future internet

MDPI

Article

Collaborative Blockchain-Based Detection of Distributed Denial of Service Attacks Based on Internet of Things Botnets

Georgios Spathoulas [1,*], Nikolaos Giachoudis [2], Georgios-Paraskevas Damiris [2] and Georgios Theodoridis [2]

[1] Department of Information Security and Communication Technology, Norwegian University of Science and Technology, 2815 Gjovik, Norway

[2] Department of Computer Science and Biomedical Informatics, University of Thessaly, 35131 Lamia, Greece, ngiachou@uth.gr (N.G.); gdamiris@uth.gr (G.-P.D.); gtheodoridis@uth.gr (G.T.)

* Correspondence: georgios.spathoulas@ntnu.no

Received: 28 August 2019; Accepted: 23 October 2019; Published: 25 October 2019

Abstract: Internet of Things is one of the most significant latest developments in computer science. It is common for modern computing infrastructures to partially consist of numerous low power devices that are characterized by high diversity in both hardware and software. Existing security models, approaches and solutions are not able to sufficiently protect such systems. In this paper we propose the use of lightweight agents installed at multiple internet of things (IoT) installations (e.g., smart-homes), in order to collaboratively detect distributed denial of service (DDoS) attacks conducted by the use of IoT devices botnets. Specifically, agents exchange outbound traffic information in order to identify possible victims of DDoS attacks. This information exchange is governed by a blockchain smart contract, that ensures the integrity of both the procedure and the information. A simulation of the operation of the proposed methodology has been conducted in order to evaluate both its detection efficiency and its resilience against malicious agents that aim to falsify results.

Keywords: IoT; DDoS; agents; botnets; security; detection; blockchain

1. Introduction

Internet of Things is one of the most evident advancements in computing through the last decade. It has enabled the embedding of computing power and network communication capabilities into a variety of devices, which can provide new kinds of services to users. Multiple domains such as health-care, industry, physical security, farming and even home automation have been revolutionized due to the use of internet of things (IoT) devices [1,2].

This shift in the form of computing has triggered significant cyber security implications. The task of protecting the internet from cyber-attacks is continuously becoming more challenging. One of the main reasons for this is the fact that the number of connected devices has escalated, while the diversity of their type has made it impossible to secure them all. IoT devices are characterized by variety regarding hardware and software, insecure design, lack of updates, and lack of interaction by users.

Both the designing procedure and the building process of IoT devices are usually approached on a minimal cost basis. It is possible to design an IoT device in such a way that it will be secure, but then the resulting product will end up being exceedingly costly from a business perspective. Additionally, it is not common for manufacturers to actively provide firmware updates for IoT devices, even in the case of significant security vulnerabilities. Furthermore, the interaction of users with IoT devices is

limited and usually nonexistent after initial installation. Users tend not to change the default settings, including the authentication credentials, and not to update the firmware of the device as this usually is a complex procedure for any novice user.

There are many problems regarding the security of IoT devices [3–6]. Some of those are:

- The installed firmware is usually out of date. Manufacturers do not provide the required support for their devices. Even if they do so, users do not systematically update the firmware of already deployed devices.
- Sophisticated security mechanisms are typically not embedded into devices. The cost of IoT devices is low and it is impossible for a manufacturer to produce secure IoT devices that will be competitive in the end-user market.
- Users usually do not interact with devices. They do not configure the device's settings and they never change the default authentication credentials. This renders IoT devices vulnerable to malicious users that want to take control of them.
- Devices are not built from scratch, even by large manufacturers. Reuse of parts produced by unknown vendors, which do not take into account basic security requirements, is a common approach between manufacturers.

The growth of the use of IoT devices, along with the aforementioned issues, have formed an ideal environment for malicious users to commit various cyber-attacks. These attacks may violate users' privacy, aim for economical profit, or just aim to disrupt the normal operation of underlying systems and services. Some of the most significant IoT related cyber-attacks, observed recently, were distributed denial of service (DDoS) attacks [7]. The rise of IoT computing has practically provided the means to enlarge existing DDoS attacks, which were previously committed through the use of normal desktop computers or servers.

During September of 2016, a DDoS attack was launched against Brian Krebs' website [8], the magnitude of which was larger than that of any other previous similar attempt. The attack was conducted by a botnet, composed of approximately one million devices, most of which were Internet protocol (IP) cameras. In a later stage, the source code of Mirai [9,10], the botnet that was employed during the attack against the Krebs website, was released and the simple principle upon which Mirai is based was revealed. It scans the internet for accessible devices protected by factory-default usernames and passwords and it takes control of these devices, in order to form the botnet network. The Mirai attack has brought into daylight the important security implications of IoT computing and the fact that insecure devices, with default credentials, are exposed to the internet in masses [11,12].

In this paper, an agent based approach to detect DDoS attacks that are executed through the use of multiple infected IoT devices is presented. In such attacks, infected devices form a botnet that sends a high load of traffic to a victim host, in order to set the latter unable to respond to normal requests. The contribution of the paper is a protocol that enables multiple agents that are installed on gateways of different sites of IoT installations, to collaborate on detecting DDoS attacks. These agents collaborate through exchanging traffic information, while at the same time they utilize a blockchain infrastructure, in order to securely reach a consensus about the information metrics that are locally calculated at the gateways of the system.

The present paper is an extended version of the paper entitled "Collaborative Agent-based Detection of DDoS IoT Botnets" by Giachoudis et al. [13]. The main contribution of the extended version is a blockchain based consensus mechanism, in order for agents to be able to collaboratively detect DDoS attacks. Specifically, Section 5 and Sections 2.2, 6.2 are completely new in the extended version of the paper. Section 2.2 presents state of the art with respect to the use of blockchain technology regarding detection and mitigation of DDoS attacks. Section 5 discusses how blockchain technology is employed in order to enhance the algorithm presented in our previous paper [13]. Section 6.2 presents the results regarding the probability of a group of malicious agents being able to create integrity

problems for the detection results produced by the proposed scheme. Additionally, Sections 1 and 7 have been updated to reflect the extended content of the paper.

The rest of the paper is structured as follows: Section 2 presents related work, Section 3 presents the proposed system, the implementation of which is then described in Section 4. Section 5 describes the consensus protocol and Section 6 presents the simulation conducted and the obtained results, while Section 7 discusses relevant conclusions.

2. Related Work

2.1. DDoS in IoT Environments

A lot of researchers have recently focused on the domain of DDoS attacks executed against or through IoT devices. For example, a novice attack that reflects the applicability of denial of service attacks in IoT is presented in [14]. Specifically, a DDoS attack method for IoT system is discussed and implemented, while its effects are evaluated by monitoring various metrics such as CPU and memory utilization.

Multiple simulation scenarios are presented in [15], to evaluate the effects of using wireless sensor networks for committing amplified reflection DDoS attacks. The results show that conducting a successful real world DDoS attack by using wireless sensor networks, requires employing a significantly large number of infected installations.

In [16] the capability of consumer IoT devices to participate in reflective DDoS attacks is discussed. Authors evaluate specific household IoT devices available on the market in terms of reflective capability, amplification factor, duration, and intensity rate for DDoS attacks.

There are also some recent research efforts regarding detection and prevention of DDoS attacks against and/or through IoT installations. In [17] an agent based system is proposed, in order to detect DDoS attacks in IoT devices. Their short technical analysis uses a blacklisting/white-listing approach, to prevent the network from being taken down. Their experimental analysis shows that they have only run minimal simulations with less than 100 nodes, which is not representative of real world networks.

An innovative approach called AntibIoTic, which is a worm-like software that infects devices in the same way that real bot-net software does, is presented in [18]. After infecting the device, AntibIoTic protects it or notifies the owner about anomalies regarding its functionality. While the idea is interesting, the paper lacks any technical justification.

In [19] a lightweight defense mechanism that is based on utilizing devices themselves is discussed. The presented approach is interesting, but it requires devices that are able to protect themselves or the network from attacks. In practice this limits the applicability of such a solution. In [20] authors point out that IoT devices' exponential growth will enforce the use of IPv6 protocol and they discuss methods to detect attacks against the most important vulnerabilities of the specific protocol. The authors present an approach that can detect specific attacks such as wormhole and flooding attack in IPv6 sensor network environment.

The use of leaning automata, in order to set up a method for detecting DDoS attack is proposed in [21]. Their approach is based on a Service-oriented architecture (SOA)design, which is a software design that moves the detection functionality away from the devices. The learning automata are used in order to decide the optimal sampling rate in which the system should calculate packet rates for traffic flows present in the system.

2.2. Blockchain and DDoS Detection

Blockchain technology promises to transform the way information and data is exchanged between untrustworthy entities. Building trust in distributed environments without the need of central authorities is a technological breakthrough and blockchain technology can be an appropriate mechanism for securing IoT systems that are by default distributed and of limited trust. The use of both IoT and Blockchain technologies requires addressing multiple challenges before being an

effective approach [22]. Nevertheless, blockchain technology has already been successfully employed, in order to enhance detection system with respect to defending traditional cyber security attacks [23]. Blockchain has also been used to improve IoT security in general [24], and it has been shown that it can be specifically used for authentication, data integrity, or secure communication between IoT devices [25].

Specifically, the prevention of DDoS attacks through the use of blockchain has been one of the most interesting approaches discussed [24]. Moreover, a collaborative architecture based on the use of blockchain and smart contracts, that enables DDoS mitigation across multiple domains has also been proposed [26]. It mainly uses white and black lists of IPs as a security mechanism against DDoS attacks. Another similar approach [27] introduces blockchain as a defense mechanism against DDoS attacks launched by rogue physical devices. Authors propose the integration of communication between IoT devices and servers with an Ethereum blockchain instance and they suggest that such attacks can be prevented by utilizing the gas limit feature of Ethereum. BloSS [28] is a blockchain system that enables signaling between existing DDoS detection and mitigation modules and enables them to create reports and match them against certain security policies. Dorri et al. [29] proposed a blockchain-based architecture for the smart vehicles ecosystem. Nodes in the proposed system are clustered and managed by overlay block managers (OBM). Furthermore, nodes are required to sign transactions and broadcast those to the OBM, that validates the signature(s). In the case of a DDoS attack, the system would be able to react at the OBM level. Finally, blockchain has also been proposed as a solution for increasing DNS services' availability [30]. DDoS attack against DNS services can cause major problems, as they can make internet inaccessible for multiple users. The use of blockchain technology can produce distributed DNS services that are resilient to such attacks. This claim is further supported in [31], along with the fact that blockchain could also improve device authentication and data integrity in IoT systems.

2.3. Our Approach

This study proposes an alternative approach, with respect to existing work, to detect and prevent DDoS attacks in an IoT network. By utilizing the gateways in each installation instead of the IoT devices, it is much easier to apply our methodology in real world scenarios. Gateways provide more flexibility in software installation than IoT devices do, while they also offer the required processing and storage resources. Our approach is also designed with large-scale adoption in mind. The agents installed on the gateways are capable of collaborating in great numbers, in order to efficiently discover active DDoS attacks in time, through information sharing.

There are other methodologies, that have been discussed in this section, that also detect IoT related DDoS attempts. All these methodologies employ different data to experiment on, either through simulation or real world systems. While it would be interesting to directly compare the proposed schemes, such a comparison is inapplicable due to either different settings hypothesized for each method or limited available technical information for each system proposed. We have simulated network traffic to test the functionality of our system and estimate its efficiency. We plan to directly compare it with other systems, if this is possible, as part of our future work. Currently we can state that our system scales efficiently to protect large networks, while the simulation described in Section 6 has resulted in high DDoS attacks' detection rates. At the same time, our methodology has been proved to be relatively resilient against groups of malicious agents trying to falsify the results.

3. Agent-Based DDoS Detection System

3.1. The Model

The model used for implementing and testing the proposed architecture is based upon the hypothesis that there are numerous IoT installations (e.g., smart homes), each of which consists of multiple IoT devices. At each installation an agent, usually installed on the network's gateway,

monitors network traffic of the devices. The agents are the nodes of a private network and they are able to communicate with each other, in a peer-to-peer fashion.

The main idea is that agents collect traffic metrics and then they relay such information between them, without flooding the entire network. By relaying enough information between agents, we can detect an ongoing DDoS attack. The agents have access to limited processing and memory resources, therefore, in order for the system to be scalable, a lightweight work-flow is employed. The agents, to which infected IoT devices correspond, can collaboratively detect an ongoing DDoS attack, by summing up the observations each one makes for the devices attached to it.

The main metric used for traffic measurement is the rate of packets moving out of the network. These packet rates are calculated per a remote host basis, in order to detect anomalies with regard to specific hosts. In practice, the proposed system aims to reconstruct the flow of packets reaching to the victim of a DDoS attack on the side of the source of the problem, at the IoT installations.

3.2. Agents

The communication of the agents is semi-synchronous. The main information structure exchanged between agents is called path-message. A path-message is a cumulative information message that travels from agent to agent. It contains information regarding the traffic of all the devices that belong to the agents, through which the message has passed. In other words, each agent adds to path-messages, going through it, information observed at the corresponding installation and then pushes those to the next agent. The actual information the path-message contains is the total (summation of all agents traffic) out-going packets rate per destination IP address.

Regarding the generation of path-messages, a bounded probability, named *generation probability* is used. Generation probability is a parameter that controls active path-messages population existing in the network. The agent collects traffic information for all installed devices and adds that to the path-message. Then a destination agent is selected randomly and the message is sent to that agent. The agent that receives the message, adds traffic metrics of the corresponding installation to it and transfers it to another agent.

To sum up, the communication of the agents is based on the approach of concatenating information measured in different agents to deduce on occurring DDoS attacks. An agent collects traffic information from related devices, merges this information with the traffic information in the received path-messages and then forwards the updated path-messages to other agents. Due to agents' limited memory, the maximum allowed length of a path-message is bounded by a constant, according to agents' population. When a path-message reaches its maximum path length it is not transferred to another agent, but it is checked and if it carries critical information indicating an attack, an alert is relayed to every other agent. In any case, it is then discarded from the network.

At each step, the next agent can be chosen through different strategies. In this work the next agent is chosen randomly using only one constraint: a message cannot pass from the same agent more than once. Therefore, a path is strictly sustained while the message is traveling through the network.

4. Implementation

We assume that the internet consists of multiple hosts, a portion of which are IoT devices and thus they are vulnerable to becoming part of a botnet. This botnet is controlled by a C&C server, which is a server that actually controls the botnet using lightweight commands, and targets a specific victim with a DDoS attack. Both the C&C server and the victim belong to the hosts that are not IoT devices.

Let the network consist of multiple IoT devices, then the set of all IoT devices is denoted as:

$$D = \{d_1, ..., d_n\} \quad (1)$$

Additionally, let the set of all other hosts (non IoT devices) be:

$$H = \{h_1, ..., h_m\} \tag{2}$$

At each installation of IoT devices, an agent monitors the installed IoT devices. However, our approach can be installed in any number of gateways per installation or even in standalone devices directly connected to the internet. For the shake of simplicity we assume a uniform scheme, according to which there is one gateway per IoT installation. The set of agents is denoted as:

$$A = \{a_1, ...a_k\}, k \leq n \tag{3}$$

Each agent a_i monitors a subset $D^i \subseteq D, \forall i$ of the global set of IoT devices D. Specifically, the set of devices monitored by agent a_i is represented by:

$$D^i = \{d_1^i, ...d_p^i\}, p \leq n \text{ where } d_j^i \in D, \forall i, j. \tag{4}$$

When a DDoS attack occurs, a host h_c acts as the C&C server and organizes an attack against a victim host h_v. Host h_c takes control of a subset $B \subseteq D$ of the IoT devices and commands them to collaboratively send high volumes of traffic to host h_v. The devices that are part of the bot-net are denoted as

$$B = \{b_1, ...b_q\}, q \leq n. \tag{5}$$

The infected devices usually correspond to multiple different agents, in order for the attacker to be able to utilize the upload bandwidth of multiple installations while at the same time keeping the attack less susceptible to detection.

A sample network topology, consisting of three installations with three corresponding agents monitoring a varying number of devices, is depicted in Figure 1. Some of these devices (denoted in bold) are infected and comprise the botnet. A number of non IoT hosts also exist on the network and one of those has been selected as the victim host against which the DDoS attack is executed.

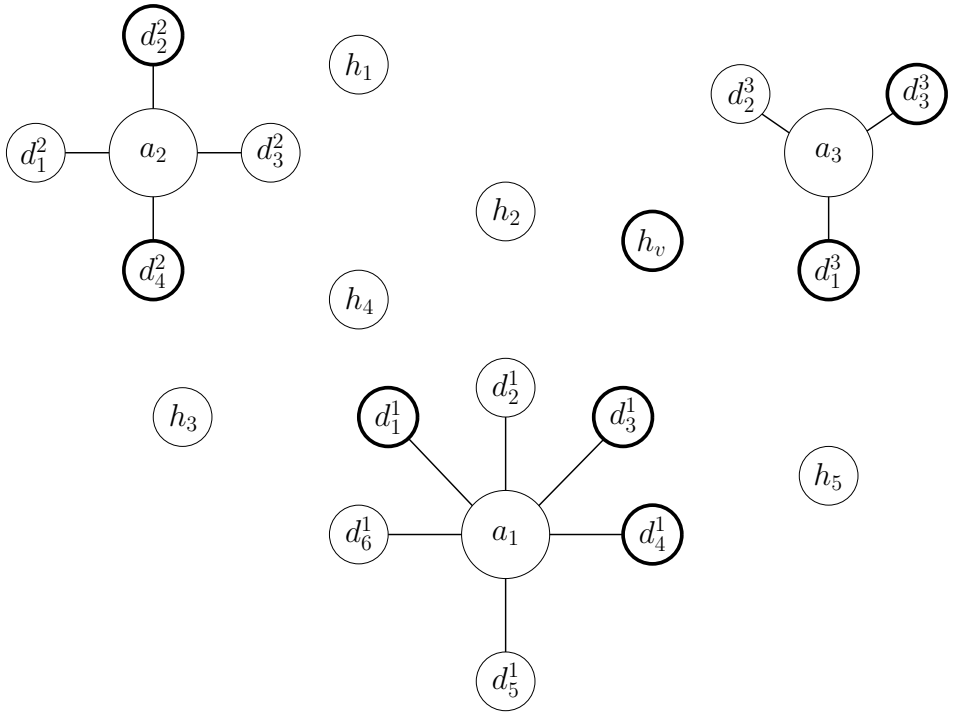

Figure 1. This is an example of a network structure in the current model.

4.1. Modeling Normal Traffic

In order to conduct the simulation, traffic regarding normal operation of IoT devices had to be simulated. To formalize this simulation, four generic profiles of IoT devices have been employed.

- **Single server devices:** These are devices which communicate with a single host to upload data. This happens periodically for most of the devices and the volume of data is usually small. Such devices communicate to a specific host $h \in H$ and the corresponding traffic packets are modelled according to a normal distribution. The mean μ and the variance σ^2 of the distribution are fixed throughout the simulation, for each different device.
- **Multiple clients devices:** These are devices which provide access to data they collect to multiple clients. Such devices communicate with more than one host and these communications are not as uniform as those of the previous profile. The traffic packets for such devices are produced by normal distributions. A different distribution is used for each different counterpart host $h \in H$. The mean μ and the variance σ^2 of the distribution for the communication of a specific device with a specific host are fixed throughout the simulation.
- **Streaming devices:** These devices are practically the equivalent of IP cameras. They stream high loads of traffic for continuous time windows to more than one host. The model for producing the packet flows for such devices is similar to the one of the previous profile. The difference is that the mean μ of the distribution is significantly higher due to higher traffic rates of such devices. Additionally, the packets produced are not scattered in time. The model produces continuous flows of packets for longer time periods, in comparison to previous models.
- **Management traffic:** Devices that do not send data but store them only locally. These devices may communicate with other hosts on the internet for management reasons, such as time syncing or DNS resolution. The model of this profile produces short occasional packet flows, that correspond to communication with a few standard hosts. The mean μ and the variance σ^2 of the distribution are both relatively small, to simulate the exchange of few packets in fixed time intervals.

The devices $d \in D$, used in the simulation, were picked randomly out of these four profiles, in order to create a realistic set of devices, with respect to real world IoT landscape.

4.2. Modelling DDoS Attack Traffic

The approach used for the simulation of the traffic generated by the DDoS attacks is described in the present subsection. One C&C server h_c is chosen randomly out of set H ($h_c \in H$) to initiate out the DDoS attack. Host h_c controls a bot-net B, that consists of multiple infected IoT devices belonging to set D, ($\forall b_i \in B, b_i \in D$). Additionally a single victim host h_v is randomly selected out of set H ($h_v \in H$). All devices belonging to B, receive commands from host h_c, and consequently they send chunks of traffic to host h_v, which eventually sum up to unbearable traffic rates for the victim host h_v.

The traffic flows that are produced in the simulation are two folded. Short (few packets) connections between host h_c and hosts $b_i \in B$ are established in the start of the DDoS attack to send the required commands to the bots. Larger continuous traffic flows from hosts $b_i \in B$ to the victim host h_v are then generated and correspond to the actual traffic of the attack. Those start early in the attack time frame, while they last until the execution of the attack is completed or interrupted. The volume of packets of the flow from a specific host's $b_i \in B$ to the victim host h_v is relatively low. These flows have as common destination the victim host h_v and eventually sum up to a large volume of traffic that concurrently reaches to the victim and the latter becomes unable to serve.

4.3. Monitoring Model

As mentioned above, each agent a_i is monitoring a number of IoT devices noted as $D^i = \{d^i_1, ...d^i_p\}$ and calculates traffic metrics for the devices in the corresponding set. The metrics calculated are accumulated to a special data structure called path-message M. This data structure is sent from agent

to agent, in order to collect data from different installations. It is called path-message, because it accumulates data from agents, along a random path.

A path-message is defined as a triple $M = (P, Cr, T_{out})$. In this triple $P = [a_1, \ldots, a_k]$ is the list of agents the path-message has passed through, where $\{a_1, \ldots, a_k\} \subseteq A$. Variable Cr is an integer representing the creation time of the path-message and T_{out} is a mapping from the set of hosts H to the set of natural numbers $\mathbb{N}$ that holds the current rate of packets being sent from the devices of the installation to the corresponding host $h \in H$. Formally,

$$T_{out} : H \rightarrow \mathbb{N}. \tag{6}$$

Variable T_{out} represents the out-going traffic of the installation. Each device in the installation d_i has its own mapping that corresponds to the outbound packets rate sent from device d_i to hosts outside the installation. These packet rates are denoted as T^d_{out}. Therefore, the agent a_i has to combine the information from its devices, that belongs to $D^i = \{d^i_1, ...d^i_p\}$ as follows:

$$T^i_{out}(h) = \sum T^{d^i_j}_{out}(h),\ \forall h \in H,\ \forall d^i_j \in D^i. \tag{7}$$

The path-message can actually be in two distinct states, the information gathering state, and the evaluation state. Every path-message starts in the information gathering state. While being in this state, the path-message is forwarded among multiple random agents, while traffic information is accumulated into it. When enough information has been accumulated, the path-message goes into the evaluation state and the data accumulated is evaluated, in order to check for ongoing DDoS attacks. Finally, after path-messages are evaluated, a consensus procedure takes place in order to detect attacks on a global scale.

4.3.1. Information Gathering

Initially, while the path-message M is in the information gathering state, an agent receives it, adds traffic information to it and then sends it to another agent. After agent a_i adds information to M another agent a_j is randomly chosen and the message M is send to it. The length of a path-message M, denoted as $|P|$, is limited by a percentage of the population of the agents in the network. This option controls the life-time of path-messages and thus prevents the flooding of the agents' network by significant traffic related to path-messages relaying. Formally, the length of the path-message (and its corresponding life span) shall always be $|P| \leq c|A|$, where $c \in (0, 1]$.

4.3.2. Evaluation

In practice when the length of the path-message $|P|$ exceeds the limit of $c|A|$, the path-message gets into the evaluation state, so it stops being transmitted from agent to agent and a metric called detection ratio is calculated, in order to detect possible attempts for DDoS attacks. In case of a DDoS attack being executed and given the fact that a high percentage of agents the path-message has passed through ($a \in P$) monitor infected devices $b \in B$, the attack should be detectable. It is expected that the $T^i_{out}(h_v)$ metric, the total packet rate towards the victim of the attack h_v, should be significantly higher than the similar metric calculated for any other hosts.

The host receiving the most packets in the path-message is denoted as h_m. Specifically h_m is the host for which:

$$T_{out}(h_m) > T_{out}(h_i), \forall i \neq m. \tag{8}$$

The main metric (detection ratio) used to detect committed DDoS attacks is the ratio of $T_{out}(h_m)$ to the corresponding value for the next busiest host of the path-message.

$$detection\_ratio(h_m) = \frac{T_{out}(h_m)}{\max_{\forall i \neq m}(T_{out}(h_i))} \tag{9}$$

If $detection\_ratio(h_m) > 2$ then h_m is a candidate for being a DDoS attack victim.

4.3.3. Consensus

The next step of the detection procedure is to accumulate information gathered from all path-messages, in order to be able to detect attacks executed from the network of protected installations. The information of each path-message, as this was filtered in the previous step, exists in different agents. In practice, it has been extracted from the agent at which each path-message has reached its maximum length. A light-weight consensus algorithm is then required, in order to collaboratively identify executions of DDoS attacks through data lying at different agents. The mechanics of this consensus mechanism are analyzed in Section 5.

Firstly, every agent checks if the packet rate of the busiest host of a specific path-message, denoted h_m, is at least twice that of the second busiest host of the same path-message. In that case the busiest host is a candidate victim and all such filtered entries (busiest hosts of path-messages) are concatenated for the running time slot. The one with the highest detection ratio is elected as the global candidate victim for the specific time slot.

If the global candidate victim is the same for at least three consecutive time slots, then that host is identified as probable DDoS victim and an alert is produced. There will be cases where a candidate victim h_m is not actually a victim but it just receives a large number of packets during its normal operation. In this case, it is statistically improbable that this host is proposed as a candidate victim in consecutive time steps, as it has been shown from the simulation in Section 6. On the other hand, when a real DDoS attack occurs, the real victim will satisfy the requirement of being proposed as a global candidate in many consecutive time slots, hence the above approach has been followed.

The described approach is based on the fact that path-messages hold accumulated information from all agents they have passed through. They do not store information only from the agent at which they reached the maximum length and at which they have been evaluated. In practice, candidate victims are the hosts to which a significant packet incoming rate has been summed up from data collected in random subset of agents. Even when a host is not being attacked but just receives a high rate of traffic from specific installations, the probability that the agents of those installations will co-exist in the path of the same path message is rather limited. Consequently, the high rates submitted by the agents corresponding to those installations will be averaged out by the rest of the agents that the path-message has passed through. On the other hand, actual victims tend to receive traffic from significantly more installations and the probability of a path mainly consisting of the agents from such installations is higher.

At some point in time it is probable that the protocol will filter in hosts, that simply happen to receive more traffic than others and are not victims of a DDoS attack. This is very unlikely to happen consistently for the same host in different path-messages and thus, the requirement of having a specific host as global candidate for consecutive time slots filters out such false alerts.

5. Consensus Mechanism

Gateways installed at different sites are vulnerable to other attacks, apart from DDoS, and the probability that the control of agents may be taken by malicious users is not negligible. In order to make the proposed protocol more robust and resilient to malicious nodes that would attempt to either hide an ongoing DDoS attack, or trigger a false alert, a validity checking mechanism has been employed.

The mechanism to protect the integrity of the information exchanged through path messages is based on a blockchain infrastructure that is co-maintained by all nodes participating in the system. Each node is checking the validity of the most recent states the path message has gone through according to specific rules. If a violation is observed, then the agent/agents related to it is/are excluded from subsequent path-messages.

Blockchain technology enables distributed nodes to collaborate without the requirement for a trusted third party. It provides the basis for collaboration of participating nodes upon a predefined

protocol. Nodes are equal to each other and interact according to predefined rules. Accidental or deliberate actions that are invalid with respect to the rules are not allowed by the protocol. An abstract blockchain infrastructure has been employed in order to present how a smart contract could orchestrate the agents collaboration and provide the required means, in order to reach consensus between agents about hosts that may be victims of DDoS attacks. No specific blockchain platform has been chosen, as this is out of the scope of this paper. The proposed scheme can be implemented in every blockchain platform that offers smart contracts functionality.

A smart contract is a set of rules that define how different nodes collaborate. Technically, it is a computer program and the main parts it consists of are a set of state variables and a set of functions which are called by nodes interacting with the smart contract. The calls to these functions are strictly ordered and the state variables of the contract have unquestionable values at any point in time, throughout the whole procedure. This makes smart contracts an appropriate scheme for supporting collaborating nodes that need to maintain a global state, without using a single central node to host this state.

5.1. The Fault-Tolerant Path-Message

In a fault-free environment, as described in Section 4, the information added to a path-message, as this is exchanged between nodes, is merged with metrics of previous agents into a single sum value. In order to tackle an environment with higher probability of malicious behavior, the information structure of the path-message has been enhanced. Apart from the last state, a number of previous states of the path message are additionally stored. Based on that, it will be feasible to detect the event of a malicious node trying to falsify information of the path message, or even of a group of more than one malicious node trying to collaborate on succeeding that.

The enhanced structure of the path-message contains the recent k states of it, by holding data about distinct states submitted by the k previous nodes. Hence, this structure provides a way of checking at least k previous steps and ensuring the data has not been falsified. Furthermore, each distinct data record of traffic information is signed by each node, securing the whole process in the case of nodes attempting to tamper with the information submitted by previous nodes. The actual value of k parameter needs to be set with respect to other parameters such as the length of the path message and the ratio of malicious collaborating agents. Related experiments are presented in Section 6.

When an agent receives a path-message, it has to sum up the metrics of outbound traffic, observed locally with the ones already existing in the path-message. The accumulated traffic metrics are defined as the new state of the path message. Each agent practically updates the state of the path message and the transition between two states is coupled to the information the agent adds to the path message.

Formally, let a path-message be denoted by $\mathbf{P}$, then the path-message is defined as a set of the previous states committed by the last k nodes.

$$\mathbf{P} \coloneqq \{\mathbf{H}, \mathbf{T}_0, \ldots, \mathbf{T}_{k-1}\} \tag{10}$$

where $\mathbf{H}$ is the list of the previous nodes, and $\mathbf{T}_i,\ \forall i \in [0, k-1]$ is the ordered list of state updates (regarding traffic information) committed by each one of the last k nodes.

The additional building block required is a formally defined validation scheme, that enables all involved parties to check if a status update is valid or not. In that sense, the rules applied are:

- An agent can only add outbound traffic during status update. No subtractions are allowed.
- An agent can add outbound traffic to more than one destination as long as the sum of this traffic is bounded by an upper limit that is relevant to the maximum upload bandwidth of the installation. For simplicity reasons, a global value is set for this limit for all agents.

These rules are mathematically defined as follows:

$$\mathbf{T}_{i+1} = \{\mathbf{t}_i^j + \mathbf{t}_{i+1}^j \mid \mathbf{t}_{i+1}^j \geq 0,\ \forall j \in I\}, \tag{11}$$

where I is a set of IP addresses and

$$\sum_{j \in I} \mathbf{t}_{i+1}^{j} \leq M, \tag{12}$$

where M is the maximum upload rate of each installation.

5.2. Smart Contract

A smart contract governs the collaboration between the agents, in order to successfully exchange information according to the rules defined above and to combine information from path-messages that have reached their maximum length at different agents. The smart contract facilitates three basic procedures for the protocol.

- It defines when path-messages are created and also defines a random path of agents for each one of the path-messages.
- It validates any reports of malicious agent behavior and blacklist related nodes
- Handles all path-messages that reach their maximum length at the same time-step and calculates the required metrics to detect any ongoing DDoS attacks

Every agent is identified by a blockchain identity. The smart contract permits access to its endpoints to a predefined set of such identities that corresponds to the agents that take part in the protocol. The workflow is designed in such a way that minimizes the interaction of the agents with the smart contract, and thus reduces performance overhead and the transaction fees in case of a public blockchain. There are two different functions that an agent may interact with the smart contract through, which are:

- **path_message_validation()**: path message transition validation
- **path_message_submission()**: submission of finalized path messages in order to conclude whether there is an ongoing attack or not

All agents follow a specific workflow regarding the handling of path-messages. The agents exchange and update path-messages, as it has been described in Section 4, but at specific points and under specific circumstances they may be required to interact with the smart-contract, as well. The main points of the workflow are:

- The smart contract triggers the generation of new path-messages and it also sets the random path to be followed for each one of these. A random approach has been opted for path construction, in order to prevent agents from choosing the next node on their own. In such a case, malicious collaborating nodes would be able to set up paths consisting exclusively of them. As the mechanism is based on the fact that each agent checks data submitted from the previous agents in the path, this would enable malicious nodes to submit invalid data, while going undetected. The path is predetermined from the generation of the path-message and known to all agents it passes through. The randomization process is not related with keeping the path secret but with forcing agents to adhere to paths that they cannot set on their own. At each time period the smart contract generates a number (equal to the number of path messages to be created) of seeds (random values). Each seed value is hashed and the modulo of the result against the number of nodes indicates the ID of the agent to initialize the path-message.

$$seed_0 = hash(seed) \tag{13}$$

$$n_0 = seed_0 \mod n \tag{14}$$

Consequently, the ID of the next node is indicated by rehashing the hashed value and calculating its modulo against the number of the nodes.

$$seed_t = hash(seed_{t-1}) \tag{15}$$

$$n_t = seed_t \mod n \tag{16}$$

If the next node n_t happens to have already been included in the path or is temporarily blacklisted, then node n_{t-1} rehashes the value until an appropriate node is selected.

- When an agent starts a new path-message it adds its own traffic metrics, adds the identity of its own and of the subsequent agent to the list and signs the updated state, before forwarding the information to the next node.
- When an agent receives a path-message from a previous node it has to check the validity of the information within the path message. The agent is required to validate the last k state transitions with respect to the validity of the signatures, the validity of the transitions according to the rules defined above, and the validity of the previous node IDs according to initial seed value.
  - If everything is okay then the agent has to update the state of the path message with its traffic information and sign the new state. If the path-message has not reached its maximum length, then the agent has to find out who the next node is, as defined in Equations (15) and (16). It adds the next corresponding agent to the list and forwards the path-message to it. If the path-message has reached its maximum length then the agent has to submit it to the contract through the **path_message_submission()** function. In this case, the contract checks the length of the path message, the validity of the last k state transitions and if no inconsistencies are found, then the path message is added to the pool of path messages that reached maximum length at the specific time interval. These are then combined as described in Section 4, to produce alarms regarding DDoS attacks.
  - If an issue is observed at a specific transition, then the agent has to report that to the smart contract, through the smart contract **path_message_validation()** function. The agent has to report the whole path message to the contract along with the index of the first (chronologically) invalid transition. The contract can automatically validate the claim of the agent and if it stands, then it is going to mark the agent that is responsible for the invalid state transition and all subsequent agents (apart from the one reporting it) as malicious. These nodes are then temporarily (or even permanently) added to a black list of inactive nodes and are not included in subsequent path-messages' paths.

The core functionality of the consensus mechanism is depicted in the activity diagram of Figure 2.

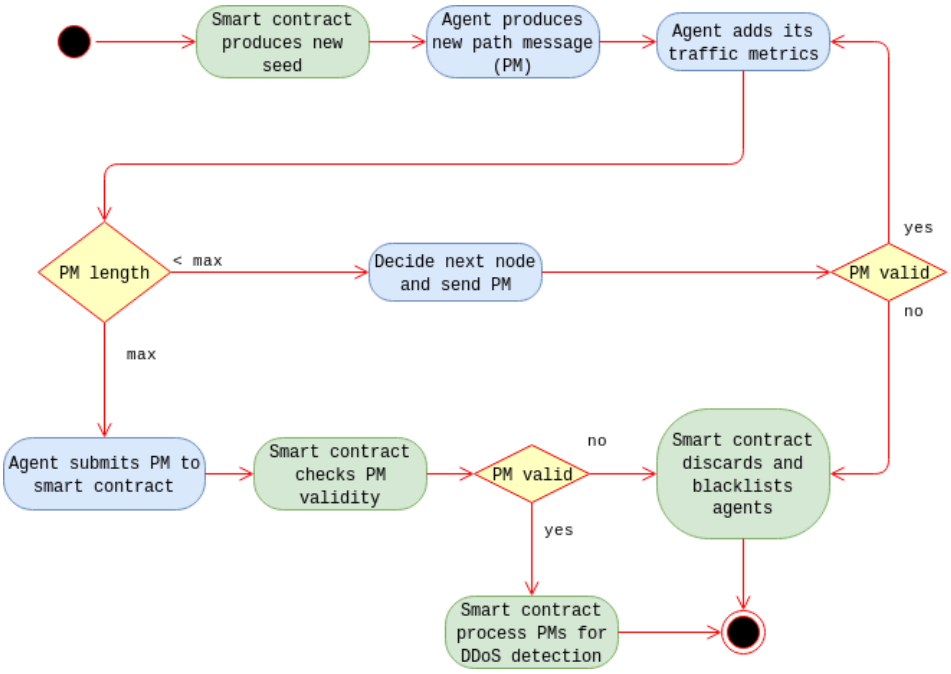

Figure 2. Consensus protocol activity diagram.

In order to further reduce the malicious behaviors that are possible, the node that conveys the path-message waits for a signed acknowledgment response from the node that receives that. If the response is never sent, then the node chooses another node to convey the path-message to.

This feature is protecting the protocol from malicious nodes that will try to stop the transmission of path-messages at some point of the path. If the node receives a response from the subsequent node, then the responsibility of monitoring the future of the path-message is transferred from the former to the latter. Each node can compute the time steps remaining for a path-message to reach the end of its life-cycle and therefore to be submitted to the smart contract. When this happens the smart contract emits an event that is accessible for all agents.

If the path-message never reaches the smart contract, then the node that sent an acknowledgment response and never forwarded the massage is assumed to be a malicious one and the dispute resolving protocol is initiated in order to find out who this node is. All nodes that have sent a path-message and received a response wait for the corresponding event that the path-message has reached the smart contract in a specific time slot. If that never happens then they can initiate a procedure in which all subsequent nodes are candidates for being the malicious nodes that stopped the path message forwarding. Each one of these can prove that they behaved according to the protocol by submitting the signed acknowledgment response they received when they actually forwarded the path-message. The agent that has not forwarded the path-message will not be able to make such a claim, so it will be added to the agents' blacklist.

As analyzed, the path-message states correspond to aggregated outbound traffic rates to specific hosts for each installation. That means that agents are able to inspect states of previous (in the path) agents and conclude on the hosts to which IoT devices of corresponding installations connect to. IoT devices usually connect to predefined hosts, the identity of which can be easily inferred (e.g., an IP camera connects to the manufacturer's storage service), thus the privacy implications are rather limited. Nevertheless, in order to further minimize privacy implications, the identity (IP address) of hosts can be replaced by a salted hash of it.

Specifically, the IP address of hosts is hashed with the use of a random salt value related to the time slot of the path-message generation. It is essential that this salt is common for all path-messages generated concurrently, as the results from those path-messages are going to be aggregated and evaluated against each other. If this procedure produces an alert regarding an ongoing DDoS attack

against a host then the salted hash of this host's IP address is produced. Then the agents that have taken part in the corresponding path-message can resolve the identity of the victim by submitting the actual IP and validating it against the produced hash value. In order to prevent dictionary attacks against this hashing scheme, an approach proposed in [32] can be used, which is designed specifically for the prevention of such attacks. Slow and difficult to compute hash functions will add a bearable overhead when updating path-messages status, but it will make brute-force attacks inefficient, as a lot of processing power will be required in order to access relatively limited information.

5.3. Security Analysis

The main aim of the protocol, analyzed in the previous subsection, is to fortify the collaboration of agents against malicious nodes that may try to falsify information. The malicious behavior is as follows, a malicious node q_m^i receives a path-message from an honest node u_h^i. The data in the path-message received can be altered either by subtracting traffic or by adding traffic. Since the path-message consists of signed traffic information of the k previous nodes, the malicious node cannot tamper with that information. It can, however, update the current state of the path-message without any restrictions.

Practically, a malicious node can try to remove evidence of an ongoing DDoS attack by reducing the values summed up by previous nodes regarding the traffic rate against the actual target of the attack. Alternatively, it can try to trigger false alerts, by adding an artificially high rate of traffic against a specific host.

The state transition validation, committed by the agents, and potentially by the contract, hinders a malicious node from submitting false information. The protocol is mainly being carried out between the agents and the smart contract is engaged only when an inconsistency is identified. Agents are required to put in processing resources to this procedure, in order to minimize the interaction with the smart contract. Figure 3 depicts the communication scheme between the nodes and the smart contract.

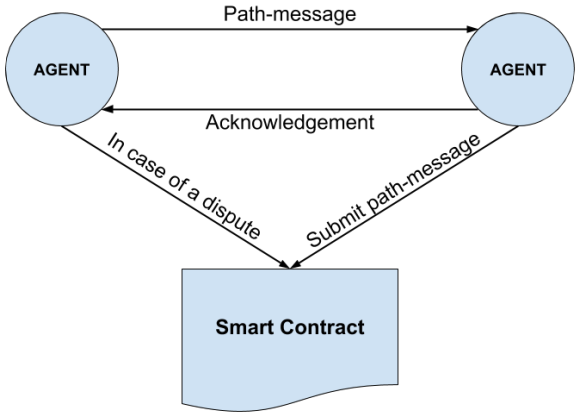

Figure 3. Communication protocol between nodes and the smart contract.

Each agent checks the validity of the most recent state changes, thus the collaboration of more than one malicious node may enable them to surpass the validation mechanism and falsify information, while going undetected. If each agent only validates data provided by the previous one, then two collaborating malicious agents that happen to be subsequent nodes for a specific path message, can break the rules defined above and either hide an actual DDoS attack or trigger a false alert.

In practice, if the state transition submitted by the first one of the two malicious nodes is invalid and the second malicious node does not report that, then the malicious behavior will go undetected.

In order to minimize the probability of this happening, the protocol requires from each agent to validate the latest k state transitions for the path-message. Practically, in order for collaborating nodes to surpass this validation mechanism, it is required that at least k of them shall be ordered continuously in the path of a path-message. Because the ordering of the nodes is decided randomly, a higher value for k parameter, minimizes the probability of malicious nodes being able to collaborate as described above.

5.4. Blockchain Limitations

There are two limitations of blockchain technology with respect to the protocol described in this section, which are related to random values generation and time measurement. Smart contracts execution is deterministic and as this happens in all nodes at the same time it is practically infeasible to generate a random value. This problem has been encountered in multiple blockchain applications and it has been solved either by using the hash of past blocks or past transactions in order to generate random values or employ external oracles. Regarding time measurement, applications for which accuracy is not a strict requirement can take into account block height. As blocks are produced in a relatively stable rate, the number of blocks that have been mined after an event give a satisfactory approximation of the time that has passed since then.

6. Simulation and Results

A simulation approach has been employed, in order to justify both the detection capability of the proposed system and the robustness of the protocol against malicious or accidental submission of invalid information.

6.1. Detection Capability

In order to justify that the detection of DDoS attacks is feasible through the use of the proposed methodology, a simulation was conducted. The simulation duration was split into 1000 continuous time slots through which agents calculate metrics and relay messages as discussed in Section 4. A network consisting of both multiple IoT devices and ordinary hosts was created. IoT devices are grouped into different installations and are monitored by different agents accordingly. Some of the IoT devices are supposed to be infected and used as part of a botnet. This botnet executes a DDoS attack against a specific victim host at a specific time interval of the simulation (starts at time slot 500 and ends at time slot 750). The normal traffic of the network along with the traffic triggered by the DDoS attack are simulated at each agent of the system, according to the model analyzed in Section 3. Then the methodology presented in Section 4 is employed, in order to detect the DDoS attack.

The various simulation setups used have been designed by setting different values for the size of the network and the infection ratio (percentage of infected IoT devices). Specifically, nine different scenarios were studied according to the combinations of three different network sizes (5000, 7500 and 10,000 IoT devices) and three different infection ratios (1%, 3% and 5%). The hosts are equal in number to the IoT devices at each scenario, while the number of IoT devices at each installation (number of devices per agent) is approximately 10. Additionally, the generation probability of path-messages is set equal to 0.1.

Finally, different values for factor c (maximum path-message length) were tested, in order to check which is the most efficient approach. The different values used for factor c were 0.1, 0.2, 0.3, 0.4, and 0.5.

Tables 1–4 depict the results obtained through the different simulation scenarios. In Table 1 the average number of path-messages that exist in the network throughout the simulation is shown. The path-messages population significantly varies, for the different choices for the c parameter value. This metric is important as it is crucial to succeed in detecting the attack by using the minimum possible

volume of path-messages and avoiding the induction of unnecessary overhead into the system. As the value of c factor grows, the average population of active path-messages increases.

Table 1. Average number of path-messages existing in the network with respect to factor c.

c	Number of Devices		
	5000	7500	10,000
0.1	423.80	734.40	1052.60
0.2	556.90	935.40	1367.80
0.3	615.90	1083.82	1694.10
0.4	709.80	1264.20	2010.80
0.5	746.00	1531.12	2372.50

Table 2. Assessment metric of all path-messages in a network with 1% infected devices and for 5000, 7500, and 10,000 devices. Bold numbers show the best cases for each number of devices.

c	Number of Devices		
	5000	7500	10,000
0.1	1.04	1.17	1.69
0.2	1.24	1.32	**1.77**
0.3	1.40	**2.17**	1.33
0.4	**1.44**	2.08	1.23
0.5	1.39	1.02	1.20

Table 3. Assessment metric of all path-messages in a network with 3% infected devices and for 5000, 7500, and 10,000 devices. Bold numbers show the best cases for each number of devices.

c	Number of Devices		
	5000	7500	10,000
0.1	2.10	2.10	2.04
0.2	2.57	3.77	2.29
0.3	**3.41**	3.88	**2.62**
0.4	3.29	**4.40**	2.08
0.5	2.68	3.87	1.99

Table 4. Assessment metric of all path-messages in a network with 5% infected devices and for 5000, 7500, and 10,000 devices. Bold numbers show the best cases for each number of devices.

c	Number of Devices		
	5000	7500	10,000
0.1	3.12	3.43	4.48
0.2	3.85	4.88	**5.58**
0.3	5.62	5.18	5.41
0.4	5.64	4.67	5.04
0.5	**6.30**	**5.40**	3.88

Additionally, in order to present the detection capabilities of the methodology an assessment metric is presented in Tables 2–4 for infection ratios 1%, 3%, and 5% respectively. Assessment metric used is the ratio of the highest observed T_{out} for the victim host over the highest observed T_{out} of all other hosts, throughout all path-messages processed. This assessment metric gives a good insight of the detection ability of the proposed methodology. The higher the assessment metric is, the easier it will be to detect the ongoing DDoS attack. This is only a general assessment metric and it is expected that the detection ratio values produced by Equation (9) for h_v will be even higher, as they are locally calculated for each path-message.

By observing the results, it is obvious that it is feasible to detect the DDoS attack. Additionally, a common efficient value for c parameter for all cases is 0.3, as it generally gives quite satisfactory results. A shorter value choice would create shorter path-messages that would not collect information from the required number of agents, in order to detect the attack. On the other hand, larger values create longer path-messages, which seem to average out the information collected for the DDoS victim host.

Figure 4 depicts the values collected by all path-messages constructed in the case of 5000 IoT devices network with a 3% infection ratio. These values have been gathered for a coefficient value $c = 0.3$. Records related to the actual DDoS victim host h_v are colored in red and can easily be identified from the rest of the records, which are colored in blue. From Figure 4 it is obvious that during the attack the detection ratios calculated for victim host h_v tend to be significantly higher levels than the corresponding values for all other hosts.

The attack takes place in time window between time slot 500 and time slot 750. The results that correspond to this attack span in a larger time window up until time slot 900. This is normal as path-messages that have gathered information during the time of attack are evaluated after the end of the attack and produce corresponding results approximately up until time slot 900.

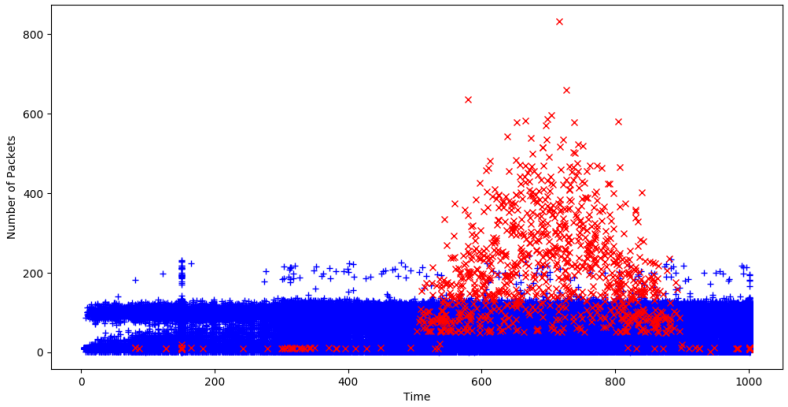

Figure 4. 5000 IoT devices, 3% infection ratio, 0.3 max path-message length.

Consequently, the path-messages detection ratio values that have been filtered in from all agents are concatenated in chronological order. The produced set of records mainly contains reports concerning the host h_v. Figure 5 shows records for all filtered in path-messages for the 10,000 IoT devices network with a 1% infection ratio ($c = 0.3$). It depicts all path-messages for which packet rate of the busiest host is at least double that of the next busiest host. The red marks correspond to path-messages of the actual victim host h_v, while the blue ones correspond to path-messages for any other host. It is spotted that during the attack most of the records filtered in relate to h_v. However, before and after the attack there are also multiple records for other hosts.

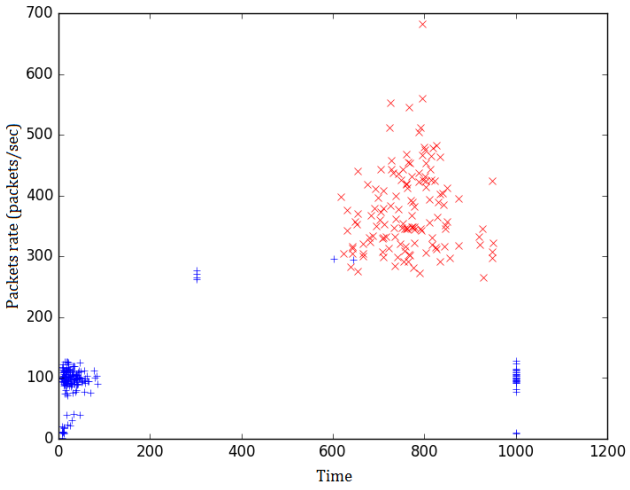

Figure 5. DDoS detection for 10,000 IoT devices (all records).

In order to efficiently detect a DDoS attack going on, a requirement for multiple consequent appearances (in consequent time slots) of the same host as candidate victim was discussed in Section 4. This step practically discards all records, irrelevant to DDoS attacks, that have been filtered in due to extreme traffic conditions or any other cause. The consequent records requirement is valid only for records relevant to actual DDoS attacks. The records that satisfy this requirement are finally outputted as DDoS attacks alerts.

The relative tests have produced zero false positives in all scenarios. Regarding the 1% infection ratio scenarios (the hardest case), the system has produced 89, 70, and 130 alerts for the 5000, 7500, and 10,000 IoT devices' networks. These results have been produced by setting a value of 0.3 for the c coefficient.

It is evident that there may be rare cases where benign traffic may be identified as part of DDoS attacks. For example, in the case of IP cameras, if multiple deployed devices utilize a single common point of storage, then the traffic created by video streams uploading procedure resembles that of a DDoS attack against the storage point. In general, it is expected that such cases are rare and do not exhibit the dispersion of source nodes that actual DDoS attacks do. Limited source hosts dispersion means limited probability of triggering an alert. Of course, such cases cannot be neglected, but on the other hand, hosts that are expected to receive such traffic are known beforehand and a list of such hosts may be maintained. In practice, alerts can be filtered as they are produced, according to the candidate victim host or agents may use a white-list of hosts expected to receive multiple connections and exclude corresponding traffic metrics from path-messages.

6.2. Robustness Against Malicious Nodes

In order to validate the blockchain based collaboration protocol against agents that intentionally or accidentally submit false information, recurring simulation experiments have been conducted. A blockchain platform agnostic consensus protocol (i.e., it can be implemented on any blockchain platform that provides smart contracts support e.g., Ethereum, Hyperledger, Corda) is proposed. As the protocol failure probability is independent of any specific implementation of the blockchain, we have conducted a simulation experiment to evaluate this probability. The goal was to estimate the probability that information of a path-message has been falsified, without this being noticed, under specific parameters such as:

- nodes population
- malicious nodes population
- maximum path-message length

- number of state checks committed at each agent

For all scenarios, the population of nodes was set to 1000. The population of malicious nodes has been varied between 100, 200, and 300. The maximum path-message length has been set to values 10, 20, and 30. The number of checks committed at each agent have been gradually increased from 2 to 7. The resulting probabilities that malicious nodes can falsify information while going undetected are depicted in Tables 5–7 for all different parameters' values. Specifically, each table shows the results obtained for each one of the three different cases with respect to the population of malicious nodes.

Table 5. 100 malicious nodes.

k	**Path-message length**		
	10	20	30
2	0.073	0.169	0.237
3	0.004	0.019	0.020
4	0	0.001	0
5	0	0	0

Table 6. 200 malicious nodes.

k	**Path-message length**		
	10	20	30
2	0.271	0.472	0.639
3	0.053	0.118	0.184
4	0.012	0.023	0.030
5	0.011	0.006	0.005
6	0	0.001	0

Table 7. 300 malicious nodes.

k	**Path-message length**		
	10	20	30
2	0.521	0.764	0.890
3	0.174	0.324	0.421
4	0.042	0.097	0.139
5	0.010	0.022	0.044
6	0	0.008	0.013
7	0	0	0.004

Figures 6–8, depict the effect of the increase in number of checks to the probability of collaborating nodes submitting invalid information while going undetected. It is evident that an additional state check vastly decreases the aforementioned probability. Specifically, when 100 out of 1000 nodes are malicious, an additional state check approximately divides the probability by a factor of 15. When 200 out of 1000 nodes are malicious, an additional state check approximately divides the probability by a factor of 5 and in the case of 300 malicious nodes the corresponding factor is 3. Of course, as the length of the path message is increased the effect of the additional check is slightly reduced, but in any case, the required number of state checks to minimize the probability is feasible and will not produce significant overhead.

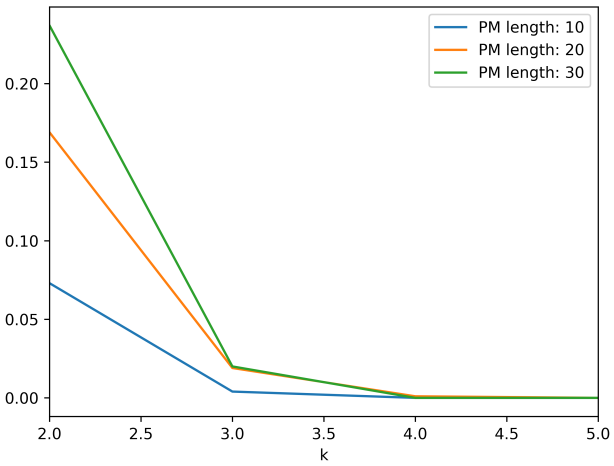

Figure 6. 100 malicious nodes.

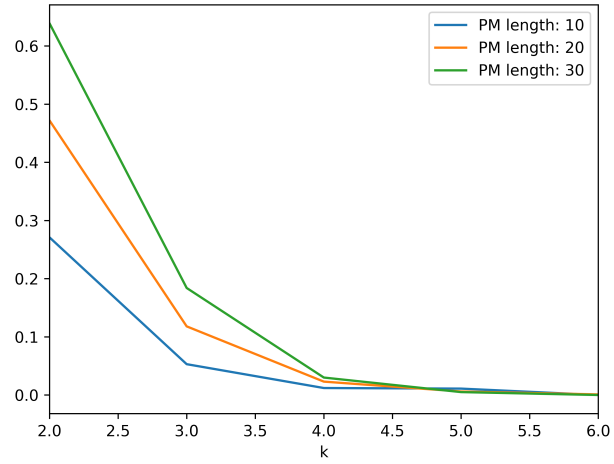

Figure 7. 200 malicious nodes.

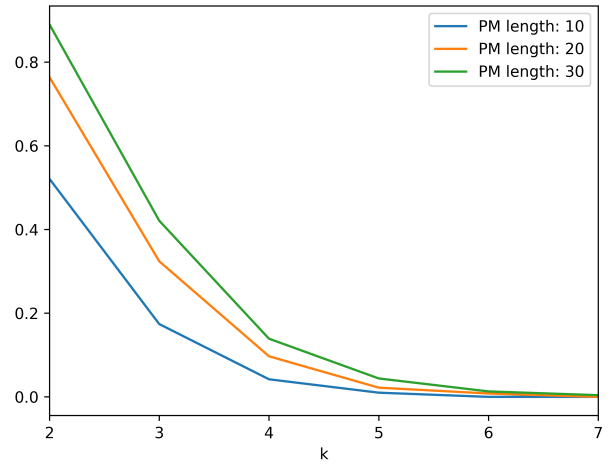

Figure 8. 300 malicious nodes.

Our approach exhibits some qualitative advantages, with respect to other collaborative DDoS detection techniques, presented in literature. While conducting a quantitative comparison was not applicable, we opted for identifying the main characteristics that each one of the published methods demonstrates. The comparison has been made upon the following main points:

- **Integrity:** Integrity is related to whether the proposed approach defends the integrity of the exchanged information and thus the validity of the detection results.
- **Availability:** Availability relates to the robustness of the proposed systems with respect to possible failures for a limited number of nodes. It describes the ability of proposed systems to remain available in such cases.
- **Responsiveness:** Responsiveness describes the ability of systems to timely detect possible attacks and produce corresponding alerts.
- **Scalability:** Scalability corresponds to whether the proposed systems can be scaled to protect larger gateways federations, without becoming inefficient.

An analysis of eight previous research efforts (traditional ones and blockchain based ones) along with the approach presented herein has been conducted. According to this analysis, a comparison depicted in Table 8 has been attempted. While our approach is blockchain based, it adheres to layer 2 scaling paradigm [33] that enables building an efficient decentralized application, which does not suffer from low transaction rates and other related issues. Our methodology utilizes the state channels approach [34] and enables agents to collaborate in a safe and efficient way. Our system is the only one that can offer integrity, availability, and responsiveness guaranties and at the same time maintain these offerings even for relatively larger networks of nodes.

Table 8. Comparison between different collaboration approaches.

Approach	Integrity	Availability	Responsiveness	Scalability
Rodrigues et al. [28]	✓	✓		
Javaid et al. [27]		✓	✓	
Rodrigues et al. [26]	✓	✓	✓	
François et al. [35]		✓	✓	
Chen et al. [36]		✓	✓	
Chin et al. [37]			✓	✓
Patil et al. [38]			✓	✓
Sagirlar et al. [39]	✓	✓		✓
Our approach	✓	✓	✓	✓

7. Conclusion and Future Work

Security of IoT devices is a major issue, as such devices usually become the means for conducting attacks against other hosts. One such case is the execution of DDoS attacks, through the use of IoT devices in constructing botnets. The results have shown that it is possible to detect the network activity of a DDoS attack if multiple agents collaborate. The main idea is to use an efficient number of path-messages that travel through agents installed in different IoT sites and cumulatively construct a traffic profile for probable DDoS attacks' victims. Additionally, obtained results have shown that a length coefficient of 0.3 is more or less effective in all cases examined.

The minimum level of collaboration required in order for such an attack to be detected was evaluated. While many open issues still exist regarding the implementation of such a system or the implementation of the underlying consensus mechanism, it is shown by our experiments that it is possible to detect large scale DDoS attacks.

More research is planned to be conducted in terms of implementing the consensus mechanism that is used among the agents, as the latter are usually installed on limited resources hardware. Implementing the proposed protocol on a specific blockchain platform, such as Ethereum, will provide

a proof of the applicability of the solution and stress out possible practical limitations that may emerge. Additionally, an enhanced version of the proposed algorithm that would have the same results, while at the same time fully protecting the privacy of engaging users, could benefit the involved parties more. Homomorhpic encryption techniques can be utilized in order to enable the construction of path-messages, without revealing traffic information of installations to other agents.

Author Contributions: All authors contributed equally to this work.

Funding: This research received no external funding.

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

References

1. Wan, J.; Yan, H.; Suo, H.; Li, F. Advances in cyber-physical systems research. *KSII Trans. Internet Inf. Syst. (TIIS)* **2011**, *5*, 1891–1908. [CrossRef]
2. Chen, M.; Wan, J.; Li, F. Machine-to-Machine Communications: Architectures, Standards and Applications. Available online: https://www.researchgate.net/profile/Jiafu_Wan2/publication/264846553_Machine-to-Machine_Communications_Architectures_Standards_and_Applications/links/550b9af60cf265693cef8967/Machine-to-Machine-Communications-Architectures-Standards-and-Applications.pdf (accessed on 24 October 2019).
3. Jing, Q.; Vasilakos, A.V.; Wan, J.; Lu, J.; Qiu, D. Security of the Internet of Things: Perspectives and challenges. *Wirel. Netw.* **2014**, *20*, 2481–2501. [CrossRef]
4. Andrea, I.; Chrysostomou, C.; Hadjichristofi, G. Internet of Things: Security vulnerabilities and challenges. In Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 6–9 July 2015; pp. 180–187.
5. Weber, R.H. Internet of Things–New security and privacy challenges. *Comput. Law Secur. Rev.* **2010**, *26*, 23–30. [CrossRef]
6. Roman, R.; Najera, P.; Lopez, J. Securing the internet of things. *Computer* **2011**, *44*, 51–58. [CrossRef]
7. Peraković, D.; Periša, M.; Cvitić, I. Analysis of the IoT impact on volume of DDoS attacks. In Proceedings of the 33rd Symposium on New Technologies in Postal and Telecommunication Traffic (PosTel 2015), Beograd, Serbia, 1–2 December 2015; pp. 295–304.
8. Krebs, B. KrebsOnSecurity Hit with Record DDoS. 2016. Available online: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/ (accessed on 25 January 2018).
9. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and other botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
10. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Invernizzi, L.; Kallitsis, M.; et al. Understanding the mirai botnet. In Proceedings of the 26th {USENIX} Security Symposium ({USENIX} Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 1093–1110.
11. Zhang, Z.K.; Cho, M.C.Y.; Wang, C.W.; Hsu, C.W.; Chen, C.K.; Shieh, S. IoT security: Ongoing challenges and research opportunities. In Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications (SOCA), Matsue, Japan, 17–19 November 2014; pp. 230–234.
12. Mahmoud, R.; Yousuf, T.; Aloul, F.; Zualkernan, I. Internet of things (IoT) security: Current status, challenges and prospective measures. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 14–16 December 2015; pp. 336–341.
13. Giachoudis, N.; Damiris, G.P.; Theodoridis, G.; Spathoulas, G. Collaborative Agent-based Detection of DDoS IoT Botnets. In Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini Island, Greece, 21–29 May 2019; pp. 205–211.
14. Liang, L.; Zheng, K.; Sheng, Q.; Huang, X. A Denial of Service Attack Method for an IoT System. In Proceedings of the 2016 8th International Conference on Information Technology in Medicine and Education (ITME), Fuzhou, China, 23–25 December 2016; pp. 360–364. [CrossRef]
15. Pacheco, L.A.B.; Gondim, J.J.C.; Barreto, P.A.S.; Alchieri, E. Evaluation of Distributed Denial of Service threat in the Internet of Things. In Proceedings of the 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 24–27 October 2016; pp. 89–92. [CrossRef]

16. Lyu, M.; Sherratt, D.; Sivanathan, A.; Gharakheili, H.H.; Radford, A.; Sivaraman, V. Quantifying the Reflective DDoS Attack Capability of Household IoT Devices. In Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '17), Boston, MA, USA, 18–20 July 2017; ACM: New York, NY, USA, 2017; pp. 46–51. [CrossRef]
17. Sonar, K.; Upadhyay, H. An Approach to Secure Internet of Things Against DDoS. In *ICT4SD 2015, Proceedings of the International Conference on ICT for Sustainable Development, Ahmedabad, India, 3–4 July 2015*; Springer: Singapore, 2016; Volume 2, pp. 367–376.
18. Donno, M.D.; Dragoni, N.; Giaretta, A.; Mazzara, M. AntibIoTic: Protecting IoT Devices Against DDoS Attacks. In *International Conference in Software Engineering for Defence Applications*; Springer: Cham, Switzerland, 2017.
19. Zhang, C.; Green, R. Communication Security in Internet of Thing: Preventive Measure and Avoid DDoS Attack over IoT Network. In Proceedings of the 18th Symposium on Communications & Networking, Society for Computer Simulation International (CNS '15), San Diego, CA, USA, 12–15 April 2015; pp. 8–15.
20. Chen, C.M.; Hsu, S.C.; Lai, G.H. Defense Denial-of Service Attacks on IPv6 Wireless Sensor Networks. In *Genetic and Evolutionary Computing, Proceedings of the Ninth International Conference on Genetic and Evolutionary Computing, Yangon, Myanmar, 26–28 August 2015*; Springer International Publishing: Cham, Switzerland, 2016; Volume 1, pp. 319–326.
21. Misra, S.; Krishna, P.V.; Agarwal, H.; Saxena, A.; Obaidat, M.S. A Learning Automata Based Solution for Preventing Distributed Denial of Service in Internet of Things. In Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, Dalian, China, 19–22 October 2011; pp. 114–122. [CrossRef]
22. Reyna, A.; Martín, C.; Chen, J.; Soler, E.; Díaz, M. On blockchain and its integration with IoT. Challenges and opportunities. *Future Gener. Comput. Syst.* **2018**, *88*, 173–190. [CrossRef]
23. Meng, W.; Tischhauser, E.W.; Wang, Q.; Wang, Y.; Han, J. When intrusion detection meets blockchain technology: A review. *IEEE Access* **2018**, *6*, 10179–10188. [CrossRef]
24. Kshetri, N. Can blockchain strengthen the internet of things? *IT Prof.* **2017**, *19*, 68–72. [CrossRef]
25. Khan, M.A.; Salah, K. IoT security: Review, blockchain solutions, and open challenges. *Future Gener. Comput. Syst.* **2018**, *82*, 395–411. [CrossRef]
26. Rodrigues, B.; Bocek, T.; Lareida, A.; Hausheer, D.; Rafati, S.; Stiller, B. A blockchain-based architecture for collaborative DDoS mitigation with smart contracts. In Proceedings of the IFIP International Conference on Autonomous Infrastructure, Management and Security, Zurich, Switzerland, 10–13 July 2017; Springer: Cham, Switzerland, 2017; pp. 16–29.
27. Javaid, U.; Siang, A.K.; Aman, M.N.; Sikdar, B. Mitigating loT Device based DDoS Attacks using Blockchain. In Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Munich, Germany, 15 June 2018; pp. 71–76.
28. Rodrigues, B.; Bocek, T.; Stiller, B. Enabling a Cooperative, Multi-Domain DDoS Defense by a Blockchain Signaling System (BloSS). Available online: https://www.ieeelcn.org/lcn42demos/1570382330.pdf (accessed on 24 October 2019).
29. Dorri, A.; Steger, M.; Kanhere, S.S.; Jurdak, R. Blockchain: A distributed solution to automotive security and privacy. *IEEE Commun. Mag.* **2017**, *55*, 119–125. [CrossRef]
30. Karaarslan, E.; Adiguzel, E. Blockchain Based DNS and PKI Solutions. *IEEE Commun. Stand. Mag.* **2018**, *2*, 52–57. [CrossRef]
31. Lu, Y. Blockchain and the related issues: A review of current research topics. *J. Manag. Anal.* **2018**, *5*, 231–255. [CrossRef]
32. Provos, N.; Mazieres, D. A Future-Adaptable Password Scheme. In Proceedings of the USENIX Annual Technical Conference, FREENIX Track, Monterey, CA, USA, 6–11 June 1999, 1999; pp. 81–91.
33. Gudgeon, L. SoK: Off The Chain Transactions. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 360.
34. Dziembowski, S.; Eckey, L.; Faust, S.; Hesse, J.; Hostáková, K. Multi-party virtual state channels. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 19–23 May 2019; pp. 625–656.
35. François, J.; Aib, I.; Boutaba, R. FireCol: A collaborative protection network for the detection of flooding DDoS attacks. *IEEE/ACM Trans. Netw. (TON)* **2012**, *20*, 1828–1841. [CrossRef]

36. Chen, Y.; Hwang, K.; Ku, W.S. Collaborative detection of DDoS attacks over multiple network domains. *IEEE Trans. Parallel Distrib. Syst.* **2007**, *18*, 1649–1662. [CrossRef]
37. Chin, T.; Mountrouidou, X.; Li, X.; Xiong, K. An SDN-supported collaborative approach for DDoS flooding detection and containment. In Proceedings of the MILCOM 2015-2015 IEEE Military Communications Conference, Tampa, FL, USA, 26–28 October 2015; pp. 659–664.
38. Patil, N.V.; Krishna, C.R.; Kumar, K.; Behal, S. E-Had: A distributed and collaborative detection framework for early detection of DDoS attacks. *J. King Saud Univ.-Comput. Inf. Sci.* **2019**, in press. [CrossRef]
39. Sagirlar, G.; Carminati, B.; Ferrari, E. AutoBotCatcher: Blockchain-based P2P Botnet Detection for the Internet of Things. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 1–8.