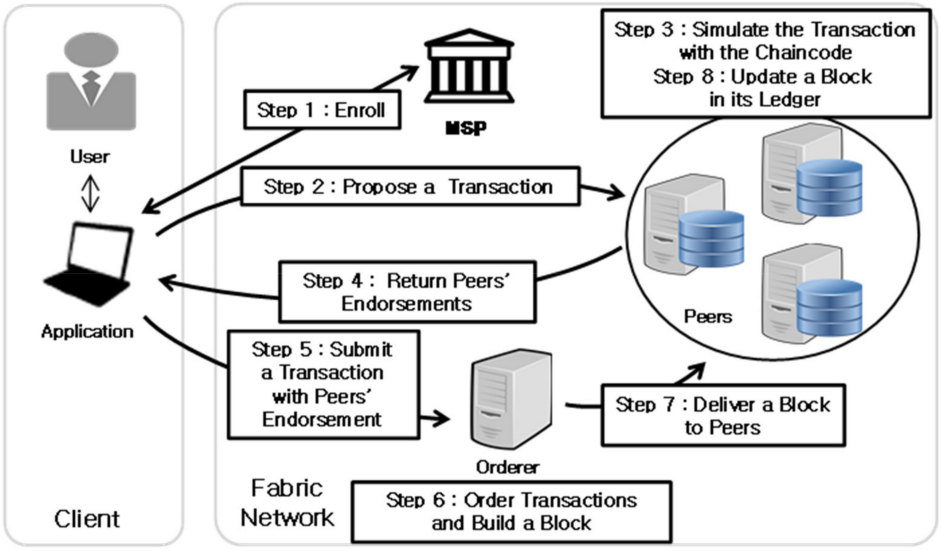


Figure 3. Working steps of creating and updating a block in the HyperLedger Fabric [29,31].

- Step 1: The user obtains a User ID with a certificate from the MSP after enrolling in the MSP by using an application.
- Step 2: The user proposes a transaction to Peers in the network.
- Step 3: Each Peer who received the transaction from the user performs a validation check according to the endorsement policy and then simulates the transaction with the pre-deployed Chaincode; at this stage, the ledger is not updated yet.
- Step 4: Each Peer returns its endorsement with the expected result value (Read/Write set) to the user when simulating the Chaincode is successfully performed.
- Step 5: The user collects the endorsements from Peers, and submits them to the Orderer.
- Step 6: The Orderer sorts the endorsed transactions received from Peers in the chronological order, creates a block containing the endorsed transactions, and distributes the created blocks to all Peers in the network.

- Step 7: The Orderer sends the created block to Peers.
- Step 8: After receiving and verifying the block, each Peer updates its ledger by adding a new block to the previous block. At this stage, the ledger is finally updated.

2.3. Related Works

In this section, we briefly introduce and discuss existing works related to our study.

First, many studies have proposed defense methods against inside attackers based on trust mechanism and reputation systems in WSNs as follows. In [22], the authors proposed Group based Trust Management Scheme (GTMS) in which each node has one of three trust states assigned from the base station and such state is shared with all sensor nodes in the network. Specifically, first, each node calculates trust values of other nodes in its cluster. Next, the cluster head collects those data and forwards them to the base station. Finally, the base station assigns one of the trust state (trusted, untrusted, and uncertain) to each node, and periodically multicasts the globally assigned trust state data to all cluster heads. In [23], some special nodes (designated agents) observe the operation of other nodes by using the watchdog mechanism and calculate their trust values. The evaluated trust rating is encrypted and broadcasted to their neighbor nodes. In [37], the authors proposed Consensus-based False Information Filtering Algorithm (CFIFA) to show better defense performance against bad-mouthing and false-praise attacks than an existing trust mechanism. In CFIFA, inaccurate or malicious observations information can be filtered out by a consensus algorithm based on a majority voting method, and thus overall trust evaluation process can be guaranteed. However, the common problem on these studies is that they assume the network and communication channel is secure and thus they did not consider a secure propagation method by which sensor nodes can share the detected inside attacker list or information with nodes in the network.

Meanwhile, in [24], the authors proposed a Blockchain Trust Model (BTM) that manages the trust evaluation data of all sensor nodes and the detection history of malicious nodes by using both the smart contract used in Ethereum and Trust Mechanism (TM). They focused on preserving all trust mechanism records in only cluster head nodes or sink nodes in WSNs by using the blockchain technology, but they did not study to use the information of detected malicious nodes to proactively defend against them by sharing the information with other sensor nodes.

3. Problem Description

To motivate this study, by using two attack cases, we explain why the proactive defense method using the detected inside attacker list is required to better defend against inside attacks in the WSN even if a trust mechanism is deployed. Consider a simple WSN topology with eight nodes (six sensor nodes, one sink node, and one base station) as shown in Figure 4; node D is a blackhole packet drop attacker [36]. We assume that the GPSR routing algorithm and a trust mechanism using the beta trust model are used in this WSN (see Section 2).

First, we consider the first attack case depicted in Figure 4a. In this WSN topology, according to the GPSR, node A will choose node D as the next hop to send its data packet toward SN (i.e., the routing path is A → D → SN). Since the inside attacker D is the blackhole attacker, node D will drop all packets received from node A. Meanwhile, node A's trust mechanism will detect node D as an inside attacker when D's trust value is lower than the detection threshold, and then remove it from A's routing table. After that, node A will find another trustful routing path (A → B → C → SN). However, assuming that node B's best next-hop is node D by the GPSR, node B will forward A's data packets to the attacker node D again since node B does not know node D is an inside attacker; this can be possible when the attacker node D launches selective forwarding attacks such that node D drops only node A's packets but correctly forward the other nodes' packets toward the SN. This problem can be solved if node A shares its detected inside attacker list with node B so that node B also removes node D in advance.

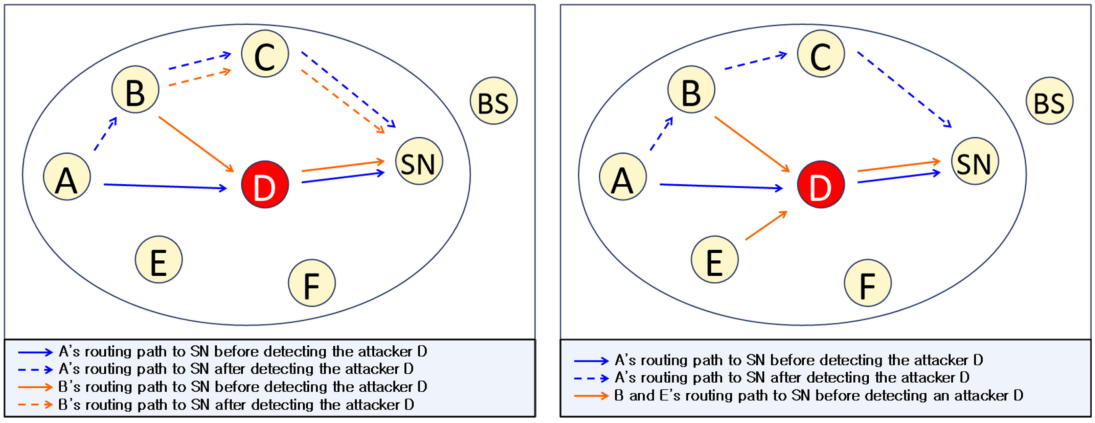


Figure 4. Motivational examples (two attack cases). (**a**) first attack case; (**b**) second attack case.

Next, we consider the second attack case as depicted in Figure 4b. In this case, node D drops all packets received from its neighbor nodes A, B, and E. When node A detects node D as an inside attacker, node B and E may not know node D is an attacker yet. Therefore, nodes B and E will keep sending their packets to node D until their trust mechanisms evaluate node D's trust value below the detection threshold. However, when node D launches on–off attacks [45] to node B and E, it will take a significant amount of time for node B and E to detect node D. This problem can be also solved if node A shares its detected inside attacker list with node B and E.

Meanwhile, the detected inside attacker list must be securely shared with and stored in all sensor nodes in the WSN. For example, if the whole procedures of sharing the attacker detection list are transparent to all inside members including the attacker node D, node D will try to avoid such a situation by modifying or dropping the packets related to the sharing procedures.

However, according to our survey as we discussed in Section 2.3, existing research works did not well address all of the above problems at the same time. Therefore, in this paper, we propose to use the HyperLedger Fabric to securely share the detected inside attacker list with all sensor nodes in the WSN.

4. Proposed Method

In this section, we propose a proactive defense model that securely sharing the detected inside attacker list with all sensor nodes based on the blockchain technique and trust mechanism even in the presence of multiple collaborating inside attackers in the network. We first show the architecture of our proposed defense model and then describe how our model works.

4.1. The Architecture of Proposed Defense Model

Figure 5 shows the network frameworks of the existing model with only a trust mechanism and our proposed model. As we explained in Section 2, blockchain technology provides a trustworthy method by which sensor nodes in the network can share data securely. Therefore, we design our defense model by adding a blockchain mechanism to the existing model as shown in Figure 5b.

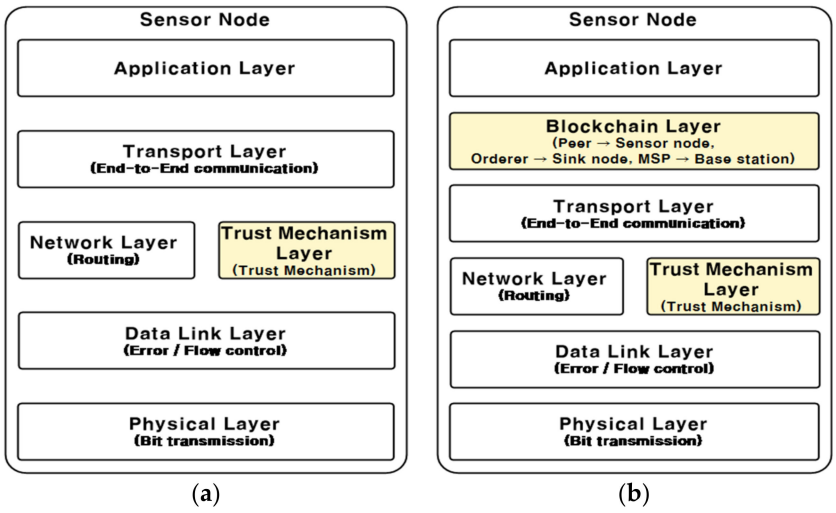


Figure 5. Networking frameworks of existing model and proposed model in a sensor node. (**a**) existing model; (**b**) proposed model.

In the existing model, if a sensor (the trust mechanism) detects an inside attacker, the sensor (the trust mechanism) will report and spread it to other sensor nodes. Such data exchange is performed at the layer of trust mechanism in a distributed manner in the WSN. As we discussed in Section 3, such data exchange cannot be secured in the presence of other collaborating inside attackers in the network, and thus can be hampered by them. On the other hand, by the favor of the blockchain technology, our proposed model securely shares the detected inside attacker list (or data) with all sensor nodes at the layer of the blockchain mechanism, which is implemented as a distributed middleware in the WSN. Thus, by the assumption that the blockchain technology provides a secure data sharing method, the detected inside attacker list can be shared securely in the WSN employed with our proposed model.

Figure 6 shows two different viewpoints for the same WSN topology where each sensor employs our proposed model; Figure 6a shows the general WSN viewpoint and Figure 6b shows the blockchain network viewpoints. When a sensor wants to share the detected inside attacker list with all nodes in the network, the blockchain mechanism in sensors will be activated. In this case, we need to view the WSN as shown in Figure 6b. Thus, in the blockchain network (e.g., HyperLedger Fabric network), each sensor node in the WSN in Figure 6a plays a role of *User* and *Peer* in the blockchain network in Figure 6b, and the Sink Node (SN) and the Base Station (BS) in Figure 6a plays a role of *Orderer* and *MSP* in the blockchain network in Figure 6b, respectively. We explain how our proposed model works to securely share the detected inside attacker list to all sensor nodes by using these two network viewpoints in the below Section 4.2.

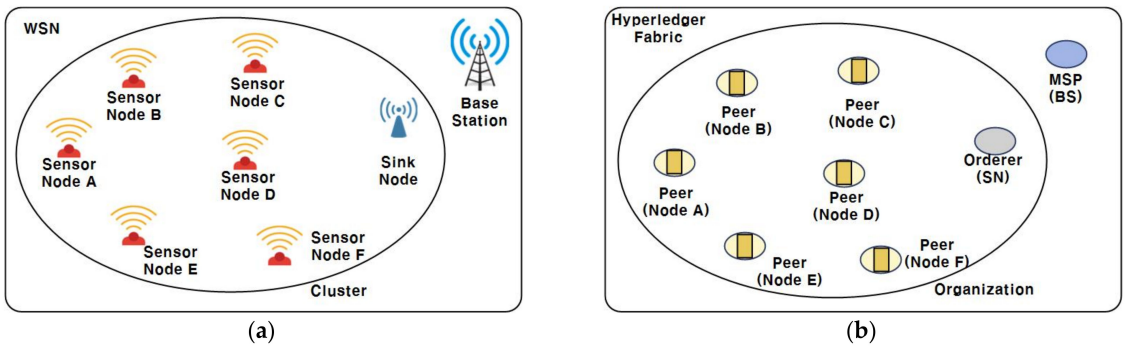


Figure 6. A WSN employed with our proposed model. (**a**) general WSN viewpoint; (**b**) blockchain (HyperLedger Fabric) network viewpoint.

4.2. Working Steps of Our Proposed Defense Model

Consider a WSN with eight nodes (six sensor nodes, one sink node, and one base station) as shown in Figure 7. We assume that node D is an inside packet drop attacker who tries to drop all packets received from the node. In this situation, node A captures an inside attacker node D by its trust mechanism and node A wants to share the detected inside attacker list (data about node A) to the other nodes in the network.

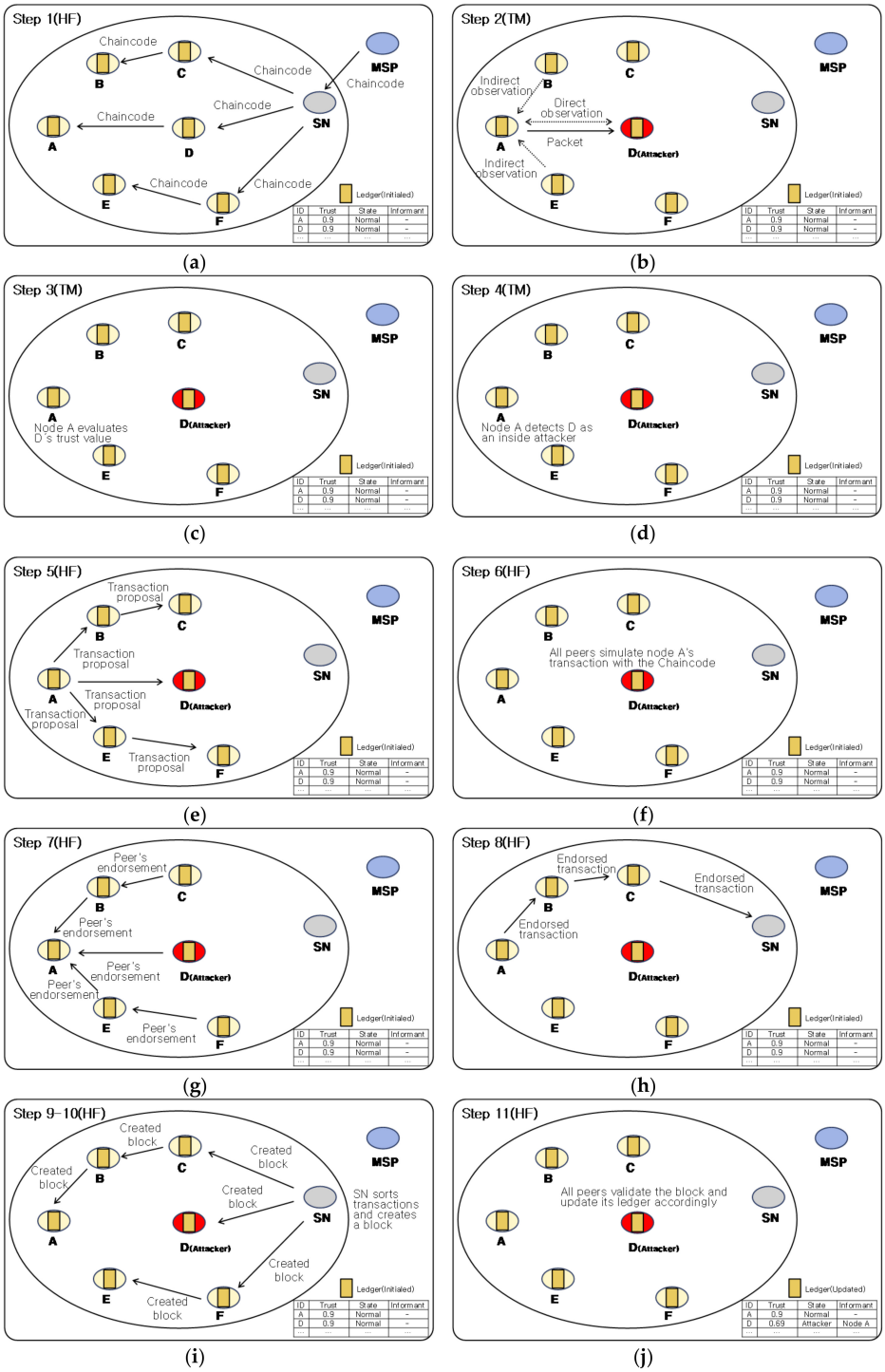


Figure 7. Description of working steps of the proposed model in a WSN; D is an inside attacker. (**a**) step 1; (**b**) step 2; (**c**) step 3; (**d**) step 4; (**e**) step 5; (**f**) step 6; (**g**) step 7; (**h**) step 8; (**i**) step 9–10; (**j**) step 11.

To proactively defend against the insider attacker A, our proposed defense model works as the following steps (Step 1–Step 11) (see Figure 7); (TM) indicates that step works in the trust mechanism and (HF) indicates that step works in the HyperLedger Fabric Platform of our proposed model.

- **Step 1(HF):** The administrator initializes the HyperLedger Fabric network of the WSN by deploying and spreading the Chaincode to all nodes in the network (Figure 7a).
- **Step 2(TM):** Node A sends its packets to node D (because D is assumed to be the best next-hop) and then observe' node D's packet forwarding behaviors. In addition, neighbor nodes B and E also observe' node D's behaviors and then provide their indirect observation information to node A (Figure 7b).
- **Step 3(TM):** Node A evaluates node D's trust value ($T_{A\rightarrow D}$) by using its direct observations and indirect observations from neighbor nodes B and E by using the trust model (e.g., by two Equations (1) and (2)) (Figure 7c).
- **Step 4(TM):** Node A detects node D is an inside attacker when $T_{A\rightarrow D}$ becomes lower than the inside attacker detection threshold θ_T (Figure 7d).
- **Step 5(HF):** To securely share its detected inside attacker list (node D is an attacker), node A sends a transaction proposal to all sensor nodes (all peers); the transaction proposal includes the malicious node ID, trust value, state, and informant (Figure 7e).
- **Step 6(HF):** All Peers validate the received transaction according to the endorsement policy and then simulate the transaction in the Chaincode (Figure 7f).
- **Step 7(HF):** When the simulation result is fine, Peers return their endorsements and expected result values (Read/Write Set) to node A (Figure 7g).
- **Step 8(HF):** Node A compares all endorsements and Read/Write Set received from Peers, and then sends them to the Orderer (the Sink Node; SN) (Figure 7h).
- **Step 9(HF):** The Orderer creates a new block that contains the detected inside attacker list about node D (Figure 7i).
- **Step 10(HF):** The Orderer sends the created block to all Peers (sensor nodes) (Figure 7i).
- **Step 11(HF):** After receiving the block, Peers validate the block and update their ledgers (Figure 7j).

Now all sensor nodes know node D is an inside attacker captured by node A, and they can avoid node D proactively and inside attackers cannot tamper with updated distributed ledgers without being detected by the HyperLedger Fabric in our proposed model.

5. Implementation and Construction of Our Proposed Model

5.1. Development Environment

To implement and construct our model, we used the HyperLedger Fabric for the blockchain platform and the Beta trust model for the trust mechanism. To install the HyperLedger Fabric, we used a PC (Intel Core i5-2500, 3.3 GHz, 16 GB RAM, and Windows 10) with Oracle VirtualBox Latest [46] and Ubuntu 16.04 LTS [47]. In addition, we installed all required packages (HyperLedger Fabric version 2.2 [48], Docker version 19.03 [49], Docker-compose version 1.25.4 [50], Go version 1.14 [51], and Node.js version 12.16.2 [52]) to construct HyperLedger Fabric network in the Linux (Ubuntu) environment. We implemented a WSN with a trust mechanism based on the Beta trust model and the GPSR routing algorithm by using C++ Programming Language.

5.2. Construction and Initialization of HyperLedger Fabric Network (HFN)

According to the topology of the WSN depicted in Figure 8, we constructed and initialized a HyperLedger Fabric Network (HFN) as follows; the constructed HFN will be used in experiments later in Section 6. Initially, we created the basic test-network with two Organizations, two Peer nodes and one orderer by executing the fabric network creation procedure. After that, by modifying the basic

test-network, we added six Peer nodes to the network. We designed a Chaincode and deployed it to the fabric network. Finally, we obtained a User ID with a certificate. Figure 8 shows the created fabric network where two Peer nodes (each belonging to org1 and org2), a single orderer, and a CA that manages certificates were created.

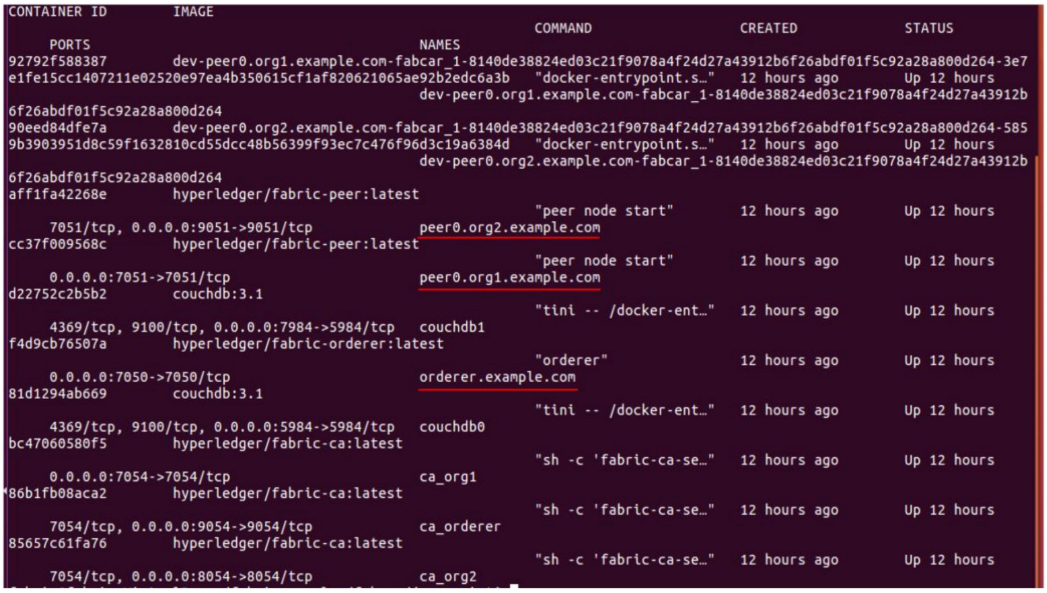
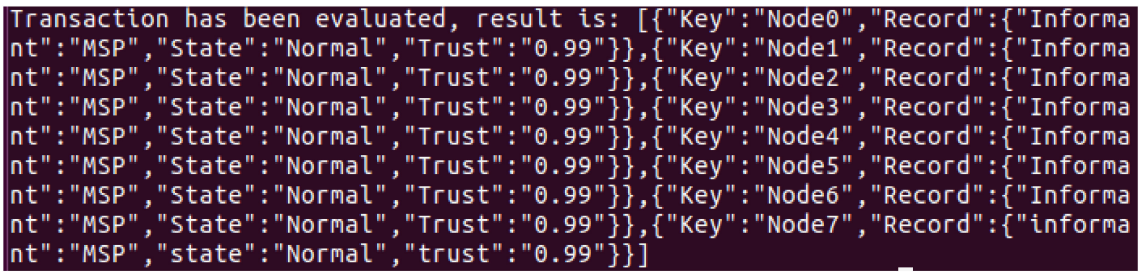
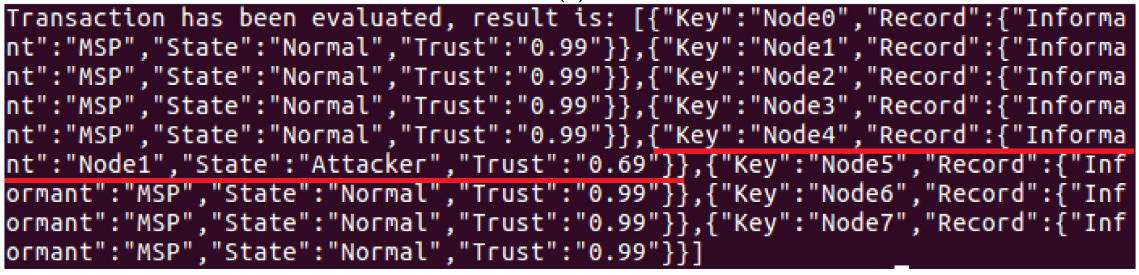


Figure 8. The generated fabric network based on the test-network.

Figure 9a shows the result of querying the initialized ledger after the fabric network is created and the Chaincode is deployed to the fabric network as we explained above; we can see the initial ledger in each node is set to {"Key": Node ID, "Informant": "MSP", "State": "Normal", "Trust": "0.99"}. Figure 9b shows the result of querying after updating the ledger by creating a new transaction by assuming that node 1 detected node 4 as an inside attacker when the detection threshold is 0.7. As a result, when executing the new transaction, the information stored in the ledger is updated as {"Key": Node ID, "Informant": "MSP", "State": "Attacker", "Trust": "0.69"}, and other nodes can use the same information in their ledgers.



(**a**)

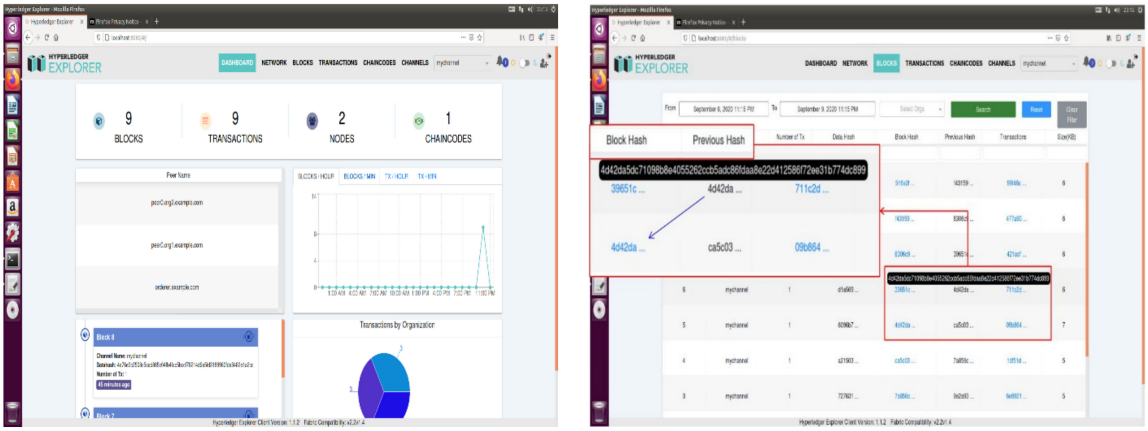


(**b**)

Figure 9. The sample result of querying. (**a**) the initialized ledger; (**b**) the updated ledger.

5.3. Managing HFN by Using the HyperLedger Explorer

HyperLedger Explorer [53] is a Graphic User Interface-based (GUI) web tool that can monitor information such as blocks, transactions, and the Chaincode generated in the fabric network. Figure 10a shows the main page of HyperLedger Explorer that corresponds to the fabric network implemented in Section 5.2. We can check the number of blocks, transactions, peer nodes, and Chaincodes that were created in and deployed to the HyperLedger Fabric network. Figure 10b is the blocks page that shows the information of each block stored in the ledger. For example, we can see the hash value of each block and also the previous hash value of block 6 is the block hash value of block 5. Using this tool, we can monitor and manage our fabric network.



(**a**) (**b**)

Figure 10. The page views of HyperLedger explorer. (**a**) main page; (**b**) blocks page.

6. Experiments

6.1. Experimental Purpose and Methods

The purpose of this experiment is to show how our proactive defense model implemented based on the HyperLedger Fabric and Trust Mechanism (HF-TM_{Beta}) can better defend against an inside attacker in WSNs than an existing Trust Mechanism (TM_{Beta}); both defense models use the Beta trust model-based trust mechanism as described in Section 2.1.

For our experiments, we considered a simple WSN topology with eight sensor nodes (Node A – Node H) as depicted in Figure 11; we put one inside attacker node D (blackhole packet drop attacker) and one sink node H. In this experimental setup, node A (source node) continues to generate data packets and send them toward the destination node H. However, due to the limited communication range of a sensor node (red line in the figure), node A needs the cooperation of intermediate nodes between node A and node H. According to the GPSR routing algorithm and the experimental network topology, the routing path from A to node H will be selected as A → D (inside attacker) → F → H. We implemented and constructed all required experimental programs including TM_{Beta} and HF-TM_{Beta} by using C++ and HyperLedger Fabric Platform by considering the sample WSN as we explained in Section 5.

We conduct experiments as follows.

(1) Node A (source node) continues to create a data packet and send it toward the sink node H according to its routing algorithm; as we explained, the default best next hop will be node D (inside packet drop attacker). Node D (as a blackhole packet drop attacker) continues to drop all received packets.

(2) Node A will defend against node D by using two defense models (TM_{Beta} and HF-TM_{Beta}); when node A captures node D, node A will find another node by using its defense model and routing algorithm.

(3) When node A's data packet finally reaches the destination node H by completely avoiding the attacker D in its trustworthy routing path to the destination node H, we terminate experiments and compare the number of dropped packets produced by TM_{Beta} and HF-TM_{Beta}.

- **DP[TM_{Beta}]**: The number of dropped packets when TM_{Beta} is used;
- **DP[HF-TM_{Beta}]**: The number of dropped packets when HF-TM_{Beta} is used;

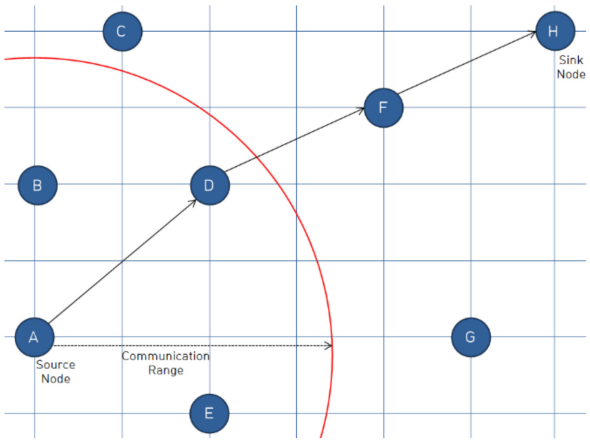


Figure 11. A WSN application for our experiment.

6.2. Experimental Results and Analysis

To better understand our experimental results, we describe how two defense models (TM_{Beta} and HF-TM_{Beta}) work against an inside packet drop attacker (node D) in our experiments by using Figure 12a,b, respectively.

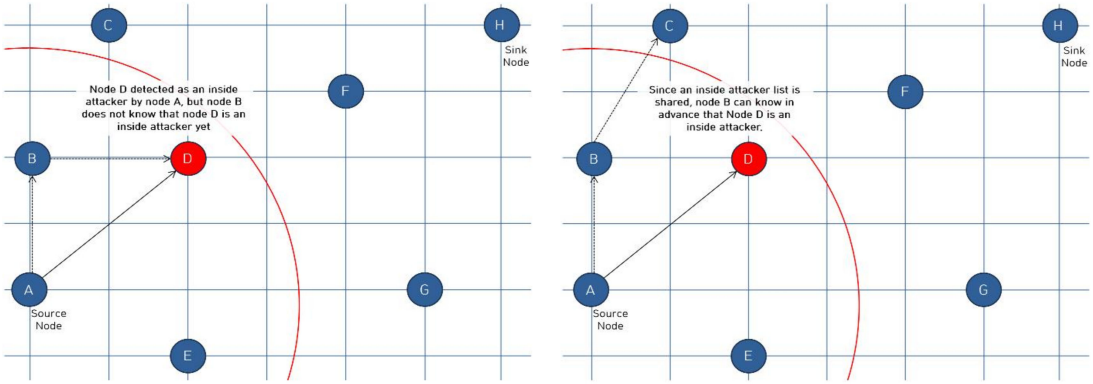


Figure 12. Two defense models in our experiments. (**a**) TM_{Beta} (existing model); (**b**) HF-TM_{Beta} (our proposed model).

(1) **When TM_{Beta} is used:** If the source node A detects the attacker node D, node A chooses node B as its next hop instead of node E according to its TM_{Beta} and GPSR. However, node B chose the attacker node D as its next hop according to its TM_{Beta} and GPSR because node B does not know node D is an inside attacker detected by node A. This is because TM_{Beta} does not have a defense mechanism that securely shares the detected inside attacker list with all sensor nodes in the WSN. Consequently, node B continued to send data packets to node D until node B detects node D as an inside attacker by its TM_{Beta}.

(2) **When HF-TM_{Beta} is used:** Like the case when TM_{Beta} is used, if the source node A detects the attacker node D, node A chose node B as its next hop instead of node E according to its HF-TM_{Beta}

and GPSR. However, unlike the case that TM_{Beta} is used, thanks to the HyperLedger Fabric platform of HF-TM_{Beta}, node B already had the detected inside attacker list (i.e., node D is an inside attacker detected by node A) and thus node B chose node C instead of node D.

Next, we compare the defense performance of two defense models by using the metric $DP[TM_{Beta}]$ and DP[HF-TM_{Beta}]. Table 2 and Figure 13 show our experimental results when the various detection threshold values are used (i.e., θ_T is used between 0.1 and 0.9).

Table 2. Experimental results.

θ_T	$DP[TM_{Beta}]$	DP[HF-TM_{Beta}]	ΔDP (= $DP[TM_{Beta}] - DP$[HF-TM_{Beta}])	Decrease Rate (DR)
0.1	2731	913	1818	67%
0.2	1216	408	808	66%
0.3	709	239	470	66%
0.4	457	155	302	66%
0.5	307	105	202	66%
0.6	205	71	134	65%
0.7	133	47	86	65%
0.8	79	29	50	63%
0.9	37	15	22	59%

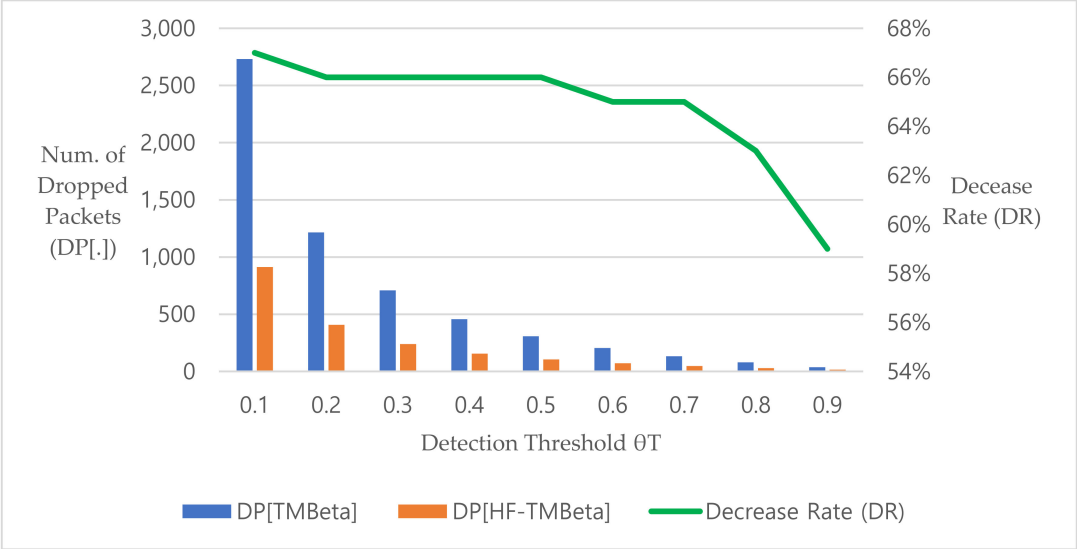


Figure 13. Experimental results.

First, $DP[TM_{Beta}]$ is much larger than DP[HF-TM_{Beta}] for all values of θ_T. This is because our proposed model (HF-TM_{Beta}) proactively avoided attacker D unlike the existing trust mechanism (TM_{Beta}) as we explained above. As a result, HF-TM_{Beta} could save a significant amount of packet loss compared with TM_{Beta} as we can see the amount of saving packet loss ΔDP.

Second, as θ_T increases, ΔDP and decrease rate (DR) also decreased. We calculated

$$DR(\%) = \frac{(DP[TM_{Beta}] - DP[HF - TM_{Beta}])}{DP[TM_{Beta}]} \times 100 \quad (3)$$

DR can be considered as the amount of dropped packets the existing trust mechanism TM_{Beta} additionally introduced or the inside attacker D could introduce damage to the network in terms of packet loss until it finds a completely reliable routing path from the source node A to the destination H compared with our HF-TM_{Beta}. As shown in Table 2 and Figure 13, DR ranged from 59% when $\theta_T = 0.9$ to 67% when $\theta_T = 0.1$ and as θ_T increases, DR decreased monotonously. When θ_T is low, the attacker is not easily detected by the trust mechanism and thus the attacker could launch packet drop attacks

longer than when θ_T is high. We note that although a very low θ_T is not used in real practices, we conducted our experiments based on various values of θ_T and showed results for research purposes.

Consequently, based on our experimental results, we verified that our proposed model can proactively avoid a detected inside attacker and thus improve significantly security and performance of the WSN.

7. Conclusions and Future Works

In this paper, we proposed a proactive defense model that integrates the blockchain technology and a trust mechanism against inside attackers by securely sharing the detected inside attacker list with all sensor nodes in the WSN. In addition, we implemented our proposed model based on the HyperLedger Fabric which is one of the most widely-used consortium blockchain platforms and the trust mechanism with the Beta trust model. Last, we showed experiment results that validate our proposed defense model can avoid the detected inside attackers proactively and thus save potential damage which can be made by those attackers; in our experimental settings, our model with the inside attacker sharing mechanism could lower the significant amount of packet drops by an inside packet drop attacker by 59 to 67% according to various detection threshold values compared with an existing trust mechanism without sharing the detected inside attacker list.

Our future research directions can be summarized as follows:

- We will construct more realistic, complicated WSN topologies with many sensor nodes on the HyperLedger Fabric and conduct extensive experiments and simulations by considering various network performance metrics such as network lifetime, energy consumption, and entire network packet delivery rate.
- We will study more sophisticated insider threats even in the presence of our proposed defense model in the WSN, especially advanced colluding insider attacks. By doing this study, we believe we can find potential vulnerabilities or limitations of our proposed method and improve our proposed model by considering them.

Author Contributions: Conceptualization, Y.C.; methodology, K.C. and Y.C.; software, K.C.; validation, K.C.; formal analysis, K.C. and Y.C.; investigation, K.C. and Y.C.; writing—original draft preparation, K.C. and Y.C.; writing—review and editing, Y.C.; visualization, K.C.; supervision, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: An earlier version of this paper was presented at the Korea Software Congress (KSC) Winter Conference, Pyeongchang, South Korea, in 18–20 December 2019. The author would like to thank the editors and reviewers for their valuable comments and constructive suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422. [CrossRef]
2. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330. [CrossRef]
3. Hande, A.; Cem, E. Wireless sensor networks for healthcare. *Comput. Netw.* **2010**, *54*, 2688–2710.
4. Mohamed, L.M. Classification of Attacks in Wireless Sensor Networks. In Proceedings of the International Congress on Telecommunication and Application '14, University of A. MIRA, Bejaia, Algeria, 23–24 April 2014.
5. Khalid, O.; Khan, S.U.; Madani, S.A.; Hayat, K.; Khan, M.I.; Min-Allah, N.; Kolodziej, J.; Wang, L.; Zeadally, S.; Chen, D. Comparative study of trust and reputation systems for wireless sensor networks. *Secur. Commun. Netw.* **2013**, *6*, 669–688. [CrossRef]
6. Wang, Y. The design of smart home system based on wireless sensor network. In Proceedings of the IEEE 4th International Conference on Electronics Information and Emergency Communication, Beijing, China, 15–17 November 2013.

7. Ko, J.; Lu, C.; Srivastava, M.B.; Stankovic, J.A.; Terzis, A.; Welsh, M. Wireless Sensor Networks for Healthcare. *Proc. IEEE* **2010**, *98*, 1947–1960. [CrossRef]
8. Pathan, A.S.K.; Lee, H.; Hong, C. Security in Wireless Sensor Networks: Issues and Challenges. In Proceedings of the 8th International Conference Advanced Communication Technology 2006, Phoenix Park, Korea, 20–22 February 2006.
9. Chelli, K. Security Issues in Wireless Sensor Networks: Attacks and Countermeasures. In Proceedings of the World Congress on Engineering, London, UK, 13 July 2015.
10. Xing, K.; Srinivasan, S.S.R.; Jose, M.; Li, J.; Cheng., X. Attacks and Countermeasure in Sensor Networks: A Survey. In *Network Security*; Springer: Boston, MA, USA, 2010; pp. 251–272.
11. Shivani, G.; Mukul, V.; Aparajita, N. Insider threats in wireless sensor networks and their countermeasures. *Int. J. Comput. Sci. Mob. Comput.* **2016**, *5*, 476–486.
12. Daia, A.S.A.; Ramadan, R.A.; Fayek, M. Sensor Networks Attacks Classifications and Mitigation. *Ann. Emerg. Technol. Comput.* **2018**, *2*, 28–43. [CrossRef]
13. Cho, Y. Trust-Based Defense Against Insider Packet Drop Attacks in Wireless Sensor Networks. Ph.D. Thesis, University of Maryland, College Park, MD, USA, 2013.
14. Cho, Y.; Qu, G. Detection and prevention of selective forwarding-based denial-of-service attacks in WSNs. *Int. J. Distrib. Sens. Netw.* **2013**, *9*, 205920. [CrossRef]
15. Trcek, D. Trust management in the pervasive computing era. *IEEE Secur. Priv.* **2011**, *9*, 52–55. [CrossRef]
16. Varadharahan, V. A note on trust-enhanced security. *IEEE Secur. Priv.* **2009**, *7*, 57–59. [CrossRef]
17. Han, G.; Jiang, J.; Shu, L.; Niu, J.; Chao, H.C. Management and applications of trust in Wireless Sensor Networks: A surveyJ. *Comput. Syst. Sci.* **2014**, *80*, 602–617. [CrossRef]
18. Lopez, J.; Roman, R.; Agudo, I.; Fernandez-Gago, C. Trust management systems for wireless sensor networks: Best practices. *Comput. Commun.* **2010**, *33*, 1086–1093. [CrossRef]
19. Cho, Y.; Qu, G. FADER: False alarm detection and recovery for trust-aware routing in wireless sensor networks. In Proceedings of the International Conference on Connected Vehicles and Expo, Las Vegas, NV, USA, 2–6 December 2013.
20. Cho, Y.; Qu, G. Insider Threats against Trust Mechanism with Watchdog and Defending Approaches in Wireless Sensor Networks. In Proceedings of the IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, 24–25 May 2012.
21. Cho, Y.; Qu, G. Enhancing Trust-Aware Routing by False Alarm Detection and Recovery. In Proceedings of the IEEE Military Communications Conference, Baltimore, MD, USA, 6–8 October 2014.
22. Shaikh, R.A.; Jameel, H.; d'Auriol, B.J.; Lee, H.; Lee, S.; Song, Y. Group-Based Trust Management Scheme for Clustered Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2009**, *20*, 1698–1712. [CrossRef]
23. Chen, H.; Wu, H.; Zhou, X.; Gao, C. Agent-based Trust Model in Wireless Sensor Networks. In Proceedings of the Eighth ACIS International Conference on Software Engineering, Qingdao, China, 30 July–1 August 2007.
24. She, W.; Liu, Q.; Tian, Z.; Chen, J.S.; Wang, B.; Liu, W. Blockchain Trust Model for Malicious Node Detection in Wireless Sensor Networks. *IEEE Access* **2019**, *7*, 38947–38956. [CrossRef]
25. Saia, R.; Carta, S. Internet of Entities (IoE): A Blockchain-based Distributed Paradigm for Data Exchange between Wireless-based Devices. In Proceedings of the 8th International Conference on Sensor Networks (SENSORNETS-2019), Prague, Czech Republic, 26–27 February 2019.
26. Lazrag, H.; Chehri, A.; Saadane, R.; Rahmani, M.D. A Blockchain-Based Approach for Optimal and Secure Routing in Wireless Sensor Networks and IoT. In Proceedings of the 2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Sorrento, Italy, 26–29 November 2019.
27. Ren, Y.; Liu, Y.; Ji, S.; Sangaiah, A.K.; Wang, J. Incentive Mechanism of Data Storage Based on Blockchain for Wireless Sensor Networks Research on Efficient Data Forwarding in Vehicular Networks. *Mob. Inf. Syst.* **2018**, *2018*, 6874158.
28. Yoon, D. Understanding of the Blockchain. In *Blockchain Learning with Hyperledger Fabric (In Korean)*; Jpub: Paju, Korea, 2018; Volume 1, pp. 6–12.
29. Hyperledger. Available online: https://www.hyperledger.org (accessed on 9 August 2020).
30. Kim, H. Analysis of Security Threats and Countermeasures on Blockchain Platforms. *J. KIIT* **2018**, *16*, 103–112. [CrossRef]

31. Mukne, H.; Pai, P.; Raut, S.; Ambawade, D. Land Record Management using Hyperledger Fabric and IPFS. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019.
32. Brad, K.; Kung, H.T. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking 2000, Boston, MA, USA, 6–11 August 2000; pp. 243–254.
33. Audun, J.; Roslan, I. The beta reputation system. In Proceedings of the 15th Bled Electronics Commerce Conference 2002, Bled, Slovenia, 17–19 June 2002.
34. Josang, A. Trust and Reputation Systems. In Proceedings of the International School on Foundations of Security Analysis and Design, Bertinoro, Italy, 9–15 September 2007.
35. Yu, Y.; Li, K.; Zhou, W.; Li, P. Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures. *J. Netw. Comput. Appl.* **2012**, *35*, 867–880. [CrossRef]
36. Farruh, I.; Yousaf, B.Z. Trust mechanisms to secure routing in wireless sensor networks: Current state of the research and open research issues. *J. Sens.* **2017**, *2017*, 4724852.
37. Suh, T.; Cho, Y. An Enhanced Trust Mechanism with Consensus Based False Information Filtering Algorithm against Bad-Mouthing Attacks and False-Praise Attacks in WSNs. *Electronics* **2019**, *8*, 1359. [CrossRef]
38. Marti, S.; Giuli, T.J.; Lai, K.; Baker, M. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking 2000, Boston, MA, USA, 6–11 August 2000; pp. 255–265.
39. Ahmed, A.; Bakar, K.A.; Channa, M.I.; Haseeb, K.; Khan, A.W. A survey on trust based detection and isolation of malicious nodes in ad-hoc and sensor networks. *Front. Comput. Sci.* **2014**, *9*, 280–296. [CrossRef]
40. Zibin, Z.; Shaoan, X.; Hongning, D.; Xiangping, C.; Huaimin, W. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2008**, *14*, 352–375.
41. Yim, J.; Yoo, H.; Gwak, J.; Kim, S. Blockchain and Consensus Algorithm (In Korean). *Electron. Telecommun. Trends* **2018**, *33*, 45–56.
42. Gramoli, V. From blockchain consensus back to Byzantine consensus. *Future Gener. Comput. Syst.* **2020**, *107*, 760–769. [CrossRef]
43. Bartoletti, M.; Lande, S.; Podda, A.S. A Proof-of-Stake protocol for consensus on Bitcoin subchains. In Proceedings of the International Conference on Financial Cryptography and Data Security, Sliema, Malta, 3–7 April 2017.
44. Longo, R.; Podda, A.S.; Saia, R. Analysis of a Consensus Protocol for Extending Consistent Subchains on the Bitcoin Blockchain. *Computation* **2020**, *8*, 67. [CrossRef]
45. Labraoui, N.; Gueroui, M.; Sekhri, L. On-Off Attacks Mitigation against Trust Systems in Wireless Sensor Networks. In Proceedings of the IFIP International Conference on Computer Science and its Applications, Oran, Algeria, 8–10 May 2018.
46. VirtualBox. Available online: https://www.virtualbox.org/wiki/Downloads (accessed on 6 October 2020).
47. Ubuntu. Available online: https://releases.ubuntu.com/16.04.7/?_ga=2.44292775.2093950133.1601944028-206050969.1600952235 (accessed on 6 October 2020).
48. Hyperledger Fabric Samples. Available online: https://github.com/hyperledger/fabric-samples (accessed on 6 October 2020).
49. Docker. Available online: https://docs.docker.com/engine/install/ubuntu/ (accessed on 6 October 2020).
50. Docker-Compose. Available online: https://docs.docker.com/compose/install/ (accessed on 6 October 2020).
51. Go Programming Language. Available online: https://golang.org/dl/ (accessed on 6 October 2020).
52. Node.js. Available online: https://nodejs.org/dist/v12.16.2/ (accessed on 6 October 2020).
53. Hyperledger Explorer. Available online: https://github.com/hyperledger/blockchain-explorer (accessed on 6 October 2020).