EDISON: A Blockchain-based Secure and Auditable Orchestration Framework for Multi-domain Software Defined Networks

Chandrasekar Balachandran, Puneet A.C, Gowri Ramachandran, and Bhaskar Krishnamachari
Viterbi School of Engineering, University of Southern California, Los Angeles, USA
{cbalacha, puneetac, gsramach, bkrishna}@usc.edu

***Abstract*—The emerging networking standards such as 5G and 6G, coupled with technologies like Software Defined Networks (SDN) and Network Function Virtualization (NFV), are increasingly moving towards a multi-tenant and multi-vendor deployment model. Under these circumstances, the hardware vendors rent their networking and computation resources to multiple service providers and application developers. Such a deployment model lets various vendors collaboratively offer networking services to the tenants and the end-users at far greater efficiency and better affordability. However, the issues around trust, ownership, and data security become a concern for tenants and vendors in such multi-tenant and multi-vendor setting. In particular, the centralized nature of SDN controllers, together with the limitations of the contemporary authentication and access control mechanisms, make multi-stakeholder SDN deployments susceptible to several Sybil and trust-related exploits. We present EDISON, a blockchain-based authentication and access control framework, for multi-stakeholder SDN infrastructure that adheres to the Zero-trust security model. It allows the network vendors and third-party service providers to securely set up a service-level agreement while enabling the concerned stakeholders to audit the network operations through an end-to-end encrypted tamper-proof ledger. EDISON creates an ecosystem structured on smart contracts, wherein the network elements rented and used by the tenants interact with the services deployed in the form of contracts to enable decentralized and transparent orchestration.**

***Index Terms*—Authentication, Zero-trust, SDN, Decentralized 5G, NFV, Blockchain, SD-WAN**

I. Introduction

The software-defined networks (SDN) are at the forefront of emerging 5G and 6G technologies, managing the configuration and control operations of networking elements such as routers, switches, and other physical hardware via software abstraction. Through such "softwarization", SDN separates the control and the data plane, wherein the former manages the network and handles issues such as congestion control and packet losses, while the latter is responsible for application data transmission [1]. Such a SDN model branched out of data centers into more distributed and broader networks, leading to the inception of Software-Defined Wide Area Networks [2], that are now considered to be the backbone of the 5G and 6G networks.

SDN creates multiple layers of network abstraction over the physical hardware-based network elements like switches and routers to run multiple virtualized services for different vendors and tenants. Network component virtualization, commonly referred to as Network Function Virtualization (NFV), is used in concert with SDN to create virtual networks through overlays. These technologies allow for service-aware network slicing that is hardware agnostic across multi-vendor, multi-tenant based network services chain.

The challenge of orchestrating businesses that involve multiple parties whose individual intent may lead to undesirable consequences others is an issue often faced by many industries, including those adopting SD-WAN [3]. Many solutions for decentralized, trust-less execution, and operation have been proposed. Still, few have gained as much traction and widespread adoption as those built using blockchain and digital ledger technologies. Such systems allow the formation and maintenance of consensus about the existence, status, progression of a set of shared facts, and even code/logic execution without any centralized authority.

These resulting trustless and auditable systems closely resemble the characteristics of zero-trust-based systems [4]. Such systems have grown in adoption with cloud and service-based operations to guarantee better security and reduction in breaches caused due to exploits in implicit trust models. zero-trust implies "trust no one implicitly" and this model relies on three core principles: Mandatory authentication for accessing any resource, adopt default least privilege access control, and always have secure, auditable log of events. Such an approach provides an advantage over traditional perimeter security, as the smaller segments present reduced attack vectors for malicious actors and allow for quicker breach detection. Various SDN and NFV features enable agile network packet processing, global view of network state, dynamic policy installation, and validation, to simplify the implementation of zero-trust micro-segmentation.

However, most of the prior mentioned solutions only aim to provide enhanced security to the vendor and instill little to no confidence within the network's end tenants, wherein tenants have zero visibility to the network management layer. Though Service Level Agreements (SLA) may be created, there are no methods for the end tenants to verify and validate the networks' actual behavior controlled by the vendors' SDN management layer. Hence, the end tenants are forced into an uneven handshake and thus require to implicitly trust the vendors completely, making them vulnerable to Sybil and

collusion-based attacks and breach in contracts with no simple means of detection and enforcement [5], [6]. Hence the need for a decentralized software defined network with the ability to provide secure audit trails within a zero-trust-based multi-vendor, multi-tenant network. Existing solutions either focus on logging transactions on the blockchain [7], [8] or consider resource orchestration in non-adversarial settings [9], [10].

We present, EDISON, (coined from "sEcure, Decentralized, and audItable Sdn OrchestratioN"), a blockchain-based orchestration framework to allow the SDN clients and vendors to create, manage and execute services through an auditable and zero-trust based solution. EDISON aims to use cryptographic key exchange that ensures forward secrecy, coupled with an identity-based authentication approach to provide control to the end tenants of a network owned and operated by multiple vendors within an adversarial setting. The system aims to re-establish this control through tenant side authentication and key generation performed through blockchain-based architecture.

II. Gap and Requirements

The emerging networking standards such as 5G and 6G are increasingly managed and controlled by software through SDN and NFV while operating in a multi-tenant and multi-vendor setting. As the demand for network increases, multiple services are expected to rely on SD-WAN. Developing a reliable and trustworthy network framework following a centralized orchestration framework is challenging due to reliance on fixed central points of operation.

Requirements: We list down the requirements for a reliable and trustworthy orchestration framework for SDNs:

- **R1: Decentralized Authentication and Access Control.** In a software-defined network, all the hardware resources responsible for providing the networking services are managed and configured by the software. It is crucial to validate identities of network elements within the vast network whilst simultaneously prevent single point failures in authentication chains.
- **R2: SLA Adherence.** Infrastructure vendors rent their resources such as switches and routers to tenants by creating a service-level agreement. It is important to monitor the services during run-time to ensure that all parties adhere to the SLAs. Furthermore, due to the size and nature of packets logged, it is imperative that there exists a mechanism to autonomously validate and audit records to detect violations.
- **R3: Transparent and Secure Logging.** When handling issues around SLA violation and network breaches, it is important to have a transparent and decentralized logging infrastructure to audit the network reliably.
- **R4: Micro-segmentation.** Due to the vast and complex nature of SD-WAN topologies, it is important to maintain fine grained control of individual network elements indeed to ensure no implicit trust is granted to any component of the network element that may be compromised.

III. Related Work

Traditionally, most SDN security efforts are targeted towards the "fortification of the network" to defend against attacks such as malicious/highjacked controllers, interception and corruption of packets (Man in the Middle), northbound Application Programming Interfaces (API) exploitation, south Distributed Denial of Services (DDoS) attacks, Packet sniffing, Address Resolution Protocol (ARP) spoofing and poisoning, traffic, and southbound DDoS [5], [6].

Multiple articles have addressed trust issues for SDN in multi-stakeholder settings. Vilalta *et al.* [9] presents a hierarchical architecture for orchestrating controllers, switches, and routers, but it relies on a centralized network orchestrator at the top level in the hierarchy, which could be breached. BSec-NFVO [7] is a blockchain-based SDN orchestration framework that enables auditability for the clients requesting services from tenants. The transactions between tenants and vendors pass through a consensus process and stored in an immutable ledger to ensure auditability. This solution provides transparent and secure logging, but it does not discuss how the authentication and access control is handled. DISCO [10] presents a distributed architecture for SDNs that can operate in a multi-domain environment. Although DISCO offers scalability and fault-tolerance, it does not describe any approaches to handle Byzantine Fault Tolerant (BFT) operation.

Qiu *et al.* [8] presents a blockchain-based approach to distribute SDN controllers, wherein the application logic executed by the controllers are verified through a consensus algorithm. It does not explicitly address SLA adherence and how the tenants can participate and log their transactions. DistBlockNet [11] introduces a distributed blockchain-based architecture for IoT, wherein a blockchain platform is used to create and modify flow tables for application traffic transparently and securely. The architecture of DistBlockNet is tailored towards flow-based security verification, which accepts or rejects flow based on the policies defined by the application, and it does not address how the nodes are authenticated and authorized. Biczok *et al.* [12] presents 5GEx, which is a service orchestration framework for multi-domain SDN, but it does not explain how 5GEx can handle adversarial scenarios.

Existing solutions either focus on logging transactions on the blockchain, consider resource orchestration in non-adversarial settings, or present a solution to a subset of requirements listed in Section II. Additionally, we also discuss how EDISON can mitigate various security attacks in Section V, which is also one of the significant advantages of our architecture. EDISON aims to develop an orchestration framework that can tolerate Byzantine behaviors through the use of blockchain technology, while guaranteeing zero-trust security model in multi-vendor and multi-tenant SD-WAN deployments.

A. Motivation

In light of the requirements listed in Section II, a new architecture based on a distributed Public Key Infrastructure (PKI) with Kerberos authentication technique is needed to mutually

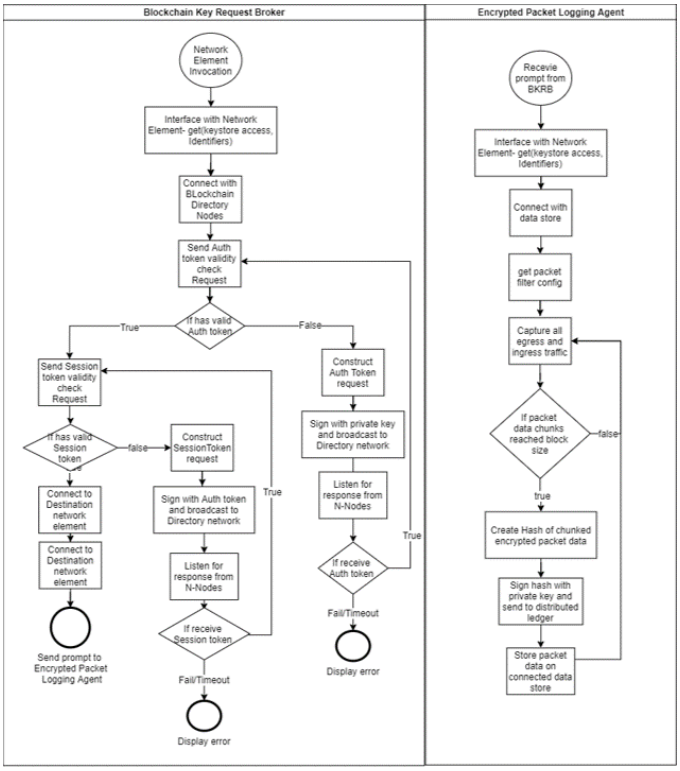

Fig. 1: Modules of the Embedded Blockchain Client Package: Left - Blockchain key request broker flow, Right - Encrypted packet logging agent flow.

authenticate and audit virtual network elements created by the vendors' hypervisor. It can be achieved by auditing the activity of the key distribution center (KDC). PKI-Kerberos hybrid designs apply the lightweight symmetric key encryption style of Advanced Encryption Standard AES, coupled with the ease and convenience of digital certificates provided by the public key infrastructure. A solution based purely on PKI and Kerberos is susceptible to attacks by a malicious vendor at the deployment level, which could cause other issues, such as SLA deviations, QoS issues, and potential for data theft through packet fingerprint sniffing. All of the prior mentioned challenges now have to be handled in a multi-stakeholder environment; wherein each stakeholder may have their agendas that do not entirely align with other parties or the overall well-being of the network.

IV. EDISON

A. System Description

In SD-WANs, overlays allow for multi-tenant network usage, which spurs new avenues for conflicting interests between competing services and applications. The network stakeholders can broadly be divided into: 1) The vendors who provide the network infrastructure service, 2) The tenants who use these services to host and provide application services, and 3) mutually interested third parties. More often than not, the network tenants have little to no visibility to the underlying orchestration of the network that the vendors own and manage. Such an operational model creates a blind trust-based system that could potentially grow to be an adversarial environment as control is not distributed equally and transparently to all network stakeholders. Thus, leading to a tenant vs. vendor trust and control imbalance that EDISON aims to solve.

EDISON presents a novel approach that uses Kerberos and PKI's strength to authenticate SDN network elements and secure northbound and southbound APIs of the vSD-WAN. At the same time, it holds vendors accountable to tenants through audit trails on cryptographically secure, immutable, and append-only ledger.

B. Building Blocks of EDISON

The EDISON framework relies on a blockchain platform with support for immutable ledger, smart contracts, and Byzantine fault-tolerance/sybil-resistant consensus algorithm. Our architecture is blockchain-agnostic and it exposes APIs for accessing services from the blockchain platform. The actual services/functionalities could be implemented on any platform.

In order to implement autonomous orchestration and operation from within the network, we would need to embed certain features and functionalities within the network elements themselves so that they are capable of acting and participating within the EDISON network. For this purpose, we propose a light-weight module called the Embedded Blockchain SDN Client Package (EBSCP) that comprises of the capability to perform secure key exchange along with creating encrypted audit logs. The certificate and signed binaries provided by the participating clients are installed by the vendors within the network elements at the time of deployment. Such a trusted module allows for the clients to interface with the network devices with the assurance of tamper resistant operation.

Another critical building block of the EDISON framework is the use of smart contracts and Decentralized Applications (DApp) to provide services such as: 1) node identification and authentication, 2) key/token generation and validation, 3) SLA verification based on the request of the Embedded Blockchain SDN Client Packages (EBSCP) in each of the network elements of the decentralized SD-WAN network. The decentralization of such services assures fault tolerance along with the distribution of responsibility and stake of the network. Each DApp implementation relies on a smart contract running on the blockchain. These smart contracts are executed on demand by the network elements via defined endpoints such as Representational State Transfer (REST) or Simple Object Access Protocol (SOAP) - based interfaces exposed by the registered DApps. Some of the contracts are responsible for data creation whilst others are intended for data retrieval and validation. All contracts require some form of Byzantine fault tolerance and consensus in order to ensure zero-trust based operation. The ecosystem of these DApps and their run time environments are described in Figure 3. The following lists their features and functionalities:

1) Embedded Blockchain SDN Client Package (EBSCP): The client packages and signs the binaries (EBSCP) to be installed by the vendors into the network elements at the time of deployment. The EBSCP allows for means of validation and authentication when connecting to the consortium of network users through the use of PKI and checksums. This client is considered to be honest, passive and act on behalf of the network users to orchestrate, monitor, and validate behavior of vendor's network elements. The key functionality of the EBSCP modules as illustrated in Figure 1 is explained in detail here: The BKRB is a blockchain client that acts as the module that performs all key exchange operations on behalf of the network element and is responsible for managing the shared key store that is used by the network elements. The BKRB acts as the interfacing broker that enables a traditional network element or virtual network element to communicate with different parties on a peer to peer network as a passive light-weight client that does not store a ledger/blockchain within itself, locally. The BKRB initially has access to the public/private key of each network element created by the vendor, along with the MAC address (or other id) of the element and the required network elements that are to be connected. This client also holds information with regard to connection information of the different blockchain networks/channels it would need to interface with. The Encrypted Packet Logging Agent (EPLA) captures all ingress and egress packets from a network element by scanning packets on all ports of the network device. It then uses the embedded certificate to sign the encrypted data blocks and publishes it to a ledger/data store. This enables the EPLA to create secure and tamper resistant audit logs of network events by scanning, collecting, and signing encrypted transactions and network packets and publishing them to a targeted storage zone for further analysis by tenants, vendors and other third parties.

2) EDISON Smart contracts and Decentralized Application Services: **Decentralized Application Services:** As part of the EDISON framework, 3 key services are needed to ensure complete functioning of the blockchain client with the various tenant and vendors nodes of the network. These DApps essentially function as interfaces to underlying smart contracts that abstract out implementation specific variances. This ensures that once the EBSCP are installed within the SDN elements, the packages will still be able to interact and perform core operation with the EDISON network even if the underlying business logic is revamped to adjust to changing demands. Such a modular design allows for loosely coupled, platform agnostic interfacing by leveraging the power of Decentralized applications and smart contracts.

Smart Contract: The smart contracts act as the primary means of interacting with the underlying blockchain. All reads and writes to the blockchain are performed through the smart contracts. This ensures that all operations are securely logged providing for a comprehensive audit trail and is generally considered to be a good practice when implementing a blockchain system.

The following sections describe the core DApps and their underlying smart contracts needed for the EBSCP to function as illustrated in Figure 3.

Authentication DApp: The BKRB client needs an authentication token (Auth token) to interact with all the DApps. The Auth token is provided by the Authentication server by looking up the ID and private key signature within the distributed Active Directory [13]. The DApp is responsible for generation of a unique Auth key and its registration in the Active Directory, for the client to use for all further secure communication with other DApps to maintain forward secrecy.

Session Management DApp: Once the BKRB client gains its initial Auth token, it then requests for a list of network elements with which it is expected to interact. The Session management DApp performs a lookup using the topology validation contract to determine sessions needed to be created for the requesting network elements. The DApp then generates a new key using the key generation contract, updates the Active Directory using the Distributed Hash Table (DHT) key contract and finally provides the BKRB with the ability to securely establish sessions with the corresponding pre-authenticated network element. Further Session management DApp also preforms simple look ups in order to validate session tokens on behalf of other DApp requests and tracks all illegal and invalidated sessions.

Key generation Contract: This contract is executed on any one of the tenant nodes to invoke a Pseudo-Random Number Generator (PRNG) in order to create a desired bit length symmetric key for the requesting network element. Keys for the same request may be generated by one or more nodes at the same time but the first to publish to the key generation channel and gain endorsement of the required tenant and vendor nodes will be considered valid for the session. This endorsed key is then passed into the DHT key store contract for mapping and assignment to the network element.

DHT key store Contract: This contract updates and maintains a global distributed hash table that contains entries to every session of every network element within the EDISON framework. This table essentially functions as a distributed Active Directory. Every read and write operation to the table requires a transaction to be endorsed by all stakeholders in the authentication cycle including vendors and tenants.

Authentication and Session validation Contract: This contract accesses the Active Directory to verify and validate authentication or session tokens given to it. Furthermore, the contract registers every requesting validation to a channel in order to facilitate better auditing.

Topology validation Contract: This contract accesses the SLA to topology mapping store which maintains a dictionary of lists detailing what each network element is connected to within the SD-WAN. This contract is directly accessed by the topology configuration DApp to service request of elements for their topology configuration. This contract is also responsible for adding, updating and logging any network element mapping when requested by a vendor or tenant node after getting consensus from all stakeholders on the same.

SLA validation Contract: This contract maintains and

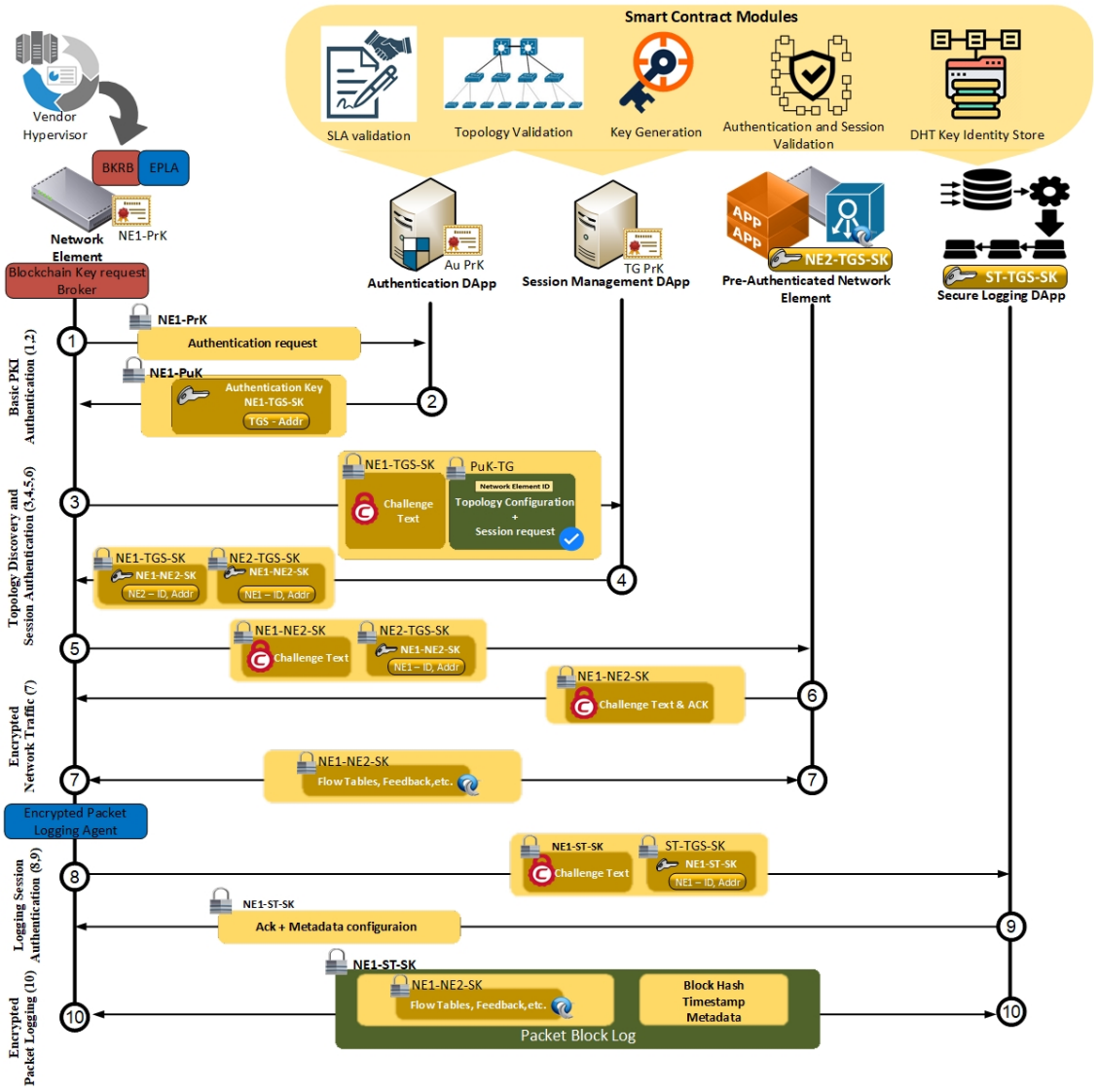

Fig. 2: The Key exchange sequence of the new network element using BKRB to the Authentication server, requesting session to desired network elements. Encrypted packet logging blocks sent between EPLA and secure logging DApp. The flow is described in detail in Section IV-C.

accesses a store of SLA key performance indicators (KPI) such as throughput and network jitter rate that have been derived using various SLA to KPI mapping techniques based on the tenant vendor negotiations. The contract is invoked to validate logged network packets captured by the EPLA during analysis and audit phase [14].

Secure logging DApp : This DApp is the primary interface that the EPLA uses to publish its captured encrypted packets to a permanent data store for analysis and auditing. This DApp can be implemented in a variety of ways based on policies and performance constraints.

DHT packet store Contract: This contract logs and stores signed packet blocks constructed by the EPLA. This contract is accessed directly by the Secure logging DApp to service write requests of each network elements EPLA after establishing its session validity. The underlying data store may vary from an actual blockchain, to a peer to peer data store like a distributed hash table, to a simple distributed data store like Amazon S3 or even a central private data store.

C. Operation Flow

This section describes the different steps and messages exchanged within the EDISON system, as described in Figure 2.

1) BKRB Authentication Request sent from Network element 1 (NE1) to Authentication DApp encrypted using Network element's private key-NE1Prk
2) Authentication DApp response Authentication token(NE1-TGS-SK) encrypted with NE1's public

key- NE1PuK

3) NE1 session request to session management DApp using challenge text encrypted in NE1-TGS-SK and identity validation messaged encrypted using PuK-TG
4) Session Management DApp response to NE1 with session tokens NE1-NE2-SK encrypted with key NE1-TGS-SK and NE2-TGS-SK for NE1 and Pre-Authenticated Network element-2 (NE2) respectively.
5) NE1 sends session request message containing challenge text signed with NE1-NE2-SK along with forwarding session token of NE2 encrypted by the Session management DApp with NE2-TGS-SK.
6) NE2 retrieves session token: NE1-NE2-SK, after decryption using NE2-TGS-SK which validates challenge text sent by NE1 using said session token and sends acknowledgment with NE1-NE2-SK.
7) NE1 and NE2 now preform secure communication using NE1-NE2-SK for the remainder of the session.
8) EPLA Auth and session request to Secure Logging DApp(SLD) using stored key NE1-ST-SK for challenge text and ST-TGS-SK for validation token holding client information - gained through Authentication and Session DApp (not shown in figure).
9) SLD decrypts token to get NE1-ST-SK token and validates credentials of requesting client. Sends acknowledgement and metadata configuration encrypted using NE1-ST-SK to the client.
10) Encrypted logging messages between NE1 and NE2 are sent from EPLA to SLD with relevant block hash, timestamp and other metadata packaged and encrypted with NE1-ST-SK.

D. Operation Phases

The overall operation of the EDISON framework can be broken down into two phases, as indicated in Figure 2: Phase 1 - the Topology validation and authentication phase and Phase 2 - the Encrypted packet logging and auditing phase. We describe the two phases assuming a network of multiple vendors and tenants hosting the DApps and smart contracts, and the BKRB and EPLA blockchain packages are installed on the vendors' network elements in the initial set up phase.

1) Phase 1: Blockchain Key Request Broker Authentication: The initial bootstrapping process involves configuring the SLA and KPI data structures by the vendor and tenant nodes. The SLA KPI mapping and network topology dictionary list are constructed after having gained the consensus of all relevant stakeholders using the topology and SLA validation contracts and the storage of initial public keys to identify all pre-authenticated network elements.

After the bootstrapping phase, the BKRB is linked with the tenant's network elements to begin the validation process illustrated in Figure 2. Each network element's BKRB broadcasts an authentication request to all tenants (nodes) hosting the Authentication DApp (functioning as the active authentication directory). Tenant nodes are responsible for validating the authentication request by utilizing the authentication and session validation contract. If successful, the tenant node broadcasts the successful authentication and generation of a secret Ticket Granting Ticket (TGT) session token (used to communicate with the session management DApp), using the Key generation contract, on a private channel to all the network user nodes. Subsequently, all nodes update their directories with the network element's status as authenticated and register the key using the DHT key store contract. This information is shared only amongst the tenant nodes in a private channel, and it is not shared with the vendors for security and to maintain secure audits against the vendors. After the tenant nodes have updated the newly authenticated network element's status in their distributed ledger, they reply to the BKRB with the agreed Auth TGT key. This process is seen from stages 1 to 2 of the basic PKI Auth in Figure 2.

The BKRB then creates a request for the required network elements that it needs to establish sessions with and broadcasts the request to all the nodes that host the Session management DApp. The requests are encrypted/signed with the active authentication (TGT) token, are illustrated in stage 3 of the topology discovery and session authentication phase. The nodes will validate the request via the Authentication and Session Validation contract. If the request is successfully validated, the corresponding list of network elements associated with the requesting element is retrieved using the topology validation contract. A set of session tokens are generated for each of these element pairs using the key generation contract and the DHT key Identity contract. Once consensus is reached, the nodes will reply to the requesting BKRB with the session's key package as seen in stage 4 of the session authentication phase. These session keys would allow the BKRB to connect with the destination network elements' BKRB and perform the session key exchange. If the destination BKRB approves the session key exchange, the destination network elements shared key store is updated with the appropriate session key. This is seen from stages 5 to 6 of the Session authentication phase. Thus, a secure connection can begin between the two network elements directly without a broker's need any longer.

2) Phase 2: Encrypted Packet logging and Auditing: During the SDN operation, all interactions between the network elements will occur using the session keys generated through the authentication channels between the BKRB and the tenant nodes in Phase 1. Another tenant referred to as the Encrypted Packet Logging Agent (EPLA) captures all ingress and egress packets from a network element that it is embedded to and posts this data as blocks to a public ledger. The EPLA would broadcast the signed block of captured encrypted packets with the session tokens to the nodes hosting the Secure Logging DApp. The key and signature are validated by nodes of the vendor and the tenants as seen in stage 8 to 9 of Figure 2. Once a session is established between the DApp and the EPLA of the network element, continuous blocks of signed, encrypted packets are sent between the two (as seen in stage 10) till the session expires. Thus creating an immutable time-bound sequence of encrypted packets that can be audited to verify any network elements behavior within the network and

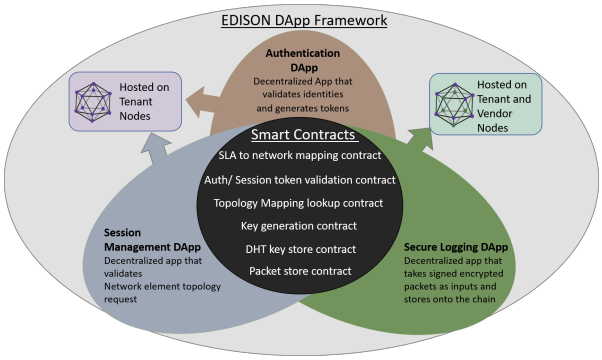

Fig. 3: The various Smart contracts used by the 3 decentralized applications within the system. A depiction of which DApp services are owned and run on which set of network nodes.

can be used to cross-validate said behavior against the SLAs agreed upon by different parties of the network. These packets are passed back into a virtual network device to re-stimulate behavior logged during the analysis phase. This behavior is then measured and compared with the SLAs via the SLA validation contract to verify appropriate behavior. Hence, this allows for an automated validation mechanism to check for the proper operation and orchestration of each network element after decorating the packets with the associated session keys and hence enabling for zero-trust-based automated auditing and operation of the entire SDN network in a multi-vendor multi-client environment.

V. Security and Trust Analysis

This section discusses the security features and abilities of the EDISON framework against common SD-WAN related issues in a multi-vendor multi-tenant environment.

Central Certifying Authority (CA) corruption: Traditional PKIs require Centralized, trusted Certifying authorities to issue and validate certificates [15]. When the CA is subverted, the entire PKI collapses. In the EDISON framework, we use PKIs for the initial phase of authentication and switch to session keys issued by the Authentication and Session management DApps, thus preventing any single point of failure.

Central Authentication and Active Directory Vulnerabilities: Most Single Sign-On systems rely on a central Active Directory to be maintained by an Authentication server, which authenticates all users of the system [16]. This Centralized Directory is vulnerable to DDoS and other common single point failure exploits. The EDISON framework employs a DHT hosted across multiple tenant nodes to store and operate as a distributed Active Directory with high availability and resilience. Thus satisfying the requirement of Decentralized authentication and access control (R1 of Section II).

SLA violation and manipulation: Traditional SLA violations are uncovered through long audit trails that take several man-hours to perform and are rarely real-time [17]. In the case of EDISON, due to the EPLA tracking every packet of all network elements, it allows for simple automated re-runs of packets within test networks for asserting network element behavior. Furthermore, due to hash values and time stamps, the logs are both immutable and verifiable. Lastly, the SLA validation services made available through DApps on the network allow any node that owns the correct set of keys to decrypt, re-run, and cross-validate packets automatically. Hence, creating a real-time feedback loop for tamper detection. Thus EDISON allows for the fulfillment of SLA adherence (R2 of Section II).

Log and Audit trail tampering: As the Vendor holds complete control of the Network element in traditional deployments, they can alter and manipulate the resulting logs generated with no chance for cross-validation by tenants [18]. The EDISON EPLA module captures all ingress and egress packets of the network element. It publishes these secure signed blocks of packets with Hash values, timestamps, and other relevant metadata to a Datastore DApp. In turn, this DApp stores and accumulates all network element management layer packets within a ledger for further automated analysis. This allows EDISON to ensure transparent and secure logging (R3 of Section II).

Log Analysis attacks: Logs hold complete information regarding the configuration and operation of the network. This often exposes sensitive operational information to attackers to exploit [19]. The EDISON framework's EPLA captures and tracks only encrypted packets which can only be decrepitude by the tenants who own the secret key.

Forward Secrecy: Key agreement and exchange protocols need to give assurances that session keys and data encrypted by it will remain secure even if the private key of the network element is compromised at some stage in the future. In EDISON each of the authentication and session tokens is generated and distributed over authentication cycles and prior session keys are never recycled. Hence, preventing future key compromises for affected previous session-related data.

Man in the middle attack: In this type of attack, the attacker can eavesdrop all the transactions between nodes. It is caused by some instances, such as ARP spoofing, DNS spoofing, or vulnerable access points [20]. The EDISON framework provides the ability to micro-segment and manage each individual network element thus monitoring and logging suspicious activities and rejecting sessions accordingly (R4 of Section II).

VI. Implementation and Evaluation

Our evaluation aims to understand the timing overhead introduced by EDISON when creating a secure network connection between two network elements, which broadly spans from steps 1 to 7 of Figure 2. The benchmark for testing is a simple SDN controller to switch connection and flow table installation without encryption. We create a test case which is subdivided into three categories:

1) Setting up of a full authentication using existing secure socket layer (SSL)–PKI authentication with a centralized

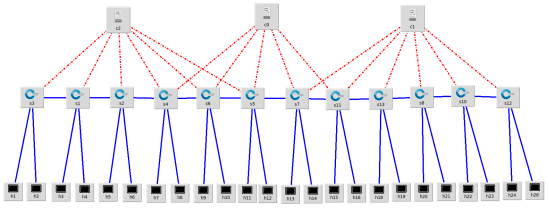

Fig. 4: Mininet topology for a 3 vendor - multi remote controller environment running across 12 switches with 24 hosts.

CA. Here we are performing only the first two steps, as mentioned in Figure 2.

2) Full authentication cycle to authenticate network elements and establish a secure session between two authenticated network elements, which covers all the steps given in Figure 2.
3) Renewing expired session on existing authenticated network elements to create a new secure session between the two network elements, which corresponds to steps starting from the fifth step in Figure 2.

Setup: To simulate the SDN, we used the Mininet network (see Figure 4) emulator [21] inside a Ubuntu 18.04 virtual machine with 8GB of RAM and a four-core processor. The switches and controllers in Mininet support OpenFlow protocol, which is a standard communication protocol for SDNs. OpenDaylight controller is used for monitoring and to induce flows in the network. The system has interconnected controllers and switches which best depict the networks in 5G/6G. Then we are using Open vSwitch to create a set of self-signed certificates for controllers and switches. Each switch was configured separately to use SSL over TCP. The network flows were induced manually using Open vSwitch to test the key exchange.

For the second and third parts of the test case, we automated the key exchange processes using a Python script to initialize the network and send messages. With the help of the PyCrypto library in Python, we implemented AES and RSA algorithms to pass messages. We are using RSA-2048 bit encryption for the session key transfer. Once the MAC IDs are validated, a session key is issued, then we use AES-128 encryption to pass the token keys, as seen in Phase 1 of Section c.

Tendermint BFT as a Blockchain Platform: Tendermint BFT is an open-source blockchain platform with support for Byzantine fault tolerance consensus. To understand the overhead added by the consensus process, we have set up a Raspberry Pi testbed with five Tendermint BFT nodes. In particular, our evaluation focused on understanding how much do the nodes to come to a consensus. The packet transfer and network delays are recorded using Wireshark packet capture.

The following are the key findings from our evaluation:

- Our benchmark evaluation cases consider Open flow without any encryption seen in Figure 5 (Top-Left) showing the minimum latency. Each data frame and acknowledgement packet's average packet size is around *150 Bytes and 70 Bytes*, respectively.
- For the first case, we are checking the delays induced by using SSL over TCP. With the help of Open vSwitch, we are setting up a controller that issues keys for secure communication. We are generating self-signed certificates that have SHA-2 with RSA encryption, as shown in Figure 5 (Top-Right). Public and private keys are issued to both switch and the controller. After configuring the switches, we begin communicating over the network. The handshaking shows more significant spikes in time and more propagation delay per message. The handshaking has a delay of 500 milliseconds where the data frames are of length 250 Bytes on average.
- Figure 5 (Bottom-Right) shows the overhead added by the consensus process, wherein five Tendermint nodes are validating a transaction and storing it on the blockchain. Here, the overhead represents the cost of authentication cycles.
- Next, we observe the EDISON framework's packet flow, as seen in Figure 5(Bottom-Left). We implement all the parts mentioned earlier for phase 1 with the addition of session management and topology lookup. At a time close to 0 seconds, the first spike depicts the initial Auth request, where RSA 2048 encryption is used. A delay of approximately 1 second is caused by the BKRB to validate the keys. The second spike depicts the session key phase, where AES-128 encryption is used, representing the third and fourth steps in Figure 5. Once the sender and the receiver have the session key, they must be validated and update their directories. This includes the new topology validation and flow table update. This whole process takes approximately 1 second. After this, a third spike shows that the connection is established and that the network elements can now begin servicing any application. This last phase depicts the steps 5 and 6 in the Figure 2. In step 7, all the flow tables are updated, bringing the authentication phase to an end. Due to the various types of encryption used, the packet size becomes significantly large. Thus the typical handshake messages are around 1500 Bytes while echo frames/ ack frames are about 140 Bytes.
- In the third part of the test case, as the two hosts are already authenticated (steps 1 and 2, as shown in Figure 2 have been completed), then the communication occurs after passing new keys. This case illustrates the packet transfer when the sender and receiver have already set up the connection line. The latency shows the new token keys generation and validation. Since only AES-128 is used for token key passing, the overall packet size and delays are less than the previous setup.
- We repeat the above tests for messages passing between all the hosts in the network and for various lengths of messages. As shown in Figure 6, we see a distribution of overheads caused by each of the phases in the process.

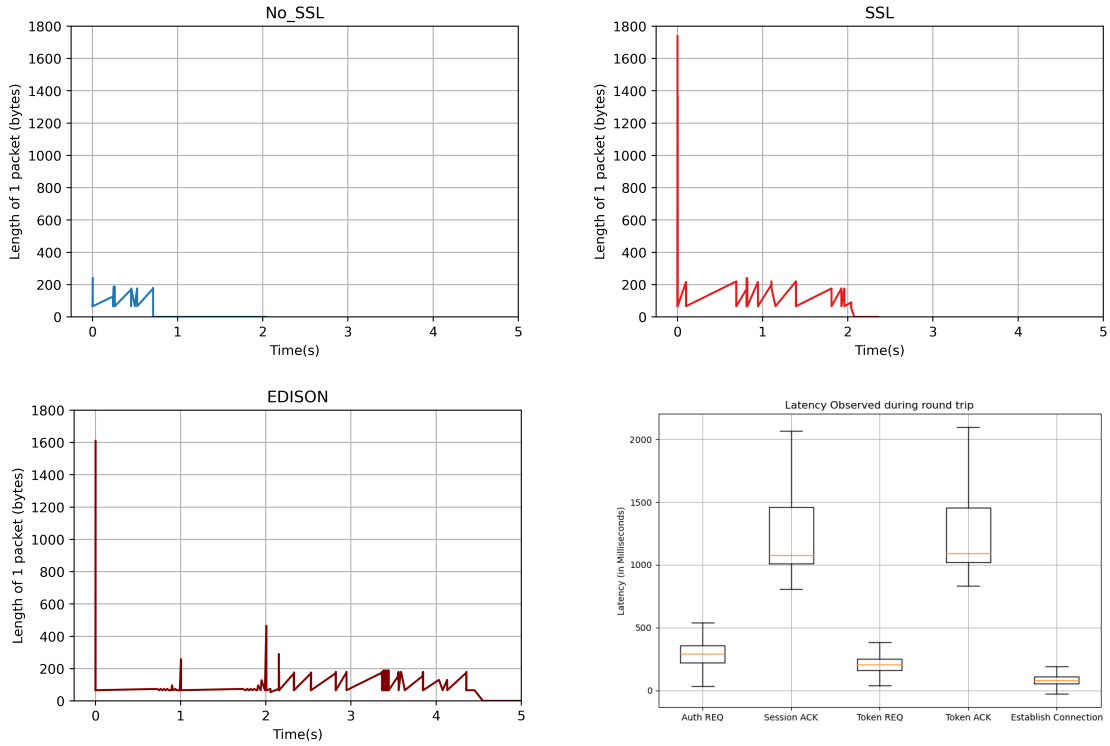

Fig. 5: Top: Packet size and arrival rate comparison created by the test rig to install a secure flow table: (left) The base line test case of flow installation without any encryption (right) Simple SSL-based encryption with self signed keys. Bottom: (Left)Packet size and arrival rate for EDISON authentication cycle and flow installation .(Right) Mean and average time taken for each process in the authentication cycle of the blockchain based authentication and key exchange.

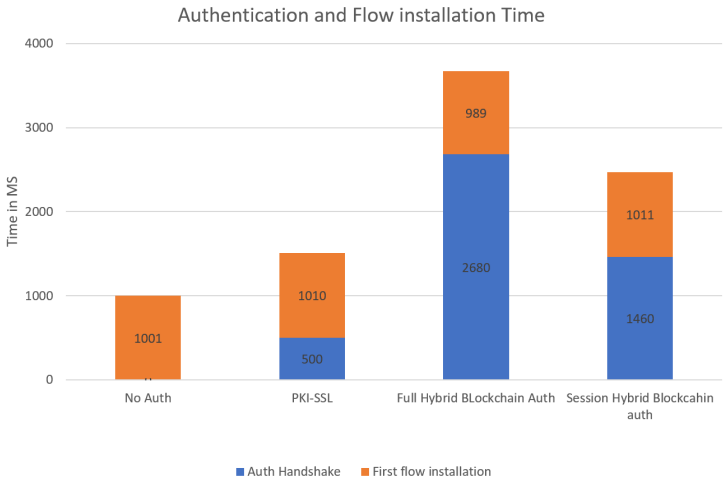

Fig. 6: Comparative evaluation of overhead caused by authentication cycles and differentiating between full and session refresh authentication cycle timings.

- In summary, the results show the delay introduced by blockchain and transaction validation is not significant for BFT blockchain, and that the messages are end to end encrypted, validated, and logged as seen in Figure 6. Further, EDISON's flexible authentication scheme allows for full and session authentication cycles to be carried out based on tenant set policies.

A. *Blockchain Throughput Analysis*

As shown in Figure 5 (Bottom-Right), the Tendermint BFT platform adds one second on average to achieve among five nodes. Multiple transactions are injected onto the blockchain network throughout EDISON's authentication and logging operations. These transactions may take on average 1 second, according to Figure 5 (Bottom-Right). When the number of consensus nodes increases, the consensus delay may also increase, as reported in [22].

Note that EDISON's architecture is blockchain-agnostic; therefore, the application developers could choose any capable platform, including Hyperledger Fabric and Ethereum. Thakkar [23] reports that the Hyperledger Fabric can handle tens of transactions per second if the block size is increased to 50 transactions per block. Bez *et al.* [24] shows that Ethereum supports 15 transactions/second, which may hamper the real-time performance of EDISON. Zheng et al. [25] evaluate the performance of well-known blockchain platforms that support smart contracts.

As reported in [25], the smart contracts with loop operations, which leads to multiple back-and-forth interactions between clients and the contract, tend to have limited throughput (2 transactions per second). The permissioned blockchain platforms with faster consensus protocol tend to outperform PoW-based public blockchain platforms such as Ethereum [25]. Therefore, we let the application developers choose the desired blockchain platform for their deployments based on their requirements.

VII. Conclusion

The SDN technology is set to change the future of connectivity, however, like most bleeding edge systems, it is still plagued by several security vulnerabilities. In this paper, we have proposed a Secure and Auditable Decentralized SDN Orchestration (EDISON) framework to provide greater authentication and auditing capabilities to tenants over the vendor's network elements. We have shown how EDISON achieves decentralized orchestration by using a peer-to-peer network comprising of tenant and vendors' nodes that host decentralized applications leveraging smart contracts. In addition, our implementation gives network elements the ability to capture encrypted packets and publish them as blocks to a secure logging DApp. We have also performed a comparative analysis on simulation environments to evaluate EDISON's performance against no-authentication and PKI-SSL and found that EDISON's flexible authentication scheme allows it to be both secure and performant, based on the needs of the network. The security analysis section also provided credence to the use case and showed how EDISON mitigates several security vulnerabilities of the SD-WAN system in adversarial environments.

Acknowledgment

This work was supported by the USC Viterbi Center for Cyber-Physical Systems and the Internet of Things (CCI).

References

[1] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5g: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE journal on selected areas in communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[2] Z. Yang, Y. Cui, B. Li, Y. Liu, and Y. Xu, "Software-defined wide area network (sd-wan): Architecture, advances and opportunities," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019, pp. 1–9.

[3] T. Hewa, G. Gür, A. Kalla, M. Ylianttila, A. Bracken, and M. Liyanage, "The role of blockchain in 6g: Challenges, opportunities and research directions," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*, 2020, pp. 1–5.

[4] E. Gilman and D. Barth, *Zero Trust Networks*. O'Reilly Media, Incorporated, 2017.

[5] "Security challenges with network functions virtualization," *Future Generation Computer Systems*, vol. 67, pp. 315 – 324, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X16302321

[6] W. Jia-si, W. Jian, L. Jian-an, and Z. Yue, "Secure software-defined networking based on blockchain," 2019.

[7] G. A. F. Rebello, I. D. Alvarenga, I. J. Sanz, and O. C. M. B. Duarte, "Bsec-nfvo: A blockchain-based security for network function virtualization orchestration," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[8] C. Qiu, F. R. Yu, F. Xu, H. Yao, and C. Zhao, "Permissioned blockchain-based distributed software-defined industrial internet of things," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–7.

[9] R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, and R. Martínez, "Hierarchical sdn orchestration for multi-technology multi-domain networks with hierarchical abno," in *2015 European Conference on Optical Communication (ECOC)*, 2015, pp. 1–3.

[10] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed sdn controllers in a multi-domain environment," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–2.

[11] P. K. Sharma, S. Singh, Y. Jeong, and J. H. Park, "Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 78–85, 2017.

[12] G. Biczok, M. Dramitinos, L. Toka, P. E. Heegaard, and H. Lonsethagen, "Manufactured by software: Sdn-enabled multi-operator composite services with the 5g exchange," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 80–86, 2017.

[13] R. Wang, J. He, C. Liu, Q. Li, W. Tsai, and E. Deng, "A privacy-aware pki system based on permissioned blockchains," in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 2018, pp. 928–931.

[14] E. Kapassa, M. Touloupou, A. Mavrogiorgou, and D. Kyriazis, "5g slas: Automated proposition and management of agreements towards qos enforcement," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2018, pp. 1–5.

[15] J. A. Berkowsky and T. Hayajneh, "Security issues with certificate authorities," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, 2017, pp. 449–455.

[16] G. Wang, J. Yu, and Q. Xie, "Security analysis of a single sign-on mechanism for distributed computer networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 294–302, 2013.

[17] I. Brandic, V. C. Emeakaroha, M. Maurer, S. Dustdar, S. Acs, A. Kertesz, and G. Kecskemeti, "Laysi: A layered approach for sla-violation propagation in self-manageable cloud infrastructures," in *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*, 2010, pp. 365–370.

[18] D. Sandler, K. Derr, S. Crosby, and D. S. Wallach, "Finding the evidence in tamper-evident logs," in *2008 Third International Workshop on Systematic Approaches to Digital Forensic Engineering*, 2008, pp. 69–75.

[19] K. Kowalski and M. Beheshti, "Analysis of log files intersections for security enhancement," in *Third International Conference on Information Technology: New Generations (ITNG'06)*, 2006, pp. 452–457.

[20] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[21] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 2014, pp. 1–6.

[22] G. S. Ramachandran, K. Wright, L. Zheng, P. Navaney, M. Naveed, B. Krishnamachari, and J. Dhaliwal, "Trinity: A byzantine fault-tolerant distributed publish-subscribe system with immutable blockchain-based persistence," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 227–235.

[23] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2018, pp. 264–276.

[24] M. Bez, G. Fornari, and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, 2019, pp. 167–176.

[25] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2018, pp. 134–143.