

Article

Detection of Security Attacks in Industrial IoT Networks: A Blockchain and Machine Learning Approach

Henry Vargas , Carlos Lozano-Garzon * , Germán A. Montoya *  and Yezid Donoso 

Systems and Computing Engineering Department, Universidad de los Andes, Bogotá 111711, Colombia; hf.vargas10@uniandes.edu.co (H.V.); ydonoso@uniandes.edu.co (Y.D.)

* Correspondence: calozanog@uniandes.edu.co (C.L.-G.); ga.montoya44@uniandes.edu.co (G.A.M.); Tel.: +57-1-339-4949 (C.L.-G.)

Abstract: Internet of Things (IoT) networks have been integrated into industrial infrastructure schemes, positioning themselves as devices that communicate highly classified information for the most critical companies of world nations. Currently, and in order to look for alternatives to mitigate this risk, solutions based on Blockchain algorithms and Machine Learning techniques have been implemented separately with the aim of mitigating potential threats in IIoT networks. In this paper, we sought to integrate the previous solutions to create an integral protection mechanism for IoT device networks, which would allow the identification of threats, activate secure information transfer mechanisms, and it would be adapted to the computational capabilities of industrial IoT. The proposed solution achieved the proposed objectives and is presented as a viable mechanism for detecting and containing intruders in an IoT network. In some cases, it overcomes traditional detection mechanisms such as an IDS.

Keywords: blockchain; Industrial Internet of Things (IIoT); intrusion; machine learning



Citation: Vargas, H.; Lozano-Garzon, C.; Montoya, G.A.; Donoso, Y. Detection of Security Attacks in Industrial IoT Networks: A Blockchain and Machine Learning Approach. *Electronics* **2021**, *10*, 2662. <https://doi.org/10.3390/electronics10212662>

Academic Editor: George Hatzivasilis

Received: 10 September 2021

Accepted: 15 October 2021

Published: 30 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Internet of Things (IoT) has managed to permeate various fields from home automation to industries with critical infrastructure, achieving in the latter cases, complement or in some cases replace the operation of industrial control systems such as SCADA. The wide applicability of the IoT devices allows to technify fields of the industry whose technological maturity has been relatively low. Some of the most relevant examples are associated with oil exploitation and electricity production, both fields directly associated with the national cyberdefence [1,2].

While the change of paradigm to IoT has brought great benefits, it has also generated new challenges in terms of cybersecurity and cyber defense. For example, the design of technology and architectural solutions associated with new production techniques has had to consider the possibility of strengthening IoT communications and devices to ensure the availability, integrity, and availability of information deployed on these resources. In these cases, it is necessary to establish methodologies and procedures in charge of establishing countermeasures, safeguards, and continuities that allow for the control of data cybersecurity [1,3].

The need to protect and classify the information inside the IoT infrastructure by the owners of the technology has awakened in the attackers the investigation of new techniques and mechanisms to obtain, manipulate, extract or eliminate the information processed there. This situation is completely unacceptable for the critical cybernetic infrastructures of a nation, where the damage to a portion of the technology can generate catastrophic consequences in legal, economic, social, and even environmental terms, for the entire state [1,4].

One more reason that justifies the need to work more on the security of the IoT devices is the lack of baselines. The IoT market is functionally similar, but its vendors manufacture

these devices in very different ways. A 2017 IoT security survey [5] showed that 96% of businesses believe there should be regulations for IoT security. If there is no standardization or verification of device security, situations like the Mirai botnet can be easily replicated.

The IoT security issues have been addressed by seeking various defense strategies according to the layers of the IoT model: Perception, Network, and Application. For the perception layer, security strategies are based on hardware security. These security strategies are described as follows [6]:

- Authentication: Apply cryptographic hash algorithms to counter side channel attacks.
- Privacy: Using symmetric or asymmetric encryption algorithms can prevent unauthorized access to sensor information while it is being captured or sent to the next layer.
- Sensitive information: With a K-Anonymity approach it is possible to protect personal or location data.
- Risk analysis: Allows to identify new threats and helps to determine the best way to prevent information leaks and ensure appropriate security strategies.

The network layer is exposed to many types of attacks, due to the transmission medium used. The techniques of the layer can be condensed into the following three [6]:

- Authentication: A point-to-point authentication process can prevent unauthorized access to the IoT network.
- Privacy: Monitoring network activity against any type of intrusion and activating information integrity mechanisms ensures defense against such attacks.
- Routing Security: Implementing routing algorithms that guarantee the use of alternative paths can help the system to determine errors and have contingency mechanisms in case of intrusion.

Finally, the application layer or software provides access to the IoTs and needs to be secured. Its categories are as follows [6]:

- Authentication: Blocking access to non-self-hosting users with strong authentication mechanisms and the creation of user profiles that allow access segregation.
- Intrusion detection: Intrusion detection techniques can provide solutions to many security threats by generating alerts in case of suspicious activity.
- Information security: Use of encryption technologies to prevent information theft and other malicious activities.

In order to manage the risk associated with exposure to these threats, it is necessary to establish sufficient controls that focus on the need to ensure cybersecurity, especially the integrity of the information distributed over these critical networks.

This paper is organized as follows: Section 2 describes several aspects of security problems for IIoT technology, such as its security layers and different methods to assure security capabilities in IoT, namely, Blockchain and Machine Learning. In Section 3, we propose a Machine Learning technique as well as a Blockchain method to detect and mitigate several types of attacks. Then, in the Section 4, we show the outcomes related to comparisons between our approaches and a traditional method for detecting attacks. Finally, in Section 5, we present attractive conclusions about using Blockchain and Machine Learning as a mixture solution for reducing security attacks.

2. Related Works

As found in the previous section, the most common strategies for dealing with security risks in IoT networks are solutions that target authenticity, data privacy, and intrusion detection. These solutions could be completely satisfied by additional mechanisms as blockchain and machine learning. Our goal in this work was to integrate these two mechanisms to achieve a superior security solution in IoT.

2.1. Blockchain in the IoT Context

By using Blockchain technology to cover the security problems of IoT communications, the aim is to close the gap by ensuring: autonomy based on the decentralization

of transactions, trust configured from the Blockchain's own cryptography, and efficiency in the delivery of integrated information. In Blockchain each party can obtain the results defined in the rules, as well as the sanctions to the protocol; this is achieved by maintaining the following properties [7]:

- **Autonomy:** Transactions can be executed independently and automatically in a prescribed manner. Even the parties involved in the transaction are the ones who make the agreement instead of executing it. There is no need to worry about manipulation and corruption by an intermediary.
- **Trust:** Records are encrypted using somewhat symmetrical encryption rates. It is difficult for a hacker to break the codes and infiltrate the smart contract.
- **Accuracy:** Records are faster, cheaper and more accurate than traditional ones. They can avoid the human errors caused by filling out forms.

2.2. Machine Learning in the Context of IoT Cybersecurity

The IoT devices have specific characteristics in terms of communication, in which the misuse of these can trigger decentralized attacks on any type of infrastructure, even internal. Such challenges make designing a detection mechanism in IoT different from those known in conventional networks [8].

One of the objectives of Machine Learning is to enable technology to learn and make predictions based on information that has been explicitly programmed. While the use of Machine Learning has been popularized for the detection of anomalous behavior, the field of intruder detection has been relegated [8]. Traditionally, anomaly detection has been approached using statistical methods. However, the rise of machine learning techniques has opened new possibilities for the detection of outlier data due to the availability of large amounts of information to be used by machine learning models. In this sense, these machine learning models are an attractive new point of view to be implemented in IoT applications, where it is difficult to use static models [9].

Depending on the location strategies of the detection mechanism, intrusion detectors can be classified into centralized, distributed and hybrid. Centralized systems require a node with large computational capabilities that is responsible for analyzing the elements of the entire network; in distributed systems, detection is a task assigned to all network nodes; and finally, hybrids take advantage of each of the above schemes to ensure an appropriate level of efficiency and resource consumption at each node [8].

2.3. Related Projects

As a point of comparison, we found some projects, which although not aimed at a comprehensive security solution, developed one of the two security mechanisms chosen for this work.

The project developed by Jeon, Kim, and Kim [10], is focused on developing a lightweight Blockchain implementation for IoT systems, which takes advantage of the use of lightweight algorithms for encryption and validation of transactions between network nodes. To do this, they use algorithms based on cryptographic currencies such as Ethereum, MySQL servers, Smart Contract, and develop a platform for registration of IoT equipment in M2M networks, called Mobius IoT. This proposal actively maintains the protection afforded by Blockchain, requiring that the network is constantly processing encrypted information, contrary to our proposal, which activates the defenses only when malicious traffic is detected.

In the work [11], a confidence validation model of the nodes of a distributed network was developed based on mathematical strategies that calculate probabilities to determine the validity of a new node and classify it as either malignant or benign. They propose a lightweight source embedding scheme that continues to track the data packet by attaching the hash of the traversed node identification. The receiving node verifies the path of the data packet to ensure the integrity of the source data. Like the previous proposal, this project actively preserves the defenses regardless of whether an attack is in progress. This

forces solutions to be in constant use of resources and does not optimize defenses, contrary to the proposal presented here.

In the document [12], a Java-based module is established where a validation of network packets captured through applications such as Wireshark or TCPDump is done. This module validates whether there are abnormal packets in the network and if so, activates the system's defenses. The proposed defenses are alerting administrators, capturing information, closing malicious connections, among others. This proposal presents a threat detection model that alerts administrators to the presence of an attack. In comparison, our proposal seeks an integral solution where additional defense mechanisms such as Blockchain are activated when intruders are detected in the IIoT network.

In [8], the authors propose an active detection system in wireless IoT networks, based on Machine Learning and active learning techniques in order to determine possible intruders in the network. The training process is based on old standard datasets such as KDD99, which makes it difficult to learn and adapt to more modern IoT networks. This proposal seeks a comprehensive active learning system as our project, however, the data set they use to train the models is very outdated and it is not feasible to obtain it in real time because of the complexity of its calculations. Our solution takes data set directly from the protection target network.

The authors in [13] propose an intrusion detection model based on a genetic algorithm and a deep belief network. They use the NSL-KDD dataset for detecting four types of attacks: DoS, R2L, Probe and U2R. This paper, in comparison with our work, uses an old dataset difficult to be applicable to modern IoT networks and does not implement blockchain in their solution as an integrated mechanism for monitoring and securing IIoT networks.

In [14], an intrusion detection technique based on statistical flow features is proposed for protecting the network traffic of Internet of Things applications. The authors in this work use three machine learning techniques to detect malicious traffic events: Decision Tree, Naive Bayes and Artificial Neural Network (ANN). They use the same dataset employed by us, the UNSWNB15 dataset; however, they do not implement blockchain in their solution as an integrated mechanism for monitoring and securing IIoT networks.

A machine learning security framework for IoT systems is proposed in [15]. They built a dataset based on the NSL-KDD dataset and evaluated their proposal in a real smart building scenario. As we said in the previous related works, an old dataset may not be suitable for modern IoT networks. They use one-class SVM (Support Vector Machine) technique for detecting four types of attacks: DDoS, Probe, U2R and R2L. However, they do not use a blockchain approach for supervising IIoT networks.

The authors in [16] developed an algorithm for detecting denial-of-service (DoS) attacks using a deep-learning algorithm. They use three approaches for detecting DoS attacks: Random Forests, a Multilayer Perceptron and a Convolutional Neural Network. They use the same dataset employed by us, but they just aim to detect one attack (DoS) and do not integrate blockchain in their solution.

In [17], the authors use many traditional machine learning techniques such as Decision Tree, SVM, K-Means, Random Forest, among others for training and testing the NSL-KDD dataset, which is an old dataset that may not be suitable for modern IoT networks. In addition, they do not incorporate blockchain in their solution.

An artificially full-automated intrusion detection system for Fog security against cyber-attacks is proposed in [18]. They use multi-layered recurrent neural networks applied to the NSL-KDD dataset for detecting four types of attacks: DDoS, Probe, U2R and R2L. As we said previously, the NSL-KDD dataset is an old dataset that may be not suitable for modern IoT networks. In addition, they do not implement blockchain in their solution as an integrated mechanism for monitoring and securing IIoT networks.

In [19], the authors present an Intrusion Detection System (IDS) for detecting various types of malicious traffic, including Port Scanning, HTTP and SSH Brute Force. They create their own dataset based on a real scenario using Raspberry Pi devices. They use

traditional machine learning techniques such as Random Forest and Local Outlier Factor. However, they do not integrate their solution with blockchain for monitoring and securing IIoT networks.

The authors in [20] propose a model using a machine learning algorithm to detect and mitigate botnet-based distributed denial of service (DDoS) attacks in IoT networks. They use different machine learning algorithms such as K- Nearest Neighbour (KNN), Naive Bayes model and Multi-layer Perception Artificial Neural Network (MLP ANN). They use the same dataset employed by us, but they just aim to detect one attack (DoS) and do not integrate blockchain in their solution.

Finally, in [21], the authors propose an intrusion and cyber attacks traffic identification model using Machine Learning (ML) algorithms for IoT security analysis. The authors in this work use four machine learning techniques to detect malicious traffic events: Random Forest, Random Tree, Decision Tree, Naive Bayes and BayesNet. They use the same dataset employed by us, but they do not integrate blockchain in their solution.

The related projects can be summarized in Table 1, where their capabilities are observed as solutions that make use of blockchain or machine learning techniques; in the case of machine learning, if the model has been tested or trained with a recent dataset and its capability to be compared with established solutions such as IDS.

Table 1. Related works.

Proposal	Uses Blockchain	Uses Machine Learning	Recent Dataset	Comparable to Regular IDS?
[10]	Yes	No	N/A	No
[11]	Yes	No	N/A	No
[12]	No	Yes	N/A	Yes
[8]	No	Yes	1999	Yes
[13]	No	Yes	1999	No
[14]	No	Yes	2015	No
[15]	No	Yes	1999	No
[16]	No	Yes	2015	No
[17]	No	Yes	1999	No
[18]	No	Yes	1999	No
[19]	No	Yes	Own Dataset	No
[20]	No	Yes	2015	No
[21]	No	Yes	2015	No
Our proposal	Yes	Yes	2015 (Suited)	Yes

3. Solution Proposal

To define the architecture to be worked on, it was decided to use the architectural models proposed for the Industrial IoT by Kirupakar and Shalinie in [22], where Cloud, Edge and Network are contemplated. Through these three layers, it is sought to standardize the processing of these kinds of distributed networks.

In this architecture (See Figure 1), the collector node is in charge of centralizing the communications of the edge and is the point where the greatest computational load is performed. This node is connected individually to each sensor node, and its main functions are to store temporary data from the sensors, process it and deliver it to the next layer of the IIoT model, the Cloud. For this proposal, the processing of the Machine Learning and Blockchain algorithms are executed in the Collector node, requiring that this node concentrates all the data and has a higher processing capacity than the sensor nodes.

Regarding the sensor nodes, those are connected individually to the collector node and report to it the data acquired. For the case study, nodes have communication channels where they receive alerts to activate the defense mechanisms against the attacks described in the test scenarios. In this way, the computational load is not affected by the need to perform reprocessing or robust information encryption processes.

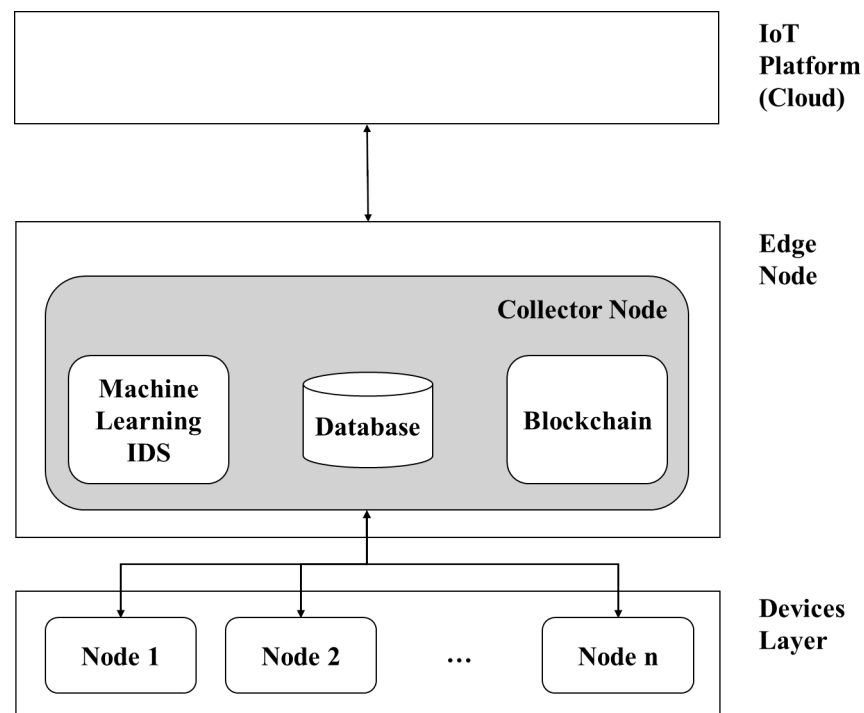


Figure 1. Proposed architecture.

3.1. Machine Learning and Blockchain Algorithm's Selection

Taking into account the low computational power of the equipment on the device layer in the deployments of IoT systems; in the algorithm selection phase, we prioritize those that will generate the least computational load for the nodes, without neglecting their main function.

Details of the selected algorithms and their implementation for both machine learning and blockchain can be found below.

3.1.1. Machine Learning Algorithm

For the selection of the Machine Learning algorithm, a comparative performance analysis was carried out among the existing algorithms, seeking to prioritize the execution time and the required computational effort. One of the design constraints for the selection of the algorithm is that it should be a supervised algorithm. This is in order to test especially the main objective of the project and obtain a functional solution, which can evolve according to the needs of a specific IIoT solution.

Taking into account the previous premise, and based on the fact that the attacks to the IIoT security are identified, the K nearest neighbors algorithm (KNN) was chosen. KNN is a supervised Machine Learning algorithm, which requires a computational calculation associated only with the distance between the nodes as can be seen in Algorithm 1 [23]. Also, this algorithm, besides being a lightweight solution is suitable for this kind of problem, since the identification of threats can be carried out considering packet parameters and network traces, this thanks to its properties as a universal classifier [24].

Algorithm 1: KNN Algorithm**Data:** Training Data Set, Test Data**Result:** Predicted class for each Test Data*Initialization of the KNN sets***begin**

- Load the training data.
- Initialize the value of K .

*Attack prediction for the test data***Offline process:****begin****while** *Are there points in the test data?* **do**

- Calculate the euclidean distance between test data point and each row of training data.
- Sort the calculated distances in ascending order based on distance values.
- Calculate the euclidean distance between test data point and each row of training data.
- Sort the calculated distances in ascending order based on distance values.
- Get top k rows from the sorted array.
- Get the most frequent class of these rows.
- Return the predicted type of class.

*Real time attack prediction for the captured packets***Online process:****begin****while** *Are there any captured packets?* **do**

- Calculate the euclidean distance for the packet.
- Sort the calculated distances in ascending order based on distance values.
- Count the number of occurrences of each class among the K nearest neighbors.
- Assign the packet to the corresponding traffic classification.

The complexity of the KNN algorithm is calculated according to the following formula: $O(n + k)$, where n is the total number of preloaded samples and k is the distance from the analyzed value (See Algorithm 1). In this sense, the algorithm is able to adapt to the needs of our solution.

The training of the machine learning model was carried out according to Figure 2; on the one hand, offline training was carried out with data from the proposed scenario attacks, as seen in the first steps of the scheme. On the other hand, and for the test scenarios, data from the attacks defined in real time were used.

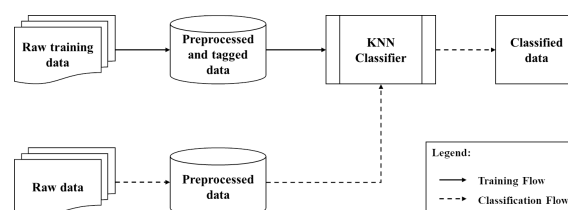


Figure 2. Classification Model.

For the training, first, the data is organized and classified according to the controlled attacks that are to be studied in the network; once the algorithm is trained with this data, it is tested on real-time traffic, and the packets are classified in defined time intervals, as observed in the second process.

The intrusion detection model is based on the identification of intruders by means of packet capture, focusing on the parameters of the TCP/IP model packets presented in Table 2. It is important to remark that these parameters were chosen for their importance in the target integrity attacks.

Table 2. Selected TCP/IP parameters.

Parameter	Relevance
Protocol	It clearly indicates the protocol of the transmission, helping to better identify the traffic of a target application.
Frame size	It helps to establish an average package size of a target application.
Source port	More clearly identifies the proprietary or application-specific port of a target application.
Destination Port	More clearly identifies the proprietary or application-specific port of a target application.
Epoch time	Allows a numerical value to be given to the time and date of transmission of a packet.
TTL Flags	Used by attackers to try to alter intercepted or altered packets. When an attacker tries to exploit open ports or services, the attacker uses specific flags.
Window size	The size of the window gives an indication of the communication established between two hosts.
Sequence number	Used by attackers to request the retransmission of packages.

3.1.2. Blockchain Algorithm

For the implementation of Blockchain, a lightweight solution was chosen [25] that respected the principles of the technology, and was coupled with the IoT technology. In this sense, a solution was defined where the block chain is located in the collector of the Edge and, through the previous registration of keys, this communicates safely with the nodes of the IIoT network. By centralizing the computational effort in the collector, the load on the nodes is reduced, thus avoiding the effect on the regular computational activities of the business served by the IIoT network.

The proposed algorithm (see Algorithm 2) makes use of SHA-256 and AES technologies to generate a centralized block chain in the collector, together with the communication channels previously established with the nodes [25]. The nodes have 128-bit keys with which they encrypt the communication in the network, and the collector stores it in the block chain. This principle was used maintaining the performance purpose of the architecture and preserving the functions of the edge without reloading the sensor nodes.

Algorithm 2: Blockchain Algorithm with AES

Data: Edge Node Encryption Key Set (K), Current transaction time, Sensor information to be recorded

Result: Secure storage of Edge sensor information

Initialization of the chain

begin

- Creation of the genesis block with the current time UTC located at position 0.

Capturing communication packets from the node

begin

- Search for the key corresponding to the node.
- Decryption of the information received with the pre-shared key.

Creating a new block

begin

- With the data in plain text, a new block is created in the string with the hash of the previous block, with the data, the hash of the data, the UTC time, and it is recorded in the i counter of the string.

Verification of chain integrity

begin

- The correct inclusion of the block in the chain is validated.

begin

- Check if each block is assigned its correct position in the chain.
- Check if the block has the hash of the immediately preceding block.
- Check if the date holds a correct hash.
- Check if the timestamp is consistent.

4. Experimental Results

In this section, we present the test scenario to be evaluated, namely, the scenario configuration, the launching attacks, and the blockchain and machine learning algorithms setup. This scenario is tested to evaluate the attack detection performance of our proposal against a traditional security solution, an Intrusion Detection System (IDS).

4.1. Scenario Configuration

To test the preservation of data integrity at the network edge, two representative attacks were chosen: a packet injection with false data (a fuzzing attack) and a denial of service attack (DoS) attack from a malicious host on the sensor node network. These controlled attacks were carried out in the test scenario implemented according to Figure 3.

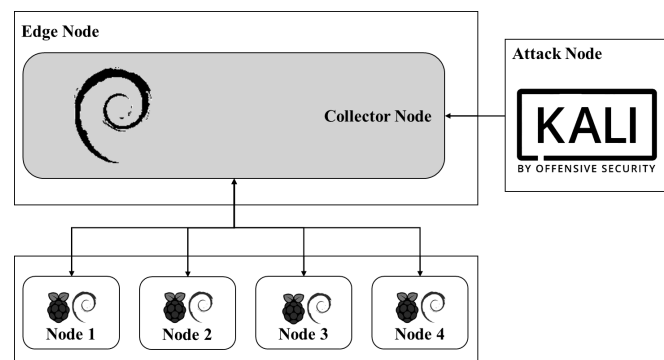


Figure 3. Test scenario.

The nodes of the proposed test scenario were configured using Debian as the operating system for the collector, Raspbian for the nodes, and Kali Linux for the malicious node. This scheme was virtualized under the VirtualBox software with a virtual network card, where the communications between the parties took place. In addition, we used Python 3, HPING3, TShark, and PyCharm.

4.2. Attacks Configuration

We selected the UNSWNB15 dataset to evaluate our proposed scheme. This dataset was generated by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [26], which corresponds to a new generation of industrial IoT (IIoT) dataset in order to evaluate and calibrate the performance of artificial intelligence/machine learning cybersecurity applications. This dataset consists of a total of 49 features and nine types of attacks [26]. These attacks include fuzzers, backdoors, analysis, reconnaissance, exploits, generic, DoS, shellcode, and worms (see Table 3). The total number of features were reduced to the features described in Table 2, that is, the following nine features: protocol, frame size, source port, destination port, epoch time, TTL, flags, window size, and sequence number [26]. This reduction was necessary to properly adapt the original dataset (the UNSWNB15 dataset) to our solution architecture explained in Section 3.1.1.

Table 3. List of attacks.

Attack Type	Amount
Normal	2,218,764
Fuzzers	24,246
Analysis	2677
Backdoors	2329
DoS	16,353
Exploits	44,525
Generic	215,481
Reconnaissance	13,987
Shellcode	1511
Worms	174

Based on the threat collection developed by the OWASP IoT group for 2018 [27], where it is established that the three most relevant threats to the IoT model are weak passwords, network threats, and insecure interfaces. Two of the most common attacks on this kind of networks were chosen: the spoofing attacks (related to insecure interfaces) and the denial of service attacks (concerning the network threats). In this sense, from the UNSWNB15 dataset, we have selected the DoS and Fuzzers attacks to represent these two of the most common attacks (see Table 3).

4.3. K-Nearest Neighbors Algorithm Setup and Results

The objective of this algorithm setup was to find the correct values for the algorithm, in order to identify, in real time, that the network is under attack. This involves identifying the malicious packets and, then, generating an alert to the nodes.

For this reason, three proof scenarios were defined: in the first, only the traces obtained from the fuzzers attack were used, in the second we used the traces generated by the denial of services attack, and for the third scenario, we combined traces from both attacks.

The tuning of the selected Machine Learning algorithm was done by adjusting the following variables:

- Number of neighbors: The KNN algorithm is based on calculating the closest distance between the data, that is, it categorizes new data according to its closeness to the others. If this value increases, it takes a greater amount of more distant elements to evaluate.
- Amount of traces: The amount of traces affects the learning process and load of the algorithm.

For each proof scenario, both the efficiency of the model and the loading time were measured. For the first performance indicator, the model was trained with 80% of the traces and the remaining were used to measure the effectiveness of detection; for the second, the time taken by the model to preload the data was calculated. Many values of the number of neighbors and traces were considered to find the best parameters configuration in order to achieve the best performance in terms of accuracy. Table 4 shows the results obtained in these tests.

Table 4. Machine learning Results.

Attack Type	Amount of Traces	Number of Neighbors	Loading Time	Accuracy
DoS	100,000	316	88.01 s	95%
DoS	50,000	224	15.75 s	97%
DoS	33,333	183	8.29 s	95%
Fuzzers	100,000	1000	133.58 s	62%
Fuzzers	100,000	2000	188.12 s	78%
Fuzzers	100,000	5000	373.45 s	99%
Fuzzers	100,000	316	85.66 s	62%
Fuzzers	50,000	224	14.64 s	62%
Fuzzers	33,333	183	8.75 s	62%
Fuzzers	20,000	200	9.44 s	62%
Fuzzers	20,000	1000	16.77 s	82%
Fuzzers	20,000	10,000	100.55 s	82%
DoS and Fuzzers	120,000	5000	339.59 s	92%
DoS and Fuzzers	120,000	7500	560.29 s	82%
DoS and Fuzzers	120,000	346	123.85 s	62%
DoS and Fuzzers	60,000	245	22.2 s	62%
DoS and Fuzzers	40,000	200	11.98 s	62%

Notice that, in Table 4, “DoS” indicates traces with normal and DoS traffic, “Fuzzers” indicates traces with normal and Fuzzers traffic, and “DoS and Fuzzers” indicates traces with normal, DoS and Fuzzers traffic. These traces were used for training and testing our KNN algorithm to obtain the best accuracy for detecting these attacks.

Many other configurations were tested (hundreds of them), but for practical reasons, we have not included more results. Anyway, the values obtained in Table 4 were the more representative results in order to select the best parameters configuration. In this sense, the best accuracy achieved (97%) for “DoS” was for 50,000 traces and 224 neighbors. The best accuracy achieved (99%) for “Fuzzers” was for 100,000 traces and 5000 neighbors.

Finally, the best accuracy achieved (92%) for “DoS and Fuzzers” was for 120,000 traces and 5000 neighbors.

As a result, it was found that for each of the attack cases tested, the effectiveness of the classification model was greater than 90%. This was achieved by adjusting the parameters “number of traces” and “number of neighbours”. The number of neighbors was based on the square root of the number of traces and decreased as the effectiveness of detection increased [28]. For each of these best cases described previously, we have tested them many additional times for obtaining statistical results. For example, for the best configuration of “DoS”, we have tested it 25 times for 50,000 traces and 224 neighbors in order to determine the confidence interval of the 97% value for accuracy obtained previously. These results are shown in Table 5. In addition, many other metrics are shown as results such as Precision, Recall, and F1-score. For most of these metrics, we have obtained a performance above 90% and upper and lower intervals were so close, showing a very good efficiency for detecting DoS and Fuzzers attacks under the best parameters configuration achieved previously.

Table 5. Confidence interval results.

	Best Scenario Normal-DoS	Best Scenario Normal-Fuzzers	Best Scenario Normal-DoS-Fuzzers
Amount of traces	50,000	100,000	120,000
Number of neighbors	224	5000	5000
Accuracy (avg)	0.9636	0.99	0.9208
Precision normal (avg)	0.97	1.00	0.97
Precision DoS (avg)	0.9368	N/A	0.8096
Precision Fuzzers (avg)	N/A	0.9432	0.8336
Recall normal (avg)	0.986	0.986	0.9868
Recall DoS (avg)	0.8808	N/A	0.7732
Recall Fuzzers (avg)	N/A	0.998	0.8192
F1-score normal (avg)	0.9796	0.99	0.98
F1-score DoS (avg)	0.9068	N/A	0.7912
F1-score Fuzzers (avg)	N/A	0.97	0.826
Percentage of normal traffic	80%	80%	68%
Percentage of DoS traffic	20%	N/A	16%
Percentage of Fuzzers traffic	N/A	20%	16%
Number of samples	25	25	25
Confidence interval for accuracy (95%)	0.00192	1.7×10^{-16}	0.0011
Standard Deviation	0.00489	4.5×10^{-16}	0.0028
Lower interval	0.9616	0.99	0.9197
Upper interval	0.9655	0.99	0.9219

4.4. Blockchain Algorithm Setup and Results

Notice that through the blockchain algorithm we want to initiate a process of encryption and verification of the integrity of the transactions carried out by the nodes of the IoT network, when the network is under attack. The tuning of the Blockchain algorithm was done by adjusting the following variables:

- Total number of blocks: This allows us to determine until what point the algorithm can support transactions without wasting time between transactions and verifying the integrity of the chain.
- Number of simultaneous nodes: This variable allows us to know if it is possible to scale the model to larger and more distributed IIoT networks

The model was tested forcing the registration of the values in the Blockchain by modifying the time between the registration of transactions, in order to generate more hashes of constant value and try to overload the algorithm. For these hashes, the correct position in the chain and the timestamp were verified. In Tables 6 and 7 we shown the different values of total and simultaneous blocks tested.

Table 6. Blockchain blocks results.

Number of Total Blocks	Error Rate	Maximum Verification Time
1	0%	7.99×10^{-6} s
2	0%	9.99×10^{-6} s
4	0%	1.59×10^{-5} s
8	0%	2.89×10^{-5} s
16	0%	5.39×10^{-5} s
32	0%	0.0001 s
64	0%	0.0002 s
128	0%	0.00042 s
256	0%	0.00087 s
512	0%	0.0023 s
1024	0%	0.0081 s
2048	0%	0.010 s
4096	0%	0.014 s
8192	0%	0.055 s
16,384	0%	0.11 s
32,768	0%	0.30 s

Table 7. Blockchain node results.

Number of Concurrent Nodes	Error Rate	Maximum Verification Time
1	0%	9.99×10^{-6} s
2	0%	1.5×10^{-5} s
4	0%	2.10×10^{-5} s
8	0%	3.5×10^{-5} s
16	0%	5.90×10^{-5} s
32	0%	0.0001 s
64	0%	0.0002 s
128	0%	0.0004 s
256	0%	0.0008 s
512	0%	0.002 s
1024	0%	0.0071 s
2048	0%	0.012 s
4096	0%	0.033 s
8192	0%	0.0819 s
16,384	0%	0.14 s
32,768	0%	0.515 s

As can be seen in Tables 6 and 7, the proposed blockchain solution supports a high load and handles excellent times. Increasing the number of concurrent nodes and/or overloading the chain with multiple transactions does not affect the response time of the solution, thus guaranteeing the scalability of our proposal.

4.5. Comparative Analysis of the Results Obtained against an IDS

An Intrusion Detection System (IDS) is a software application that attempts to identify malicious network activity. This tool is being widely used as one of the solution mechanisms

to the problem raised in this work. In order to compare the effectiveness of our proposal against this kind of solution, we selected a Snort implementation, specifically EasyIDS [29].

As mentioned above, once the detected real-time traffic is analyzed by our machine learning algorithm, it can generate two possible results. If the traffic is classified as normal traffic, the nodes make transparent communication of the information associated with their sensor devices, as shown in Figure 4. On the contrary, if the captured traffic is classified as malicious traffic, the collector sends an alert to the nodes and requires them to make use of the pre-shared keys to secure the transmission of the information from their nodes, as can be seen in Figure 5.

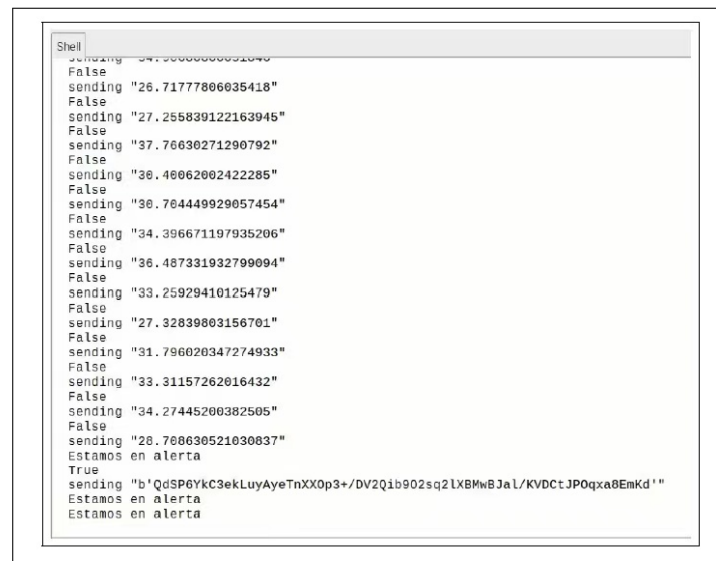


```

Run: server x
/usr/bin/python3.6 /home/henry/Documents/TESIS/Red/server.py
./Trazas/preprocessed_100inj_labeled.csv
True
30
279
True
Accuracy of K-NN classifier on training set: 0.86
3.278794
starting up on 0.0.0.0 port 60000
Capturing on 'vboxnet0'
420 packets captured
File created: preprocessed_captured.csv
{'normal': 420}
Capturing on 'vboxnet0'

```

Figure 4. Normal Capture.



```

Shell
False
sending "26.71777806035418"
False
sending "27.255839122163945"
False
sending "37.76630271290792"
False
sending "36.40062002422285"
False
sending "36.764449929057454"
False
sending "34.396671197935206"
False
sending "36.487331932799094"
False
sending "33.25929410125479"
False
sending "27.32839003156701"
False
sending "31.796020347274933"
False
sending "33.31157262016432"
False
sending "34.27445200382505"
False
sending "28.700630521030837"
Estamos en alerta
True
sending "b'QdSP6YkC3ekLuyAyeTnXXOp3+/DV2Q1b902sq2LXBmWBJaL/KVDCtJP0qxa8EmKd'"
Estamos en alerta
Estamos en alerta

```

Figure 5. Ciphered transmission.

In order to have a baseline for comparison, we executed the attack scripts were executed from the malicious node (Kali Linux) and was installed and configured the IDS on the edge node. As a result, for the DoS denial attacks the IDS was able to identify 100% of the malicious traffic generated as can be seen in Figure 6; whilst for the packet injection attack, by means of a Fuzzing attack, the IDS was not able to recognize the injected traffic as is shown in Figure 7.

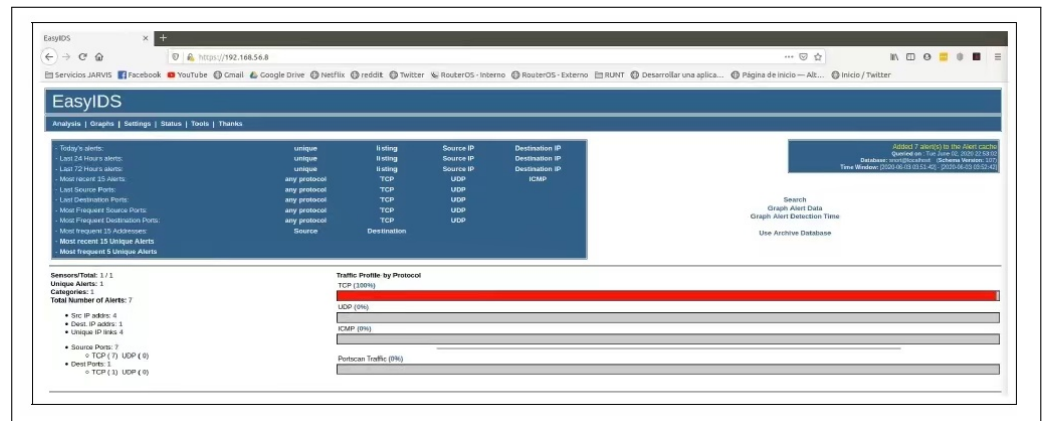


Figure 6. Snort DoS attack result.

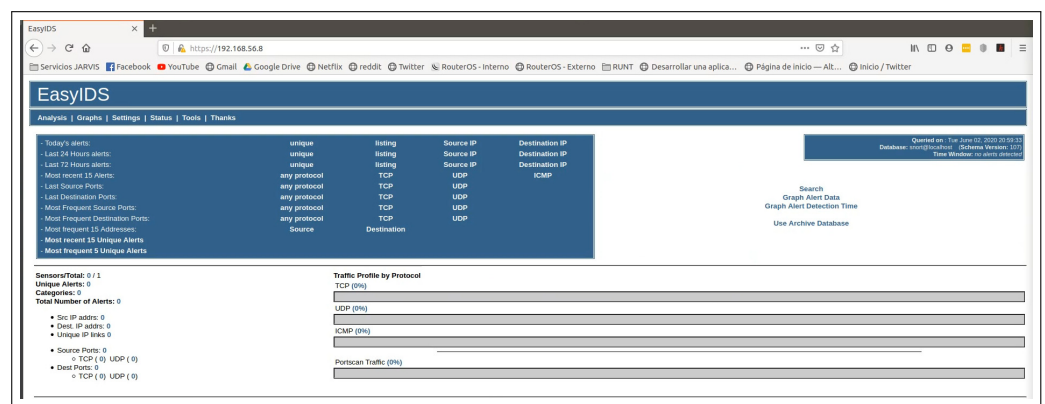


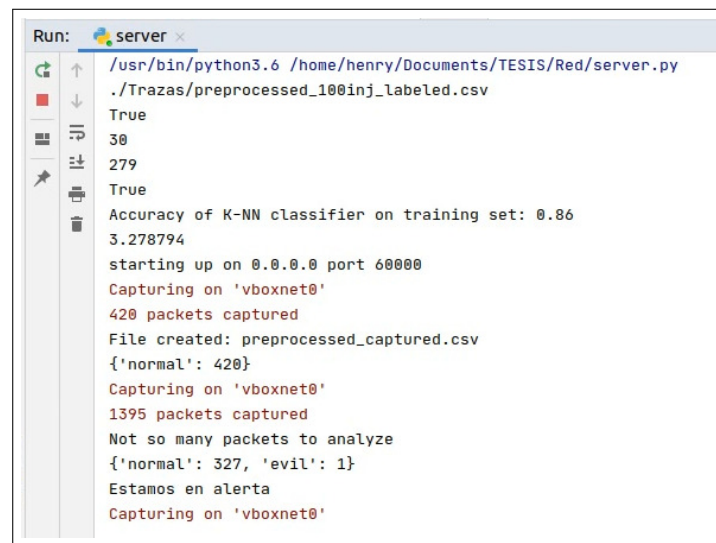
Figure 7. Snort Injection Attack result.

Subsequently, the same attack scripts were executed (once again using the Kali Linux node) and the defense mechanism proposed in this article was activated. The following results were obtained: for the Denial of Services attack, our algorithm was able to detect 100% of the malicious traffic, just like the IDS (See Figure 8). On the other hand, for the case of the packet injection attack (Fuzzing attack), our machine learning algorithm was able to identify the malicious traffic and send alerts to the nodes which began to transmit in encrypted mode, as you can see in Figure 9.

```

Run: server x
/usr/bin/python3.6 /home/henry/Documents/TESIS/Red/server.py
./Trazas/preprocessed_100DoSLabel.csv
True
30
214
True
Accuracy of K-NN classifier on training set: 0.64
4.704562
starting up on 0.0.0.0 port 60000
Capturing on 'vboxnet0'
1650 packets captured
File created: preprocessed_captured.csv
{'normal': 1650}
Capturing on 'vboxnet0'
167293 packets captured
File created: preprocessed_captured.csv
{'normal': 56412, 'dos': 110881}
Estamos en alerta
    
```

Figure 8. Machine Learning DoS attack result.



```

Run: server x
/usr/bin/python3.6 /home/henry/Documents/TESIS/Red/server.py
./Trazas/preprocessed_100inj_labeled.csv
True
30
279
True
Accuracy of K-NN classifier on training set: 0.86
3.278794
starting up on 0.0.0.0 port 60000
Capturing on 'vboxnet0'
420 packets captured
File created: preprocessed_captured.csv
{'normal': 420}
Capturing on 'vboxnet0'
1395 packets captured
Not so many packets to analyze
{'normal': 327, 'evil': 1}
Estamos en alerta
Capturing on 'vboxnet0'

```

Figure 9. Machine Learning Injection attack result.

As can be seen, Snort's chances against a spoofing attack like the one proposed are null; however, they can identify attacks such as denial of service. On the contrary, both types of attacks are identified satisfactorily with the Machine Learning algorithm developed.

5. Conclusions

Combining machine learning and blockchain strategies allowed us to establish a strategy for identifying and mitigating attackers in real time in an IIoT network. Similarly, it reduces the computational efforts of the network nodes, as long as the network is free of intruders and no additional encryption processes are being executed. The proposed solutions found in the state of art tend to focus on independent security solutions and do not contemplate an integrated mechanism for monitoring and securing IIoT networks. This work proposes an integrated system for the detection and containment of intruders on the Edge.

The model developed is capable of identifying diverse attacks under the premise that they must be reproduced and identified previously to train the model. The model is versatile and responds very well to load and scalability increases, where performance fluctuation is very low.

The algorithm proved to be superior in some aspects to traditional solutions such as IDS in the identification of more sophisticated attacks. Thus, contributing to the traditional security of IIoT devices, its real-time identification component is crucial to stop attacks on critical infrastructure and; therefore, avoid significant impact on national security. Taking into account its versatility, it is possible to carry out real implementations that can identify intruders in real time and contain them with little computational effort in networks as small and not very scalable as IoT networks.

Author Contributions: All authors contributed to the fulfillment of this paper. In detail, the paper conceptualization and methodology were accomplished by all the authors. The original draft preparation, the software tests, and the validations were carried out by H.V., while the technique review was accomplished by C.L.-G. and G.A.M. The final review and editing phases were completed by C.L.-G., G.A.M. and Y.D. All authors have read and agreed to the published version of the manuscript.

Funding: All authors acknowledge financial provided by the Vice Presidency for Research & Creation publication fund at the Universidad de los Andes, Bogotá, Colombia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original UNSWNB15 dataset [26] used to analyze attacks in IoT networks can be found at <https://research.unsw.edu.au/projects/unsw-nb15-dataset> (accessed on 1 October 2020).

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

References

1. Mcginthy, J.M.; Michaels, A.J. Secure Industrial Internet of Things Critical Infrastructure Node Design. *IEEE Internet Things J.* **2019**, *6*, 8021–8037. [CrossRef]
2. Lee, S.K.; Bae, M.; Kim, H. Future of IoT Networks: A Survey. *Appl. Sci.* **2017**, *71*, 1072. [CrossRef]
3. Lee, I.; Bae, M.; Kim, H. Internet of Things (IoT) Cybersecurity: Literature Review and IoT Cyber Risk Management. *Future Internet* **2020**, *12*, 157. [CrossRef]
4. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and Security: Challenges and Solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
5. The State of IoT Security. October 2017. Available online: <https://www.gemalto.com/m2m/documents/iot-security-report> (accessed on 5 September 2020).
6. Abdullah, A.; Hamad, R.; Abdulrahman, M.; Moala, H.; Elkhediri, S. CyberSecurity: A Review of Internet of Things (IoT) Security Issues, Challenges and Techniques. In Proceedings of the 2nd International Conference on Computer Applications and Information Security, ICCAIS 2019, Riyadh, Saudi Arabia, 1–3 May 2019; pp. 1–6. [CrossRef]
7. Yu, Y.; Li, Y.; Tian, J.; Liu, J. Blockchain-Based Solutions to Security and Privacy Issues in the Internet of Things. *IEEE Wirel. Commun.* **2018**, *25*, 12–18. [CrossRef]
8. Yang, K.; Ren, J.; Zhu, Y.; Zhang, W. Active Learning for Wireless IoT Intrusion Detection. *IEEE Wirel. Commun.* **2018**, *25*, 19–25. [CrossRef]
9. Erhan, L.; Ndubuaku, M.; Mauro, M.D.; Song, W.; Chen, M.; Fortino, G.; Bagdasar, O.; Liotta, A. Smart Anomaly Detection in Sensor Systems: A Multi-Perspective Review. *Inf. Fusion* **2021**, *67*, 64–79. [CrossRef]
10. Jeon, J.H.; Kim, K.H.; Kim, J.H. Blockchain based data security enhanced IoT server platform. In Proceedings of the International Conference on Information Networking, Chiang Mai, Thailand, 10–12 January 2018; pp. 941–944. [CrossRef]
11. Suhail, S.; Hong, C.S.; Lodhi, M.A.; Zafar, F.; Khan, A.; Bashir, F. Data trustworthiness in IoT. In Proceedings of the International Conference on Information Networking, Chiang Mai, Thailand, 10–12 January 2018; pp. 414–419. [CrossRef]
12. Kshirsagar, D.D.; Sale, S.S.; Tagad, D.K.; Khandagale, G. Network Intrusion Detection based on attack pattern. In Proceedings of the ICECT 2011-2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, India, 8–10 April 2011; pp. 283–286. [CrossRef]
13. Zhang, Y.; Li, P.; Wang, X. Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network. *IEEE Access* **2019**, *7*, 31711–31722. [CrossRef]
14. Moustafa, N.; Turnbull, B.; Choo, K.R. An Ensemble Intrusion Detection Technique based on proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet Things J.* **2018**, *6*, 4815–4830. [CrossRef]
15. Bagaa, M.; Taleb, T.; Bernal, J.; Skarmeta, A. A machine learning Security Framework for Iot Systems. *IEEE Access* **2020**, *8*, 114066–114077. [CrossRef]
16. Susilo, B.; Sari, R. Intrusion Detection in IoT Networks Using Deep Learning Algorithm. *Information* **2020**, *11*, 279. [CrossRef]
17. Liu, J.; Kantarci, B.; Adams, C. Machine Learning-Driven Intrusion Detection for Contiki-NG-Based IoT Networks Exposed to NSL-KDD Dataset. In Proceedings of the ACM Workshop on Wireless Security and Machine Learning, Linz, Austria, 13 July 2020. [CrossRef]
18. Almiani, M.; AbuGhazleh, B.; Al-Rahayfeh, A.; Atiewi, S.; Razaque, A. Deep Recurrent Neural Network For IoT Intrusion Detection System. *Sci. Direct Simul. Model. Pract. Theory* **2020**, *101*, 102031. [CrossRef]
19. Eskandari, M.; Janjua, Z.H.; Vecchio, M.; Antonelli, F. Passban IDS: An Intelligent Anomaly Based Intrusion Detection System for IoT Edge Devices. *IEEE Internet Things J.* **2020**, *7*, 6882–6897. [CrossRef]
20. Pokhrel, S.; Abbas, R.; Aryal, B. IoT Security: Botnet detection in IoT using Machine learning. *arXiv* **2021**, arXiv:2104.02231.
21. Shafiq, M.; Tian, Z.; Sun, Y.; Du, X.; Guizani, M. Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city. *Future Gener. Comput. Syst.* **2020**, *107*, 433–442. [CrossRef]
22. Kirupakar, J.; Shalinie, S.M. Situation aware intrusion detection system design for industrial IoT gateways. In Proceedings of the 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), Gurugram, India, 21–23 February 2019. [CrossRef]
23. Ray, S. A Quick Review of machine learning Algorithms. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 35–39. [CrossRef]
24. Ponmaniraj, S.; Rashmi, R.; Anand, M.V. IDS Based Network Security Architecture with TCP/IP Parameters using Machine Learning. In Proceedings of the 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 28–29 September 2018; pp. 111–114. [CrossRef]
25. Au, E. Building a Minimal Blockchain in Python: Understanding Blockchain by Coding. 2019. Available online: <https://towardsdatascience.com/building-a-minimal-blockchain-in-python-4f2e9934101d> (accessed on 18 July 2020).

26. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the IEEE Military Communications and Information Systems Conference, Canberra, ACT, Australia, 10–12 November 2015. [CrossRef]
27. OWASP IoT Security. 2018 OWASP IoT Top 10. Available online: <https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf> (accessed on 23 June 2020).
28. Zhan, Y.; Chen, H.; Zhang, G.C. An optimization algorithm of K-NN classification. In Proceedings of the 2006 International Conference on Machine Learning and Cybernetics, Dalian, China, 13–16 August 2006; pp. 2246–2251. [CrossRef]
29. Skynet Solutions “EasyIDS 0.2”. 2007. Available online: <https://skynetsolutions.com/easyids-0-2-released/> (accessed on 11 June 2020).