# Blockchain-Enabled Collaborative Intrusion Detection in Software Defined Networks

Wenjun Fan*, Younghee Park†, Shubham Kumar†, Priyatham Ganta†, Xiaobo Zhou* and Sang-Yoon Chang*
*Computer Science Department, College of Engineering and Applied Science
University of Colorado Colorado Springs, Colorado Springs, United States, CO 80918
Email: {wfan, xzhou, schang2}@uccs.edu
†Computer Engineering Department,
San Jose State University, San Jose, United States, CA 95192
Email: {younghee.park, shubham.kumar, priyatham.ganta}@sjsu.edu

***Abstract*—Collaborative intrusion detection system (CIDS) shares the critical detection-control information across the nodes for improved and coordinated defense. Software-defined network (SDN) introduces the controllers for the networking control, including for the networks spanning across multiple autonomous systems, and therefore provides a prime platform for CIDS application. Although previous research studies have focused on CIDS in SDN, the real-time secure exchange of the detection-relevant information (e.g., the detection signature) remains a critical challenge. In particular, the CIDS research still lacks robust trust management of the SDN controllers and the integrity protection of the collaborative defense information to resist against the insider attacks transmitting untruthful and malicious detection signatures to other participating controllers. In this paper, we propose a blockchain-enabled collaborative intrusion detection in SDN, taking advantage of the blockchain's security properties. Our scheme achieves three important security goals: to establish the trust of the participating controllers by using the permissioned blockchain to register the controller and manage digital certificates, to protect the integrity of the detection signatures against malicious detection signature injection, and to attest the delivery/update of the detection signature to other controllers. Our experiments in CloudLab based on a prototype built on Ethereum, Smart Contract, and IPFS demonstrates that our approach efficiently shares and distributes detection signatures in real-time through the trustworthy distributed platform.**

***Index Terms*—Blockchain, Intrusion Detection, Detection Signature, IDS, SDN, Ethereum, Smart Contract, Snort**

## I. Introduction

An intrusion detection system (IDS) is designed to detect malicious behavior to protect the target system/network. Usually, a standalone IDS is able to protect one target organization, however, in order to protect large networks or IT ecosystems from the high distributed attacks or the attack spreading across multiple domains, the isolated IDSes will not be able to establish connections for communication and sharing alerts and incident information. Such task can be facilitated by Collaborative IDS (CIDS) [1]. A CIDS can provide a global view for the large-scale networks/systems in contrast to the standalone IDS only observing the intrusion events occurring at one place. The strength of the CIDS is that it enables the data sharing of the intrusion events happening in multiple domains through sharing the attack information between the distributed IDSes.

Software Defined Networking (SDN) has proved itself to be a backbone in the new era's network design and is increasingly becoming an industry standard [2]. A regular SDN controller is designed to provide typically centralized intelligence for controlling the network flow traversing on the data plane. Such centralized controller lacks the capability to protect large networks. Nowadays, CIDS scheme is also pervasively applied to the SDN-based networks using collaborative scheme [3], [4] to share information and update parameters of the flow-table of the controllers across multiple domains.

Because the cyber threat information (CTI)[1] sharing is the critical process to support the collaborative security intelligence for protecting a distributed networks, a secure approach is necessary, especially for sharing security data. There are a number of cloud computing-based centralized security data outsourcing solutions for establishing collaborative protection over multiple domains, e.g. the EU H2020 C3ISP project [5], while that needs a centralized cloud server for CTI data collection and sharing. That scheme is not feasible for full decentralized peer-to-peer (P2P) network contexts. A full decentralized and collaborative P2P network has no centralized server, and all the peers on the network share their knowledge and information between each other directly.

To this end, we are motivated to apply blockchain technology to facilitate the secure data sharing since blockchain has a number of inherent security properties to support such application (see Section V-A). In particular, our blockchain-enabled approach aims to share the new intrusion detection signature generated by collaborative IDS in order to update local IDS engine. Every individual SDN controller plays a centralized intelligence role for its network domain, while the collaborative SDN controllers themselves constitute a P2P network for sharing information so that they can learn new detection signatures with each other.

The contribution of this work can be summarized as follows:

[1]Cyber threat information (CTI) is the general term used in the context of network security area for data sharing

- We build a blockchain-enabled collaborative intrusion detection scheme (called Detection-Signature-Chain or DS-Chain) for sharing, updating, and attesting detection signatures across controllers P2P network.
- We design the blockchain-based distributed PKI scheme (called Certificate-Chain or C-Chain) to establish trust over the participating nodes.
- We integrate C-Chain and DS-Chain for secure updates on CIDS in SDN networks.
- We present a smart-contract-based practical byzantine fault tolerance (PBFT) distributed consensus to support "$n$-compromise resistance" against insider attacks.
- We prototype the system design of our approach integrating SDN Ryu controllers, Snort network intrusion detection, Ethereum as well as its smart contract, and IPFS for detection signature file store.
- We conduct experiments to demonstrate that our approach is effective and efficiency.

## II. Related Work

### A. *Blockchain-enabled Collaborative IDS*

Applying blockchain to the context of CIDS has several positive impacts in the aspects of data sharing, alerts exchange and trust computation [6]. Alexopoulos et al. [7] introduced a generic blockchain-based CIDS architecture and some design considerations with the purposed of using blockchain's properties to improve the trust, accountability and consensus between the participating monitors. However, the authors didn't solve those specific problems like how to store and manage the big size alert data, while just mentioned to store the hashes of the raw alert data for reducing communication overhead. Such problem is addressed by our approach. CBSigIDS [8] was designed as a consortium blockchain-based CIDS aiming to establish trust of the participating IDS nodes to defender against inside attacker who can transmit untruthful detection signature rules to other nodes. CBSigIDS applies challenge-based trust mechanism to build node's trustworthiness, while uses node's private key to sign the detection signature rules to be transmitted and needs other majority of nodes to verify the signed rules by using the sender node's public key. CBSigIDS still relies on a centralized certificate authority (CA) to register participants, which suffers the single point of failure. That is addressed by our approach by using a distributed PKI. Ajayi et al. [9] proposed a permissioned blockchain approach to secure the detection signature extraction, storage and distribution stages. However, the authors did not explicitly describe their trust mechanism of registering/authorizing the nodes who can write on the ledger, which is addressed by our approach. Meng and Li engaged in a serial of research focusing on improving the trust management in collaborative intrusion networks using blockchain, such as using blockchain (nodes) to verify the received feedback replying the challenge request [10], and also building a verified chain of malicious feedback to establish a blockchain-based trust [11]. Both research rely on a trusted centralized 3rd party CA to issue the unique proof-of-identity (e.g., a pair of public and private key) to the participating nodes, while we use blockchain-based distributed PKI.

### B. *Collaborative IDS in Software-defined Networks*

During the past years, several research were conducted for integrating CIDS into OpenFlow based SDN. CIPA [12] presented an artificial neural network-based CIDS dispersing the neurons (the computational power) over the programmable SDN switches. Hameed and Khan [13] designed a secure controller-to-controller (C-to-C) protocol enabling SDN controllers of different autonomous systems (AS) to exchange attack information. Their C-to-C protocol is flexible in deployment in linear order, P2P, or centralized scheme to collaboratively disseminate attack information. SeArch [4] is comprised of a hierarchical layer of intelligent IDS nodes working in collaboration to detect anomalies and formulate policy into the SDN-based IoT gateway devices to stop attacks. All the above research did not consider the trust of the collaborative nodes, while our approach addresses this issue by using blockchain.

### C. *Blockchain-based SDN Control Plane*

The research about the deployment of blockchain technology in software-defined networks have emerged for tackling the security issues existing in prior SDN contexts [14], in particular, the research of blockchain-based distributed SDN control plane for securing the information update between controllers is related to our work. A case in point is the proposal [15] that applies a permissioned blockchain to maintain the a list of updated system activities and time stamps in each controller. However, the work lacks a consensus protocol to add new participants although it uses Simplified Practical Byzantine Fault Tolerance (SPBFT) consensus to broadcast message/request, and also, it overlooks the insider attacks of the controllers. In contrast, our approach covers these issues.

### D. *Information Sharing Using Blockchain with IPFS*

In fact, blockchain is deficient to store large size data/file, while InterPlanetary File System (IPFS) [16] provides a high throughput content-addressed block storage model with content addressed hyper links. IPFS often cooperates with blockchain for providing a distributed data storage. To share privacy/sensitive information through public blockchain integrated with IPFS, the confidentiality issue has to be taken into account. Wang et al. [17] proposed a blockchain-based framework for data sharing, which uses IPFS to store the encrypted file, and uses Ethereum permisionless blockchain to store the file location returned by IPFS. The framework applies attribute-based encryption mechanism which allows the data owner to specify the data access policy based on the user's identity and attributes to achieve fine-grained access control over data. By contrast to encrypt the file, Steichen et al. [18] proposed an alternative way that is to use the Ethereum smart contract to conduct an access control list (ACL) of the permissions to access the file stored in IPFS, and allow the IPFS to enforce the ACL. The drawback of this solution is that it needs to modify the original IPFS code to integrate the ACL. In our approach, we use permissioned blockchain that builds the trust of the participants and distribute detection signature files which have no sensitive/privacy information.

## III. System Model

### A. Network with Multiple SDN Controllers

In this paper, we aim to decentralize the SDN control plane through leveraging multiple controllers with collaborative detection communication to gain a coordinated defense. We define the (virtual) network boundary of the coordinated defense as one autonomous system (AS) overseeing multiple controllers which could cross geographical domains, or as a single-domain with multiple controllers/firewalls enabling multiple networking inbound points. By this definition, we assume that there is no outsider attacks threatening the coordination communication between the controllers, and we focus on the insider attacks which can compromise the controller nodes and exploit the communications.

### B. Threat Model

Now that we defined our network boundary and coordinated defense sustained by our approach using blockchain-based controllers P2P network with collaborative detection, we can also build the threat model that motivates us to defend against. The first threat is against the availability of the conventional centralized intelligence using one SDN controller, which suffers the single point of failure, and that is considered and addressed by using decentralized and coordinated controllers. The second threat is against the integrity of the collaborative defense communication between controllers, without any protection, an (inside) attacker can tamper the transmitted information. We apply blockchain to provide tamper-resistance to the transactions (containing the detection information) of the collaborative defense communication. Because we assume our collaborative defense communication will not exfiltrate the defined network boundary, we consider there is no outside attacker can access such inside defense information. In this regard, there is no confidentiality problem since the access to the information should be from the insiders, but we need to register all the insiders/participants. Therefore, we use permissioned blockchain to control the participation in the coordinated defense riding on the blockchain transaction creation and processing. Further, we consider the compromised controllers transmitting untruthful or malicious defense information launched by inside attackers who use the legitimate identities or registered as a legal users at the participant registration stage. To address this issue, we propose the "$n$-compromise resistance" using PBFT consensus protocol, whereby every transaction containing defense information (i.e. detection signature) must be attested by a quorum of other participants to resist against even $n$ participants are compromised or controlled by the attackers.

## IV. System Design

We present our approach by describing the system design in this section and the blockchain scheme in the next section.

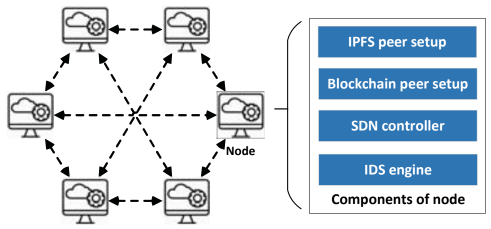

Fig. 1: Full distributed P2P network topology in our approach

### A. Architecture

Figure 1 graphically shows a high level full distributed P2P network topology of the participants in our approach, whereby each participant node can communicate with every other node registered in the permissioned blockchain-based P2P network. Further, every node consists of multiple functional components, and they are *IDS engine*, *SDN controller*, *Blockchain peer setup*, and *IPFS peer setup*.

The IDS engine is used to detect intrusion using existing signature-based rules, and generate new detection signature using anomaly-based approach. The SDN controller is used to control the underlying data plane through installing flow entries, which include the security related entries to filter and process the malicious traffic according to the detection signature information given by the IDS engine aforementioned. The blockchain peer setup is used to generate and process the transactions for propagating the detection signature from one SDN controller to other controller in order to facilitate the coordinated defense. Because it is not economically practical to store large amounts of data (like our detection signature file) on the blockchain, we apply InterPlanetary File System (IPFS) [16] to store the file, while use the blockchain to record and trace the detection signature file ID (SFID), which is the detection signature file address given by IPFS essentially.

Figure 2 shows the system design which includes the time sequence of the four steps walking by detection signature Generate, Import, Fetch and Update (from left to right in the diagram). We describe these four steps as follows: 1) in Generate step, the new detection signature is generated by the IDS engine on a source node, which informs the controller component; 2) in Import step, the source controller imports the detection signature file on the IPFS, the IPFS returns a SFID to the controller and the controller calls the smart contract to upload a transaction containing the SFID as payload to the blockchain; 3) in Fetch step, the destination controller reads the SFID delivered through the blockchain, and uses the SFID to request to the IPFS in order to get the corresponding detection signature file; 4) in Update step, the destination controller downloads the detection signature file from IPFS, and updates the detection signature on the destination IDS engine. Note that this diagram does not show the SFID attestation process that is described in Section IV-C.

### B. Chain Types

This subsection describes the blockchain types used in our approach. According to the proposed system design, we have two blockchains in terms of the types of payload ($T$) to

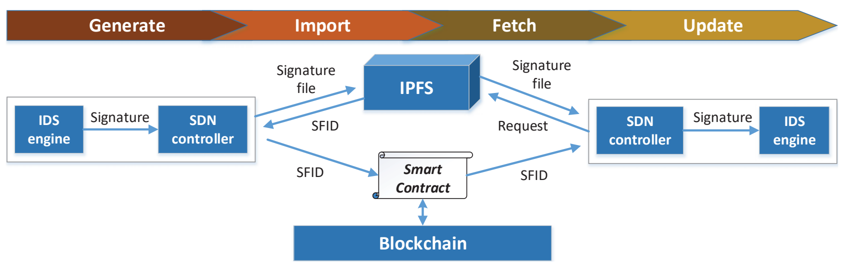

Fig. 2: Our system design for the blockchain-enabled CIDS in SDN with four steps: Generate, Import, Fetch, and Update

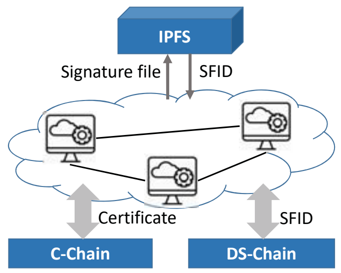

Fig. 3: Our system architecture involving controllers, IPFS, and blockchains

be transmitted over the blockchains respectively in order to realize the purpose of the collaborative intrusion detection across SDN networks.

We define *C-Chain* as the *Certificate Chain* for transmitting the certificate payload, and *DS-Chain* as the *Detection Signature Chain* for propagating the detection signature (ID) payload (see Figure 3). Both chains have the controllers as the peer nodes, while the C-Chain is used to control the participation of the controller, e.g. authorize or revoke the controller's certificate, and the DS-Chain is for broadcasting the detection signatures and the detection signature file attestation. Also, both blockchains are permissioned, since they limit the participation in the blockchain to only the certified nodes whose certifications are provided by C-Chain. So, the DS-Chain is more frequently-used than the C-Chain. Also, We use two separate blockchains rather than a single blockchain since they are for two different applications, in addition, they involve different distributed consensus protocols (which are described in greater details in Section V-B).

The payload type defined as $T$ is used for distinct functionalities of our blockchain scheme, where C-Chain is for $T = 1$ and DS-Chain is for $T = 2$ and $T = 3$. Note that $T = 1$ indicates that the payload is for the digital certificate (see Section V-D), $T = 2$ is for the SFID for updating the collaborative intrusion detection signature (see Section V-E), and $T = 3$ corresponds to the payload type for the attestation of the SFID update (see Section V-F).

## C. Detection Signature File Attestation

The DS-Chain automatically distributes the SFID for the controllers network detection signature update. Blockchain provides eventual consensus and is designed to be tolerant against temporary node failures; blockchain continues to operate when some nodes are experiencing failures to execute the protocol (in our case, when less than 1/3 of the controller nodes are experiencing such failure), while those nodes experiencing temporary failure, e.g., unavailable due to networking disconnection, can re-connect and re-synchronize at any time in the future[2]. Because blockchain allows such temporary de-synchronization although designed to eventually get synchronized, it by itself is insufficient to guarantee that the SFID has been delivered or is being used by the controllers at the time.

To allow attestation of the detection signature update, we build such additional functionality on the DS-Chain. The DS-Chain includes explicit attestation of the events of the delivery of the SFID to a controller and the controller's accessing the IPFS to retrieve the corresponding detection signature file. The attestation information is uploaded on the DS-Chain so that other controllers become aware that the controller (the subject of the attestation) updated its security rules in terms of the detection signature. Section V-E provides more details about the DS-Chain.

The attestation is critical since, when there is networking between the controller domains, the controllers have assurances that they are at the same security level. For example, if the explicit attestation is lacking on one side, a controller updated with the latest detection signatures is aware that a node managed by the controller lagging in detection signature updates can be vulnerable against the latest intrusion threat and can disallow certain networking or executions because it is less trustworthy.

## V. Blockchain Scheme

### A. Blockchain Properties motivating our approach

As stated, our approach leverages the properties of blockchain, this subsection points out those properties enabling our approach as follows.

- **Decentralization**: Multiple entities (in our case, the SDN controllers) are involved in the generation of the transactions associating with the detection signature. Only a quorum of controllers achieve consensus, will the transaction become viable. Therefore, our scheme is resistant against the insider compromises up to $n$ controllers, which is termed "$n$-compromise resistance".
- **Immutability**: All the effective transactions put into the distributed ledger will remain immutable, which can provide a strong capability to monitor and track the process of detection signature generation by the corresponding controllers.
- **Modularity**: The blockchain provides an additional layer of implementation and therefore can be modular to the rest of the SDN controllers operations. Such modularity

[2]Such blockchain design to tolerate temporary node failures but eventually agree on the blockchain state is generally applicable across the blockchain applications. For example, in permissionless Bitcoin, a node can re-connect with the blockchain network any time and download and synchronize the blockchain state at that time of re-connection, or there can be temporary soft forks causing distinct blocks which are both valid but eventually gets resolved by choosing one via the longest-chain rule.

**Algorithm 1** Smart-contract-based PBFT

```
Require: $(3n + 1)$ $C$ participate in
Ensure: $(2n + 1)$ $C$ agree
  $C_i$ submits a transaction
  $count = 1$
  $i' = 1$
  while $i' < (3n + 1)$ && $count < (2n + 1)$ do
    if $i' \neq i$ then
      $C_{i'}$ submits a transaction for vouching
    end if
    if $C_{i'}$ agree then
      $count = count + 1$
    end if
    $i' = i' + 1$
  end while
  if $count >= (2n + 1)$ then
    Transaction is vouched in terms of PBFT
    Block is generated
  else
    Transaction is not vouched
  end if
```

TABLE I: Comparison of different blockchain schemes

| Blockchain | Payload type, $T$ | Consensus protocol |
|---|---|---|
| C-Chain | $T = 1$ | Smart-contract-based PBFT |
| DS-Chain | $T = 2$ | Smart-contract-based PBFT |
| DS-Chain | $T = 3$ | PoA-based broadcasting |

is critical for our design goal of flexibility enabling the involvement of heterogeneous SDN controllers.

### *B. Distributed Consensus Protocols*

In our approach, we use two distributed consensus protocols for different purposes. First, we apply Proof-of-Authority (PoA) distributed consensus protocol. This consensus is used where it only needs the broadcasting of the transactions using PoA which is largely driven by identities and registration and is thus applicable in permissioned environments. The authority for PoA can be a controller in our approach. Second, we use PBFT in our approach to resist against byzantine failure of upto $n$ controllers, which is defined as $n$-compromise resistance. Because we utilize PBFT, we need $3n + 1$ participants in total while reach $2n + 1$ as the quorum of participants in order to achieve the $n$-compromise resistance.

We define $C$ as the participating controller node in the blockchain P2P network, and use smart contract to facilitate the PBFT consensus. The smart-contract-based PBFT consensus protocol is described by Algorithm 1. That is to say, every transaction issued by one participant (i.e. the controller in our case) using the smart contract, another $2n$ at least (plus the participant sending the transaction itself, the total number becomes $2n + 1$) participants must achieve an agreement by voting for it to make that transaction viable.

Table I summaries these two distributed consensus protocols used for distinct purposes by our C-Chain (see Section V-D) and DS-Chain (see Section V-E) respectively. Note that comparing such PoA-based broadcasting with smart-contract-based PBFT, any transaction transmitted by the PBFT smart contract is a voting-based PoA broadcast essentially. In other words, PoA-based broadcasting needs $3n$ unicasts to propagate one transaction effectively, while smart-contract-based PBFT needs $(2n + 1) \cdot 3n$ unicasts to make the transaction effective. Regarding the variables of the type of the payload, refer to Section V-C.

TABLE II: Definition of Variables

| Variable | Definition |
|---|---|
| $i$ | The index of the controller |
| $j$ | The index of time |
| $e$ | The index of block |
| $x$ | The index of the detection signature file |
| $t$ | The number of generated private-public key pairs |
| $C_i$ | The $i$-th controller |
| $I_i$ | The identity of $i$-th controller |
| $B_e$ | The $e$-th block |
| $F_x$ | The $x$-th detection signature file |
| $T$ | The type of payload where $T \in \{1, 2, 3\}$ |
| $P_T$ | The payload of type $T$ |
| $K_{i,j}$ | The public key of $i$-th controller at time $j$ |
| $k_{i,j}$ | The private key of $i$-th controller at time $j$ |
| $n$ | The $n$ of $n$-compromise resistance |
| $H$ | The hash function |
| $E$ | The encryption function |
| $D$ | The decryption function |
| $G$ | The digital certificate generation function |

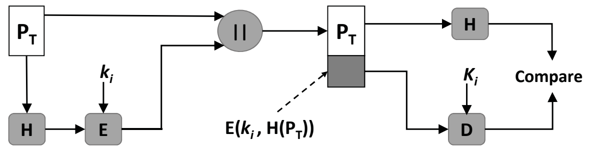

Fig. 4: Standard digital signature ($i$ is the sender controller)

### *C. Block Construction*

Given a payload (also called transactions or records, depending on the blockchain applications), this section describes the construction of the block containing the payload. Table II described the variables used in the context. The block contains the followings: the hash of the previous block (in plaintext), the payload $P_T$ (in plaintext), the type of payload $T$ (in plaintext), and the digital signature[3] (processed using the aforementioned data included in the block). In other words, the $e$-th block $B_e = \{H(B_{e-1})||P_T||T||E(k_i, H(P_T))\}$, where $T \in \{1, 2, 3\}$. We adopt the standard digital signature scheme (see Figure 4). The hash output of the previous payload $H(P_T)$ is the input of the digital signature algorithm using the sender controller's $k_i$, so the digital signature is $E(k_i, H(P_T))$. The concatenation of $E(k_i, H(P_T))$ and $P_T$ will be used for verification using the sender controller's $K_i$. So, the recipient controller takes the concatenation message and produces a hash code. The recipient controller also decrypts the digital signature using the sender's $K_i$. If the calculated hash code matches the decrypted digital signature, the digital signature is accepted as valid. Because only the sender controller knows the $k_i$, only the sender controller could have produced a valid digital signature.

The block is therefore specific to the controller generating the block and has strong source integrity, e.g., other controllers can verify the digital signature to check that the block got generated by the source controller. This block construction is applicable for both of our blockchains described in the

[3]The digital signature is the cryptographic mechanism based on public-key ciphers for integrity and authentication protection and not the signature used for IDS. For clarity, we refer to the IDS signature as *detection signature*.

following sections. Note that since in our case we use one block to contain only one transaction, so the term "block" and "transaction" are exchangeable in the context, which is essentially the payload.

## *D. C-Chain for Distributed PKI ($T = 1$)*

A blockchain can be used for permissioned if there is an off-line registration. The permissioned blockchain controls the participation in the blockchain transaction generation and processing. In our approach, the blockchain for the SDN controllers P2P network is permissioned, and only the registered controllers are allowed to transmit payload. In contrast to the permissionless blockchain-based cryptocurrency, such as Bitcoin, our permissioned blockchain-based approach reduces the networking overhead for practicality.

To this end, the C-Chain is used to facilitate the blockchain-based distributed PKI to replace the centralized CA and provides trust to the participating controllers that need to be established. For the blockchain, the payload is the certificate since $T$=1. In this regard, we define that for any controller $C_i$, $i \in \{1, 2, ....., 3n + 1\}$, $3n + 1$ indicates the total number of controllers. For any time $j$, $j \in \{1, 2, ....., t\}$, $t$ represents the entire number of the created public-private key pairs. Based on that, we define $K_{i,j}$ as the public key of $C_i$ at time $j$, while $k_{i,j}$ is the private key of $C_i$ at time $j$, and $I_i$ is the identity of $C_i$. So, $P_1$=$\{I_i||K_{i,j}\}$. We define $G$ as the function for generating digital certificate, so the digital certificate is $G(K_{i,j}, I_i, k_{i,j}) = \{I_i||K_{i,j}||E(k_i, H(I_i||K_{i,j}))\}$. Because the payload structure (i.e., $P_1$=$\{I_i||K_{i,j}\}$) and the type of payload (i.e., $T$=1) for the C-Chain are fixed, so the block can be constructed by the means described in Section V-C. Note that the C-Chain uses the smart-contract-based PBFT consensus, which means one certificate transaction must get voted by more than 2/3 of the whole participating controllers, otherwise, the certificate will not be viable. Also, an effective certificate can be revoked, and the revocation needs to be compliance with the PBFT consensus.

## *E. DS-Chain for SFID Delivery and Update ($T = 2$)*

This DS-Chain realizes two purposes: the SFID delivery requiring PBFT consensus and the SFID attestation requiring PoA-based broadcasting where the PoA-validators are the sender controllers themselves. This section focuses on describing the SFID delivery and update, and Section V-F will depict the SFID attestation in greater detail.

As mentioned, for this chain, the payload is the SFID since $T$=2, and the SFID is returned by IPFS when the controller imports a detection signature file into IPFS. Therefore, the payload (i.e., $P_2$=SFID) and the type of payload (i.e., $T$=2) of the DS-Chain for SFID delivery are fixed, the block can be constructed by the means described in Section V-C as well. Also, the SFID delivery performed by the DS-Chain applies smart-contract-based PBFT consensus. In other words, every SFID transaction riding on one PoA-based broadcasting needs at least another 2/3 votes of the entire participating controllers and each vote requires one PoA-based broadcasting.

In addition, one detection signature file can contain one detection signature or multiple ones (e.g., with a certain number, or with a fixed update frequency[4]). If using one file to include only one detection signature, the controller can instantly broadcast the defense information to inform other controllers, which is appropriate to defend against the urgent and large-scale attacks to have a immediate detection signature update over the participating controllers in order to protect the SDN networks. However, this method increases the networking overhead since every SFID transaction needs the PBFT-based voting. So, an alternative way is to allow one file to contain multiple detection signatures and upload them together into the IPFS, and then only one transaction containing the given SFID (which represents the file including multiple detection signatures) needs to be transmitted by the DS-Chain using PBFT consensus. The latter method is more networking efficient than the former one, while the latter can not transmit an immediate detection signature until it collects enough ones. For example, assuming one detection signature file can contain $X$ detection signatures, and the blockchain P2P network includes 3 controllers. If we use the method of one file containing X detection signatures, the system will have 3 transaction broadcasts following PBFT consensus, while if we use the method of one file containing only one detection signature, then the system will have $3{\cdot}X$ transaction broadcasts in terms of PBFT consensus.

## *F. DS-Chain for SFID Attestation ($T = 3$)*

As aforementioned, DS-Chain is also in charge of transmitting SFID attestation. This type of payload is transmitted using PoA-based broadcasting where the PoA-validator is the controller itself who sends the payload.

Upon receiving the SFID-delivery block in Section V-E, the controller retrieves and updates the corresponding detection signature file. Given a new SFID $x$, the controller retrieves the detection signature file $F_x$, computes the hash of the file $H(F_x)$ for the block payload, i.e., $P_3 = H(F_x)$. The payload as well as the type of payload (i.e., $T$=3) are used to construct the block, including the digital signature, as described in Section V-C. Therefore, the block is specific to the controller (with the source integrity provided by the digital signature) and to the detection signature file $F_x$.

In contrast to the type of payload for SFID delivery described in Section V-E requiring multiple distinct transactions signed by multiple controllers (more than 2/3 of the controllers), the type of payload for attestation requires one block using PoA as the controller being the subject of the attestation and creating the attestation block being the validator.

## VI. Implementation and Experiment

In this section, we describe the implementation detail of our prototype in terms of the design, and present the experimental results built on the prototype.

[4]PulledPork for Snort and Suricata rule management updates the rulesets of detection signatures twice a week

TABLE III: VM specification in CloudLab

| | |
|---|---|
| Virtualization | Xen |
| Operating system | Ubuntu 16.04.1 LTS (64-bit) |
| CPU | Intel(R) Xeon(R) CPU E5-4620 0 @ 2.20GHz |
| Processor number | 1 |
| Memory | 4 GB |
| Network Adapter | Ethernet Physical |

TABLE IV: Measurement of deploying payloads in blockchain

| **Type of payload** | **Gas used** | **ETH cost** | **Block size (bytes)** | **Block verification time (ms)** |
|---|---|---|---|---|
| $T = 1$ | 27594 | 0.00055188 | 1027 | 998.530 |
| $T = 2$ | 25185 | 0.0005037 | 1013 | 998.419 |
| $T = 3$ | 38620 | 0.0007724 | 924 | 998.308 |

### A. Implementation

We use Ethereum (Geth version 1.9.16-stable) as well as smart contract (Solidity version 0.5.16) to facilitate the blockchain-enabled approach. We use Truffle (version 5.1.34) to deploy the smart contract on the Ethereum blockchain. Also, we select Ryu (version 4.34 with OpenFlow 1.3) as the SDN controller which has built-in Snort library. So, the Ryu framework integrated with the Snort library can facilitate our IDS engine and SDN controller components, and the Snort library supports developing detection program that can inform detection alert to the controller [19].

We employ CloudLab as the underlying platform to build the blockchain P2P network to prototype our approach. The virtual machine's specification is described in Table III. We initially create a three-nodes-based P2P network, and each node is hosted by a VM where we install the above mentioned softwares to prototype our design. We implement the three-nodes prototype testbed on Emulab cluster for carrying out basic measurement. Specifically, built on the testbed, the average round trip times (RRT) between every two nodes are 0.942 ms (node A to node B), 0.968 ms (node A to node C), and 1.012 ms (node B to node C).

### B. Experimental Results

*1) Computation Overhead:* This subsection presents the computational overhead based on our implementation. Table IV shows the measurement of pushing the different types of payload into corresponding blockchain using smart contract. We observe the cost of blockchain resource in terms of different payloads. The certificate payload takes the most blockchain resource of the three types when the Ethereum blockchain processes the payload, since the certificate payload is much greater and more complex than the other two types, e.g. using X.509 as the certificate format in our approach. In our case, the gas price is 20 Gwei, so $gas{\cdot}(20{\cdot}Gwei) = ETH$, where $10^9 \cdot Gwei = 1ETH$. Also, because we configure the Ethereum to use PoA consensus to broadcast the payloads, where our smart-contract-based PBFT consensus builds on and every broadcast is a PoA-based broadcast. So, for broadcasting each payload regardless of the scale of the blockchain P2P network, the block verification time is always around 998 ms.

In addition, Table V shows the CPU and memory usages as well as the time of executing different tasks using blockchain. We find that running the Ethereum blockchain costs CPU 1.18% and deploying smart contract costs CPU 3.06%, while

TABLE V: CPU, MEM and execution time when carrying out different tasks using blockchain

| **Task** | **CPU (%)** | **MEM (MB)** | **Exec. Time (ms)** |
|---|---|---|---|
| Run Geth | 1.18 | 303.35 | – |
| Deploy smart contract | 3.06 | 652.46 | 1015 |
| Push certificate ($T = 1$) | 4.99 | 63.72 | 126.2 |
| Push SFID ($T = 2$) | 4.54 | 60.94 | 121.1 |
| Push SFID attestation ($T = 3$) | 5.3 | 53.31 | 180 |

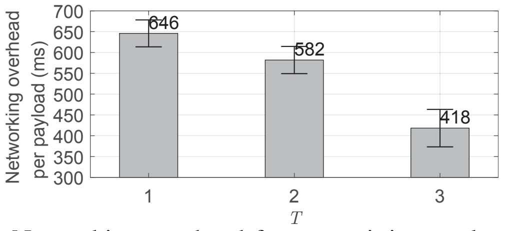

Fig. 5: Networking overhead for transmitting payloads ($T$)

they take RAM memory 303.35 MB and 652.46 MB respectively. Note that the cost of running Geth is persistent as long as the Ethereum blockchain is running, while the cost of deploying smart contract only lasts around 1.015 second as the Table shows.

Now that the blockhcain is running and the smart contract has been deployed on the blockchain, we test the tasks for pushing different types of payloads on the blockchain by calling the smart contract. We can see that the task for $T = 3$ to push the attestation takes the most CPU usage (5.3%) and execution time (180 ms), since it keeps track of the number of controllers which have sent back their attestation values after downloading the detection signature files using a local count variable, and increases the count of these controllers which have sent back their attestation values. Because the attestation payload size is the smallest of the three types, the task costs the least memory usage. Also, the tasks for $T = 1$ and $T = 2$ are very similar, just to construct the corresponding payload and submit it. However, the certificate payload is larger than the SFID payload, that is why the task for $T = 1$ takes more computational overhead than the task for $T = 2$. Note that we do not show the underlying PBFT consensus cost for both payloads over here.

*2) Networking Overhead:* This subsection presents the networking overhead measurement results based on our prototype. We first test the average time for transmitting every specific type of payload (see Figure 5, which shows the average time values with 95% confidence interval). We find that transmitting one certificate payload ($T = 1$) takes network latency 646 ms on average, one SFID payload ($T = 2$) takes 582 ms on average, and one attestation ($T = 3$) takes 418 ms on average.

Further, because both certificate payload ($T = 1$) and SFID payload ($T = 2$) use PBFT consensus for ensuring every payload viable, from a node-level perspective, they both need $(2n + 1) \cdot 3n$ transmissions/unicasts to get the votes for endorsing the payload according to our $n$-compromise resistance, where $3n$ is the rest number of controllers excluding the sender controller and $2n + 1$ indicates a quorum of controllers to achieve majority decision. By contrast, for the

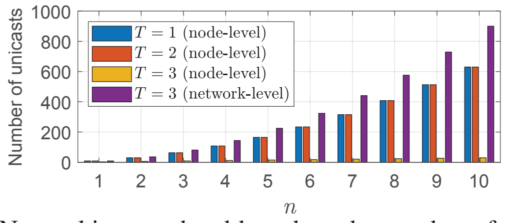

Fig. 6: Networking overhead based on the number of unicasts

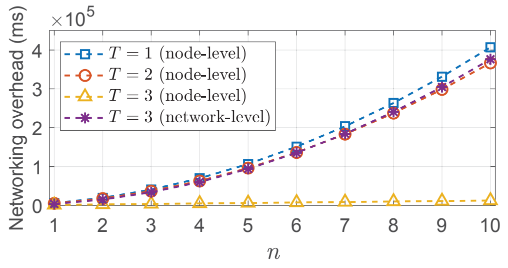

Fig. 7: Networking overhead in terms of latency

attestation payload ($T = 3$), from a node-level perspective, it only needs every receiver node to tell the sender node that it already downloads and uses the new detection signature, thus it includes $3n$ transmissions/unicasts; while from a network-level perspective, it needs $3n \cdot 3n$ transmissions/unicasts since every other controller ($3n$ in total) needs to tell other $3n$ controllers that it already updates the detection signature. Figure 6 shows the increasing number of unicasts for making one certifcate and SFID payload viable, and finishing one SFID attestation in terms of the increasing $n$ (of $n$-compromise resistance).

Therefore, using the average network latency per payload transmission shown by Figure 5 to multiply the number of transmissions shown by Figure 6, we can get the total networking overhead (latency) in terms of the different types of payload (see Figure 7). We can see that the node-level networking overhead for the attestation is orders of magnitude less than the certificate and SFID, while the network-level networking overhead for the attestation is relatively comparable to the other two payloads' node-level networking overheads.

## VII. Conclusion

Integrating collaborative intrusion detection with the software defined networks is a promising research direction, since SDN has become the standard network management which separates the flow control intelligence from the data plane while the CIDS strength can improve the security of the individually centralized SDN networks by forming a coordinated defense across multiple domains. In this paper, we propose a blockchain-enabled collaborative intrusion detection in SDN networks, which provides the trust management of the controllers and the integrity of the detection signature sharing over the controllers to gain a coordinated defense and defend against insider attacks supported by $n$-compromise resistance. We develop our prototype using Ethereum, smart contract, Ryu SDN framework and IPFS in CloudLab, and the experiments show that our approach is effective and efficient for sharing detection signatures in real-time through the trustworthy distributed platform.

## Acknowledgment

This research was supported in part by Colorado State Bill 18-086. This research was also partially supported by the grant from the National Science Foundation (NSF-#1723804).

## References

[1] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Comput. Surv.*, vol. 47, no. 4, May 2015.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] S. Hameed and H. A. Khan, "Sdn based collaborative scheme for mitigation of ddos attacks," *Future Internet*, vol. 10, p. 23, 2018.

[4] T. G. Nguyen, T. V. Phan, B. T. Nguyen, C. So-In, Z. A. Baig, and S. Sanguanpong, "Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks," *IEEE access*, vol. 7, pp. 107 678–107 694, 2019.

[5] D. W. Chadwick, W. Fan, G. Costantino, R. de Lemos, F. D. Cerbo, I. Herwono, M. Manea, P. Mori, A. Sajjad, and X.-S. Wang, "A cloud-edge based data security architecture for sharing and analysing cyber threat information," *Future Generation Computer Systems*, vol. 102, pp. 710 – 722, 2020.

[6] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: A review," *IEEE Access*, vol. 6, pp. 10 179–10 188, 2018.

[7] N. Alexopoulos, E. Vasilomanolakis, N. R. Ivánkó, and M. Mühlhäuser, "Towards blockchain-based collaborative intrusion detection systems," in *Critical Information Infrastructures Security*, 2018, pp. 107–118.

[8] S. Tug, W. Meng, and Y. Wang, "Cbsigids: Towards collaborative blockchained signature-based intrusion detection," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1228–1235.

[9] O. Ajayi, O. Igbe, and T. Saadawi, "Consortium blockchain-based architecture for cyber-attack signatures and features distribution," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2019, pp. 0541–0549.

[10] W. Li, Y. Wang, J. Li, and M. H. Au, "Towards blockchained challenge-based collaborative intrusion detection," in *International Conference on Applied Cryptography and Network Security*, 2019, pp. 122–139.

[11] W. Meng, W. Li, L. T. Yang, and P. Li, "Enhancing challenge-based collaborative intrusion detection networks against insider attacks using blockchain," *International Journal of Information Security*, pp. 1–12, 2019.

[12] X.-F. Chen and S.-Z. Yu, "Cipa: A collaborative intrusion prevention architecture for programmable network and sdn," *Computers Security*, vol. 58, pp. 1 – 19, 2016.

[13] S. Hameed and H. Ahmed Khan, "Sdn based collaborative scheme for mitigation of ddos attacks," *Future Internet*, vol. 10, no. 3, p. 23, 2018.

[14] T. Alharbi, "Deployment of blockchain technology in software defined networks: A survey," *IEEE Access*, vol. 8, pp. 9146–9156, 2020.

[15] Z. Shao, X. Zhu, A. M. Chikuvanyanga, and H. Zhu, "Blockchain-based sdn security guaranteeing algorithm and analysis model," in *International Conference on Wireless and Satellite Systems*. Springer, 2019, pp. 348–362.

[16] J. Benet, "Ipfs-content addressed, versioned, p2p file system (draft 3)," *arXiv preprint arXiv:1407.3561*, 2014.

[17] 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. 38 437–38 450, 2018.

[18] M. Steichen, B. Fiz, R. Norvill, W. Shbair, and R. State, "Blockchain-based, decentralized access control for ipfs," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1499–1506.

[19] W. Fan and D. Fernandez, "A novel sdn based stealthy tcp connection handover mechanism for hybrid honeypot systems," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–9.