A Blockchain-Based Architecture for Collaborative DDoS Mitigation with Smart Contracts

Bruno Rodrigues, Thomas Bocek, Andri Lareida, David Hausheer, Sina Rafati, Burkhard Stiller

▶ **To cite this version:**

Bruno Rodrigues, Thomas Bocek, Andri Lareida, David Hausheer, Sina Rafati, et al.. A Blockchain-Based Architecture for Collaborative DDoS Mitigation with Smart Contracts. 11th IFIP International Conference on Autonomous Infrastructure, Management and Security (AIMS), Jul 2017, Zurich, Switzerland. pp.16-29, 10.1007/978-3-319-60774-0_2 . hal-01806063

HAL Id: hal-01806063
https://hal.inria.fr/hal-01806063

Submitted on 1 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

A Blockchain-based Architecture for Collaborative DDoS Mitigation with Smart Contracts

Bruno Rodrigues[1], Thomas Bocek[1], Andri Lareida[1], David Hausheer[2], Sina Rafati[1], Burkhard Stiller[1]

[1] Communication Systems Group (CSG)
Department of Informatics (IfI)
University of Zürich (UZH)
[2] P2P Systems Engineering Lab
Department of Electrical Engineering and Information Technology
TU Darmstadt
{rodrigues,bocek,lareida,rafati,stiller}@ifi.uzh.ch
hausheer@ps.tu-darmstadt.de

Abstract. The rapid growth in the number of insecure portable and stationary devices and the exponential increase of traffic volume makes Distributed Denial-of-Service (DDoS) attacks a top security threat to services provisioning. Existing defense mechanisms lack resources and flexibility to cope with attacks by themselves, and by utilizing other's companies resources, the burden of the mitigation can be shared. Emerging technologies such as blockchain and smart contracts allows for the sharing of attack information in a fully distributed and automated fashion. In this paper, the design of a novel architecture is proposed by combining these technologies introducing new opportunities for flexible and efficient DDoS mitigation solutions across multiple domains. Main advantages are the deployment of an already existing public and distributed infrastructure to advertise white or blacklisted IP addresses, and the usage of such infrastructure as an additional security mechanism to existing DDoS defense systems, without the need to build specialized registries or other distribution mechanisms, which enables the enforcement of rules across multiple domains.

Keywords: Distributed Denial-of-Service (DDoS), Security, Blockchain, Software-defined Networks (SDN), Network Management

1 Introduction

In the past years, a rise in DDoS attacks could be observed [1]. DDoS attacks have the simple goal of interrupting or suspending services available on the Internet and its motivations range from personal grudges over blackmail to political reasons [10]. A recent example is an attack conducted against Domain Name System (DNS) servers responsible for domains such as Twitter, PayPal, and

Spotify [20] in October 2016. As a consequence, those services became unavailable to many US (United States) users for several hours. Besides the frequency, also the strength and duration of DDoS attacks are growing making them more efficient and dangerous. One reason for the increasing size of attacks is the availability of many reflectors, and *i.e.*, weakly secured or configured IoT (Internet of Things) devices or home gateways [20].

By exploiting legal services on those devices, *e.g.*, the Simple Service Discovery Protocol, the power of a DDoS attack is amplified, and the problem of defense is made more complicated. Thus, the impact of DDoS varies from minor inconvenience to severe financial losses for enterprises that rely on their online availability [14]. Various mitigation techniques have been proposed. However, only a few have been considered for widespread deployment because of their effectiveness and implementation complexities. An ongoing IETF (Internet Engineering Task Force) proposal discusses the development of a collaborative protocol called DOTS (DDoS Open Threat Signaling) to advertise DDoS attacks [13]. However, this paper proposes an infrastructure of blockchains and smart contracts, which provide the required instrumentation without the need to maintain design and development complexities of such a new protocol.

As with a different direction, the adoption of DDoS protection services, offered by companies such as Akamai [1] or CloudFlare [3], is increasing [7]. Those cloud-based solutions can absorb DDoS attacks by increasing capacity and taking the burden of detection away from the device under attack by exporting flow records from edge routers and switches. Additional analysis is performed in the cloud and packet filtering is used to balance, reroute, or drop the traffic inside the cloud. However, those solutions requires a third party DDoS Protection Service (DPS) provider, which is implying in additional costs and a decrease in service performance.

This paper presents the architecture and design of a collaborative mechanism using smart contracts and investigates the possibility of mitigating a DDoS attack in a fully decentralized manner. Thus, service providers interested in shared protection, can not only signal the occurrence of attacks but also share detection and mitigation mechanisms. The objective is to create an automated, and easy-to-manage DDoS mitigation. Three major building blocks are identified to build such a mechanism.

Blockchains and Smart Contracts. This approach proposes an architecture and an implementation of an approach to signaling white or blacklisted IP addresses across multiple domains based on blockchains and smart contracts. The advantage of using smart contracts in a blockchain is: (a) to make use of an already existing infrastructure to distribute rules without the need to build specialized registries or other distribution mechanisms/protocols, (b) to apply rules across multiple domains, which means that even if the AS (Autonomous System) of the victim is not applying these rules, some traffic can still be filtered, and (c) the victim or its AS can control which customers get blocked. The only central element remaining is to show proof of IP ownership.

Software-defined Network (SDN) is an effective solution to enable customizable security policies and services in a dynamic fashion. The centralized network control and its deployment based on the OpenFlow [11] protocol facilitates the enforcement of high-level security policies moving away from current approaches based on SNMP (Simple Network Management Protocol) and CLI (Command Line Interface). With SDN, flow-rules can be applied to block DDoS attacks, and the closer these rules are applied, and those malicious packets can be dropped, the less DDoS traffic occurs. This work uses SDN-based networks as a use case to perform in a more rapid fashion in ASes the definition and verification of flows to mitigate DDoS attacks. However, the presented solution is not limited to the usage of an SDN-based network, being compatible with detection/monitoring tools able to export attack information to be published in the blockchain.

This paper is structured as follows. Section 2 introduces basic concepts and related work on blockchain and smart contracts. Section 3 presents related collaborative DDoS mitigation strategies. Section 4 presents the architecture detailing its components and basic functioning, as well as describing the implementation details of the proposed solution. Section 5 provides a discussion on the development and results obtained so far. The work is concluded in Section 6 highlighting the significant contributions and discussing future work.

2 Background

Smart contracts are a piece of software made to facilitate the negotiation or performance of a contract, being able to be executed, verified or enforced on its own. A smart contract alone is not "smart" as it needs an infrastructure that can implement, verify, and enforce the negotiation or performance of a contract by particular computer protocols. It has gained attention in the context of blockchains that provide a fully decentralized infrastructure to run, execute, and verify such smart contracts [2]. Therefore, smart contracts need to run on a blockchain to ensure (a) its permanent storage and (b) obstacles to manipulate the contracts content. A node participating in the blockchain runs a smart contract by executing its script, validating the result of the script, and storing the contract and its result in a block.

Although the Bitcoin [12] blockchain was the first fully decentralized distributed ledger, it is primarily designed for transfer of digital assets, and it is not Turing-complete (*e.g.*, it does not support loops). Such a Turing-complete contract language allows defining rules to allow or block IP addresses that can be interpreted by an SDN controller. While several projects try to address these issues, the Ethereum [23] blockchain is the most popular that supports a Turing-complete contract language, empowering more sophisticated smart contracts. In Ethereum, smart contracts run in a sand-boxed Ethereum Virtual Machine (EVM) and every operation executed in the EVM has to be paid for to prevent Denial-of-Service (DoS) attacks.

SDN characteristics provide better network visibility by decoupling the control plane from the data plane and by the centralized management to perform

tasks such as network diagnosis and troubleshooting [9]. In addition to SDN, the OpenFlow protocol [11] leverages network management by providing a programmable and standardized interface between the data plane and the control plane. It has been recognized that the decoupling of the data plane and the control plane makes SDN a promising solution to enable the enforcement of customizable security services and policies. Various SDN-based solutions have been proposed to deal with DDoS attacks [24]. A survey on these issues is provided in [17]. However, each security/concern category can be sub-divided in fine-grained aspects *e.g.*, authentication, integrity, network communications. In the following are presented mainly research efforts addressing DDoS attacks in SDN networks.

To analyze the impact of DDoS attacks on network performance, the works in [18] and [8] have shown how such attacks may impact on several parameters like the control plane bandwidth (*i.e.*, controller-switch channel), latency, switches flow tables and the controller performance. Other works as [22] and [4] use the SDN capabilities to implement schemes that allow to detect and mitigate DDoS attacks through packet analysis and filtering. These solutions reduce the impact of attacks, but they may cause an overhead in the flow-tables and the SDN controller. Also, they do not provide any solution to address these particular SDN performance issues as proposed in [5] (*e.g.*, flow-tables, and controller overloading). Furthermore, they also do not consider DDoS attacks and the collaboration with AS customers as [16].

SDN-based solutions allow greater agility to enforce decisions that require a global network view. Therefore, intra-domain security policies and mechanisms to prevent and react to DDoS attacks can be made agiler. By combining the intra-domain capabilities provided by SDN and the inter-domain advantages provided by blockchains and smart contracts, the efficiency to mitigate DDoS attacks in both inter- and intra-domains can be improved.

3 Related Work

There are four broad categories of defense against DDoS attacks according to [14]: (1) attack prevention, (2) attack detection, (3) attack source identification, and (4) attack reaction.

(1) Tries to prevent attacks before they become a problem, *i.e.* as close to the sources as possible. The obvious method to achieve this for amplified or reflected attacks is for the access provider to filter spoofed packets;
(2) Can be a difficult task since certain attacks mask themselves as legitimate user traffic or use various traffic types. Due to this complexity, it can be hard to make a confident decision if traffic is part of an attack or special user behavior, *e.g.* a flash crowd;
(3) Is applied after an attack was detected. This step is important to efficiently contain or re-route the attack as close to its source as possible;
(4) The final step involves taking concrete measures against the attack. The better the result from (3) the more efficiently this can be done.

Among the collaborative DDoS mitigation techniques, there are two main approaches using resource management to react against bandwidth attacks [14]. The first takes effect within the victim's domain and the second within the domain of the victims ISP, *i.e.* the AS. Both techniques apply traffic classification and define specific actions for those classes. Both customer and AS resource management schemes need to classify traffic into several types, and then treat them differently. However, it is rather difficult to give an accurate classification as DDoS attacks can mimic any legitimate traffic. In this regard, some sophisticated techniques can be implemented to classify traffic, but a unified reaction strategies implemented both at the AS and the customer can be more efficient than applying just one.

Other works exist for cooperative defense against DDoS attacks. However, it is still an open issue since DDoS attacks are growing in scale, sophistication, duration and frequency [10]. The IETF is currently proposing a protocol [13] called DOTS (DDoS Open Threat Signaling) covering both intra-organization and inter-organization communications to advertise attacks. The protocol requires servers and clients DOTS agents, which can be organized in both centralized and distributed architectures to advertise black or whitelisted addresses. A DOTS client should register to a DOTS server in advance sending provision and capacity protection information and be advertised of attacks. Then, the DOTS protocol is used among the agents to facilitate and coordinate the DDoS protection service as a whole. Also, a similar approach to the IETF proposal is presented in [19]. The authors use a similar architecture but using an advertising protocol based on FLEX (FLow-based Event eXchange) format, which is used to simplify the integration and deployment of the solution and facilitate the communication process between the involved domains.

The proposed standard advertises the need for defensive measures in anticipation of or response to attack. The main drawback compared to the approach presented herein is the requirement of additional infrastructure requiring trust and collaboration between ISPs. A collaborative defense approach using VNF (Virtual Network Functions) is presented in [15]. The authors propose a cooperation between domains that implements VNFs to alleviate DDoS attacks by redirecting and reshaping excessive traffic to other collaborating domains for filtering. In [24], a gossip-based communication mechanism is proposed to exchange information about attacks between independent detection points to aggregate information about the overall observed attacks. The system is built as a peer-to-peer overlay network to disseminate attack information to other listening users or systems rapidly.

A similar approach was presented in [21], formalizing a gossip-based protocol to exchange information in overlay network using intermediate network routers. A different approach is presented in [16], which proposes a collaborative framework that allows the customers to request DDoS mitigation from ASes. However, the solution requires an SDN controller implemented at customer side interfaced with the AS, which can change the label of the anomalous traffic and redirect them to security middle-boxes. In the approach presented in this paper cus-

tomers and ISPs can take action to mitigate an attack by interfacing directly with a blockchain providing the necessary trust.

Instead of making use of an existing infrastructure such as the blockchain and smart contracts, approaches mentioned above proposes the development of specific gossip-based protocols. In this sense, the deployment and integration of such solutions become complex since existing solutions need to be modified to support these protocols. The IETF proposal focuses on standardizing a protocol to facilitate its deployment. However, its implementation complexity still exists in distributed and centralized architectures to support the different types of communication. Instead, some of the requirements can be inherited from the natural characteristics of blockchains, smart contracts, and SDN, avoiding the complexities of development and adoption of new protocols.

4 Proposed System Architecture

This section presents the design principles considered in the architectural design. First, Section 4.1 exemplifies a deployment scenario. Section 4.2 provides a detailed description of its main components. Implementation details are presented in Section 4.3.

4.1 Application Scenario

A scenario is presented in Figure 1 illustrating the system architecture. A web server hosted at AS C is under a DDoS attack from devices hosted at various domains (ASes A, B, and C). With a non-collaborative DDoS mitigation approach, the web server relies on defense mechanisms that are implemented at the AS where it is allocated, which in many cases may be distant from the origin of the attack traffic and therefore overloading several domains with attack traffic.

Participants of the collaborative defense (ASes and customers) first need to create a smart contract, that is promptly linked with a registry-based type of smart contracts. Therefore, when attackers overload web server, the customer or the AS under attack stores the IP addresses of attackers in the smart contract. In an Ethereum blockchain a new block is created every 14 seconds, so subscribed ASes will receive updated lists of addresses to be blocked and confirm the authenticity of the attack by analyzing the traffic statistics and verifying the authenticity of the target's address.

Once other ASes retrieve the list of attackers and confirm the attack, different mitigation strategies can be triggered according to the security policies and mechanisms available in the domain. Also, it can block malicious traffic near of its origin. Near-source, defense is ideal for the health of the Internet because it can reduce the total cost of forwarding packets which, in the case of DDoS attacks mostly consist of useless massive attack traffic [13].

In scenarios involving multiple domains, once collaborative defense nodes receive information about attacks, these can apply mitigation actions in agreement

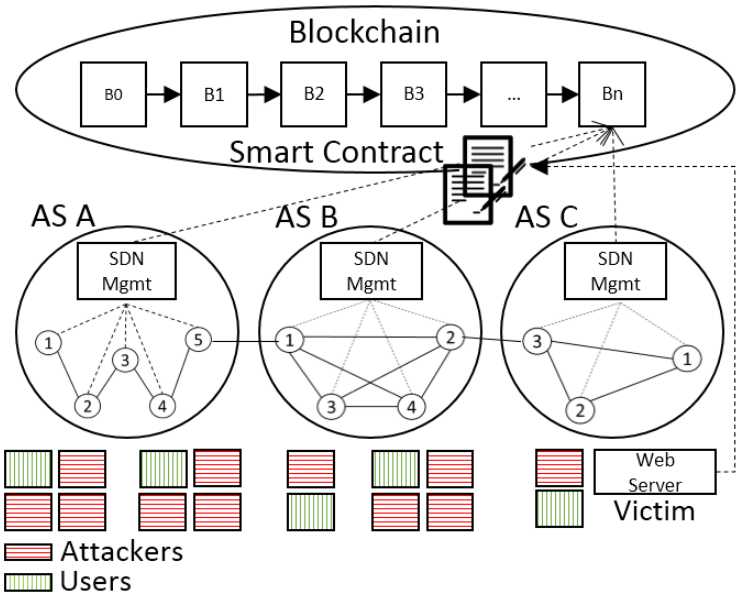

Fig. 1. Application Scenario

with their security policies. In this sense, an incentive mechanism is necessary to prevent domains from abusing cooperative defense.

4.2 Architectural Design

As DDoS attacks continue to increase and vary in their patterns, the need for coordinated responses also increases to detour the attacks efficiently. However, it is important to note that only the collaboration between customers and ASes is an additional approach to existing defense mechanisms. The architecture depicted in Figure 2 is composed of three components:

- **Customers**: may report white or blacklisted IP addresses to the Ethereum blockchain via smart contracts;
- **ASes**: may publish white or blacklisted IP addresses and retrieve lists containing the published IP addresses, and may implement their DDoS mitigation mechanisms;
- **Blockchain/Smart Contract**: the public Ethereum blockchain (Ethereum Virtual Machine nodes) running Solidity smart contracts, which comprises the logic to report IP addresses in the blockchain.

The architecture is built considering the following principles:

(1) DDoS detection and mitigation countermeasures are provided as on-demand services by either the ASes or third-party services;

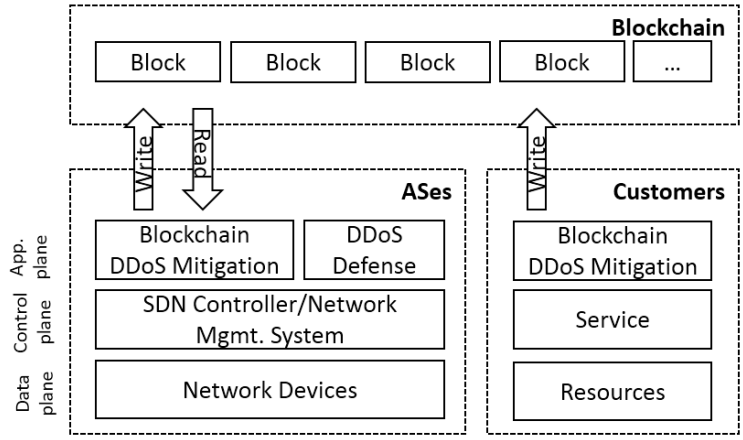

Fig. 2. Proposed System Architecture

(2) To report/receive attack information, it is necessary for the domain to dedicate a node connected to the blockchain. This can be dedicated hardware exclusively for this purpose or virtualized to minimize resource consumption;
(3) To efficiently aid coordinated attack responses, Blockchain DDoS Mitigation modules are running on the entities (customers or ASes) reporting IP addresses and listening to the blockchain;
(4) Only customers or ASes with proof of ownership of their IP may report addresses to the smart contract;
(5) Different domains implement different security policies as well as different underlying management systems. Once notified of a DDoS attack in which the customer has its authenticity confirmed, countermeasures are defined according to the domain security policies and available actions.

To mitigate DDoS attacks (1) different techniques can be used upon the detection by ASes or customers, which typically involves analyzing Internet traffic with sophisticated attack detection algorithms, followed by filtering. In this regard, a collaborative approach decreases the overhead of such algorithm in the detection phase using information from other domains. Blockchain DDoS Mitigation appliances (2) both on the customer and ASes are simpler as Ethereum is public and already available technology, which can be used to perform rapid and widespread DDoS advertisement using smart contracts. Services with challenge/response authentication can be utilized by an AS to ensure that the IP address (3) of the customer reporting the attack is the customer under attack, and to enforce the necessary countermeasures (4) by the security policies implemented in the domain.

The smart contract logic illustrated in Figure 3 is deployed as a complementary solution to existing DDoS mitigation mechanisms. However, domains implementing the system should consider the principles mentioned above in its design. First, any domain (*e.g.*, customers or ASes) participating must create

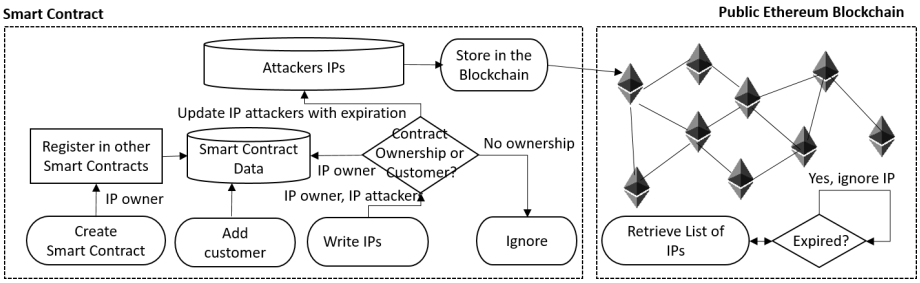

Fig. 3. Proposed System Flowchart

a smart contract identified with an IP address or range of addresses certified by an authority. Then, the smart contract is registered in a registry-based type of smart contract so that participation can be easily tracked and thus relevant smart contracts can be identified.

Traffic arriving at both the customer and AS can be analyzed and filtered using existing monitoring tools (*e.g.*, NetFlow, sFlow, custom SDN implementations). The Blockchain appliance can be deployed as an additional security feature to any system that implements an apparatus to advertise black or whitelisted IP addresses to the blockchain. The analysis of traffic in a gateway is facilitated by SDN, and therefore the approach is intended to use a monitoring framework based on the OpenFlow protocol.

4.3 Implementation Details

Listing 1.1 and 1.2 outlines current implemented features of the smart contract to store source IP addresses that should be blocked or allowed. For simplicity, only IPv4 addresses are shown here. Either the customer or the AS can create the smart contract. In any case, a certificate of IP ownership is required. For the customer, the certificate can be created with an automated challenge-response system, while the AS requires a certificate matching their entry in the AS registration.

The one that created the smart contract (owner of the account that created the contract) can add other addresses that are also allowed to add IPs to block. Before such address is added, it is checked if the address matches its parent subnet. Both AS and the customer can store src IP with an expiration time. The time is measured in blocks, and the access to the stored data is public and can be viewed by anyone.

Before retrieving a list of IP pairs (source/destination), the verifyIP() function needs to be called to make sure that the target IP address has a proof of ownership. The issuing of a certificate (certOwnerIPv4) is the only remaining central entity in the architecture. After that, any AS (does not need to be the customers AS) can use these IPs to block traffic on its network.

The smart contract needs first to register itself in another smart contract Registry, which stores all relevant smart contracts that should be watched. Thus

an AS listens for these changes, and any addition can be monitored and assessed against the network properties of the AS and apply a blocking rule if necessary.

```
contract SDNRulesAS {
  struct ReportIPv4 {
    uint32 expiringBlock;
    uint32 src_ip;
    DstIPv4 dst_ip;
  } ReportIPv4[] report_src_ipv4;

  struct DstIPv4 {
     uint32 dst_ipv4;
     uint8 dst_mask;
  } DstIPv4 dstIPv4;
  bytes certOwnerIPv4; address owner;
  mapping (address => DstIPv4) customerIPv4;
  bool flag; //Indicate black or whitelisted addresses
  function SDNRulesAS(uint32 dst_ipv4, uint8 dst_ipv4_mask,
      bytes _certOwnerIPv4, bool _flag) {
    owner = msg.sender;
    certOwnerIPv4 = _certOwnerIPv4;
    dstIPv4 = DstIPv4(dst_ipv4, dst_ipv4_mask);
    flag = _flag;
    //TODO: register in a registry contract
  }
  //suicide and deregistering function here
  function createCustomerIPv4(address customer, uint32
      dst_ipv4, uint8 dst_ipv4_mask) {
    if(msg.sender == owner &&
        isInSameIPv4Subnet(dst_ipv4, dst_ipv4_mask)) {
      customerIPv4[customer] =
        DstIPv4(dst_ipv4, dst_ipv4_mask);
    }
  }
  function isInSameIPv4Subnet(uint32 dst_ipv4, uint8 dst_mask
      ) constant returns (bool) {
    // true if customer IP is in same subnet
  }
}
```

Listing 1.1. Smart contract structures and core functionality

```
function reportIPv4(uint32[] src, uint32 expiringBlock) {
  if (msg.sender == owner) {
    for (uint i = 0; i < src.length; i++) {
      drop_src_ipv4.push(ReportIPv4(
        expiringBlock, src[i], dstIPv4));
    }
  }
  DstIPv4 customer = customerIPv4[msg.sender];
  if(customer.dst_ipv4 != 0) {
```

```
10      for (i = 0; i < src.length; i++) {
11        report_src_ipv4.push(ReportIPv4(
12          expiringBlock, src[i], customer));
13      }
14    }
15  }
16
17  function verifyIP(bytes pubKey) constant returns (bool) {
18    //check if signature in certOwnerIPv4 is correct
19  }
20
21  function reportedIPv4() constant returns (uint32[] src_ipv4,
        uint32[] dst_ipv4, uint8[] mask) {
22    uint32[] memory src; uint32[] memory dst; uint8[] memory
          msk;
23    for (uint i = 0; i < report_src_ipv4.length; i++) {
24      if(drop_src_ipv4[i].expiringBlock > block.number) {
25        src[src.length] = report_src_ipv4[i].src_ip;
26        dst[dst.length] = report_src_ipv4[i].dst_ip.dst_ipv4;
27        msk[msk.length] = report_src_ipv4[i].dst_ip.dst_mask;
28      }
29    }
30    return (src, dst, msk);
31  }
```

Listing 1.2. Smart contract IP reporting functions

5 Discussion

The use of the Ethereum Virtual Machine (EVM) allows the multiple domains involved in an attack scenario to invoke functions in a smart contract reporting attacks or maintaining a list of trusted addresses to be operating in case of attack. The support of white or blacklisted IP addresses is a decision that depends on the policies and security mechanisms available in each domain. Therefore, the smart contract was developed to support both lists using a flag indicating which type of address is being reported. The existing and distributed storage infrastructure reduces the complexity in the development and adoption of the approach as it supersedes the design and standardization process of a gossip-based protocol, which needs to be embraced by the various ASes and customers. Also, the EVM smart contracts support in a decentralized and native way the logic to control who is reporting an attack and who are the attackers.

Through a high-level comparison with the ongoing IETF proposal (the DOTS protocol) [13], instead of making use of an existing infrastructure such as the blockchain and smart contracts, the IETF proposes from scratch the development of such protocol with several requirements (*e.g.*, extensibility, resilience) to be deployed in a distributed architecture. In this sense, the protocol development becomes complex since it must be deployed in distributed and centralized

architectures to support different types of communication (inter and intra domain, *i.e*, inside the domain of an AS and between ASes). Instead, it is argued that some of the requirements can be inherited from the natural characteristics of blockchains, smart contracts and SDN, avoiding the complexities of development and adoption of new protocols.

However, this smart contract works well for a small number of attacks, while for large-scale attacks, the approach is currently costly the contract size, but will be addressed this issue in a future work to make reference to a larger list of IP addresses. Therefore, to keep the complexity of the architecture low, only the data (*e.g.*, IP addresses) should be stored in the contract, and it may become necessary to add a reference as shown in Listing 1.3, where the full list of addresses can be retrieved. The cost of adding 50 source IPs directly in a freshly deployed contract is 2.5 mio gas (gas is the internal pricing for running a transaction or contract in Ethereum) at the current gas price [6] of 20 gwei, which is 0.05 ETH at the current market price of 9.3 USD is in total 0.46 USD, while 100 source IPs cannot be mined in one contract and multiple contracts have to be used as it exceeds the 4 mio gas limit.

```
1 struct ReportIPv4 {
2 uint32 expiringBlock;
3 uint32 src_ip;
4 //e.g. https://example.com/blockedips.txt
5 string src_ipv4_ref;
6 DstIPv4 dst_ip;}
7 ReportIPv4[] report_src_ipv4;
```

Listing 1.3. Storing references

6 Summary and Future Work

This paper proposes a collaborative architecture using smart contracts and blockchain to enable DDoS mitigation across multiple domains. As a distributed and primarily public storage, the blockchain determines a straightforward and efficient structure to develop a collaborative approach toward DDoS attacks mitigation. The proposed architecture can be deployed as an additional security mechanism to existing DDoS protection schemes. Therefore, it is not intended to dictate how security mechanisms and policies should be implemented in a particular domain. Instead, it can be combined with existing solutions to reduce the DDoS detection and mitigation overhead by involving multiple domains in the process. Coupled with current solutions, the DDoS detection and mitigation overhead process comprising multiple domains can be reduced.

The architecture enables ASes to deploy their DPS and generate added value for their customers without transferring control of their network to a third party. The main contributions of this new approach are summarized as (a) the design and development of an architecture based on blockchains to advertise DDoS attacks across multiple domains, (b) the adoption and integration of the approach

is facilitated since Ethereum and smart contracts are publicly available, and the ability to enforce rules on the ASes-side by the use of SDN, (c) can be utilized as an additional security mechanism without modifying existing ones.

Future work will investigate ways to compress the list, *e.g.*, with a bloom filter, and its advantages and disadvantages. Another limitation is that blocking of destination IPs should be possible only for static IPs. Thus, automated services issuing these certificates of IP ownership need to check for dynamic IPs first, *e.g.*, using services such as SORBS (dul.dnsbl.sorbs.net). Also, the current smart contract supports only one hierarchy. Thus, createCustomerCertIPv4() in Listing 1.1 needs to be extended to allow more hierarchies to map subnets accordingly.

Another major factor towards the practicability of the approach is the fairness among the cooperative domains. If an AS is targeted more times than others, means that one would be using resources of others to protect themselves. Therefore, this relevant aspect will be detailed in a future work to propose a reputation scheme based on the participation of the domains in the cooperative architecture.

References

1. Akamai: How to Protect Against DDoS Attacks - Stop Denial of Service. `https://www.akamai.com/us/en/resources/protect-against-ddos-attacks.jsp` (2016), [Online, accessed 2017-1-10]
2. Bocek, T., Stiller, B.: Smart Contracts - Blockchains in the Wings, pp. pp. 1–16. Springer, Tiergartenstr. 17, 69121 Heidelberg, Germany (jan 2017)
3. CloudFare: Cloudflare advanced ddos protection (2016), `https://www.cloudflare.com/static/media/pdf/cloudflare-whitepaper-ddos.pdf`
4. Dao, N.N., Park, J., Park, M., Cho, S.: A feasible method to combat against ddos attack in sdn network. In: 2015 International Conference on Information Networking (ICOIN). pp. pp. 309–311 (Jan 2015)
5. Dridi, L., Zhani, M.F.: Sdn-guard: Dos attacks mitigation in sdn networks. In: 2016 5th IEEE International Conference on Cloud Networking (Cloudnet). pp. pp. 212–217 (Oct 2016)
6. Fund, E.: Ether unit converter (jan 2016), `http://ether.fund/tool/converter`
7. Jonker, M., Sperotto, A., van Rijswijk-Deij, R., Sadre, R., Pras, A.: Measuring the Adoption of DDoS Protection Services. In: Proceedings of the 2016 ACM on Internet Measurement Conference. IMC '16, Santa Monica, California, USA (2016)
8. Kandoi, R., Antikainen, M.: Denial-of-service attacks in openflow sdn networks. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). pp. pp. 1322–1326. IEEE (2015)
9. Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: A comprehensive survey. Proceedings of the IEEE 103(1), pp. 14–76 (2015)
10. Mansfield-Devine, S.: The Growth and Evolution of DDoS. Network Security (10), pp. 13–20 (2015)
11. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review 38(2), pp. 69–74 (2008)

12. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
13. Nishizuka, K., Xia, L., Xia, J., Zhang, D., Fang, L., Gray, C.: Inter-organization cooperative ddos protection mechanism. Draft (December 2016), `https://tools.ietf.org/html/draft-nishizuka-dots-inter-domain-mechanism-02`, draft
14. Peng, T., Leckie, C., Ramamohanarao, K.: Survey of network-based defense mechanisms countering the dos and ddos problems. ACM Computing Surveys (CSUR) 39(1), pp. 3 (2007)
15. Rashidi, B., Fung, C.: Cofence: A collaborative ddos defence using network function virtualization. In: 12th International Conference on Network and Service Management (CNSM 16) (October 2016)
16. Sahay, R., Blanc, G., Zhang, Z., Debar, H.: Towards autonomic ddos mitigation using software defined networking. In: SENT 2015: NDSS Workshop on Security of Emerging Networking Technologies. Internet society (2015)
17. Scott-Hayward, S., O'Callaghan, G., Sezer, S.: Sdn security: A survey. In: Future Networks and Services (SDN4FNS), 2013 IEEE SDN For. pp. pp. 1–7. IEEE (2013)
18. Shin, S., Gu, G.: Attacking software-defined networks: A first feasibility study. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. pp. pp. 165–166. ACM (2013)
19. Steinberger, J., Kuhnert, B., Sperotto, A., Baier, H., Pras, A.: Collaborative ddos defense using flow-based security event information. In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. pp. 516–522 (April 2016)
20. The Associated Press: Hackers Used 'Internet of Things' Devices to Cause Friday's Massive DDoS Cyberattack. `http://www.cbc.ca/news/technology/hackers-ddos-attacks-1.3817392` (Oct 2016), [Online, accessed 2017-1-10]
21. Velauthapillai, T., Harwood, A., Karunasekera, S.: Global detection of flooding-based ddos attacks using a cooperative overlay network. In: Network and System Security (NSS), 2010 4th International Conference on. pp. pp. 357–364. IEEE (2010)
22. Wei, L., Fung, C.: Flowranger: A request prioritizing algorithm for controller dos attacks in software defined networks. In: 2015 IEEE International Conference on Communications (ICC). pp. pp. 5254–5259. IEEE (2015)
23. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger (jan 2016), `https://goo.gl/LG7adX`
24. Zhang, G., Parashar, M.: Cooperative defence against ddos attacks. Journal of Research and Practice in Information Technology 38(1), pp. 69–84 (2006)