IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 7, 2019, accepted July 19, 2019, date of publication July 24, 2019, date of current version August 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930715

Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract

ZAKARIA ABOU EL HOUDA[1,2], **ABDELHAKIM SENHAJI HAFID**[1], **AND LYES KHOUKHI**[2]

[1]Department of Computer Science and Operational Research, University of Montreal, Montreal, QC H3T 1J4, Canada
[2]ICD/ERA, University of Technology of Troyes, 10300 Troyes, France

Corresponding author: Zakaria Abou El Houda (zakaria.abou.el.houda@umontreal.ca)

ABSTRACT With the exponential growth in the number of insecure devices, the impact of Distributed Denial-of-Service (DDoS) attacks is growing rapidly. Existing DDoS mitigation schemes are facing obstacles due to low flexibility, lack of resources, and high cost. The new emerging technologies, such as blockchain, introduce new opportunities for low-cost, efficient and flexible DDoS attacks mitigation across multiple domains. In this paper, we propose a blockchain-based approach, called Cochain-SC, which combines two levels of mitigation, intra-domain and inter-domain DDoS mitigation. For intra-domain, we propose an effective DDoS mitigation method in the context of software defined networks (SDN); it consists of three schemes: (1) Intra Entropy-based scheme (I-ES) to measure, using sFlow, the randomness of data inside the domain; (2) Intra Bayes-based scheme (I-BS) to classify, based on entropy values, illegitimate flows; and (3) Intra-domain Mitigation (I-DM) scheme to effectively mitigate illegitimate flows inside the domain. For inter-domain, we propose a collaborative DDoS mitigation scheme based on blockchain; it uses the concept of smart contracts (i.e., Ethereum's smart contracts) to facilitate the collaboration among SDN-based domains (i.e., Autonomous System: AS) to mitigate DDoS attacks. For this aim, we design a novel and secure scheme that allows multiple SDN-based domains to securely collaborate and transfer attack information in a decentralized manner. Combining intra- and inter-domain DDoS mitigation, Cochain-SC allows an efficient mitigation along the path of an ongoing attack and an effective mitigation near the origin of the attack. This allows reducing the enormous cost of forwarding packets, across multiple domains, which consist mostly of useless amplified attack traffic. To the best of our knowledge, Cochain-SC is the first scheme that proposes to deal with both intra-domain and inter-domain DDoS attacks mitigation combining SDN, blockchain and smart contract. The implementation of Cochain-SC is deployed on Ethereum official test network Ropsten. Moreover, we conducted extensive experiments to evaluate our proposed approach; the experimental results show that Cochain-SC achieves flexibility, efficiency, security, cost effectiveness, and high accuracy in detecting illegitimate flows, making it a promising approach to mitigate DDoS attacks.

INDEX TERMS DDoS, entropy, Bayes classifier, SDN, intra-domain mitigation, inter-domain collaboration, blockchain technology, smart contracts.

I. INTRODUCTION

In recent years, security threats of DDoS attacks have been increasing causing severe collateral damage to network operators as well as Internet service providers (ISPs). Some recent dramatic incidents (e.g., DDoS attacks against US banks [1]) show that DDoS attacks are becoming powerful, devastating and more destructive plaguing network operators and ISPs in a stealthy way [2]. Indeed, the rapid growth in the number of insecure devices, with an estimated 50 billion devices by the end of 2020 [3], can facilitate and enhance the capability of attacks. For example, on the 2nd of October 2016, a huge and dramatic attack, exceeding a rate of 1 Tbit/s, was conducted against Dyn Domaine Name System (DNS) services. As a consequence, many popular Internet services, e.g., Amazon and GitHub [4] became disconnected for several hours [5]. Such incidents damage ISPs and cost millions of dollars of lost revenues for enterprises [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Raúl Lara-Cabrera.

Recently, SDN has attracted tremendous attention as a novel technology that facilitates network management and provides new capabilities to manage and deploy networks dynamically [7]–[10]. SDN separates data and control planes; this separation allows for more control over the network and brings a new way to deal with various form of DDoS attacks [11].

To deal with DDoS attacks inside a domain (i.e., AS), we propose an effective DDoS mitigation method in the context of SDN; it consists of 3 schemes: (1) Intra Entropy-based scheme (I-ES) to measure, using sFlow [12], the randomness of data inside the SDN based domain; (2) Intra Bayes-based scheme (I-BS) to classify, based on entropy values, illegitimate flows; and (3) Intra-domain Mitigation (I-DM) scheme to effectively mitigate illegitimate flows inside the domain. However, an intra-domain DDoS mitigation alone cannot mitigate DDoS attacks on the paths to the victim because the attackers and the target are not necessarily in the same AS (see Fig. 1). Thus, there is a need to an efficient inter-domain collaboration to: (a) effectively reduce the enormous cost of forwarding packets, across multiple domains, which are mostly useless amplified attack traffic; and (b) block the attack close to its source. Existing collaborative DDoS mitigation schemes suffer from low flexibility and high cost; more importantly, they are centralized. The centralized solution, by its nature, causes single-point-of-failure and is vulnerable to DDoS attacks that can make it difficult, or even infeasible, to share information, among ASs, and make effective decisions to mitigate the attacks.

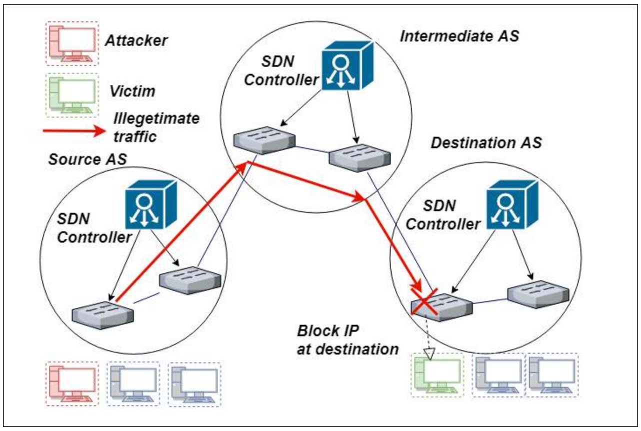

FIGURE 1. **Concept of DDoS attacks across multiple SDN domains.**

The new emerging technologies, such as blockchain and smart contract, open new opportunities for low-cost, efficient and flexible collaboration across multiple ASs to mitigate DDoS attacks. Indeed, blockchain has been shown to provide a decentralized collaboration in trustless network environments. Blockchain technology (e.g., Bitcoin [13], Ethereum [14] and Zcash [15]) has proven its effectiveness and success in achieving high level of security and transparency in financial field [16] as well as non-financial fields (e.g., Healthcare [17], decentralized IoT [18], decentralized data sharing [19], and online advertising [20]). The inter-domain collaboration process may exploit the power of the decentralized approach by replicating data on each network's node, thus enforcing integrity and reliability of the whole ecosystem.

For inter-domain, we propose a collaborative DDoS mitigation scheme based on blockchain. This scheme uses the concept of smart contracts (i.e., Ethereum's smart contracts) to facilitate the collaboration among SDN-based domains (i.e., AS) to mitigate DDoS attacks. We design a novel and secure scheme that allows multiple SDN-based domains to securely collaborate and transfer attack information in a decentralized manner. This allows an SDN-based domain to effectively collaborate and inform its neighboring domains about ongoing DDoS attacks. Thus, by combining our scalable intra-domain mitigation approach based on SDN and inter-domain DDoS mitigation scheme, we are not only able to efficiently mitigate DDoS attacks within the target's domain, but also to share attack information in a fully decentralized manner resulting in effective decision making, by multiple domains, to mitigate against DDoS attacks.

This paper presents the design, specification and implementation of a blockchain-based approach called Cochain-SC in which two levels of mitigation are combined (i.e., intra-domain and inter-domain DDoS mitigation). Cochain-SC provides an efficient mitigation along the path of an ongoing attack and effective mitigation near of source of the attack. The implementation of Cochain-SC is deployed on Ethereum official test network Ropsten [21], an open blockchain platform.

The main contributions of this paper can be summarized as follows:

- We design a decentralized secure DDoS collaboration scheme (Cochain-SC) based on blockchain using smart contract; it supports two levels of mitigation: intra-domain and inter-domain DDoS mitigation.
- We propose a smart contract-based scheme, that makes use of Ethereum's smart contract technology, to realize a decentralized secure, flexible and low-cost collaboration, among multiple SDN-based domains, to mitigate against DDoS attacks.
- We propose an Intra Entropy-based scheme (I-ES), to measure the randomness of data inside the domain using sFlow.
- We propose an Intra real-time detection scheme, called Intra Bayes-based scheme (I-BS), to automatically detect network traffic anomalies inside the domain.
- We propose an Intra-Domain Mitigation (I-DM) scheme to effectively mitigate illegitimate flows inside the domain.
- We evaluate the performance of Cochain-SC in terms of flexibility, efficiency, security and cost effectiveness. The experiments results show that Cochain-SC can effectively mitigate the attack inside the domain with high accuracy, low false positive rate and low overhead, and achieves the requirements of the new generation

of flexible, secure, efficient and low-cost inter-domain DDoS collaboration schemes.

The rest of the paper is organized as follows. Section II presents related work. Section III presents an overview of Cochain-SC. Section IV presents our intra-Domain DDoS mitigation approach, which consists of intra Entropy-based scheme (I-ES), Intra Bayes-based scheme (I-BS) and Intra-Domain Mitigation (I-DM) scheme. Section V presents inter-domain DDoS collaboration scheme. Section VI presents the implementation of Cochain-SC. Section VII evaluates Cochain-SC. Finally, Section VIII concludes the paper.

II. BACKGROUND AND RELATED WORK

Several schemes have been proposed in the literature to mitigate DDoS attacks. In this section, we briefly introduce DDoS attacks. Then, we overview existing DDoS mitigation schemes that can be classified in two categories: (1) Intra-domain DDoS mitigation schemes; and (2) Inter-domain DDoS mitigation schemes.

A. DDOS ATTACKS TECHNIQUES

DDoS attacks aim to overwhelm target's resources, such as network bandwidth and computer CPU, with illegitimate flows. To launch DDoS attacks, the attacker (e.g., bot master), generally, needs to control a large number of compromised devices (called zombies). Each zombie sends a huge volume of illegitimate traffic, to deny services, to legitimate users of the target (see Fig. 2). There are two major categories of DDoS attacks: real source IP-based attacks (typical DDoS attacks) and DRDoS (Distributed Reflection Denial of Service) attacks. In a typical DDoS attack, the bot master orders a large number of zombies to directly flood the target. DRDoS attacks consist of bot master, zombies and reflectors. Each zombie is ordered by the bot master to send a large number of packets, in which the source IP address is replaced with the victim's IP address, to other devices known as reflectors; upon receipt of these packets, the reflectors send the victim a huge volume of illegitimate traffic. Our previous works [22]–[24] did show their effectiveness in protecting permissioned blockchain against DRDoS attacks; in this paper, we consider typical DDoS attacks. Moroever, we combine an intra-domain DDoS mitigation scheme using machine learning scheme (i.e., I-BS) to enhance the accuracy of the detection model and an inter-domain DDoS collaboration allowing for an efficient mitigation along the path of an ongoing attack and effective mitigation near to the origin of attack.

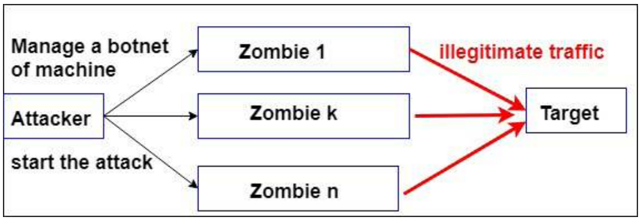

FIGURE 2. DDoS attacks.

B. INTRA DDOS MITIGATION SCHEMES

Several intra DDoS mitigation schemes have been proposed; in the following, we present some of the most prominent as well as their limitations.

The BCP 38 standard [25] proposes a filtering method that requires every ISP to: (a) verify that the packets from its network use valid prefixes (IP addresses); and (b) filter these packets that use forged source addresses that are outside its range of legitimate addresses. This type of solution has proven to be effective against IP-spoofing attacks. However, it does absolutely nothing to deal with real source IP-based attacks. In [26], Rodrigo et al. proposed a flow-based intrusion detection scheme (i.e., Self Organizing Maps) using OpenFlow (OF) protocol to gather traffic flow statistics. This scheme [26] did not consider the overhead to the control plane caused by the flow collecting process using OF protocol; moreover, performance analysis does not include the overall system performance. In [27], Mehdi et al. proposed an anomaly detection scheme in the context of SDN using OF protocol. However, the scheme was focused only on the home environment (i.e, small-scale setup); in large-scale environments, a high rate data traffic to SDN controller may overload the control plane. In [28], Yu et al. proposed OpenSketch, a software defined traffic measurement scheme. OpenSketch provides an efficient method to collect measurement data through a three-stage pipeline (hashing, filtering, and counting). However, OpenSketch sends all the counters to the SDN controller for analysis, which may overload the control plane. In [29], Wang et al. proposed an entropy-based scheme in OF switches; it focuses only on detection, but it cannot find the victim or the illegitimate hosts to, eventually, block them. In [30], Lim et al. proposed a DDoS mitigation scheme for botnet-based attacks that runs on SDN controller; it requires a large amount of communications between the control plane and data plane to protect the victim. Moreover, this scheme [30] not only makes SDN controller vulnerable to DDoS attacks against the SDN controller but also requires high latency to cooperate with SDN controller. In [31], K. Giotis et al. combined sFlow protocol with OF protocol in order to detect DDoS attacks reducing the communication overhead between data plane and control plane. This scheme [31] works well; however, it has high false positive Rate.

To address the weaknesses of existing solutions [25]–[31], we propose an efficient and scalable intra-domain DDoS mitigation scheme to detect and mitigate DDoS attacks. In our scheme, we combine entropy calculation using sFlow with SDN functionalities to block illegitimate traffic. The proposed solution employs sFlow protocol to separate flow monitoring from the forwarding logic; this makes it much more scalable compared to existing native OF schemes [25]–[31]. And using machine learning scheme (i.e., I-BS), our intra-domain DDoS mitigation scheme is much accurate in comparison with the ones using sampling technology [31].

C. INTER DDOS MITIGATION SCHEMES

As DDoS attacks evolve rapidly and become more devastating, cooperation among several domains has become necessary to ensure sophisticated mitigation in order to cope with large scale DDoS attacks.

Several mitigation schemes have been proposed to effectively collaborate in order to deal with DDoS attacks. However, only a few of them have been considered for widespread deployment because of their implementation complexity and limited effectiveness. Guo et al. [32] proposed a mechanism that deploys filters at the border of the network to block incoming packets with source IP addresses that do not belong to the network. However, the effectiveness of this method depends on global deployment across the Internet. This method has ''neighborhood policy'' that requires every ISP to participate in order to provide the list of IP addresses that does not belong to its network. In [33], IETF (Internet Engineering Task Force) proposed the development of a new collaborative protocol called DOTS (DDoS Open Threat Signaling) in order to advertise DDoS attacks. The inter-domain collaboration scheme used in DOTS leverages the mitigation by sharing resources among organizations. DOTS protocol consists of customers (DOTS clients) and controller (DOTS server). When an attack is detected, the customer requests the mitigation service from the controller which is responsible for inter-domain communication and coordination. The implementation of the inter-domain DDoS mitigation can be either distributed or centralized. However, the effectiveness of DOTS depends on global deployment which may be disregarded due to implementation complexity. Moreover, the process of collaboration can be easily compromised. To counter this, digital certificates can be used; however, it is costly to setup and maintain certificates. In addition, if centralized implementation is used, it will cause single-point-of-failure.

In [34], Steinberger et al. proposed a similar scheme to DOTS [33]; it uses flow-based event exchange format to simplify the deployment and the collaboration between domains. This scheme [34] also requires a secure public-key infrastructure (PKI). PKI-based systems are costly to setup and maintain. In [35], Giotis et al. proposed a collaborative DDoS mitigation scheme across multiple SDN based domains. They extend Border Gateway Protocol (BGP) protocol to repost incidents as URIs in BGP signals. However, any modification to BGP is a challenging endeavor. Moreover, the incident report latency may be large given that domains do not report in real-time. More importantly, this scheme [35] does not verify the authenticity of incident reports resulting in a scheme that is vulnerable to spoofed incident reports from illegitimate domains. In [36], Bahman et al. proposed CoFence, a DDoS defense mechanism that facilitates collaboration among network function virtualization (NFV) based domains. In CoFence, when a NFV-based domain is under attack, it redirects the traffic to other NFV-based domains to filter the packets. First, CoFence has a privacy issue since it redirects the traffic to other NFV-based domains. Moreover, this process of redirection will also increase incident report latency. Many other schemes have been proposed (e.g., [6], [37]); however, the complexity of deployment and overhead, that is generated, remain challenging issues in these schemes. In [38], Rodrigues et al. proposed a scheme, which uses blockchain and smart contracts, to advertise blacklisted IP addresses. However, this scheme [38] requires a central entity to issue certificates of ownership of IP addresses, when calling (i.e., storing data) smart contracts. Moreover, it is concerned only with inter-domain DDoS mitigation and not with intra-domain DDoS mitigation. In [39], Mathis et al. proposed an OpenFlow-based firewall to provide security to blockchain nodes. They implemented their solution as a module, in SDN controller, that uses SDN functionalities to filter network traffic; it provides access control functionality and protects blockchain nodes from DoS attacks. However, a very high packet rate from switches to SDN controller may overload the control plane.

To address the shortcomings of existing solutions [32]–[39], we propose an efficient, secure, low-cost, easy-to-deploy and decentralized inter-domain collaboration scheme; it allows multiple SDN based domains to securely collaborate and transfer attack information (i.e., suspicious IP addresses) in a decentralized manner based on blockchain using smart contract. The use of new technologies (e.g., SDN, blockchain and smart contract) introduces new opportunities for secure, efficient and flexible DDoS attacks collaboration across multiple SDN based domains. Indeed, with these technologies, one can avoid the complexity of developing new protocols and/or the modification of existing ones (e.g., [35]); in addition, it removes the need to use a central entity in contrast to [38] and enforces permissions to participate in the collaboration.

III. COCHAIN-SC: AN OVERVIEW

In this section, we present an overview of Cochain-SC. More specifically, we explain how Cochain-SC can combine two levels of mitigation, intra-domain and inter-domain DDoS mitigation, allowing for an efficient mitigation along the path of an ongoing attack and an effective mitigation near to the origin of attack. Inter-domain DDoS mitigation scheme assumes multiple SDN based domains (e.g., ASs: A, B, C, D, E and F) to collaborate as shown in Fig. 3. SDN based domains communicate with each other via our proposed inter-domain collaboration scheme based on blockchain using smart contract. First, the organization (i.e., owner of the smart contract) needs to create the collaboration contract. Then, it adds the authorized participants (i.e., collaborators). Therefore, when attackers, which are distributed across multiple domains, generate an attack towards the victim (e.g., hosted at AS C; see Fig. 3), our intra-domain DDoS mitigation scheme detects and mitigates the attack inside the domain; it also stores the suspicious IP address, denoted ip_address, in the smart contract. When the next block is

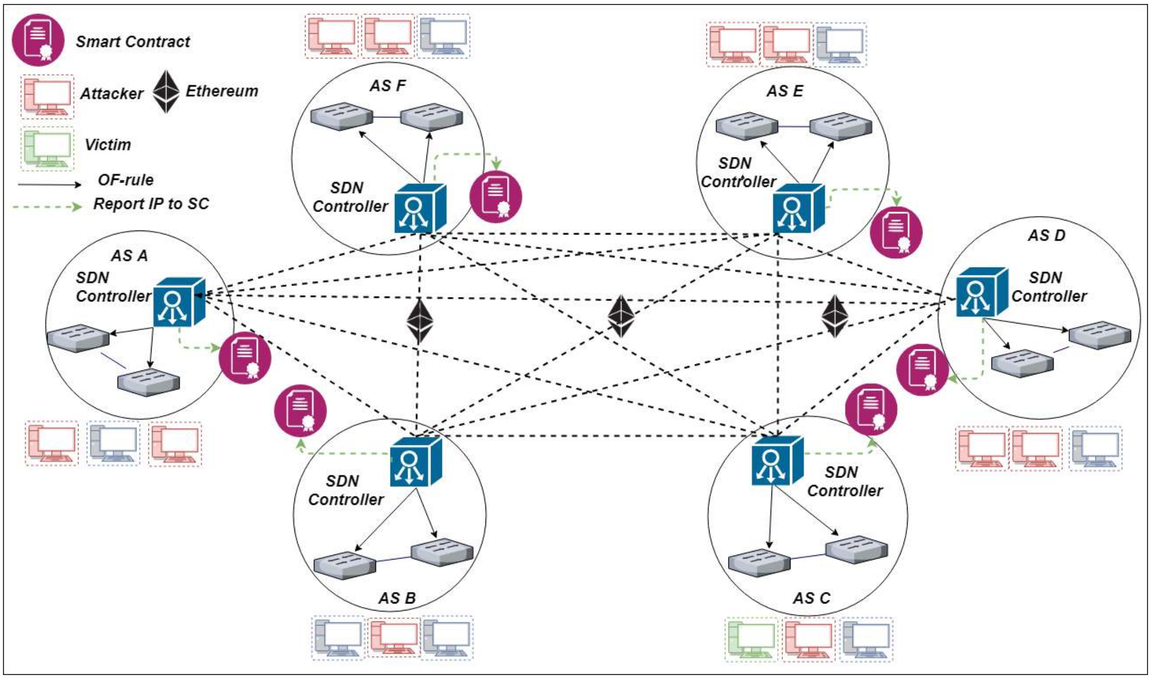

FIGURE 3. Cochain-SC blockchain-based framework.

mined, each of the authorized participants of the collaborative scheme will have access to the list of suspicious IP addresses to be blocked; this will allow for an efficient mitigation along the path of an ongoing attack and an effective mitigation near to the origin of attack. To report/receive attack informations, each SDN based domain runs an Ethereum client (e.g., geth client [40]). In the following, we describe Cochain-SC in more details. First, we describe our intra-domain DDoS mitigation scheme which is composed of 3 schemes (i.e., I-ES, I-BS and I-DM). Then, we describe our inter-domain DDoS mitigation scheme.

IV. INTRA-DOMAIN DDOS MITIGATION SCHEME

A. DESIGN OVERVIEW

When designing the intra-domain DDoS mitigation scheme, we did consider the following objectives: (a) the scheme should ensure a full protection and should be accurate in detecting attacks inside the domain; this is ensured via a robust detection scheme (i.e., I-BS), based on Intra Entropy-based scheme (I-ES); (b) the attacks should be effectively mitigated using Intra-Domain Mitigation (I-DM) scheme; and (c) the method should be scalable.

B. SYSTEM ARCHITECTURE

The architecture of intra-domain DDoS mitigation method consists of four main modules (see Fig. 4): (1) Intra Entropy-based scheme (I-ES) to measure the randomness of data inside the domain using sFlow; (2) Intra Bayes-based scheme (I-BS) to classify, using entropy values, illegitimate flows; (3) Intra-Domain Mitigation (I-DM) scheme to effectively mitigate illegitimate flows inside the domain; and (4) blockchain layer, used in inter-domain DDoS mitigation (see Section V).

Intra-domain DDoS mitigation scheme has two main phases : (1) an intra-domain machine learning DDoS detection and mitigation module. This module has the objective to detect, in real-time, illegitimate flows and consists of I-ES, I-BS and I-DM ; and (2) blockchain module, used in inter-domain DDoS mitigation (see Section V). I-ES aims to measure the randomness of data inside the victim's domain (e.g., AS C; see Fig. 3) using network traffic flow features. I-BS has the objective to detect, in real-time, illegitimate flows based on stateful network traffic features (I-ES calculation). It is running as an application on the top of the SDN controller (i.e., application layer) and is using entropy values; it collects traffic information and detects automatically illegitimate flows. We use also in our process of detection/mitigation, the REST API [41] to manage any SDN controller and block illegitimate traffic. I-DM aims to effectively mitigate illegitimate traffic inside the domain. OF was not designed to support QoS features; however, OF 1.3 introduces *m*eters [42] to the OF protocol (see Section IV.D). Each flow entry specifies *meter*; *meter* entries with different *Meter_id* are deployed to monitor the speed of the classified illegitimate flows by I-BS; if the flow rate exceeds *band* (rate limiter), I-DM drops suspected flows.

C. MACHINE LEARNING DDOS DETECTION AND MITIGATION MODULE

In this section; first, we describe the details of our information collection method based on flow packet sampling using sFlow; then, we describe I-ES, to measure the randomness of

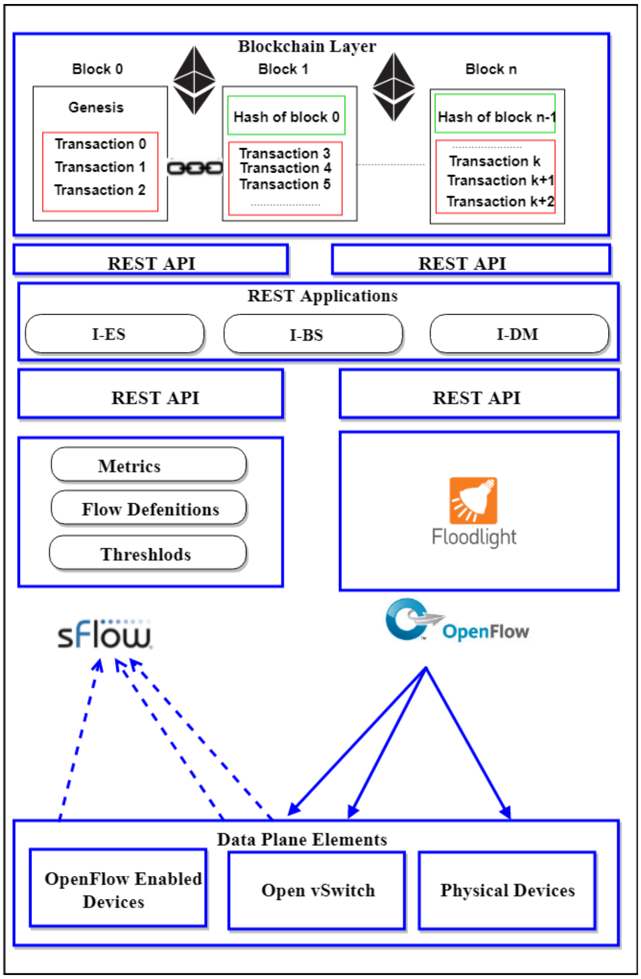

FIGURE 4. System architecture.

data inside the domain and extract network features; finally, we describe I-BS, to detect illegitimate flows.

1) FLOW STATISTICS COLLECTION

There are two commonly methods for collecting information: the first method is based on OF protocol and the second method is based on flow monitoring. In this paper, we choose to use flow monitoring methods based on sFlow protocol. In the following, we describe each of these methods and justify our choice.

To detect DDoS attacks in SDN, most existing solutions [25]–[31] propose to collect and send, periodically, the features of the flows (e.g., number of received packets and duration of matched flows) to SDN controller using OF protocol. Collection of features, using OF protocol, can be initiated when SDN controller sends a feature request *(ofp_flow_stats_request)* to OF switches which respond by sending the flow table content *(ofp_flow_stats_reply)*. This method can collect the overall traffic of flow information passing through the data plane. However, this method can overload the control plane and exhausts the bandwidth between OF controller and OF switches; furthermore, it may exhaust Ternary Content Addressable Memory (TCAM) in OF switches. Therefore, OF based method is not adequate for detecting high rate DDoS attacks.

To address the shortcomings of the method described above, we decided to monitor flows using flow monitoring method based on sFlow protocol. This is more efficient, scalable and does not consume bandwidth between SDN controller and OF switches. sFlow performs flow aggregation that is required during DDoS attacks when the number of flow entries is very high. The sFlow collector (sFlow-RT [43]) receives periodically packet samples from each sFlow agent embedded in data plane (data plane devices) and updates the counters of each flow during the monitoring interval. Afterwards, periodically, I-ES calculates entropy values and I-BS detects automatically illegitimate flows.

2) I-ES

Table 1 shows the list of notations used to describe I-ES. The basic idea behind I-ES comes from Shannon's information theory [44]. The entropy calculation measures the disorder/randomness of incoming data (i.e., the incoming flow for a given time period). I-ES runs as an application on the top of the controller and uses sFlow protocol; it collects traffic information and computes the entropy of each flow. When a victim's domain (e.g., AS C, see Fig.3) is under DDoS attacks, the number of packets that have the same IP address destination, denoted ipdst (i.e., victim's IP address) increases resulting in a concentrated distribution of ipdst; while the normal state of victim's network leads to a more dispersed probability distribution of ipdst. High entropy values mean highly dispersed probability distribution of ipdst, while low entropy values mean a concentration of ipdst. Therefore, we use I-ES to measure the changes of traffic information inside the victim's domain during monitoring interval ΔT.

TABLE 1. Notations.

Notations	Definition
MAC_{src}	The MAC source address of a packet
MAC_{dst}	The MAC destination address of a packet
IP_{src}	The IP source address of a packet
IP_{dst}	The IP destination address of a packet
$Port_{src}$	The $Port$ source of a packet
$Port_{dst}$	The $Port$ destination of a packet
IP_{proto}	The transport protocol of a packet (TCP/UDP)
S_j	OF switch S_j
ΔT	The monitoring interval
$F_{i,j}$	Flow f_i at a local OF switch S_j
$p_{i,j}$	The probability of flow f_i over all flows at local OF switch S_j
N	The total number of flows at local OF switch S_j
R	Set of real numbers
I	Set of positive integers

In this work, we define a flow as a seven tuple: $\{MAC_{src}, MAC_{dst}, IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, Proto\}$

Let $F_{i,j}$ denote flow f_i at local OF switch S_j; it is defined as follows:

$$F_{i,j}(IP_{dst_i}, S_j) = \{< IP_{dst_i}, S_j, t > | S_j \in S, i, j \in I, t \in R\} \tag{1}$$

where IP_{dst} is the destination IP address of f_i, t it the current timestamp, and $S = \{S_j, j \in I\}$ the set of OF switches.

Let $|F_{i,j}(IP_{dst_i}, S_j, t)|$ be the count number of packets of flow $F_{i,j}$ at time t. The variation of the number of packets for flow f_i at local OF switch S_j during ΔT is defined as follows:

$$\begin{aligned} &|N_{F_{i,j}}(IP_{dst_i}, S_j, t + \Delta T)| \\ &\quad = |F_{i,j}(IP_{dst_i}, S_j, t + \Delta T)| - |F_{i,j}(IP_{dst_i}, S_j, t)| \end{aligned} \tag{2}$$

The probability $p_{i,j}$ of flow f_i over all flows at local OF switch S_j is expressed as follows:

$$p_{i,j}(IP_{dst_i}, S_j) = \frac{N_{F_{i,j}}(IP_{dst_i}, S_j, t + \Delta T)}{\sum_{i=1}^{N} N_{F_{i,j}}} \tag{3}$$

where $\sum_{i=1}^{N} p_{i,j}(IP_{dst_i}, S_j) = 1$.

Let IP_{dst} be a random variable that represents the number of flows during the time interval ΔT. We define the entropy of flow f_i at local OF switch S_j as follows:

$$H(IP_{dst}) = -\sum_{i=1}^{N} p_{i,j}(IP_{dst_i}, S_j) \log_2 p_{i,j}(IP_{dst_i}, S_j) \tag{4}$$

In order to have a measurement metric that is independent from the number of distinct values, we divide the entropy values by the upper bound value that is $\log_2 N$. Therefore, the normalized entropy values are between [0, 1] and are defined as below:

$$H(IP_{dst})' = \frac{H(IP_{dst})}{\log_2 N} \tag{5}$$

The attribute (e.g., IP_{src}, IP_{dst}) to aggregate flows depends on the attack (e.g., DRDoS, DDoS) under investigation. When the network suffers from DDoS attack, the number of the flows that have the same destination IP address IP_{dst} (i.e., target of attack) increases sharply leading to a significant decrease of entropy values; while the source IP addresses entropy values are relatively concentrated. Attackers generates illegitimate traffic from the same UDP/TCP port source number, as legitimate users generate legitimate traffic from random UDP/TCP port source number; therefore, this attribute also can better represent the DDoS attack characteristics. Consequently, we use $\{IP_{dst}, Port_{src}$ and $Port_{dst}\}$ as the attribute to aggregate flows. Finally, we represent the network traffic features at the k^{th} time period as:

$$X_k = \{H(IP_{dst})'_k, H(Port_{src})'_k, H(Port_{dst})'_k\} \tag{6}$$

3) I-BS

I-BS is a binary classifier in the field of machine learning; it uses stateful traffic features (i.e., traffic features vector X_k) in order to classify vector X_k as either legitimate or illegitimate. I-BS receives vector X_k and classifies it using the probability of illegitimacy (see Section 3.b). Once X_k is classified as illegitimate, I-BS notifies I-DM to deploy the mitigation action against this illegitimate vector X_k. In the following, we detail I-BS; first, we briefly describe the flow representation; then, we discuss in more details the criterion classification of I-BS.

a: FLOW REPRESENTATION

In I-BS, each sample is represented by a vector $x = (x_1, x_2, x_3)$ where x_1, x_2, x_3 are values taken, respectively, by random variables $H(IP_{dst})'$, $H(Port_{src})'$ and $H(Port_{dst})'$. Each of these random variables indicates, respectively, entropy values of destination IP address, UDP/TCP port source and UDP/TCP port destination.

b: CRITERION OF CLASSIFICATION

I-BS considers two classes of vectors: (1) legitimate vectors denoted by *leg* and (2) illegitimate vectors denoted by *illeg*. The class of k^{th} vector X_k, denoted by c, can be either *leg* or *illeg* and is represented as follows:

$$c = \underset{c \in \{leg, illeg\}}{\arg\max}\ p(c|X_k)$$

We have $p(leg|X_k) + p(illeg|X_k) = 1$; thus, the selection criterion is defined as follows:

$$X_k \text{ is illegitimate iff } p(illeg|X_k) \geq 0.5 \tag{7}$$

Using Bayes theorem [45], the probability of vector X_k to belong to class c is defined as follows:

$$p(C = c|X = X_k) = \frac{p(C = c).p(X = X_k|C = c)}{p(X = X_k)} \tag{8}$$

According to the total probability theorem, we have:

$$\begin{aligned} &p(C = c|X = X_k) \\ &\quad = \frac{p(C = c).p(X = X_k|C = c)}{\sum_{c \in \{leg, illeg\}} p(C = c)p(X = X_k|C = c)} \end{aligned} \tag{9}$$

Therefore, the selection criterion is equivalent to: X_k is illegitimate iff

$$\begin{aligned} &p(C = c|X = X_k) \\ &\quad = \frac{p(C = c).p(X = X_k|C = c)}{\sum_{c \in \{leg, illeg\}} p(C = c)p(X = X_k|C = c)} \geq 0.5 \end{aligned} \tag{10}$$

$H(IP_{dst})'$, $H(Port_{src})'$ and $H(Port_{dst})'$ are conditionally independent variables given class c. Let $p_k(leg)$ and $p_k(illeg)$ denote the conditional probabilities that the k^{th} vector X_k is receptively legitimate and illegitimate. Using Eq.(10), the selection criterion can be expressed as follows: X_k is illegitimate iff (11), as shown at the bottom of the next page.

When $p(leg) = p(illeg)$, the selection criterion becomes package: X_k is illegitimate iff (12), as shown at the bottom of this page.

I-BS is trained and then used to classify k^{th} vector X_k as either legitimate or illegitimate. By combining I-ES and I-BS, Cochain-SC can accurately detect the attack in real time with low False positive rate while maintaining a high detection Rate (see Section VII).

D. INTRA-DOMAIN MITIGATION (I-DM) SCHEME

When I-BS detects DDoS attack, a mitigation action is performed to protect the victim. For this aim, new OF rules are installed, using the API of the SDN controller, into the OF switch under attack; these rules have a high priority to match suspicious packets and monitor their speed. I-DM has the purpose to effectively mitigate illegitimate traffic. A flow entry can specify a meter; meter entries with different Meter_id are deployed to monitor the speed of the classified illegitimate flows by I-BS; if the packet rate surpasses the band (rate limiter), then we drop suspected packets (see Fig. 5). In our simulations, we set an adaptive band; at the beginning the band is fixed to 1000 packets per second, then it can be adjusted based on the capacity of OF table, traffic rate and workload of both data and control plane.

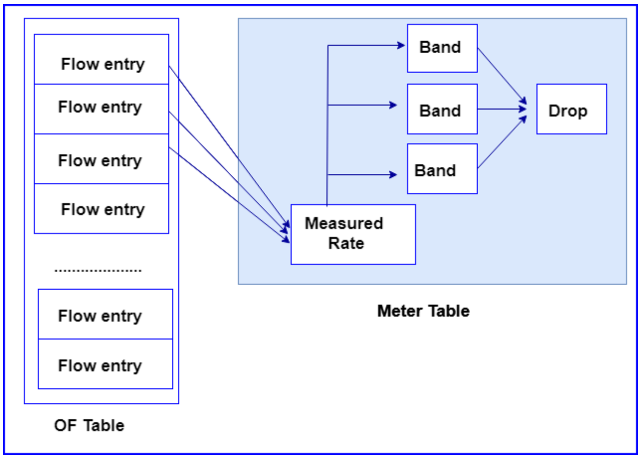

FIGURE 5. Table model in OF switches.

V. INTER-DOMAIN DDOS MITIGATION SCHEME

A. OVERVIEW

This scheme has the objective to ensure sophisticated mitigation across multiple domains and cope with large scale DDoS attacks. Inter-domain DDoS mitigation assumes multiple SDN based domains to collaborate. Fig. 6 shows the high-level architecture of our inter-domain DDoS mitigation scheme. ASs (SDN based domains) are classified into 3 types of network domains, source domain, intermediate network domains and destination domain. The source domain is the network (i.e., AS) in which the attacker starts the attack. Intermediate network domains forward illegitimate traffic. The destination domain is the domain where the victim is hosted. The SDN controller of each AS can either report or retrieve the list of illegitimate IPs (see Fig. 6). We have leveraged SDN and blockchain to both mitigate the attack inside the domain (as presented in Section IV) and effectively collaborate to: (a) reduce the enormous cost of forwarding packets, across multiple domains, that compose amplified attack traffic; and (b) block the attack close to its source.

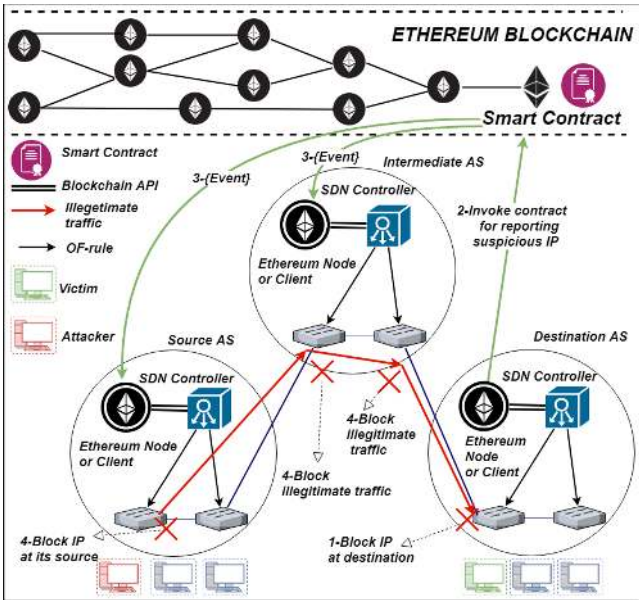

FIGURE 6. High-level architecture of inter-domain DDoS mitigation scheme.

Ethereum blockchain is a decentralized platform for developing smart contracts. The language to write these contracts is Turing complete (e.g., Solidity [46]); this allows users to create and run smart contracts (of any complexity) on the blockchain. From the computing prospective, bitcoin network provides distributed data storage for managing transactions while the Ethereum network not only provides distributed

$$\frac{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k}p(illeg)}{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k}p(illeg)+\prod_{k=1}^{n} p_k^{X_k}(leg)(1-p_k(leg))^{1-X_k}p(leg)} \geq 0.5 \tag{11}$$

$$\frac{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k}}{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k}+\prod_{k=1}^{n} p_k^{X_k}(leg)(1-p_k(leg))^{1-X_k}} \geq 0.5 \tag{12}$$

data storage but also offers computing resources; each node of the Ethereum network executes the code (e.g., smart contract) that is deployed by participants using Ethereum Virtual Machine (EVM). Fig. 6 shows the main steps executed by the proposed approach: (1) Once our intra-domain DDoS mitigation scheme detects the attack, using I-ES and I-BS, it mitigates the attack inside the domain using I-DM and blocks the illegitimate traffic; (2) The SDN controller of the domain under the attack (Destination AS), sends a transaction, if it is authorized, to the smart contract in order to report suspicious IP adresses; (3) once the transaction is confirmed (i.e., the transaction is in the new block appended to the blockchain), an event is emitted by the contract and is received by the registered/authorized collaborators of the smart contract (e.g., SDN controller of source and intermediate network domains); and (4) upon receipt of the event, the collaborators block the illegitimate traffic.

B. COCHAIN-SC'S SMART CONTRACT

1) APPLICATION SCENARIO

We consider an organization (i.e., contract owner) that would like to manage a collaboration process between different ASs around the world. First, it creates Cochain-SC's smart contract and deploys it on the Ethereum blockchain. The use of blockchain in the collaboration process allows for transparency while maintaining “pseudonymity". Then, the organization adds, via the smart contract, the collaborators into the system. It includes the collaborator's address and some other information (e.g., collaborator notes). When a collaborator (e.g., victim's network) detects and mitigates an attack, it sends a transaction to the smart contract to add the suspicious IP address. The smart contract allows (1) the organization (owner of the contract) to add collaborators to the contract; (2) the organization to manage and modify the collaboration process in a transparent manner; (3) the organization to delete collaborators from the collaboration process if needed; (4) collaborators to report suspicious IP address in a secure and efficient manner; (5) collaborators to delete the reported IP address from the contract if needed. In the following, we present the design of our smart contact.

2) COCHAIN-SC'S DESIGN

The smart contract is programmed using solidity language. In our smart contract, we use the following global variables: now: the time passed in seconds since 1970; msg.sender: the msg object represents the transaction that is sent; and the msg.sender is the address of the user that sent the transaction to the smart contract. In Ethereum, there are two types of accounts: (1) Externally Owned Account (EOA): it has public and private keys; it can send transactions to transfer ether or to smart contracts and (2) contract account: it runs code and has no public/private key. The smart contract, called Collaboration contract, is deployed by the organization. In the following, we first describe the Collaboration Contract Initialization; then, we provide the functions of the Collaboration Contract. Collaboration Contract Initialization: This process defines the state variables of the contract.

1) The Collaboration Contract Owner of address types, which defines the address of organization responsible of the collaboration process.
2) The status of the Collaboration Contract: active or inactive. The contract owner is responsible for activating or deactivating the contract.
3) The name of the contract owner.
4) numberOfcollaborators: defines the number of collaborators.
5) numberOfrecords: defines the number of records (i.e., ip_addresses).
6) collaboratorsAdr: stores the addresses of the collaborators. The aim of this array is to reduce the cost of finding and removing a specific colloborator from the collaborators mapping (see variable 8).
7) recordsAdr: stores the records of suspicious ip_address. The aim of recordsAdr is to reduce the cost of finding and removing a specific record from the records mapping (see variable 9).
8) collaborators: defines a mapping collection from the address of the collaborator to a corresponding Collaborator struct.
9) Records: defines a mapping collection from the ip_address to a corresponding record struct.
10) OnlyOwner of modifier type. We applied this modifier to the function that (adds/removes) the collaborators (to/from) the smart contract and the function that either activates or deactivates the smart contract; thus, only the owner of the contract can invoke these functions (adds/removes) the colloborators or (activates/deactivates) the contract.
11) OnlyCollaborators of modifier type. It takes as input the address of the caller and checks if the caller is authorized to execute the function that the modifier is applied to it. We applied this modifier to the function that (adds/removes) the records (to/from) the smart contract; thus, only the collaborators(owner included) of the contract can invoke (adds/removes) the records.

The Collaboration Contract mainly provides the following functions where c denotes an instance of Collaborator and r an instance of record:

isCollaborator(c.EOA): This function takes as input the Externally Owned Account (c.EOA) of a collaborator and returns true if the collaborator exists in the collaboration contract; otherwise, it returns false. This function will be invoked at the moment when the owner tries to add/ remove the collaborator to/from the collaboration contract (see Algorithm 3 and 4). Algorithm 1 illustrates the logic of this function.

isRecord(r.IP): This function takes as input ip_address of a record and returns true if the ip_address exists in the collaboration contract; otherwise, it returns false. This function will be invoked at the moment when one of the collaborators tries to add/ remove the record to/from the collaboration contract

Algorithm 1 isCollaborator

```
Input  : c.EOA
Output: bool
if collaboratorsAdr.length == 0 then
    return false;
else
    if collaboratorsAdr[collaborators[c.EOA].index]==c.EOA
    then
        return true;
    else
        return false;
    end
end
```

(see Algorithms 5 and 6). Algorithm 2 illustrates the logic of this function.

Algorithm 2 isRecord

```
Input  : r.IP
Output: bool
if recordsAdr.length == 0 then
    return false;
else
    if recordsAdr[records[r.IP].index] == r.IP then
        return true;
    else
        return false;
    end
end
```

addCollaborator(c.EOA, c.Infos): This function can only be invoked by the owner of the smart contract to add collaborators; it takes as input the Externally Owned Account (c.EOA) of the collaborator and the information about the collaborator (c.Infos) and adds the collaborator to smart contract (i.e., to collaboratorsAdr array (see variable 6 in the Initialization part)) as well as the timestamp of when the collaborator was added. This happens if the contract is activated and the colloborator' identity is authenticated. Algorithm 3 illustrates the logic of this function.

removeCollaborator(c.EOA): This function can only be invoked by the owner of the smart contract to remove collaborators; it takes as input the Externally Owned Account (c.EOA) of the collaborator and removes the collaborator from the smart contract. Algorithm 4 illustrates the logic of this function. addRecord(r.IP): This function can only be invoked by either the owner of the smart contract or the collaborator that has already been added in the smart contract to report suspicious ip_address. It takes as input the suspicious ip_address and adds the record to the smart contract (i.e., to recordsAdr array (see variable 7 in the Initialization part)). Algorithm 5 illustrates the logic of this function. removeRecord(r.IP): This function can only be invoked by either the owner of the smart contract or the collaborator to remove records; it takes as input an IP address and removes the corresponding record, if it exists, from the smart contract. Algorithm 6 illustrates the logic of this function. ChangeStatus(bool status): This function can only be invoked by the owner of the smart contract to either activate or deactivate the smart contract. Algorithm 7 illustrates the logic of this function.

Algorithm 3 addCollaborator

```
Input  : c.EOA, c.Infos
Output: null
if msg.sender is not owner then
    throw;
end
if status is not true then
    throw;
end
if isCollaborator(c.EOA) == true then
    throw;
else
    length ← collaboratorsAdr.push(c.EOA) ;
    collaborators[c.EOA]←
    Collaborator(c.EOA,c.Infos, now, length-1) ;
    emit CollaboratorAdded(c.EOA, c.Infos) ;
    numberOfCollaborators++ ;
end
```

Algorithm 4 removeCollaborator

```
Input  : c.EOA
Output: null
if msg.sender is not owner then
    throw;
end
if status is not true then
    throw;
end
if isCollaborator(c.EOA) == false then
    throw;
else
    rowToDelete ← collaborators[ c.EOA ].index ;
    keyToMove← collaboratorsAdr[length-1] ;
    collaboratorsAdr[rowToDelete]=keyToMove ;
    collaborators[keyToMove].index=rowToDelete ;
    collaboratorsAdr.length - - ;
    emit CollaboratorRemoved(c.EOA) ;
    numberOfCollaborators - - ;
end
```

VI. IMPLEMENTATION

In this section, we present the evaluation of our intra-domain DDoS mitigation scheme. Then, we describe the process of the implementation and deployment of the Collaboration Contract.

Algorithm 5 addRecord

```
Input  : r.IP
Output: null
if msg.sender is not Collaborator then
    throw;
end
if status is not true then
    throw;
end
if isRecord(r.IP) == true then
    throw;
else
    length ← recordsAdr.push(r.IP) ;
    records[r.IP]← Record(r.IP, msg.sender, now, length-1) ;
    emit RecordAdded(r.IP, msg.sender);
    numberOfRecords++ ;
end
```

Algorithm 6 removeRecord

```
Input  : r.IP
Output: null
if msg.sender is not Collaborator then
    throw;
end
if status is not true then
    throw;
end
if isRecord(r.IP) == false then
    throw;
end
if records[r.IP].submitter is not equal msg.sender then
    throw;
else
    rowToDelete ← records[ r.IP ].index ;
    keyToMove← recordsAdr[length-1] ;
    recordsAdr[rowToDelete]=keyToMove ;
    records[keyToMove].index=rowToDelete;
    recordsAdr.length- - ;
    emit RecordRemoved(r.IP, msg.sender);
    numberOfRecords- - ;
end
```

Algorithm 7 ChangeStatus

```
Input  : status
Output: null
if msg.sender is not owner then
    throw;
end
if status is true then
    status ←false ;
    emit StatusChanged(“Smart Contract Deactivated”) ;
end
if status is false then
    status ← true ;
    emit StatusChanged(“Smart Contract activated”) ;
end
```

A. EXPERIMENTATION VALIDATION OF INTRA-DOMAIN DDOS MITIGATION SCHEME

In the following, we describe the experimental setups of our intra-domain DDoS mitigation scheme.

We implemented I-ES and I-BS as applications on the top of the SDN controller. Using sFlow protocol, I-ES extracts network traffic information and I-BS detects automatically illegitimate traffic inside the domain. To emulate a real network environment, we use mininet [47], a popular SDN emulation tool. Mininet uses Linux containers and virtual OF-switches (e.g., OpenVswitch [48]) to allow realistic virtual networks of hosts and switches to be constructed using a virtual machine. Mininet is installed on a VirtualBox [49] VM; VM is connected to the Internet through Network Address Translation (NAT) (for software installation and updates), and a host-only adapter is configured on VM to enable it to communicate with the host-system. Additionally, Secure Shell (SSH) is used to allow access to the VM for running different software at the same time. In our testing environment, the network monitor (i.e., sFlow-RT) and the SDN controller (i.e., Floodlight [50]) are installed on the host-system and run Ethereum geth client (1.8.20-stable client). We run our experiments on a PC with CPU Intel Core i7-8750H-2.2 GHz and 16GB RAM.

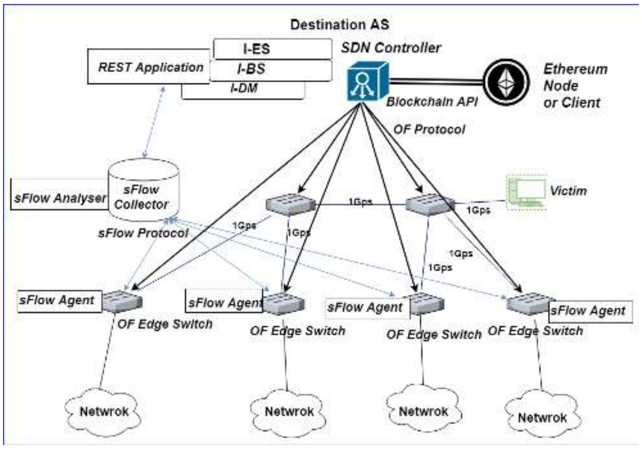

FIGURE 7. Experimental environment.

Fig. 7 shows that our intra-domain DDoS mitigation scheme is implemented in the victim network. In our experiment, we conduct a set of DDoS attacks towards a victim hosted in this domain. When I-BS detects the attack: (a) I-DM protects the victim inside the domain and blocks illegitimate traffic using OF protocol; and (b) the SDN controller

TABLE 2. Transaction details of Cochain-SC.

Details of Cochain-SC Creation Transaction in Ropsten test network	
TxHash	0xf68f79e7cb678645d3b5837bda8b33e4a3d9ebcfd78709a3df255c0a23059b25
Block Height	4738376 (182 Block Confirmations)
Timestamp	(Jan-01-2019 08:13:32 AM +UTC)
From	0xa70836a9a115f774cb848134d0f8b2473e27d181
To	0xcd0df692d251b0a82e63e7fafda2c72aa6b0a8f9
Gas Used by Tx	2225130

invokes the smart contract for reporting suspicious IP. Fig. 7 shows the 4 components of the testbed: (a) OF controller (i.e., Floodlight): it provides elementary connectivity which can be canceled using the Static Flow Pusher API; (b) sFlow network monitor (i.e., sFlow-RT): it performs monitoring of 7500 switch ports in data center networks; (c) REST application: it executes I-ES and I-BS; and (d) 6 OF switches, the bandwidth of each link is set to 1 Gbps.

Each OF network contains more than 20 hosts; multiple hosts are simulated to launch the attack towards the victim hosted inside the domain. The rate of the attack varies from 100 to 500 Mbps; the objective is to test the scalability of the proposed solution. The sampling rate used in sFlow is 1/64. For the attack script, Scapy's Python library [51] is used to generate DDoS flooding attack traffic from zombie hosts towards the victim as well as the simulated legitimate traffic. We use Hping3 [52] command line to Simulate DDoS attacks (i.e., UDP/ICMP DDoS attacks). We present the experiment results of our Intra-Domain DDoS mitigation scheme in Section VII.

B. DEPLOYMENT OF THE COLLABORATION CONTRACT

Once the smart contract is deployed, it can be self-executed without any human intervention. The deployment process is elaborated using truffle framework [53], a decentralized application development framework. First, we code the collaboration contract using the high-level language programming solidity. Then, we compile the contract into EVM byte code; once the contract gets compiled, it generates the EVM byte code and Application Binary Interface (ABI). Afterwards, we deploy the smart contract to the blockchain. Initially, we have deployed the smart contract on a private blockchain using Ganache [54], an Ethereum simulator used for testing the smart contract in a fast way. Then, we have deployed the smart contract on Ethereum official test network Ropsten. The contract lifecycle is shown in Fig. 8. Once deployed, the contract can be invoked using ABI definition and the address of the contract. If needed, the contract can be deleted (cannot be invoked anymore). We tested the implementation of Cochain-SC using both private (Ganache simulator) and public blockchain (Ethereum official test network Ropsten). Table 2 shows Cochain-SC creation transaction in Ropsten official test network. The details of a given transaction can be found using Ropsten Etherscan [55].

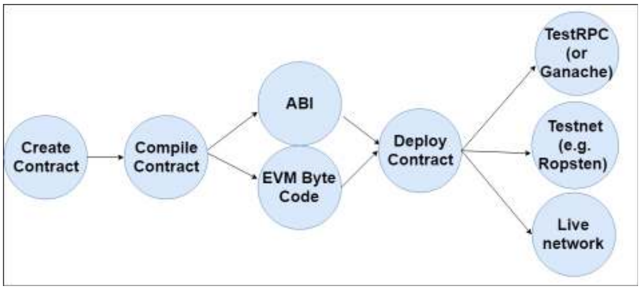

FIGURE 8. The contract lifecycle.

VII. COCHAIN-SC: EXPERIMENTAL RESULTS AND CHARACTERISTICS

A. EXPERIMENTAL RESULTS

Fig. 9 shows that, inside the domain of the victim and when the control is disabled (i.e., without Cochain-SC), the illegitimate traffic generated from a large scale of zombies sustains over 2000 packets attacks per second. However, when the control is enabled (i.e., with Cochain-SC), the illegitimate traffic is blocked when I-BS detects illegitimate traffic; indeed, I-BS notifies the controller that mitigates the attack using I-DM.

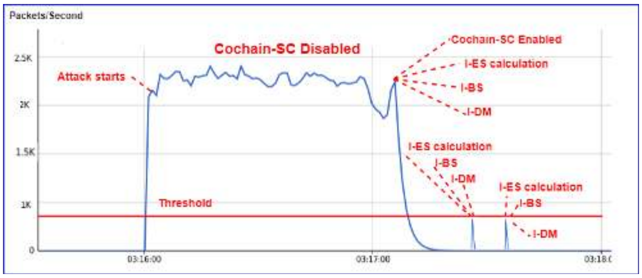

FIGURE 9. DDoS attacks before and after enabling cochain-SC.

The time taken by Cochain-SC operations is smaller than 13 seconds as shown in Fig. 10:

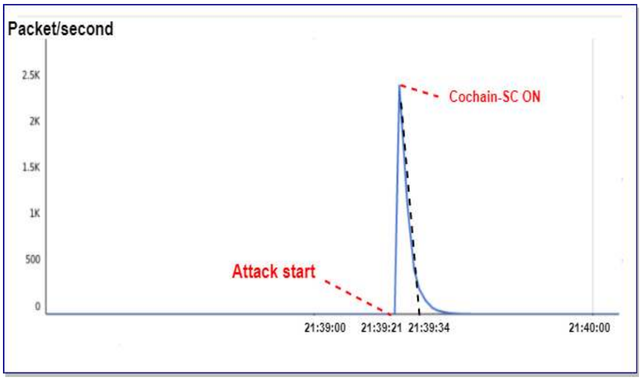

FIGURE 10. attack mitigation.

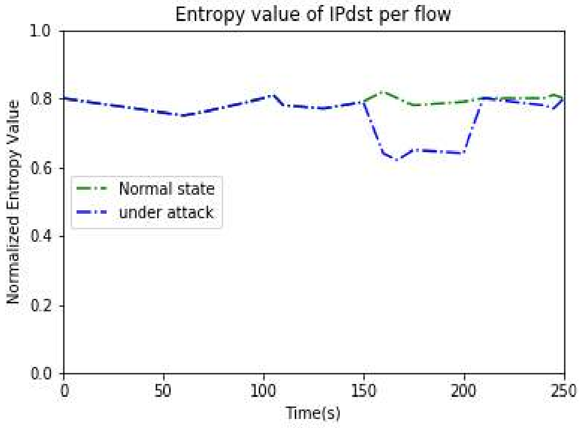

FIGURE 11. Normalized entropy value of IPdst per flow.

To test I-ES, we simulate the attack within an interval of 250 seconds. We launch the attack during the interval [150, 200]; Fig. 11 shows that the normalized entropy decreases rapidly at the start of the interval.

B. PERFORMANCE EVALUAION

To measure the performance of I-BS, we define the detection rate (DR) and false positive rate (FPR) as follows:

$$DR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{TN + FP}$$

where, TP (True Positives) represent the illegitimate flows that are correctly identified as illegitimate, FN (False Negatives) represent the illegitimate flows that are classified as legitimate, FP (False Positives) represent the legitimate flows that are identified as illegitimate, and TN (True Negatives) represent the legitimate flows that are classified as legitimate. The performance of I-BS is evaluated using ROC curve. Moreover, we conducted two experiments (i.e., 100Mbps and 500Mbps) and we compared I-BS in terms of accuracy and FPR with our previous work [22] and one of the most prominent related schemes [31]. Fig. 12 shows that Cochain-Sc achieve around 100% detection rate for 100 Mbps while it has just 26 % of FPR, while [22] and [31] achieve the same detection rate but with respectively 31% and 40% of FPR. Fig. 13 shows that Cochain-Sc achieves around 100% detection rate for 500 Mbps while it has just 23% of FPR, while [22] and [31] achieves the same detection rate but with respectively 30% and 34% of FPR.

C. CHARACTERISTICS

The main objective of Cochain-SC is to provide a secure, easy-to-deploy, low-cost, efficient and flexible DDoS attacks mitigation scheme based on Ethereum using smart contract. In this section, we answer the question: how does Cochain-SC provide these features? Note that efficiency of cochain-SC has been shown in the experiments presented above.

1) FLEXIBILITY/EASY_TO_DEPLPY

Cochain-SC provides two levels of flexibility: (1) Cochain-SC provides the organization (contract owner) with the flexibility to easily add/remove collaborators to/from the system using addCollaborator()/removeCollaborator() functions. Similarly, collaborators can easily add/remove records to/from the system using addRecord() /removeRecord() functions; (2) Cochain-SC provides the organization (contract owner) with the flexibility to easily join or leave the system. To join the system, the organization needs to deploy the Collaboration contract. To leave the system, the organization can easily deactivate the contract using ChangeStatus() function. All these updates can be verified by anyone in the network (i.e., Ethereum).

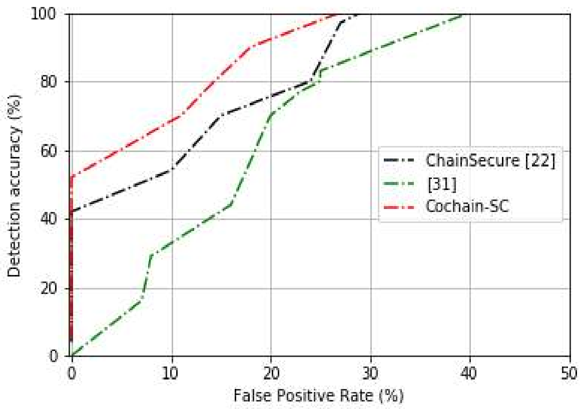

FIGURE 12. ROC curves for the 100 Mbps case.

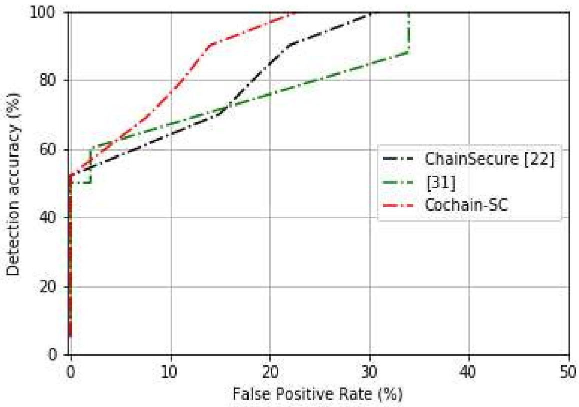

FIGURE 13. ROC curves for the 500 Mbps case.

2) SECURITY/ELIGIBILITY

Only authorized collaborators that have permissions can report suspicious IP address. This is achieved by Cochain-SC using modifiers. For example, the modifier "OnlyOwner" allows only the owner of the contract to execute the addCollaborator(), removeCollaborator() and changeStatus() functions. If a malicious/compromised user tries to execute these functions in order to either add illegitimtae collaborators to report fake IP or remove legitimate collaboartors, the execution will fail and no action will be recorded on the blockchain. The same restriction rule applies for the "OnlyCollaborators" modifier for the execution of addRecord() and removeRecord() functions; only collaborators (and also the contract owner) can add/remove the records.

3) LOW COST

In this section, we estimate the cost of the creation of the collaboration contract as well as the execution of each function used in Cochain-SC. When we conducted the experiment, the gasPrice was set to $1Gwei$, where $1Gwei = 10^9$ $wei = 10^{-9}ether$, and 1 ether was equal to 133.42 USD. Table 3 shows the cost of the execution of different functions in Cochain-SC. We observe that the highest cost corresponds to the creation of Cochain-SC at 0.296 USD. However, it is performed only once to setup the collaboration system. All functions, provided by the smart contract, have low costs. Thus, Cochain-SC is cost effective compared to exiting related schemes.

TABLE 3. Cochain-SC creation and functions costs.

Function	Gas Used	Actual Cost(ether)	USD
create Cochain-Sc	2225130	0.00222513	0.296
addCollaborator()	140000	0.00014	0.018
removeCollaborator()	38450	0.00003845	0.005
addRecord()	102669	0.000102669	0.013
removeRecord()	39367	0.000039367	0.005
changeStatus()	28929	0.000028929	0.003

4) ANALYSIS

First, Cochain-SC preserves pseudonymity and does not allow traceability of identities of collaborators (e.g., IP address of the collaborator). It is important to preserve the pseudonymity of collaborators; otherwise, they may be a target of DDoS attacks. Moreover, Cochain-SC does not suffer from single point of failure problem since it runs on Ethereum. Furthermore, it is decentralized scheme; thus, there is no need to a centralized authority (or a third party) to maintain the collaboration system; the reliability and availability of the records, recorded on the blockchain, are guaranteed. [31] is the only scheme that uses blockchain to advertise blacklisted IP addresses. However, this scheme [31] requires a central entity to issue certificates of ownership of IP addresses. Moreover, this scheme [31] is concerned only with inter-domain DDoS mitigation and not with intra-domain DDoS mitigation. However, Cochain-SC combines two levels of mitigation, intra-domain and inter-domain DDoS mitigation.

VIII. CONCLUSION

In this paper, we proposed a blockchain-based framework called Cochain-SC which combines two levels of mitigation, intra-domain and inter-domain DDoS mitigation. For intra-domain, we combined I-ES with I-BS in order to detect in real-time illegitimate flows, and I-DM to effectively mitigate illegitimate flows inside the domain. For inter-domain, we proposed a smart contract-based framework that makes use of Ethereum's smart contract technology to facilitate the collaboration among SDN-based domain peers. The collaboration contract has been tested/evaluated and deployed on Ethereum official test network Ropsten; the appendix shows Cochain-SC address in Ropsten. Note that other blockchains, such EOS [56], can be used to implement our scheme. We decided to use Ethereum because it is the most popular blockchain, that supports the concept of smart contract, with most devout developers and is the second largest in terms of market value.

APPENDIX

The collaboration contract was deployed on the Ropsten Testnet of Ethereum with the following address: Organization Owner of account address:
0xa70836a9a115f774cb848134d0f8b2473e27d181
Cochain-SC address:
0xCd0Df692D251B0a82E63e7FaFdA2c72aa6B0A8f9
Using this address, the transactions can be seen at:
https://ropsten.etherscan.io/

REFERENCES

[1] *DDoS Attacks Against U.S. Banks*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.computerworld.com/article/2493861/ddos-attacks-against-u-s–banks-peaked-at-60-gbps.html

[2] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 7, pp. 1838–1853, Jul. 2018.

[3] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," CISCO, San Jose, CA, USA, White Paper 1, 2011, vol. 1, pp. 1–11.

[4] S. Sharwood. *GitHub Wobbles Under DDOS Attack*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.itsecurityguru.org/2015/08/26/github-wobbles-under-ddos-attack/

[5] DDoS Attack. *Lessons From the Dyn DDoS Attack*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.schneier.com/blog/archives/2016/11/lessons_from_th_5.html

[6] S. Simpson, S. N. Shirazi, A. Marnerides, S. Jouet, D. Pezaros, and D. Hutchison, "An inter-domain collaboration scheme to remedy ddos attacks in computer networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 3, pp. 879–893, Sep. 2018.

[7] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 623–654, 1st Quart., 2016.

[8] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and J. Liu, "A survey on large-scale software defined networking (SDN) testbeds: Approaches and challenges," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 891–917, 2nd Quart., 2016.

[9] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 325–346, 1st Quart., 2017.

[10] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic, "A survey on data plane flexibility and programmability in software-defined networking," *IEEE Access*, vol. 7, pp. 47804–47840, 2019.

[11] R. Mohammadi, R. Javidan, and M. Conti, "SLICOTS: An SDN-based lightweight countermeasure for TCP SYN flooding attacks," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 2, pp. 487–497, Jun. 2017.

[12] P. Phaal. *sFlow*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.ietf.org/rfc/rfc3176.txt

[13] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[14] *Ethereum: A Secure Decentralised Generalised Transaction Ledge*. Accessed: Jul. 1, 2019. [Online]. Available: https://ethereum.org/

[15] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," Zerocoin Electr. Coin Company, Oakland, CA, USA, Tech. Rep. 2016-1.10, 2016.

[16] *Blockchain for Financial Services*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.ibm.com/blockchain/financial-services

[17] S. Wang, J. Wang, X. Wang, T. Qiu, Y. Yuan, L. Ouyang, Y. Guo, and F. Wang, "Blockchain-powered parallel healthcare systems based on the ACP approach," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 4, pp. 942–950, Dec. 2018.

[18] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[19] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[20] M. Pärssinen, M. Kotila, R. C. Rumin, A. Phansalkar, and J. Manner, ''Is blockchain ready to revolutionize online advertising?'' *IEEE Access*, vol. 6, pp. 54884–54899, 2018.

[21] Etherscan. *The Ethereum Block Explorer: Ropsten (Revival) Testnet*. Accessed: Jul. 1, 2019. [Online]. Available: https://ropsten.etherscan.io

[22] Z. A. El Houda, L. Khoukhi, and A. Hafid, ''ChainSecure—A scalable and proactive solution for protecting blockchain applications using SDN,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[23] Z. A. El Houda, A. Hafid, and L. Khoukhi, ''Brainchain—A machine learning approach for protecting blockchain applications using SDN,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019.

[24] Z. A. El Houda, A. Hafid, and L. Khoukhi, ''Co-IoT—A collaborative DDoS mitigation scheme in IoT environment based on blockchain using SDN,'' to be published.

[25] P. Ferguson and D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing (BCP 38)*, document RFC 2827, 2000. [Online]. Available: http://tools.ietf.org/html/rfc2827

[26] R. Braga, E. Mote, and A. Passito, ''Lightweight DDoS flooding attack detection using NOX/OpenFlow,'' in *Proc. IEEE Local Comput. Netw. Conf.*, Oct. 2010, pp. 408–415.

[27] S. A. Mehdi, J. Khalid, and S. A. Khayam, ''Revisiting traffic anomaly detection using software defined networking,'' in *Recent Advances in Intrusion Detection*, R. Sommer, D. Balzarotti, and G. Maier, Eds. Berlin, Germany: Springer, 2011, pp. 161–180.

[28] M. Yu, L. Jose, and R. Miao, ''Software defined traffic measurement with OpenSketch,'' in *Proc. 10th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, Berkeley, CA, USA, 2013, pp. 29–42. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482631

[29] R. Wang, Z. Jia, and L. Ju, ''An entropy-based distributed DDoS detection mechanism in software-defined networking,'' in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug. 2015, pp. 310–317.

[30] S. Lim, J. Ha, H. Kim, Y. Kim, and S. A. Yang, ''A SDN-oriented DDoS blocking scheme for botnet-based attacks,'' in *Proc. 6th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2014, pp. 63–68.

[31] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, ''Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments,'' *Comput. Netw.*, vol. 62, no. 5, pp. 122–136, 2014. doi: 10.1016/j.bjp.2013.10.014.

[32] F. Guo, J. Chen, and T.-C. Chiueh, ''Spoof detection for preventing DoS attacks against DNS servers,'' in *Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Washington, DC, USA, Jul. 2006, p. 37. doi: 10.1109/ICDCS.2006.78.

[33] K. Nishizuka, L. Xia, J. Xia, D. Zhang, and C. L. Fang, *Inter-Organization Cooperative DDoS Protection Mechanism*, document draft-nishizuka-dots-inter-domain-mechanism-02, 2017. [Online]. Available: https://tools.ietf.org/html/draft-nishizuka-dots-inter-domain-mechanism-02

[34] J. Steinberger, B. Kuhnert, A. Sperotto, H. Baier, and A. Pras, ''Collaborative DDoS defense using flow-based security event information,'' in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2016, pp. 516–522.

[35] K. Giotis, M. Apostolaki, and V. Maglaris, ''A reputation-based collaborative schema for the mitigation of distributed attacks in SDN domains,'' in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2016, pp. 495–501.

[36] B. Rashidi, C. Fung, and E. Bertino, ''A collaborative DDoS defence framework using network function virtualization,'' *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2483–2497, Oct. 2017.

[37] Y. Chen, K. Hwang, and W. S. Ku, ''Collaborative detection of DDoS attacks over multiple network domains,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 12, pp. 1649–1662, Dec. 2007.

[38] B. Rodrigues, T. Bocek, A. Lareida, D. Hausheer, S. Rafati, and B. Stiller, ''A blockchain-based architecture for collaborative ddos mitigation with smart contracts,'' in *Security of Networks and Services in an All-Connected World*, D. Tuncer, R. Koch, R. Badonnel, and B. Stiller, Eds. Cham, Switzerland: Springer, 2017, pp. 16–29.

[39] M. Steichen, S. Hommes, and R. State, ''ChainGuard—A firewall for blockchain applications using SDN with OpenFlow,'' in *Proc. Principles, Syst. Appl. IP Telecommun. (IPTComm)*, Sep. 2017, pp. 1–8.

[40] *Go Ethereum*. Accessed: Jul. 1, 2019. [Online]. Available: https://geth.ethereum.org/

[41] *REST API*. Accessed: Jul. 1, 2019. [Online]. Available: http://www.sflowrt.com/reference.php

[42] *OpenFlow Switch Specification*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[43] *sFlow-RT*. Accessed: Jul. 1, 2019. [Online]. Available: http://www.sflow-rt.com

[44] C. E. Shannon, ''Prediction and entropy of printed english,'' *Bell Labs Tech. J.*, vol. 30, no. 1, pp. 50–64, 1951.

[45] P. D. Hoff, *A First Course in Bayesian Statistical Methods*. New York, NY, USA: Springer, 2009.

[46] *Solidity*. Accessed: Jul. 1, 2019. [Online]. Available: https://solidity.readthedocs.io/en/develop/

[47] *Mininet*. Accessed: Jul. 1, 2019. [Online]. Available: http://mininet.org

[48] *Open vSwitch*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.openvswitch.org/

[49] Oracle. *Virtualbox*. Accessed: Jul. 1, 2019. [Online]. Available: https://www.virtualbox.org/

[50] *Floodlight*. Accessed: Jul. 1, 2019. [Online]. Available: http://www.projectfloodlight.org/

[51] *Scapy*. Accessed: Jul. 1, 2019. [Online]. Available: http://www.secdev.org/projects/scapy

[52] *Hping3*. Accessed: Jul. 1, 2019. [Online]. Available: https://tools.kali.org/information-gathering/hping3

[53] *Truffle*. Accessed: Jul. 1, 2019. [Online]. Available: https://truffleframework.com/

[54] *Ganache*. Accessed: Jul. 1, 2019. [Online]. Available: https://truffleframework.com/docs/ganache/overview

[55] *Ropsten*. Accessed: Jul. 1, 2019. [Online]. Available: https://ropsten.etherscan.io/

[56] *EOSIO: The Most Powerful Infrastructure for Decentralized Applications*. Accessed: Jul. 1, 2019. [Online]. Available: https://eos.io/

ZAKARIA ABOU EL HOUDA received the Engineer's degree in computer science from the National School of Applied sciences Marrakech, Marrakech, Morocco, and the M.Sc. degree in computer networks from Paul Sabatier University, Toulouse, France. He is currently pursuing the Ph.D. degree with the University of Montreal, Montreal, QC, Canada, and the University of Technology of Troyes, Troyes, France. His current research interests include DDoS, Blockchain, and SDN.

ABDELHAKIM SENHAJI HAFID is currently a Full Professor with the University of Montreal, where he founded the Network Research Lab (NRL). He is also a Research Fellow with CIRRELT, Montreal, and the Founding Director of the Montreal Blockchain Lab. Prior to joining University of Montreal, he has spent several years, as a Senior Research Scientist with Telcordia Technologies (formerly Bell Communications Research), NJ, USA, working in the context of major research projects on the management of next generation networks, including wireless (e.g., mobile ad-hoc networks) and optical networks. He was also an Assistant Professor with Western University (WU), Canada, the Research Director of the Advance Communication Engineering Center (venture established by WU, Bell Canada, and Bay Networks), Canada, a Researcher at CRIM, Canada, a Visiting Scientist at GMD-Fokus, Berlin, Germany, and a Visiting Professor at the University of Evry, France. He has extensive academic and industrial research experience in the area of the management and design of next generation networks. His current research interests include the IoT, Fog/edge computing, blockchain, and intelligent transport systems.

LYES KHOUKHI received the Ph.D. degree in electrical and computer engineering from the University of Sherbrooke, Canada, in 2006. From 2007 to 2008, he was a Postdoctoral Researcher with the Department of Computer Science and Operations Research, University of Montreal. He is currently a Professor with the University of Technology of Troyes, France. He has published over 100 journal and conference papers. His current research interests include cloud, the IoT and 5G, performance evaluation, and security.