Designing Fine-grained Access Control for Software Defined Networks using Private Blockchain

Durbadal Chattaraj, *Member, IEEE*, Basudeb Bera, Ashok Kumar Das, *Senior Member, IEEE*, Joel J. P. C. Rodrigues, *Fellow, IEEE*, and YoungHo Park, *Member, IEEE*

***Abstract*—Emerging next-generation Internet yields proper administration of a wide-ranging dynamic network to assist rapid ubiquitous resource accessibility, whilst providing higher channel bandwidth. Since its inception, the traditional static network infrastructure-based solutions involve manual configuration and proprietary controls of networked devices. It then leads to improper utilization of the overall resources, and hence experiences various security threats. Although transport layer security (TLS)-based solution is presently advocated in the said framework, it is vulnerable to many security threats like man-in-the-middle, replay, spoofing, privileged-insider, impersonation, and denial-of-service attacks. Moreover, the current settings of the said tool do not facilitate any secure and reliable mechanisms for data forwarding, application flow routing, new configuration deployment, and network event management. Also, it suffers from the single point of controller failure issue. In this paper, we propose a new private blockchain-enabled fine-grained access control mechanism for the SDN environment. In this regard, attribute-based encryption (ABE) and certificate-based access control protocol are incorporated. This proposed solution can resist several well-known security threats, and alleviate different system-level inconveniences. The formal and informal security inspections and performance-wise comparative study of the proposed scheme endorse better qualifying scores as compared to the other existing competing state-of-art schemes. Besides, the experimental testbed implementation and blockchain simulation show the implementation feasibility of the proposed mechanism.**

***Index Terms*—Access control, blockchain, consensus, next generation network, software-defined networking, security.**

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education under Grant 2020R1I1A3058605. This work was partially supported by by FCT/MCTES through national funds and when applicable co-funded EU funds under the Project UIDB/50008/2020; and by Brazilian National Council for Scientific and Technological Development - CNPq, via Grant No. 313036/2020-9. This work was also supported by the Ripple Centre of Excellence Scheme, CoE in Blockchain (Sanction No. IIIT/R&D Office/Internal Projects/001/2019), IIIT Hyderabad, India, and also by the Mathematical Research Impact Centric Support (MATRICS) project funded by the Science and Engineering Research Board (SERB), India (Reference No. MTR/2019/000699).

D. Chattaraj is with the JIS College of Engineering, Barrackpore - Kalyani Expy, Phase III, Block A, Kalyani 741 235, Nadia, West Bengal, India, and also with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India (email: durbadal.chattaraj@jiscollege.ac.in).

B. Bera and A. K. Das are with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India (e-mail: basudeb.bera@research.iiit.ac.in; iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in).

J. J. P. C. Rodrigues is with the Federal University of Piauí (UFPI), 64049-550 Teresina-Pi, Brazil and Instituto de Telecomunicações, 6201-001 Covilhã, Portugal (e-mail: joeljr@ieee.org).

Y. Park is with the School of Electronics Engineering, Kyungpook National University, Daegu 41566, South Korea (e-mail: parkyh@knu.ac.kr).

(Corresponding authors: Ashok Kumar Das and YoungHo Park).

A preliminary version of this paper has appeared in the IEEE International Conference on Computer Communications (INFOCOM) Workshops: BlockSecSDN: Blockchain for Secure Software Defined Networking in Smart Communities, Toronto, ON, Canada, 2020, pp. 237-242, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162669.

I. Introduction

In recent years, an extensive advancement of "Information and Communication Technology (ICT)" in various domain like "cloud computing", "Internet of Things (IoT)", "Big data" and "Next Generation Internet (NGI)" urges a proper administration of a large-scale dynamic network [1]. To address such issues, Software-Defined Networking (SDN) performs a great responsibility [2]. Since its inception, SDN dissociates the conventional data plane from the control/forwarding layer of a proprietary network asset. It provides higher security, scalability, dynamism, efficiency, and reconfiguration over "static network infrastructure" [3]. In contrast, OpenFlow is a widely advocated *de facto* open-source SDN framework that fortifies better security and programmability towards the development of new innovative next-generation network applications.

Presently, the profound adaptations of emerging SDN framework over the "static network infrastructure" exhibits different security concerns [2], [4], [5]. Since its inception, most of the security threats arose into two different layers in SDN, that is, infrastructure layer (say, data plane) and control layer (called a control plane) [4]. Without adopting any robust "authentication and access control" mechanisms between these two layers, it is very difficult to protect the underlying network assets in SDN [2], [3]. For example, suppose an external adversary injects a malicious configuration into a switch/router from a controller node, then the deployed network infrastructure would probably malfunction, and hence, it leads to "network breakdown". Such kind of setting may reduce the overall performance of SDN. More precisely, it will affect the dependability (mostly, availability and reliability) of the entire network [4]. Also, proper "accountability and detectability" of various implementation flows using single (or multiple) controllers (s) is/are not addressed in the existing state-of-arts. In contrast, detection of various attack models through proper network troubleshooting mechanisms is still missing in the existing literature [4].

Intuitively, with the current built-in configuration of network control layer in the SDN framework, it is vulnerable to "single point of compromisation (SPC)" and "single point of failure (SPF)" [6]. More precisely, failure of a single point of administration and control unit (i.e., controller node) makes

the associated infrastructure layer or data forwarding plane jeopardized. This kind of fabrication eventually makes the whole network vulnerable to the "denial-of-service (DoS)" attack. A single-point failure of the SDN controller can devise the entire network to collapse and also such a situation makes the network vulnerable to DoS attacks.

Due to unavailability of a robust access control mechanism between the application and controller nodes, an adversary can easily mount different attacks, like "man-in-the-middle (MITM)", "replay", "impersonation", "spoofing", "denial-of-service (DoS)" and "privileged-insider" attacks [4]. Furthermore, the integration of the "configuration-complex" transport layer security (TLS) protocol for governing access control among the network switches and SDN controllers is a very cumbersome and not at all a user-friendly process. Further, such a mechanism is prone to different security threats (e.g., MITM, DoS, distributed denial-of-service (DDoS), and eavesdropping attacks [4]). Moreover, with the TLS-based access control, it is also feasible for an adversary to physically capture the network switch's identity, and then launch different attacks such as spoofing, DoS, and DDoS [3].

A. Motivation and Objectives

To alleviate the aforesaid security threats in each plane of the SDN framework, various state-of-the-art solutions have been proposed in the existing literature [4], [7], [8]. But, such approaches did not properly address the access control issues whilst considering individual entities inside the SDN framework. To deal with the situation, different blockchain-based secure authentication and access control mechanisms have appeared in the recent literature [4], [9], [10]. But, these fine-grained solutions are either vulnerable to various known attacks and not considered to be a provably secure scheme or lag to provide a concrete foundation for plane-wise access control policy. Hence, this work focus on the following objectives:

- To propose a robust and secure access control scheme to alleviate several well-known security threats in the SDN framework namely, spoofing, impersonation, denial of service, man-in-the-middle, replay, and privileged-insider attacks.
- To provide a robust approach using the distributed ledger technology (DLT) for addressing the single point of SDN controller failure issue.
- To advocate real-time monitoring and debugging of network events in the underlying SDN framework, proper accounting, and auditing mechanism is necessary.
- To ensure provable security of a newly propose access control scheme, a formal as well as informal treatment, and simulation-based protocol verification are needed to be carried out.
- Finally, to prove the real-time efficacy of a newly introduced access control scheme, a testbed implementation is needed to be demonstrated.

The proposed mechanism vis-a-vis the aforementioned objectives is demonstrated as follows. In this work, two different access control mechanisms are introduced. Initially, an "attribute-based encryption" (ABE) enabled access control protocol is suggested to build a secure channel between the business application and controller layers. Later, a key agreement-based access control policy is provided between the controller and infrastructure layer. These two proposed access control strategies are capable to resist various security attacks between application and control layers, and between control and infrastructure planes. For monitoring different network activities among various components of software-denied networking framework, a P2P Cloud Server Network (called, P2P CS Networks)-enabled distributed accounting and auditing principle is adopted, where all the transactions (specifically, interaction messages) between controller and applications (or controller and switches) are aggregated into a controller node; forming a partial block and sends the same block to the cloud server in a P2P Cloud Server (CS) Networks. Then, through a consensus algorithm executed by all the peer-to-peer nodes inside the P2P CS Networks finally mined the final block into the blockchain for future purposes. After mining the block into the blockchain, an SDN admin can easily audit (or verify) the transaction details, but a privileged insider of the cloud server cannot modify the block by any means. Finally, to accomplish the fourth and fifth objectives, the Real-Or-Random (ROR) model-based formal verification and informal security inspections are carried out. The "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" based testbed experiments and real-time blockchain simulation are also executed to demonstrate the performance efficacy.

B. Research Contributions

The contributions made in this paper are listed below.

1) With the proposed access control mechanism, each principal would be able to verify the legitimacy of the other entities before initiating any action like data forwarding, application flow routing, installing a new security configuration, and network event management. In addition, with the proposed blockchain-based access control scheme, an administrator can easily verify the transactions from the cloud-assisted blockchain center (CABC) to make real-time decisions for future actions like traffic re-routing, flow management, and reconfiguration of security patches.
2) The formal security analysis using the widely used "Real-Or-Random (ROR) oracle model" [11], non-mathematical inspection, and formal security assessment using the widely-recognized "Automated Validation of Internet Security Protocols and Applications (AVISPA) tool" [12] shows that the protocol is provably secure against various potential attacks that are incurred in SDN framework. We show that the proposed access control scheme alleviates different security threats namely, denial-of-service (DoS), man-in-the-middle (MITM), spoofing, impersonation, privileged-insider, and replay attacks, apart from the session key security.
3) The testbed experiments using "MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library" [13] for different cryptographic

primitives have been conducted to measure computational time under both a server and a Raspberry PI 3 settings. Furthermore, a detailed comparative analysis shows that the proposed scheme exhibits superior security provides more functionality features and needs low communication and computation costs as compared with those for other existing co-related schemes.

4) The blockchain-based implementation on the proposed protocol has been also performed to measure the computational time required for a varied number of transactions per block and also for a varied number of blocks mined in the blockchain.

C. Paper orientation

The rest of the paper is oriented in the following manner. Section II discusses different state-of-art access control mechanisms particularly reported in software-defined networking and its real-time usage in the domain of Internet of Things (specifically, Internet of Drones (IoD), and Internet of Vehicles (IoV)). Section III presents the proposed system architecture; typically the system model as well as adversary models. This section also highlights various issues of those schemes and their concomitant challenges. Section IV demonstrates the suggested private blockchain-based access control mechanism. Section V put forward various security inspections such as formal, informal, and simulation-based validations to substantiate the efficacy of the proposed scheme. Section VI highlights a real-time testbed implementation and experimental results of the proposed scheme. Section VII presents the performance of the proposed protocol through a rigorous comparative analysis with the existing state-of-art schemes. Section VIII presents the simulation-based blockchain implementation and its computation complexity while running the proposed scheme. Finally, Section IX concludes the paper.

II. Related Work

Chattaraj *et al.* [3] proposed a blockchain-based access control mechanism for the classical SDN framework. In this scheme, the authors have put forward an attribute-based encryption/decryption technique for securing the communication between the SDN application and controller plane. Meanwhile, to achieve secure access control between the SDN controller and infrastructure layer a "polynomial-based key distribution mechanism" has been utilized. Without loss of generality, this scheme shows its effectiveness in terms of communication, and security, and functional features. However, the performance efficacy in terms of rigorous formal security analysis, real-time testbed implementation, and blockchain simulation has not been directed properly.

Iqbal *et al.* [14] designed an authentication protocol for SDN enabled smart homes application. In their scheme, smart devices authenticate each other by verifying the freshness of the communicated messages. During the entity's mutual authentication, the devices use a pre-loaded constant session for every session, for that case their scheme is vulnerable to the "Ephemeral Secret Leakage (ESL)" attack under the CK-adversary threat model and also does not provide a blockchain-based solution. Note that the blockchain technology helps to improve better security in a storage system as compared to other traditional storage systems.

Ever [15] proposed an authentication framework for Internet of drone applications, where Unmanned Aerial Vehicle (UAVs) are addressed as the mobile-sinks in the wireless sensor networks (WSN) architecture. Their scheme is presented as "one-time user authentication for sensor nodes, cluster head and mobile sinks (UAVs)". But their protocol is not resistant against "ESL attack under the CK-adversary model", "anonymity and untraceability". Moreover, their protocol needs more communication and computational costs and also does not provide blockchain-based solutions.

Ali *et al.* [16] improved an authentication scheme based on Srinivas *et al.* [17] scheme in Internet of Things environment. Their scheme is not secure against the IND-CPA security model, and the established session key is not secure under the "CK-adversary model" that is their scheme does not protect the "Ephemeral Secret Leakage (ESL)" attack.

Extensive security solutions and their countermeasures for individual planes (application, control, data plane) of SDN framework are reported in the recent literature [4], [7], [8]. Cognition is indigenous frameworks that are proposed to deal with the application plane security of SDN framework [18]. Note that, the first one was directly plugged in with the *de facto* OpenFlow framework to provide security in the application plane for SDN application whereas the later one discussed several security enforcement policies through different "cognitive functions".

Towards the solution of the existing Single Point of Failure Issue (SPFI) of the controller in the SDN framework, several state-of-the-art studies are reported in the recent literature [6], [7], [8], [19]. To protect network assets or resources in such types of application from unauthorized entities, various access control mechanisms are reported in the literature [4], [20], [21], [22], [23], [24], [25]. Also, for authentication, auditing, and tracing out different activities among various planes namely, application, control, and data planes of SDN framework, various studies have been conducted in the existing state-of-art [3], [4]. Flover and Veriflow were introduced to visualize the performance impact of SDN applications by inspecting different flow policies. Further, few authentication schemes, like FSL, HiAuth [2] and FortNox were proposed in the SDN environment to enhance the overall security of the said domain.

Moreover, in the current setting of SDN, to achieve mutual authenticated key agreement between switch and controller, Transport Layer Security (TLS) has been put forward. However, due to the outrageous pitfall of TLS, it is optional and inadequate to fulfill the underlying security requirement between the controller-switch communication in SDN applications.

In spite of this, several other security solutions are reported in the recent literature [26], [27], [28], [29], [30], [31] for SDN environment. A summary of the state-of-art access control mechanisms in IoT and SDN platforms is also provided in Table I.

TABLE I
SUMMARY OF THE STATE-OF-ART ACCESS CONTROL MECHANISMS IN IOT AND SDN PLATFORMS

State-of-art schemes (Year)	Specification and loopholes
Jindal et al. [10], (2019)	* The authors have proposed a blockchain-based secure energy trading scheme for electric vehicles (EVs) scheme called "BEST". * Their scheme is resilient against the single point of failure issue. * SSL/TLS mechanism is used for message communication between two principals.
Chaudhary et al. [9], (2019)	* The authors have proposed a blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled V2G environment called "SURVIVOR". * Their scheme is resilient against the single point of failure issue. * SSL/TLS mechanism is used for message communication between two principals.
Abdullaziz et al. [2], (2019)	* The authors have proposed a lightweight authentication protocol called "HiAuth" for SDN platform. * Their scheme incorporates information hiding technique into OpenFlow to provide superior security against Denial of Service (DoS) attack. * During authentication protocol design, the authors have used Exclusive-OR (XOR)-based logical operations only, hence the practical performance of the scheme is better.
Jiasi et al. [4], (2019)	* The authors have introduced a blockchain-enabled access control scheme in SDN environment. * Their proposed scheme advocated an attribute-based access control mechanism between the application and control plane of SDN. * In addition, a combination of public key cryptography and digital signature based entity authentication mechanisms have been advocated between the control and data planes of the underlying SDN framework.
Luo et al. [32], (2018)	* They proposed an access control scheme for WSN environment. * Their scheme is also expensive in computation due to involvement of identity-based cryptographic technique and bilinear pairing. * Furthermore, it involves the gateway node for achieving the access control process among any two neighboring IoT smart devices.
Bonola et al. [5], (2017)	* The authors have proposed a data-plane programming abstraction for software-defined stream monitoring called "Streamon". * The authors have designed a pragmatic application programming interface (API) for developing stream-based monitoring tasks in SDN platform. * Their proposed API can have the ability to detect and mitigate distributed DoS (DDoS) attacks.
Li et al. [33], (2016)	* Their access control scheme is for IoT-enabled WSN environment that is based on heterogeneous signcryption. * Their scheme is expensive in computation due to involvement of identity-based cryptographic technique and bilinear pairing. * It involves the gateway node for achieving the access control process among any two neighboring IoT smart devices.
Xia et al. [1], (2014)	* The authors have advocated a rigorous survey on the security and other functional aspects of SDN framework. * This review emphasizes on the existing challenges of source authentication and access control policies in SDN. * The proposed study shows that spoofing, MITM, DoS, DDoS attacks are the most critical vulnerabilities in the three-layered SDN architecture.
Matsumoto et al. [8], (2014)	* The authors have proposed a fine-grained defensive mechanism called "Fleet" to protect SDNs from malicious administrators". * Fleet solved the existing malicious administrator's accessibility problem in SDNs. * This approach eliminates several issues of multiple controllers usage in the control plane of the underlying SDN platform.
Phemius et al. [7], (2014)	* The authors have introduced an extensible distributed SDN control plane called "DISCO". * Their proposed plane is able to cope up with the distributed and heterogeneous nature of modern overlay networks. * The authors have claimed that DISCO is a resilient, scalable and easily extensible SDN control plane.

III. SYSTEM SPECIFICATIONS

This section demonstrates the adopted network architecture as well as the adversary model (so-called the overall system abstraction) to present the proposed access control scheme.

A. Network model

The proposed access control architecture for a generic software-defined network [2] is shown in Fig. 1. This model consists of five different entities: like a trusted registration authority (RA), SDN-applications ($\mathbf{SA}$), SDN-controllers ($\mathbf{SC}$), SDN-switches ($\mathbf{SSW}$), and a peer to peer cloud servers network (**P2P CS network**). These entities and their responsibilities are demonstrated as follows:

1) **RA:** This entity is considered as a trusted authority and is responsible to register the entities belongs to each plane of the SDN framework.
2) **SA:** An amalgamation of business logic and software application called SA is solely responsible to manage the underlying infrastructure layer (or data plane), and preserve the dynamicity of the entire network through

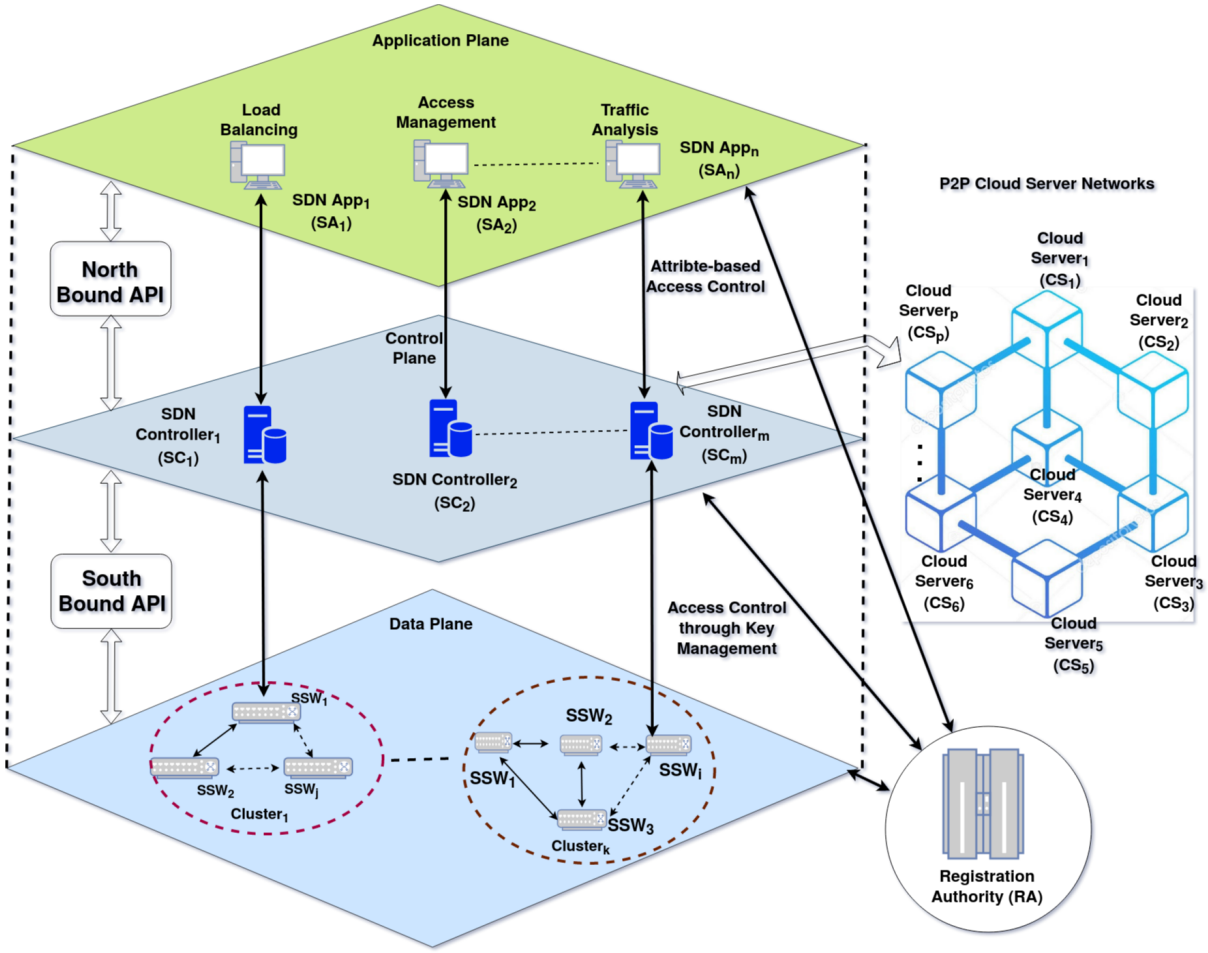

Fig. 1. Network model used in the proposed scheme (Source: [2])

SC. An example of such an entity is "load balancer", "firewall", "traffic analyzer", "intrusion detector", etc.

3) **SC:** This entity is liable for supervising and controlling the application flow as well as data traffic from its associated (SA) and network switch (SSW) respectively. In this connection, SC captures the raw data related to uninterrupted network traffic, switching events, and application flow. It then constructs the blocks followed by building transactions from the raw data. Finally, SC sends the block to other controllers node in the "P2P controller network".
4) **SSW:** This principal is considered as the sole component in the underlying data layer of the SDN framework, and it is primarily responsible for data delivery from a source node to a destination node. A distinct collection of such entities forms a cluster (CL) in data plane, and each CL is connected and managed by the SC. Note that for each distinct cluster $CL_j, \forall j = 1, 2, \cdots, m$, one or more switches are deployed. It may be noted that the controller node SC manages and controls the data communications among SSWs inside a cluster CL_j, and assesses a new configuration installation into the switch, inter-switch data forwarding, and network event monitoring. The detail description of cluster formation and its related key ingredients can be found in Remark 1.
5) **P2P CS Network:** A set of cloud servers (i.e., $CS_1, CS_2, \cdots, CS_p$) forms a "P2P Cloud Server Network" wherein a distributed ledger is maintained for block addition and mining, and maintaining all the partial blocks that are generated by each SDN controller (SC). These servers are collaboratively providing uninterrupted, fault-tolerant and ubiquitous network event monitoring, debugging, accounting and auditing facilities inside a particular realm of a SDN framework.

Remark 1. *Without any loss of generality, the core architecture of both device and data planes in the SDN framework conveys that there exists a Core Backbone Network (CBN) in the data layer, and it comprises several inter connected switches. Furthermore, the switches are integrated with a pre-defined network topology (i.e., star, mesh, ring, etc.) as reported in [34]. In addition, each switch in the CBN behaves like a coordinator node for a variety of networks like "3rd Generation Partnership Project (3GPP)", WiFi (IEEE 802.11), "Long-Term Evolution (LTE)", and IEEE 802.15.4 [4] [34] at device layer. We represent such an immensely large and scalable network configuration as a distinct cluster* CL_j $\forall j = 1, 2, \cdots, m$ *(refer to Fig. 1). Presently, in this study, we*

grossly ignore the detailed description of the cluster formation logic and algorithms, and also the key ingredients of its practical implication to reduce the inherent complexity of the paper. However, in future work, we plan to provide a spotlight on this issue in a very rigorous manner.

B. Threat Model

In this study, we have adopted the well-known "Dolev-Yao (DY) threat model" [35], [36] wherein this model allows an adversary $\mathcal{A}$ to intercept the authentication request/response messages between two intended parties (i.e., either between SA and SC or between SC and SSW) involved into communication, and even more he/she can insert, modify, delete and relay the said eavesdropped messages. Moreover, due to the existence of software bugs, it is also possible to insert malicious content into SAs. Utilizing such an act, the adversary $\mathcal{A}$ may mount denial-of-service and spoofing attacks on SC into SDN. Under this threat model, the end communicating party such as, SA and SSW are not at all considered as a trusted entity in the deployed network but the RA and the SDN controller (SC) in the SDN framework is assumed to be a trusted entity. Nonetheless, the cloud server $(CS_i|1 \leq i \leq p)$ in the P2P CS Networks is partially trusted.

We have also adopted the Canetti and Krawczyk's adversary model (CK-model) [37] which is recognized as a *de facto* adversary model for designing security protocol, and it is widely used in the literature for security inspection [38], [39], [40], [41]. This model enhances the capability of the DY model by adding the following new clause. The said clause conveys that the capability of an adversary $\mathcal{A}$ is not limited to intercept, modify, delete or insert the communication messages of different entities (i.e., SA, SC, and SSW) involved in the network as specified in the said DY model, but $\mathcal{A}$ can also able to capture long-term secrets, short-terms keys and session states in case this information are stored inside an insecure memory of the said principals at the time of mutual authentication phase [37].

Besides, it may not be always possible to monitor the deployed SAs in the 24 $\times$ 7 times. However, there is a possibility of physical capture of some SAs by a maintenance engineer acting as an internal adversary $\mathcal{A}$. Thereafter, utilizing such the extracted secret credentials stored in the physically captured SAs with the help of power analysis attacks [42], the $\mathcal{A}$ may attempt to mount other attacks, such as identity compromisation and impersonation attacks.

TABLE II
NOTATIONS AND THEIR MEANINGS

Notation	Description
App_{id}, Con_{id}, Swi_{id}	Real identities of SDN application (SA), controller (SC) and switch (SSW)
App_{type}	Type of SDN application (SA) (i.e., traffic management, access management and security control)
$Cluster$, $Cluster_{id}$	A set of network assets (combination of controller and switches), real identity of cluster
$Cert_{app}$, $Cert_{con}$, $Cert_{swi}$	Certificates of SA, SC and SSW generated by the RA (Registration Authority)
$ECDSA$.	Elliptic Curve Digital Signature Algorithm (ECDSA)
$Sig(s_K : [M])$	signature generation on a message M with private key s_K of an entity X
$ECDSA.Ver$ $(Pub_K : [M])$	ECDSA verification algorithm on message M with public key Pub_K of an entity X
T_x^{id}	Transaction identity created for principal x
$Flow_{id}$	Real identity of an SDN application flow
$state$	Current state of the whole network
$content$	Flow content (specifically, configuration instruction) with respect to $Flow_{id}$
$Event_{id}$	Real identity of a network event
$E_q(u, v)$	A "non-singular elliptic curve" with parameters u, v over finite (Galois) field $GF(q)$; q being a prime
G	A base point in $E_q(u, v)$ of order n
$P + Q$, $k.P$	Elliptic curve point addition and multiplication, respectively, $P, Q \in E_q(u, v)$ and $k \in Z_q^* = \{1, 2, \cdots, q-1\}$
$x * y$	Ordinary modular multiplication in Z_q
ΔT	"Maximum transmission delay associated with message"
$h(\cdot)$	"Collision-resistant cryptographic one-way hash function"

IV. Proposed Approach

This section discusses the proposed private blockchain-based access control scheme for a generic SDN framework, called PBAC-SDN. Initially, a brief overview of the proposed scheme is demonstrated, and after that, the detailed phase-wise illustration of each phase of the proposed scheme is illustrated. To discuss the proposed access control scheme few notations are used throughout this paper. These annotations and their corresponding meaning are highlighted in Table II.

In the system initialization phase (*SI-PBAC-SDN*), all domain-specific public-private key pairs and configuration parameters are getting finalized. In this connection different security-specific domain attributes such as "one-way cryptographic hash function", "non-singular elliptic curve and its base point", "secret and public keys", and initialization parameters of consensus protocols are chosen for all the principals involved in the SDN framework.

After successful execution of *SI-PBAC-SDN*, the proposed framework undergoes an entity registration process, called *REG-PBAC-SDN*. In this connection, the RA enrolls all the principals by utilizing *REG-PBAC-SDN* task. During this process, RA computes various parameters, and stores these into the corresponding entities database before their deployment into the network. It may be noted that these said pre-stored parameters are being utilized at the time of entity authentication and access control phase (*ACC-PBAC-SDN*).

In the proposed *ACC-PBAC-SDN* phase, three different access control policies are introduced. According to the first policy, a business/network application (SA) that deployed on the top-tier plane of the SDN framework adopts an attribute-based encryption/decryption policy to access the controller node (SC). Through this approach, a legitimate SA would able to achieve secure flow routing through SC, and install a new security feature and/or software patches into a single or multiple switches $(SSWs)$. However, in the second policy, a certificate-based "authenticated key agreement protocol" is put forward to establish a private channel between SC and SSW to assess secure data forwarding and network event monitoring. Nonetheless, in the third scenario, a private blockchain-enabled secure data collection, aggregation, and storing in terms of blocks have been carried out utilizing the "block verification and addition operations into the P2P CS

network" provided in the *BADD-PBAC-SDN* phase.

In the *BADD-PBAC-SDN* phase, each SDN-controller accumulates the raw data (in terms of interaction messages) from its associated application and network infrastructures (i.e., switches). More precisely, the interaction messages that are securely (using the said access control mechanism) traversed through each controller node are primarily being captured and aggregated into the controller node itself, and later this node constructs a partial block. It may be noted that, the controller node builds the partial block by encapsulating a batch of encrypted transactions wherein each transaction consists of a collection of different interaction messages. It is worth noticing that these transactions are being encrypted using the transaction builder's (specifically, the controller's) own public key. After that, the controller sends those partial blocks to its associated cloud server in the "P2P CS Networks". After receiving the partial block by the cloud server it verifies the Merkle tree root on the encrypted transactions present in that particular block. If it is valid, then only the cloud server continues to convert the partial block into a full block by adding block version, public key of signer, current block hash, and the digital signature on the current block hash using its own private key. Next, through a voting-based consensus mechanism based on the "Practical Byzantine Fault Tolerance (PBFT)" [43] is executed among those cloud servers in a P2P network. Eventually, the cloud servers (not an individual cloud server) in the peer-to-peer network verify and mine the block for future use.

A. System Initialization Phase (SI-PBAC-SDN)

According to this phase, the registration authority (RA) selects a "non singular elliptic curve $E_q(u,v)$ of the form: $y^2 = x^3 + ux + v \pmod q$ over a finite field GF(q) with a point of infinity $\mathcal{O}$, where $(u,v) \in Z_q = \{0,1,2,\cdots,q-1\}$ and $4u^3 + 27v^2 \neq 0 \pmod q$". The RA then also chooses a generator point "$G \in E_q(u,v)$ of order n as large as q", where $n.G = G + G + \cdots G$ (n times). After that, the RA picks a "one way cryptographic hash function" $h(\cdot)$ (say for instance, SHA-256 [44] can be utilized to ensure sufficient security for blockchain technology).

B. Entity Registration Phase (REG-PBAC-SDN)

The step-wise enumerations of the entity enrollment phase are highlighted as follows:

Step 1: Initially, a trusted third party (RA) enrolls all the entities (i.e., SA, SC, SSW, and CS) prior to their deployment. In this connection, the RA chooses distinct identities for those principals as App_{id}, Con_{id}, Swi_{id}, and CS_{id}, respectively. Then RA selects a public-private key pair say, $Pub_{RA} = s_{RA} \cdot G$ and $s_{RA} \in Z_q^*$ for itself. RA also chooses the public-private key pairs for SA, SC, and SSW as $\{Pub_{app} = s_{app} \cdot G, s_{app} \in Z_q^*\}$, $\{Pub_{con} = s_{con} \cdot G, s_{con} \in Z_q^*\}$, and $\{Pub_{swi} = s_{swi} \cdot G, s_{swi} \in Z_q^*\}$ respectively, and go to Step 2.

Step 2: After generating the public-private key pair, the RA computes certificates for each participant (specifically, SA, SC, and SSW) of the network such as, $Cert_{app} = s_{app} + h(App_{id} \,||App_{type}||\, Pub_{app}||\, Pub_{RA}) * s_{RA} \pmod q$, $Cert_{con} = s_{con} + h(Con_{id} \,||Cluster_{id}||\, Pub_{con}||\, Pub_{RA}) * s_{RA} \pmod q$, and $Cert_{swi} = s_{swi} + h(Swi_{id} \,||Cluster_{id}||\, Pub_{swi}||\, Pub_{RA}) * s_{RA} \pmod q$, and go to Step 3.

Step 3: RA picks a "t-degree symmetric bivariate polynomial" over $GF(q)$ say, $f_{con-swi}(x,y) = \sum_{i=0}^{t}\sum_{j=0}^{t} a_{ij}x^i y^j, \forall a_{ij} \in Z_q$, and computes the polynomial shares for each controller (SC) and switch (SSW) as $f_{con-swi}(Con_{id}, y)$ and $f_{con-swi}(Swi_{id}, y)$.

Step 4: RA assigns $\{App_{id}, s_{app}, Cert_{app}\}$, $\{Con_{id}, s_{con}, Cert_{con}, f_{con-swi}(Con_{id}, y)\}$, and $\{Swi_{id}, s_{swi}, Cert_{swi}, f_{con-swi}(Swi_{id}, y)\}$ to individual SA, SC and SSW, respectively, and go to Step 5.

Step 5: RA broadcasts the public parameters such as, $\{Pub_{app}, Pub_{con}, Pub_{swi}, Pub_{RA}, Con_{id}, App_{id}, Swi_{id}$, and $CS_{id}\}$ to the corresponding entities, and deletes the secret keys specifically, s_{app}, s_{con} and s_{swi} from its own database and temporary credential caches.

Finally, after the successful accomplishment of the registration process, each cloud server (CS) generates its own private key $s_{cs} \in Z_q^*$, and calculates the respective public key $Pub_{cs} = s_{cs} \cdot G$. CS then publishes its public key Pub_{cs}. The access control tasks between application and control plane, and control and data plane have been carried out after completion of the said enrollment process. The detailed discussion of these phases are illustrated in Sec. IV-C.

C. Access Control Phase (ACC-PBAC-SDN)

This section discusses the access control mechanisms between application (SA_k) and controller (SC_i), and network switch (SSW_j) and controller (SC_i) node in the underlying SDN framework.

1) Access Control Policy between SA_k and SC_i: The access control policy based on the "Attribute-Based Encryption (ABE)" [45] to provides security for the message communication among SDN application and its associated controller. The purpose of adopting this ABE is to allow the SDN controller to recover its exclusive application(s).

In general, an ABE scheme can be categorized by the following phases: a) Setup phase, b) Encrypt phase, c) KeyGen phase, and d) Decrypt phase [46] and stated as follows:

In the **Setup** phase, a "security parameter λ" and the "universe of attributes A_i", say $\mathcal{U} = \{A_1, A_2, \cdots, A_n\}$ takes as inputs and generates a "master secret" and "public key" pair, say (MK, PK) as output.

In the **Encrypt** phase ($Enc(\mathcal{P}, Msg)$) it takes an "access policy $\mathcal{P}$", a "plaintext message Msg" and the "public key PK" as inputs, and it produces the "ciphertext C as output".

The **KeyGen** phase requires an "attribute set $\mathbb{A}$", the "master secret key MK" and "public key PK" as inputs values and produces the "user secret key (i.e., decryption key), say K_U" of $\mathbb{A}$ for the user U as output value.

Finally, the **Decrypt** phase ($Dec(C, \mathcal{P}, K_U, \mathbb{A})$) needs a "ciphertext C generated with $\mathcal{P}$", the "secret key K_U" associated with $\mathbb{A}$, and the value of the public key PK as inputs. After executing this phase, it produces the message Msg or $\perp$ (null) as output.

The transactions can be transmitted among the controller and switch by encrypting with the help of the ABE **Encrypt** policy as mention above by their "access policy $\mathcal{P}$" and "public key PK". After receiving the message, the receiver decrypts the received message (or transactions) applying the **Decrypt** policy based on "secret key K_U" associated with $\mathbb{A}$ and the public key PK.

2) Access Control Policy between SC_i and SSW_j: In this section, the authenticated key agreement process between (SSW_j) (inside a particular cluster say, CL_k) and (SC_i) is demonstrated. The detailed summary of this task can be visualized from Fig. 2. Also, the step-wise enumeration of the same process is demonstrated below. It may be noted that the basic concept of "polynomial-based key distribution mechanism" by Blundo *et al.* [47] is partly utilized for forming the session key between two intended communicating parties.

• *Step $ACCCNSW_1$:* Initially, SC_i chooses a random number $r_1 \in Z_q^*$, picks system's timestamp TS_1, and computes $f_{con-swi}(Con_{id}, Swi_{id})$, $A_{con} = h(r_1 \,||TS_1) \oplus h(f_{con-swi}(Con_{id}, Swi_{id}) \,||TS_1)$ and $B_{con} = h(Con_{id} \,||Swi_{id} \,||A_{con} \,||TS_1 \,||Cert_{con} \,||f_{con-swi}(Con_{id}, Swi_{id}))$, and construct a message $msg_1 = \{Con_{id}, Swi_{id}, Cert_{con}, A_{con}, B_{con}, TS_1\}$ to SSW_j. SC_i then sends msg_1 through an open media.

• *Step $ACCCNSW_2$:* Receiving the message msg_1 from SC_i at time $TS_1^{'}$, the network switch SSW_j checks the condition $|TS_1^{'} - TS_1| < \Delta T$ wherein ΔT signifies the "maximum transmission delay". If the condition returns true then SSW_j computes $f_{con-swi}(Swi_{id}, Con_{id})$, $h(r_1||TS_1) = A_{con} \oplus h(f_{con-swi}(Swi_{id}, Con_{id}) \,||TS_1)$, and $B'_{con} = h(Con_{id} \,||Swi_{id} \,||A_{con} \,||TS_1 \,||Cert_{con} \,||f_{con-swi}(Swi_{id}, Con_{id}))$; otherwise rejects SC_i's request. Further, if the condition $B'_{con} = B_{con}$ is not satisfied then SSW_j closes the session.

• *Step $ACCCNSW_3$:* In the third-level of verification, SSW_j checks the SC_i's certificate as $Cert_{con} \cdot G = Pub_{con} + h(Con_{id}|| \; Cluster_{id}|| \; Pub_{con}|| \; Pub_{RA}) \cdot Pub_{RA}$ or not using the previously known domain-specific public data. If the above condition is satisfied then SC_i is known to be a legitimate entity for SSW_j; otherwise rejects SC_i's request message. Mean while, SSW_j picks a random number r_2 and system's timestamp TS_2, and computes $A_{swi} = h(r_2 \,||TS_2) \oplus h(f_{con-swi}(Swi_{id}, Con_{id}) \,||TS_2)$. Further, SSW_j also computes the "shared session secret key" with SC_i as $SK_{Swi,Con} = h(f_{con-swi}(Swi_{id}, Con_{id}) \,||h(r_1 \,||TS_1) \,||h(r_2 \,||TS_2) \,||Cert_{con} \,||Cert_{swi})$ and a "session key verifier" $SKV_{Swi,Con} = h(SK_{Swi,Con} \,||A_{swi} \,||Cert_{swi} \,||Con_{id} \,||Swi_{id} \,||TS_2)$. After that, SSW_j construct a message as $msg_2 = \{Con_{id}, Swi_{id}, Cert_{swi}, A_{swi}, SKV_{Swi,Con}, TS_2\}$ to SC_i, and sends the same message to SC_i via a open media.

• *Step $ACCCNSW_4$:* After receiving msg_2 at time $TS_2^{'}$ from SSW_j, SC_i checks that $|TS_2^{'} - TS_2| < \Delta T$ is true or false. If the condition returns true then SC_i verifies $Cert_{swi}$ by inspecting the below condition, that is, $Cert_{swi} \cdot G = Pub_{swi} + h(Swi_{id} \,||Cluster_{id} \,||Pub_{swi} \,||Pub_{RA}) \cdot Pub_{RA}$. If the condition is satisfied or the certificate is valid then SC_i computes $h(r_2|| \; TS_2) = A_{swi} \oplus h(f_{con-swi}(Swi_{id}, Con_{id}) \,||TS_2)$, a shared symmetric session key $SK_{Con,Swi} = h(f_{con-swi}(Con_{id}, Swi_{id}) \,||h(r_1 \,||TS_1) \,||h(r_2 \,||TS_2) \,||Cert_{con} \,||Cert_{swi})$, and a session key verifier as $SKV_{Con,Swi} = h(SK_{Con,Swi} \,||A_{swi} \,||Cert_{swi} \,||Con_{id} \,||Swi_{id} \,||TS_2)$. Next, SC_i substantiates the below condition as $SKV_{Con,Swi} = SKV_{Swi,Con}$ is true or not. If it pass the verification, then go to Step $ACCCNSW_5$.

• *Step $ACCCNSW_5$:* SC_i selects the system's timestamp TS_3, and calculate $ACK_{Con,Swi} = h(SKV_{Con,Swi}||TS_3)$. Then, SC_i builds the message $msg_3 = \{ACK_{Con,Swi}, TS_3\}$, and sends it to the SSW_j via an open media.

• *Step $ACCCNSW_6$:* After receiving the message (msg_3), SSW_j verifies that the condition $|TS_3^* - TS_3| < \Delta T$ is true or false. If it returns true then SSW_j calculates $ACK_{Swi,Con} = h(SKV_{Swi,Con}||TS_3)$. Moreover, SSW_j also checks the condition: $ACK_{Swi,Con} = ACK_{Con,Swi}$. If it returns true then only SSW_j accepts the SC_i's request; otherwise discard the message msg_3. Eventually, both SC_i and SSW_j keep the "shared secret session key" $SK_{Con,Swi}$ $(= SK_{Swi,Con})$ into their custody for "secret communications".

Thus, the authenticated key agreement task is accomplished between the controller node (SC_i) and a network switch hosted inside a cluster CL_k, and the shared secret session key between themselves is $SK_{Con,Swi}$ $(= SK_{Swi,Con})$.

D. Secure Data Aggregation, Block Verification and Mining Phase

Using the access control between a SDN application SA_k and a SDN controller SC_i (discussed in Section IV-C1), SA_k will communicate securely with the SC_i. Similarly, using the established session key $SK_{Con,Swi}$ between a SDN controller SC_i and a SDN switch SSW_j during the access control phase (discussed in Section IV-C2), SSW_j will securely communicate with SC_i. In this way, secure data aggregation takes place by SC_i.

Various transactions as shown in Fig. 3 are securely gathered by SC_i. Once a threshold number of transactions, say t, are collected, SC_i forms a partial block consisting of the information as shown in Fig. 4. For this purpose, SC_i will first encrypt the transactions Tx_i, for $i = 1, 2, \cdots, t$, using its own ECC-based public key Pub_{con}. After that the Merkle tree root (MTR) on all the t encrypted transactions $\{E_{Pub_{con}}(Tx_i) \; | i = 1, 2, \cdots, t\}$ is created and inserted in the partial block $ParBlock_m$. Once these steps are done, SC_i securely sends this partial block $ParBlock_m$ to its associated cloud server node, say CS by encrypted the block with the public key Pub_{cs} of the CS.

After receiving a partial block $ParBlock_m$ by the CS, it verifies the Merkle tree root on the encrypted transactions present in that block. If it is valid, then only CS continues to convert the partial block $ParBlock_m$ into its full block $Block_m$ by adding block version (BV), public key of signer (Pub_{cs}), current block hash $(CBHash)$ and then the "elliptic curve digital signature algorithm (ECDSA)" [48] signature on $CBHash$ using its own private key s_{cs}. The structure of the full block $Block_m$ is shown in Fig. 5.

The block verification and mining phase can be conducted through the voting-based consensus mechanism which is described in Algorithm 1. The cloud server nodes $\{CS_i$

SDN-Controller (SC_i)	**Network switch** (SSW_j)
Pick a random number $r_1 \in Z_q^*$	
Pick system's timestamp TS_1	
Calculate $f_{con-swi}(Con_{id}, Swi_{id})$, $A_{con} = h(r_1 \| TS_1) \oplus h(f_{con-swi}(Con_{id}, Swi_{id}) \| TS_1)$, $B_{con} = h(Con_{id} \| Swi_{id} \| A_{con} \| TS_1 \| Cert_{con} \| f_{con-swi}(Con_{id}, Swi_{id}))$ $msg_1 = \{Con_{id}, Swi_{id}, Cert_{con}, A_{con}, B_{con}, TS_1\}$ $\xrightarrow{\text{(via an open media)}}$	Verify if $\|TS_1' - TS_1\| < \Delta T$? If valid then calculate $f_{con-swi}(Swi_{id}, Con_{id})$, $h(r_1 \| TS_1) = A_{con} \oplus h(f_{con-swi}(Swi_{id}, Con_{id}) \| TS_1)$, $B'_{con} = h(Con_{id} \| Swi_{id} \| A_{con} \| TS_1 \| Cert_{con} \| f_{con-swi}(Swi_{id}, Con_{id}))$
Verify if $\|TS_2' - TS_2\| < \Delta T$? If valid, check if $Cert_{swi} \cdot G = Pub_{swi} + h(Swi_{id} \| Cluster_{id} \| Pub_{swi} \| Pub_{RA}) \cdot Pub_{RA}$? If valid, calculate $h(r_2 \| TS_2) = A_{swi} \oplus h(f_{con-swi}(Swi_{id}, Con_{id}) \| TS_2)$, Shared secret session key $SK_{Con,Swi} = h(f_{con-swi}(Con_{id}, Swi_{id}) \| h(r_1 \| TS_1) \| h(r_2 \| TS_2) \| Cert_{con} \| Cert_{swi})$, session key verifier $SKV_{Con,Swi} = h(SK_{Con,Swi} \| A_{swi} \| Cert_{swi} \| Con_{id} \| Swi_{id} \| TS_2)$	Check if $B'_{con} = B_{con}$? If valid, check if $Cert_{con} \cdot G = Pub_{con} + h(Con_{id} \| Cluster_{id} \| Pub_{con} \| Pub_{RA}) \cdot Pub_{RA}$? Pick system timestamp TS_2 and a random number $r_2 \in Z_q^*$ Calculate $A_{swi} = h(r_2 \| TS_2) \oplus h(f_{con-swi}(Swi_{id}, Con_{id}) \| TS_2)$, shared secret session key $SK_{Swi,Con} = h(f_{con-swi}(Swi_{id}, Con_{id}) \| h(r_1 \| TS_1) \| h(r_2 \| TS_2) \| Cert_{con} \| Cert_{swi})$, session key verifier $SKV_{Swi,Con} = h(SK_{Swi,Con} \| A_{swi} \| Cert_{swi} \| Con_{id} \| Swi_{id} \| TS_2)$ $msg_2 = \{Con_{id}, Swi_{id}, Cert_{swi}, A_{swi}, SKV_{Swi,Con}, TS_2\}$ $\xleftarrow{\text{(via an open media)}}$
Check if $SKV_{Con,Swi} = SKV_{Swi,Con}$? Either accept or reject	
Pick system timestamp TS_3	
Calculate $ACK_{Con,Swi} = h(SKV_{Con,Swi} \| TS_3)$ $ACK_{Con,Swi} = h(SKV_{Con,Swi} \| TS_3)$ $msg_3 = \{ACK_{Con,Swi}, TS_3\}$ $\xrightarrow{\text{(via an open channel)}}$	Check if $\|TS_3^* - TS_3\| < \Delta T$? Either accept or reject Calculate $ACK_{Swi,Con} = h(SKV_{Swi,Con} \| TS_3)$ Check $ACK_{Swi,Con} \stackrel{?}{=} ACK_{Con,Swi}$ Either accept or reject
Store shared secret session key $SK_{Con,Swi} (= SK_{Swi,Con})$ between controller SC_i and switch SSW_j	

Fig. 2. Summary of access control between controller SC_i and switch SSW_j

$|i = 1, 2, \cdots, n_{CS}\}$ form a distributed peer-to-peer (P2P) network, where each cloud server (CS) is considered as a peer node having the responsibility to add a full block $Block_m$ by performing the voting-based "Practical Byzantine Fault Tolerance (PBFT)" consensus algorithm. The whole process of verifying the block $Block_m$ and adding into the blockchain by the P2P cloud servers network is described in Algorithm 1. At first, a leader from the P2P cloud servers network is selected. The leader selection can be securely done either using the existing leader selection algorithm provided in [49] or simply by round-robin fashion. Next, the leader will send a block addition request consisting of the block $Block_m$, and encrypted voting request and fresh timestamp to its other peer nodes in the network. Each peer node then decrypts the encrypted part of the message, and checks validity of timestamp. If the timeliness is passed, the peer node (follower) verifies the received block by means of verifying the Merkle tree root on the encrypted transactions $\{E_{Pub_{con}}(Tx_i) |i = 1, 2, \cdots, t\}$, hash of the block and signature using the ECDSA signature verification algorithm. If all the validations are passed, each follower then sends the reply message $VotL$ and block verification status $BVStatus$ as $\{E_{Pub_{CS_l}}(VotR', VotL, BVStatus)\}$ to the leader CS_l. If all the checks by the leader CS_l are done successfully, it then proceeds to transmit the commit response to all the follower nodes. Finally, the block $Block_m$ is added into the blockchain.

Remark 2. *Blockchain Technology (BT) is recognized as one of the reliable solutions in developing a robust way out of different security problems, specifically authentication, accounting and auditing, and access control. This technology is treated as a most promising approach for transferring the digital information among nodes belong to a peer-to-peer network [50]. The applicability of the said technology towards various security solutions for SDN are summarized in [51]. However, the main purpose to adopt the blockchain technology into our proposed access control mechanism is not only restricted to improve better security in a storage, but also to deal with the following characteristics:*

1) ***Transparency:*** *In blockchain, all the data is transparent and available to be traced by any nodes in the P2P network, but the confidentiality of the data can be achieved by encryption. In other words, the confidential data is preserved into the blockchain in an encrypted format, and the decryption keys can be shared only with authorized individuals. In our proposed protocol, we adopt a private blockchain based secure data aggregation and data storing mechanism wherein several interaction messages (so called the raw data) among SA and SC, and SC and SSW are encrypted and encapsulated into several transactions. Later, these transactions are permanently stored into the blockchain. As a result, an administrator associated with the SC can access the data and take various business related decisions. Hence, our proposed scheme supports transparency.*
2) ***Ledger irreversibility****: Blockchains are developed to be immutable. Once a block is written to a blockchain, it cannot be modified, even if someone updates the data in the block, it does not get overwritten; instead, it is stored in a new block. In our proposed scheme, a block contains specifically a set of encrypted transactions $(Tx_i | i = 1, 2, 3, \cdots, t)$, timestamp (TS), hash values (i.e., previous block hash $(PBHash)$ and current block hash $(CBHash)$), Merkle Tree Root (MTR), and ECC-based digital signatures $(ECDSA.Sig(\cdot))$ which*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3088115, IEEE Internet of Things Journal

Transactions Details
(1) Transaction format of individual component (SA, SC, SSW) in SDN framework
• $T_{SA} \rightarrow \{T_{SA}^{id}, App_{id}, Pub_{app}, App_{type}, Cert_{app}, Sign_{app}\}$ • $T_{SC} \rightarrow \{T_{SC}^{id}, Con_{id}, Pub_{con}, Cluster_{id}, Cert_{con}, Sign_{con}\}$ • $T_{SSW} \rightarrow \{T_{SSW}^{id}, Swi_{id}, Pub_{swi}, Cluster_{id}, Cert_{swi}, Sign_{swi}\}$, where $Sign_{app} = ECDSA.Sig(s_{app} : [App_{id}\|\| App_{type}])$, $Sign_{con} = ECDSA.Sig(s_{con} : [Con_{id}\|\| Cluster_{id}])$, $Sign_{swi} = ECDSA.Sig(s_{swi} : [Swi_{id}\|\| Cluster_{id}\|\| Con_{id}])$
(2) Interaction between application (SA) and controller (SC)
• $T_{SA-SC} \rightarrow \{T_{SA-SC}^{id}, T_{SA}^{id}, T_{SC}^{id}\}$
(3) Interaction between controller (SC) and switch (SSW)
• $T_{SC-SSW} \rightarrow \{T_{SC-SSW}^{id}, T_{SC}^{id}, T_{SSW}^{id}\}$
(4) Application flow request to the controller (SC) before install (or uninstall) a new (or old) configuration to (or from) the switch (SSW) for a particular reason
• $T_{appflw2con}^{bf} \rightarrow \{T_{appfl_{id}}^{bf}, App_{id}, Flow_{id}, Con_{id}, Pub_{app}, content, Sign_{appfl}^{bf}\}$, where $Sign_{appfl}^{bf} = ECDSA.Sig(s_{app} : [App_{id}\|\|Flow_{id}\|\|Con_{id}\|\|content])$.
(5) Action taken by the controller (SC) with respect to the application flow request (as mentioned in previous transaction), and updation of network state
• $T_{appflw2con}^{af} \rightarrow \{T_{appfl_{id}}^{af}, Con_{id}, Flow_{id}, Swi_{id}, state, Sign_{appfl}^{af}\}$, where $Sign_{appfl}^{af} = ECDSA.Sig(s_{con} : [App_{id}\|\|Flow_{id}\|\|Con_{id}\|\|Swi_{id}\|\|state])$
(6) Current status of the network after execution of an application flow (considering both transactions $T_{appflw2con}^{bf}$ and $T_{appflw2con}^{af}$)
• $T_{appflow} \rightarrow \{T_{flow}^{id}, T_{appfl_{id}}^{bf}, T_{appfl_{id}}^{af}\}$
(7) A network event occur between controller and switch
• $T_{swi-event} \rightarrow \{T_{swi-event}^{id}, Event_{id}, Pub_{swi}, Swi_{id}, Con_{id}, event\}$
$T_{SA}, T_{SC}, T_{SSW}, T_{SA-SC}, T_{SC-SSW}, T_{appflw2con}^{bf}, T_{appflw2con}^{af}, T_{appflow}$, $T_{swi-event}$ represent different transactions constructed by the controller (SC)

Fig. 3. Formats of various transactions and their details

Block Header	
Merkle Tree Root	MTR
Timestamp	TS
Owner of Block	OB
Public key of encryptor	Pub_{con}
Block Payload (Encrypted Transactions)	
List of Encrypted Transactions $\#i$ (Tx_i)	$\{E_{Pub_{con}}(Tx_i) \| i = 1, 2, \cdots, t\}$

Fig. 4. Formation of a partial block $ParBlock_m$ by a controller node (SC)

Block Header	
Block Version	BV
Previous Block Hash	$PBHash$
Merkle Tree Root	MTR
Timestamp	TS
Owner of Block	OB
Public key of encryptor	Pub_{con}
Public key of signer	Pub_{cs}
Block Payload (Encrypted Transactions)	
List of Encrypted Transactions $\#i$ (Tx_i)	$\{E_{Pub_{con}}(Tx_i) \| i = 1, 2, \cdots, t\}$
Current Block Hash	$CBHash$
Signature on $CBHash$	$BSign = ECDSA.Sig(s_{cs} : [CBHash])$

Fig. 5. Formation of a full block $Block_m$ by a P2P cloud server node (CS)

are distinct and obtained after spending some computational power. The integration of such mathematical primitives makes it impossible to compromise the block vis-a-vis the entire chain of blocks even if it is outsourced into the cloud. Now, if some insiders try to modify the transactions inside a block, it is needed to alter the PBHash, CBHash, MTR, and digital signature, and also should have the knowledge about the decryption keys. To make it valid, a privileged-insider or an adversary requires a large computational power. As a result, the data that is stored into the blockchain can not be changed. Therefore, our proposed scheme satisfies the ledger irreversibility property.

3) ***Decentralization****: In blockchain, decentralization represents to the movement of decision-making and control process from a centralized principal (specifically, an individual or organization) to a distributed network. Although a blockchain is essentially distributed (i.e., many entities hold instances of the ledger) but it is not necessarily decentralized. In a nutshell, whether*

Algorithm 1 Voting-based consensus for block verification along with addition in blockchain

Input: A block $Block_m$ as shown in Fig. 5
Output: Commitment for block addition

1: Let a cloud server node (CS_l) is elected as a leader and hold the block $Block_m$.
2: CS_l picks a fresh timestamp TS_{CS_j} for other follower cloud server nodes CS_j, $(l \neq j)$.
3: Generate a voting request $VotR$ and encrypt it as $E_{Pub_{CS_j}}(VotR,\ TS_{CS_j})$ using its own public key Pub_{CS_j}, and send a block addition request with the same block and $E_{Pub_{CS_j}}(VotR, TS_{CS_j})$ to other cloud server nodes CS_j, $(j = 1, 2, \cdots, n_{CS}, \forall j \neq l)$.
4: Let the request be received by the nodes CS_j from CS_l at time $TS^*_{CS_j}$.
5: **for** each follower CS_j **do**
6: Decrypt the request by $(VotR',\ TS_{CS_j}) = D_{s_{CS_j}}[E_{Pub_{CS_j}}(VotR, TS_{CS_j})]$ using its private key s_{CS_j}.
7: Validate the timestamp, Merkle tree root, block hash, and signature using ECDSA signature verification algorithm on the block $Block_m$.
8: If all validations are successful, send the reply message $VotL$ and block verification status $BVStatus$ as $\{E_{Pub_{CS_l}}(VotR',\ VotL,\ BVStatus)\}$ to CS_l.
9: **end for**
10: Let $VTCount$ denote the valid votes counter. Initialize $VTCount \leftarrow 0$.
11: **for** each received response message $\{E_{Pub_{CS_l}}(VotR',\ VotL,\ BVStatus)\}$ from the responded follower CS_j **do**
12: Compute $(VotR',\ VotL,\ BVStatus) = D_{s_{CS_l}}[E_{Pub_{CS_l}}\ (VotR',\ VotL,\ BVStatus)]$ by decrypting the response message using its own private key s_{CS_l}.
13: **if** $((VotR' = VotR)$ and $((VotL = valid)$ and $(BVStatus = valid)))$ **then**
14: Set $VTCount = VTCount + 1$.
15: **end if**
16: **end for**
17: **if** $(VTCount > 2n_{fCS} + 1)$ **then**
18: Send the commit response to all follower cloud server nodes.
19: $Block_m$ is now added to the blockchain.
20: **end if**

a blockchain is decentralized or centralized solely directs to the access privileges of the participants on the ledger, and hence it entirely depends on the matter of application design. To conceptualize the decentralization mechanism in our proposed scheme, individual SDN controller initially accumulates the raw data (in terms of interaction messages) from its associated application and network infrastructures (i.e., switches). The controller node then builds a partial block by encapsulating a batch of encrypted transactions wherein each transaction consists of a collection of different interaction messages. After that, the controller securely sends the partial blocks to its associated cloud servers $\{CS_i|1 \leq i \leq p\}$ in the "P2P CS network". After receiving a partial block by the CS, it verifies the Merkle tree root on the encrypted transactions present in that block. If it is valid, then only the cloud server continues to convert the partial block into a full block by adding block version, public key of signer, current block hash, and the digital signature on the current block hash using its own private key. Next, through a voting-based consensus mechanism using the "Practical Byzantine Fault Tolerance (PBFT)" [43] is executed among those cloud servers in a P2P network. Eventually, a set of cloud servers (not an individual server) verify and mine the block for future usage. Thus, we could remark that the proposed scheme support decentralization.

V. Security Analysis

This section initially carried out the formal verification of the proposed protocol PBAC-SDN by utilizing the widely used Real-Or-Random (ROR) model. After that, the potentiality of PBAC-SDN in terms of different internal and external security threats prevention is carried out through an informal security inspection. Finally, a formal security evaluation through a simulation study is put forward by utilizing the widely-used AVISPA tool.

A. Formal Security Analysis Under ROR Oracle Model

Abdalla *et al.* [11] proposed the widely-accepted Real-Or-Random (ROR) model for analyzing the formal security of an authenticated key agreement protocol $(AKAP)$. Since its inception, a probabilistic game theory-based formal security modeling and verification techniques are advocated, and utilizing such a model one can easily infer that a newly designed AKAP provides the session-key (SK) security against a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ or not. Before providing the proof of session-key (SK) security for the proposed PBAC-SDN in Theorem 1, the following key ingredients of the ROR model need to be specified.

Intuitively, the proposed protocol PBAC-SDN consists of four key players namely, application (SA), controller (SC), switch (SSW), and cloud server (CS) in the P2P CS networks. Therefore, the following configurations are needed to be advocated in the ROR model:

(a) Participants. Suppose λ^u_{SA}, λ^t_{SC}, λ^v_{SSW}, and λ^w_{CS} are the instances u, t, v and w of the participants SA, SC, SSW and CS, respectively. These instances are also termed as the oracles.

(b) Accepted state. Let an instance say, λ^t is coined as an accepted state if it reaches an accept state after receiving the end message of the proposed protocol. The session identity namely, sid is established by sequential concatenating of all the transmit and received messages of λ^t for the recent executing session.

(c) Partnering. Two instances namely, λ^t and λ^v are assumed to be partnered to each other if, and only if, all the below three conditions are valid: i) both λ^t and λ^v reaches an

TABLE III
SUMMARY OF QUERIES AND THEIR UTILITIES

Queries	Utilities
$Send(\lambda^t, M)$	By executing this query, $\mathcal{A}$ can send a request message M to λ_t and receive a response message of the message M.
$Execute(\lambda^t_{SC}, \lambda^v_{SSW})$	Using this query, $\mathcal{A}$ eavesdrops all the exchanged messages between communicating entities.
$CorruptSSW(\lambda^v_{SSW})$	Utilizing such a query, $\mathcal{A}$ is permitted to get the previously stored secret credentials of a physically compromised SSW.
$CorruptSC(\lambda^t_{SC})$	Utilizing such a query, $\mathcal{A}$ is permitted to get the previously stored secret credentials of a physically compromised SC.
$Reveal(\lambda^t)$	This query allows $\mathcal{A}$ to get a session key shared between λ^t and its respective partner.
$Test(\lambda^t)$	By executing this query, $\mathcal{A}$ can ask λ^t for verifying the "originality of a session key (SK)", and λ^t can provide a "random outcome of a flipped unbiased coin, say cn". More precisely, if $\mathcal{A}$ runs this query and SK is fresh, λ^t results SK when $cn = 1$ or a random output in the same domain when $c = 0$; otherwise, it outputs a null value $(\perp)$

accepted state, ii) both λ^t and λ^v mutually authenticates each other, and meanwhile share a common sid, and iii) both λ^t and λ^v are mutual partners of each other.

(d) Freshness. If the established session key between two players say, SC_i and SSW_j is not disclosed during the execution of the $Reveal(\lambda^t)$ query as mentioned in Table III then the instance say, λ^t_{SC} or λ^v_{SSW} is termed as a fresh instance.

(e) Adversary. The communications that are taking place between two intended players can be monitored and controlled by a PPT adversary $\mathcal{A}$. This player is having the capability of eavesdropping, modifying, updating, relaying, deleting, and delaying the communicating messages. Also, the adversary $\mathcal{A}$ can also have access to few other queries [52] as highlighted in Table III.

(f) Session-Key (SK) security. The semantic security (or, SK security) under ROR model defines the capability of a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ to distinguish an instance's "real session key" from a "random key". To pursue such an objective, $\mathcal{A}$ may run several $Test(\cdot)$ queries on either λ^t_{SC} or λ^v_{SSW}. Further, the same task may be repeat for either λ^u_{SA} or λ^t_{SC} instance also. Finally, at the end of the experiment, $\mathcal{A}$ guesses a bit cn', and he/she will achieve success in this experiment if $cn' = cn$. Suppose, $\mathcal{S}$ be an event which signifies the probability of winning the above experiment by the PPT adversary $\mathcal{A}$. Further, assume $\mathcal{A}$'s advantage in breaking the SK security of the proposed scheme $(PBAC-SDN)$ is $Adv^{AKE}_{PBAC-SDN} = |2 \cdot Pr[\mathcal{S}]-1|$. If $Adv^{AKE}_{PBAC-SDN} \leq \theta$, for a adequately small $\theta > 0$, then it is uttered that the proposed scheme $(PBAC-SDN)$ ensures SK-security (or semantic security).

(g) Random oracle. It is assumed that a cryptographic one-way hash function is available to all the players (or participants) including the adversary $\mathcal{A}$. Further, it is also assumed that the same function is modeled as a random oracle say, $\mathcal{H}$ [52].

Theorem 1 substantiates the SK-security of the proposed authenticated key exchange (AKE) scheme (PBAC-SDN).

Theorem 1. *Let $\mathcal{A}$ be a PPT adversary executing in polynomial time t against the proposed AKE scheme $PBAC-SDN$ in the ROR model; and q_h, $|\mathbb{H}|$, and $Adv^{ECDLP}_{\mathcal{A}}(t)$ represent the number of $Hash$ queries, the bit length of hash function, and the advantage of $\mathcal{A}$ in cracking $ECDLP$, then $Adv^{AKE}_{PBAC-SDN} \leq \frac{q_h^2}{|\mathbb{H}|} + 2Adv^{ECDLP}_{\mathcal{A}}(t)$.*

Proof: Initially, a series of three sequential games say, κ_j $(j = 0, 1, 2)$ is described as specified in [36], [41], [52], [53]. It may be noted that, the initial game is κ_0 and the end game is κ_2. Suppose, $\mathcal{S}_i$ represents an event in which the adversary $(\mathcal{A})$ tries to guesses the actual bit cn by running the *Test* query analogous to the game κ_j. The complete illustration of individual game is discussed below:

Game κ_0: This is an initial game in the experiment. This game "corresponds to a real attack" against $PBAC-SDN$ in the ROR model. At the starting of this game κ_0, the bit cn needs to be selected by the adversary $\mathcal{A}$. Under the ROR model, both κ_0 and the actual AKE protocol are said to be identical. Therefore, it follows that

$$Adv^{AKE}_{PBAC-SDN} \quad = \quad |2Pr[\mathcal{S}_0] - 1|. \tag{1}$$

Game κ_1: In this game the adversary $\mathcal{A}$ devices an eavesdropping attack by running the $Execute$ query. At the end of this experiment, $\mathcal{A}$ needs to run the $Test$ query. The result of the $Test$ query determines whether $\mathcal{A}$ avails the "real session key" or a "random key in the same domain". The session key between the controller (SC) and switch SSW (or vice versa), that is, $SK_{RO_m} (= SK_{O_m R})$ is computed as $SK_{Con,Swi} = h(f_{con-swi}(Con_{id}, Swi_{id}) \,||h(r_1 \,||TS_1) \,||h(r_2 \,||TS_2) \,||Cert_{con} \,||Cert_{swi}) = h(f_{con-swi}(Swi_{id}, Con_{id}) \,||h(r_1 \,||TS_1) \,||h(r_2 \,||TS_2) \,||Cert_{con} \,||Cert_{swi}) = SK_{Swi,Con}$. To figure out this session key, the adversary $\mathcal{A}$ needs to have the knowledge about few random secrets like r_1, r_2, $f_{con-swi}(Swi_{id}, Con_{id})$, s_{swi}, s_{RA}, and s_{con}. Since, these secret parameters are encapsulated indirectly into the messages msg_1 and msg_2, eavesdropping of such messages are not at all helpful in getting the session key $SK_{Con,Swi} (= SK_{Swi,Con})$. Therefore, by applying the eavesdropping attack the adversary $\mathcal{A}$'s advantage to win the game κ_1 is not at all increased. Thus, both the game κ_0 and κ_1 are indistinguishable. As a result, it then follows that

$$Pr[\mathcal{S}_1] \quad = \quad Pr[\mathcal{S}_0]. \tag{2}$$

Game κ_2: This game executes the $Hash$ and $Send$ queries, and transforms κ_1 to κ_2. Further, this game is modeled as an active attack wherein $\mathcal{A}$ intercepts all the communication

messages say, msg_j $(j = 1, \cdots, 3)$. It may be noted that, due to the usages of random timestamps and nonces in individual message, $\mathcal{A}$ needs to find collision in hash outputs (or message digests) while performing the $Send$ query. Moreover, suppose $\mathcal{A}$ tries to derive the session key $SK_{Con,Swi}(= SK_{Swi,Con})$ utilizing all the eavesdropped messages say, msg_1, msg_2, and msg_3 between SC_i and SSW_j, and other credentials obtained from the aforesaid games. Since the session key $SK_{Con,Swi}(= SK_{Swi,Con})$ is derived by both SC_i and SSW_j as $SK_{Con,Swi} = h(f_{con-swi}(Con_{id},\ Swi_{id})\ ||h(r_1\ ||TS_1)\ ||h(r_2\ ||TS_2)\ ||Cert_{con}\ ||Cert_{swi}) = h(f_{con-swi}(Swi_{id},\ Con_{id})\ ||h(r_1\ ||TS_1)\ ||h(r_2\ ||TS_2)\ ||Cert_{con}\ ||Cert_{swi}) = SK_{Swi,Con}$, therefore it is computationally hard for $\mathcal{A}$ to compute the session key "due to the difficulty of solving $ECDLP$". Now, in the absence of solving ECDLP and hash collision, the games κ_1 and κ_2 are indistinguisable. By applying the birthday paradox results the following output:

$$|Pr[\mathcal{S}_1] - Pr[\mathcal{S}_2]| \leq \frac{q_h^2}{2|\mathbb{H}|} + Adv_{\mathcal{A}}^{ECDLP}(t). \tag{3}$$

After successfully simulating all the queries in κ_2, $\mathcal{A}$ executes the $Test$ query for winning the game. In this connection, $\mathcal{A}$ guesses the bit cn. Then, it follows that,

$$Pr[\mathcal{S}_2] \quad = \quad \frac{1}{2}. \tag{4}$$

Considering both the equations Eqs. (1) and (2), it is followed that,

$$\begin{aligned} \frac{1}{2} \cdot Adv_{PBAC-SDN}^{AKE} &= |Pr[\mathcal{S}_0] - \frac{1}{2}| \\ &= |Pr[\mathcal{S}_1] - \frac{1}{2}|. \end{aligned} \tag{5}$$

Also, from Eqs. (4) and (5), it is followed that,

$$\begin{aligned} \frac{1}{2} \cdot Adv_{PBAC-SDN}^{AKE} &= |Pr[\mathcal{S}_1] - Pr[\mathcal{S}_2]| \\ &\leq \frac{q_h^2}{2|\mathbb{H}|} + Adv_{\mathcal{A}}^{ECDLP}(t). \end{aligned} \tag{6}$$

After multiplying both sides of Eq. (6) by a factor of 2, we obtain $Adv_{PBAC-SDN}^{AKE} \leq \frac{q_h^2}{|\mathbb{H}|} + 2Adv_{\mathcal{A}}^{ECDLP}(t)$. ■

B. Informal Security Analysis

This section provides a detailed informal security inspection of the proposed access control mechanism. To achieve the same we adopt the earlier discussed adversary model (refer, Section III-B) for better illustrations.

1) Replay Attack: In this attack, an adversary $\mathcal{A}$ can send old messages during the mutual authentication among the controller and a switch say, SC_i and SSW_j discussed in Section IV-C2, the messages $msg_1 = \{Con_{id},\ Swi_{id},\ Cert_{con},\ A_{con},\ B_{con},\ TS_1\}$ $msg_2 = \{Con_{id},\ Swi_{id},\ Cert_{swi},\ A_{swi},\ SKV_{Swi,Con},\ TS_2\}$, and $msg_3 = \{ACK_{Con,Swi}, TS_3\}$ via open channels. Since the random nonces (i.e., r_1, r_2) and timestamps (e.g., TS_1, TS_2, TS_3) are being tagged in each communicating message and the timeliness of individual message are being checked at the receiver-end hence it substantiates the "message-freshness" property. Therefore, the proposed PBAC-SDN is resilient against "replay attack".

2) Spoofing Attack: The spoofing attack can be possible when the adversary $\mathcal{A}$ impersonates a authorized node (for example a switch (SSW_q)) and tries to bypass the legitimate access control policy among SC_i and SSW_j [4] via its deployed malicious (SSW_p). Lets $\mathcal{A}$ commanding SSW_p spies over the all transmitted messages $msg_1 = \{Con_{id},\ Swi_{id},\ Cert_{con},\ A_{con},\ B_{con},\ TS_1\}$ and $msg_2 = \{Con_{id},\ Swi_{id},\ Cert_{swi},\ A_{swi},\ SKV_{Swi,Con},\ TS_2\}$. $\mathcal{A}$ may attempt to construct A_{swi} and $SKV_{Swi,Con}$ with the compromised information to impersonate SSW_j for a successive sessions. Therefore, to computes a valid A_{swi} and $SKV_{Swi,Con}$, $\mathcal{A}$ needs to know the values of $\{f_{con-swi}(Con_{id}, Swi_{id}),\ r_1,\ r_2\}$. Since the information $\{f_{con-swi}(Con_{id}, Swi_{id}),\ r_1,\ r_2\}$ are secret, hence, PBAC-SDN resists spoofing attack.

3) Denial-of-Service (DoS) Attack: In the proposed scheme, SSW builds up a secret session key with the controller (SC) by satisfying the mutual authentication policy during the access control phase. An unauthorized switch can not impersonate another legitimate switch by the spoofing attack as elaborated in Section V-B2. Moreover, any switch can access the data by verifying its authenticity, and after establishes a session key, which can reduce the random access to the real-time data. Also, an ABE policy that is running during a secure transmission among SA and SC helps to reduce the feasibility of random access or overloaded execution. Therefore, PBAC-SDN resists the DoS attack.

4) Man-in-the-Middle Attack: In this attack, the adversary $\mathcal{A}$ has a provision to hijack the communicated message msg_1 from the public channel during the access control request and try to generate another valid message $Msg_1^{'}$. Without knowledge of the credentials $f_{con-swi}(Con_{id}, Swi_{id})$ and r_1, $\mathcal{A}$ is unable to distinguishes the values from A_{con} and B_{con} or can not generate the values of A_{con} and B_{con}. In the same way, without understand of the secret information $f_{con-swi}(Swi_{id}, Con_{id})$ and r_2, $\mathcal{A}$ cannot construct the valid A_{swi} and $SKV_{Swi,Con}$ by capturing the message msg_2. Thus, PBAC-SDN is secure against man-in-the-middle attack.

5) Blockchain-based verification: In the blockchain verification phase, we adopted the wide-accepted voting-based consensus that is the PBFT algorithm. During the execution of this algorithm, if a verifier tries to verify the block, he/she needs to calculate the Merkle tree root MTR' for all the encrypted transactions in that block and current hash block $CBHash'$. Whenever these all are validated successfully, then only the verifier can proceed with the ECDSA signature verification. Once the signature is validated then the proposed block can be added to the chain otherwise the block can be rejected. Therefore, the block addition goes to the three-step verification process. Hence, it is a very difficult task for an adversary to update, delete, or modify a block in the blockchain due to the previous hash value which is injected in the current block as to modify that hash value is an impossible task.

6) Impersonation attack: In this attack, an adversary $\mathcal{A}$ behaves like a legitimate entity. During the access control process $\mathcal{A}$ can intercept the communicated messages $msg_1 = \{Con_{id},\ Swi_{id},\ Cert_{con},\ A_{con},\ B_{con},\ TS_1\}$, $msg_2 = \{Con_{id},\ Swi_{id},\ Cert_{swi},\ A_{swi},\ SKV_{Swi,Con},\ TS_2\}$, and

$msg_3 = \{ACK_{Con,Swi}, TS_3\}$. After that, $\mathcal{A}$ try to generate another valid message msg_1^* on behalf of the SC_i. To achieve this goal, $\mathcal{A}$ may pick a timestamp TS_1^*, and a random number r_1^* and derive $A_{con}^* = h(r_1^* \ ||TS_1^*) \oplus h(f_{con-swi}(Con_{id}, Swi_{id}) \ ||TS_1^*)$ and $B_{con}^* = h(Con_{id} \ ||Swi_{id} \ ||A_{con}^* \ ||TS_1^* \ ||Cert_{con} \ ||f_{con-swi}(Con_{id}, Swi_{id}))$. Without knowledge of secret bivariate polynomial $f(x, y)$, $\mathcal{A}$ cannot derive such values. Therefore, $\mathcal{A}$ cannot construct another valid message msg_1^* on behalf of the SC_i. Hence, the proposed scheme is secure against the impersonation attack.

7) Privileged-insider attack: In the entity registration phase in Section IV-B, a trusted third party RA register all the entities namely, SA, SC, and SSW prior to their functioning. After the registration process is over, RA deletes all the secret credentials for those entities from its memory. Therefore, any privileged-insider of RA treated as an adversary cannot access any secret loaded information for its database. Thus, the proposed scheme is secure against the privileged-insider attack.

C. Formal Security Simulation Under AVISPA Tool

This section yields the formal security verification of the proposed scheme PBAC-SDN by adopting the widely-used automated software-based security protocol simulator, termed as "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [12]. It may be noted that, from the last two decades, automated software-based formal security verification tool has attained a vast approbation among the researchers and security practitioners. Besides AVISPA [12], various other related formal security analyzers are also found in the literature [54].

AVISPA [12] is advocated as a "push-button tool for the automated validation of Internet security protocols as well as applications" and it provides a "modular as well as expressive formal language for specifying protocols along with their security properties". This tool integrates different "back-ends" that "execute a heterogeneity of state-of-the-art automatic analysis methods" [39]. Since its inception, "On-the-fly model-checker (OFMC)", "Constraint-logic-based Attack Searcher (CL-AtSe)", "SAT-based Model Checker (SATMC)" and "Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)" are the four different back-ends presently available with the AVISPA tool. The detailed description of these back-ends can be availed from [12]. The standard procedure for testing a newly proposed authenticated key exchange (AKE) protocol under AVISPA is stepwise illustrated as follows: (i) initially, the proposed scheme is required to be codified using the "High-Level Protocol Specification Language (HLPSL)", (ii) the written HLPSL code is then executed using the specified command in the said platform, (iii) a built-in HLPSL2IF translator under the said tool is then converted the code into "Intermediate Format (IF)", (iv) finally, the generated IF is provided into one of the aforesaid back-ends to get the "Output Format (OF)" or the attack traces of the AKE protocol. The final result or the OF consists of the following key modules:

- **SUMMARY** module tells that "whether the tested protocol is safe, unsafe, or the analysis is inconclusive".
- **DETAILS** module "either explains under what condition the tested protocol is declared safe, or what conditions have been used for finding an attack or finally why the analysis is inconclusive".
- **PROTOCOL**, **GOAL** and **BACKEND** modules are "the name of the protocol, the goal of the analysis and the name of the back-end used", respectively.
- Nonetheless, "trace of an attack (if any)" is also exhibited in "the standard Alice-Bob format after some comments along with statistics".

We have implemented the proposed scheme for the access control part using the HLPSL language. For this purpose, we have defined the basic roles for the registration authority (RA), a controller node (SC_i), and a switch (SSW_j). Apart from the basic roles, the mandatory roles for the session and goal & environment. The formal security verification of the proposed scheme has been simulated under the "SPAN, the Security Protocol ANimator for AVISPA" [55], and the results are demonstrated in Fig. 6. We have chosen the widely-accepted OFMC and CL-AtSe backends for simulation purposes. The other two backends (SATMC and TA4SP) are omitted because they do not presently support bitwise XOR operation implementation. Through such an analysis it is evident that the proposed scheme is secure against passive and active attacks, namely the replay and man-in-the-middle attacks.

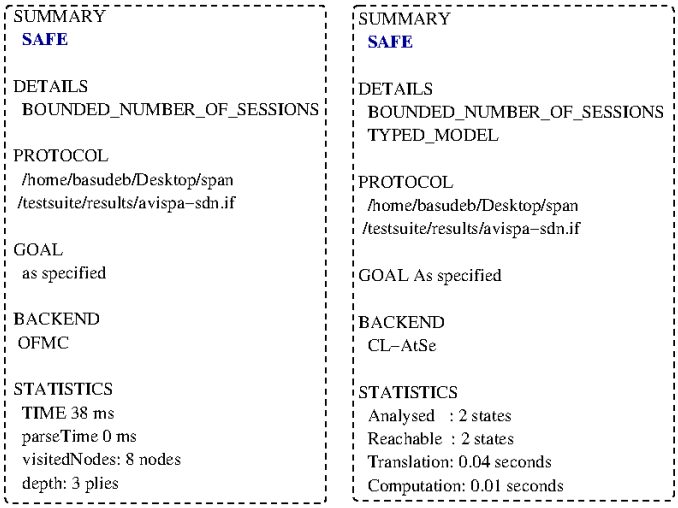

Fig. 6. The simulation results under OFMC and CL-AtSe backends

VI. MIRACL-Based Testbed Experiments

In this section, we provide a real-time experimental result for the execution time of various cryptographic primitives. The entire testbed has been executed over the two platforms, where for each scenario the every cryptographic primitive has been executed up to 100 times, and finally, we take the average running cost (run time) for such primitives based on these 100 trials.

- **Platform 1:** Utilizing MIRACL, the cryptographic primitives have been executed for a controller/server side in this platform under the system configuration "Ubuntu 18.04.4 LTS, with 7.7 GiB memory, Intel® Core™

processor- 8565U, CPU @ 1.80GHz×8, 64-bit OS type and disk size 966.1 GB". The performance results are shown in Table IV.

TABLE IV
EXECUTION TIME (IN MILLISECONDS) UNDER A SERVER

Primitive	Max. time (ms)	Min. time (ms)	Average time (ms)
T_h	0.149	0.024	0.055
T_{mtp}	0.199	0.092	0.114
T_{senc}	0.008	0.002	0.003
T_{sdec}	0.005	0.002	0.003
T_{ecm}	2.998	0.284	0.674
T_{eca}	0.002	0.001	0.002
T_{bp}	7.951	4.495	4.716
T_{mul}	0.035	0.002	0.004
T_{add}	0.004	0.001	0.002

- **Platform 2:** The cryptographic primitives have been executed using MIRACL for the smart devices/switches side by Raspberry PI 3 and the system configuration considered as follows: "Raspberry PI 3 B+ Rev 1.3, Ubuntu 20.04 LTS, 64- bit OS, 1.4 GHz Quad-core processor, cores 4, 1 GB RAM". The execution results are also displayed in Table V.

TABLE V
EXECUTION TIME (IN MILLISECONDS) UNDER RASPBERRY PI 3

Primitive	Max. time (ms)	Min. time (ms)	Average time (ms)
T_h	0.643	0.274	0.309
T_{mtp}	0.406	0.381	0.385
T_{senc}	0.038	0.017	0.018
T_{sdec}	0.054	0.009	0.014
T_{ecm}	4.532	2.206	2.288
T_{eca}	0.021	0.015	0.016
T_{bp}	32.79	27.606	32.084
T_{mul}	0.016	0.009	0.011
T_{add}	0.013	0.008	0.01

VII. COMPARATIVE ANALYSIS

In this phase, we elaborate a execution analysis on communication cost, computation cost of the access control protocol among a controller SC_i and a switch SSW_j which is mentioned in Fig. 2 in the PBAC-SDN and security and functionality features among the proposed scheme PBAC-SDN and Ali *et al.* [16], Iqbal et al. [14] and Ever [15].

A. Computation overheads

We considered that T_h, T_{senc}/T_{sdec}, T_{bp}, T_{fe}, T_{eca}, T_{ecm}, and T_{mtp} are represent the time required for the execution of the cryptographic primitives "one-way cryptographic hash function", a "symmetric key encryption/decryption (AES-128)", a "bilinear pairing", a "fuzzy extractor", an "elliptic curve point addition", an "elliptic curve point multiplication", and a "map to point", respectively. It is assumed that T_{fe} ($\approx T_{ecm}$) [56].

Here the time means the average time required for executing the cryptographic primitives in our real time experimental testbed results utilizing MIRACL in Section VI. The experimental outcomes shown in Table IV consider for a server/GSS/controller side, and for the other side the time needed for the IoT smart device or swithch, the same experiment happened under the Raspberry PI 3 setting given in the Table V. Now the computation cost for the access control phase required $8T_h + 2T_{ecm} + T_{eca} + n(T_{mul} + T_{add})$ $\approx 7.064 + 0.021n$ ms for the switch side and $7T_h + 2T_{ecm} + T_{eca} + n(T_{mul} + T_{add}) \approx 1.735 + 0.006n$ ms for the controller side, where n represent the degree of the bivariate polynomial and $T_{poly} = n(T_{mul} + T_{add})$. From the Table VI, it is observed that the computational cost for the PBAC-SDN needed less computation cost as compared to Ali *et al.* [16], and Ever [15].

TABLE VI
COMPARATIVE STUDY ON COMPUTATION COSTS

Scheme	Smart device/Swithch	GSS/Server/Controller
Iqbal *et al.* [14]	$4T_h$ ≈ 1.236 ms	$4T_h + 2T_{senc}/T_{sdec}$ ≈ 0.224 ms
Ali *et al.* [16]	$18T_h + T_{fe} + T_{senc}$ ≈ 7.868 ms	$7T_h + 3T_{senc}/T_{sdec}$ ≈ 0.394 ms
Ever [15]	$9T_h + 2T_{bp} +$ $2T_{mtp} + 3T_{ecm}$ ≈ 74.583 ms	$6T_h + 3T_{bp} +$ $2T_{mtp} + 3T_{ecm}$ ≈ 16.728 ms
PBAC-SDN	$8T_h + 2T_{ecm} + T_{eca}$ $+n(T_{mul} + T_{add})$ $\approx 7.064 + 0.021n$ ms	$7T_h + 2T_{ecm} + T_{eca}$ $+n(T_{mul} + T_{add})$ $\approx 1.735 + 0.006n$ ms

B. Communication overheads

In the proposed scheme PBAC-SDN, we consider for the "identity", "random number", hash output (using SHA-256 hashing algorithm), "elliptic curve point" $P = (P_x, P_y) \in E_q(u, v)$ where x and y coordinates of P are P_x and P_y respectively, and timestamp are 160, 160, 256, 320, and 32 bits, respectively. Therefore, the communication cost for the exchanged messages $msg_1 = \{Con_{id}, Swi_{id}, Cert_{con}, A_{con}, B_{con}, TS_1\}$, $msg_2 = \{Con_{id}, Swi_{id}, Cert_{swi}, A_{swi}, SKV_{Swi,Con}, TS_2\}$, and $msg_3 = \{ACK_{Con,Swi}, TS_3\}$ are (160 + 160 + 160 + 256 + 256 + 32) = 1024 bits, (160 + 160 + 160 + 256 + 256 + 32) = 1024 bits, and (256 + 32) = 288 bits, respectively, and hence altogether needs 2336 bits, which is given in the Table VII. It is worth noticing that the PBAC-SDN is required less communication cost in term of number of bits during the access control phase as compare to the others schemes Iqbal *et al.* [14], Ali *et al.* [16] and Ever [15].

TABLE VII
COMPARATIVE STUDY ON COMMUNICATION COSTS

Scheme	No. of messages	Total cost (in bits)
Iqbal *et al.* [14]	4	2848
Ali *et al.* [16]	3	3424
Ever [15]	6	5344
PBAC-SDN	3	2336

C. Security and Functionality Features

This section provides a comparative assessment over the several security and functionality features ($FSA_1 - FSA_{16}$) for the proposed scheme PBAC-SDN along with the existing schemes and shown in the Table VIII. The Table VIII reveals that the existing schemes Iqbal *et al.* [14], Ali *et al.* [16] and Ever [15] does not support FSA_9, FSA_{10}, and FSA_{12} where as the PBAC-SDN meets all the functionality attributes.

TABLE VIII
COMPARATIVE STUDY ON FUNCTIONALITY & SECURITY ATTRIBUTES

Attribute	Iqbal *et al.* [14]	Ali *et al.* [16]	Ever [15]	PBAC-SDN
FSA_1	✓	✓	✓	✓
FSA_2	✓	✓	✓	✓
FSA_3	✓	✓	✓	✓
FSA_4	×	✓	✓	✓
FSA_5	✓	✓	✓	✓
FSA_6	✓	✓	✓	✓
FSA_7	✓	✓	✓	✓
FSA_8	✓	✓	✓	✓
FSA_9	×	×	×	✓
FSA_{10}	×	×	×	✓
FSA_{11}	×	✓	×	✓
FSA_{12}	×	×	×	✓
FSA_{13}	×	✓	×	✓
FSA_{14}	✓	×	✓	✓
FSA_{15}	×	×	✓	✓
FSA_{16}	✓	×	×	✓

FSA_1: "replay attack"; FSA_2: "man-in-the-middle attack"; FSA_3: "mutual authentication"; FSA_4: "key agreement"; FSA_5: "device/drone impersonation attack"; FSA_6: "GSS/server impersonation attack"; FSA_7: "malicious device deployment attack"; FSA_8: "resilience against drone/device physical capture attack"; FSA_9: "formal security verification using AVISPA tool"; FSA_{10}: "ESL attack under the CK-adversary model"; FSA_{11}: "support dynamic drone/device addition phase"; FSA_{12}: "support blockchain-based solution"; FSA_{13}: "support formal security analysis under ROR model"; FSA_{14}: "resilience against IND-CPA security model"; FSA_{15}: "free from design flaws"; FSA_{16}: "anonymity and untraceability".
✓: "a scheme is secure or it supports an attribute"; ×: "a scheme is insecure or it does not support an attribute"; N/A: not applicable.

VIII. BLOCKCHAIN IMPLEMENTATION: RESULTS AND DISCUSSIONS

In this section, we have implemented the PBFT consensus algorithm for block addition into a blockchain under the algorithm defined in Algorithm 1 to estimate its impact on the performance parameters. The experiments have been done over a platform having "Ubuntu 18.04, 64-bit OS with Intel® Core™ i5-4210U CPU @ 1.70GHz, 4 GB RAM" under Node Js language with VSCODE 2019. We assume that in the P2P cloud server network, the number of cloud server nodes is 13 and the block structure is contemplated the same as mentioned in Fig. 5, which forms a connected complete network.

The size of a block mentioned in Fig.5 is calculated under the following assumptions. The block version, previous block hash, Merkle tree root, timestamp (epoch time), owner of the block, public key of the signer (ECC based public key), current block hash (using SHA-256 hashing algorithm), and ECDSA signature are taken as 32, 256, 256, 43, 160, 320, 256, and 320 bits, respectively. Moreover, each transaction Tx_i was encrypted using ECC encryption which outputs two elliptic curve points, and as a result, an encrypted transaction requires (320+320) = 640 bits. Therefore, the total block size for a block $Block_m$ becomes $(32+\ 256+\ 256+\ 42+\ 160+\ 320+\ 320+\ 256+\ 320+\ 640t) = (1962+\ 640t)$ bits.

Table IX lists the performance measures relevant to the voting-based PBFT consensus algorithm. The total number of messages required in Algorithm 1 is $O(n^2)$, where n is the total number of P2P cloud server nodes.

TABLE IX
PERFORMANCE MEASUREMENTS OF PBFT

Characteristics	Consensus algorithm (PBFT)
Byzantine fault tolerance	33%
Crash fault tolerance	33%
Verification speed (transactions per millisecond)	70-80 ms
Message complexity	$O(n^2)$

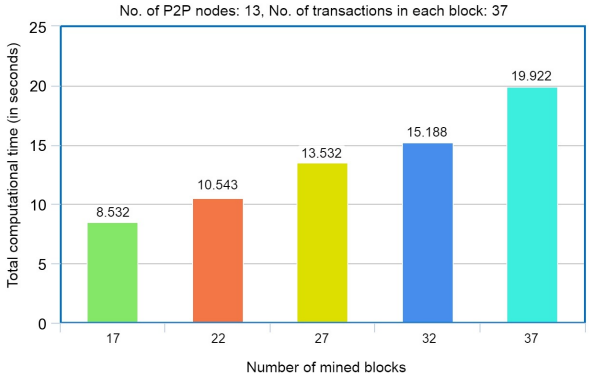

Fig. 7. The simulation results of blockchain implementation for Scenario 1

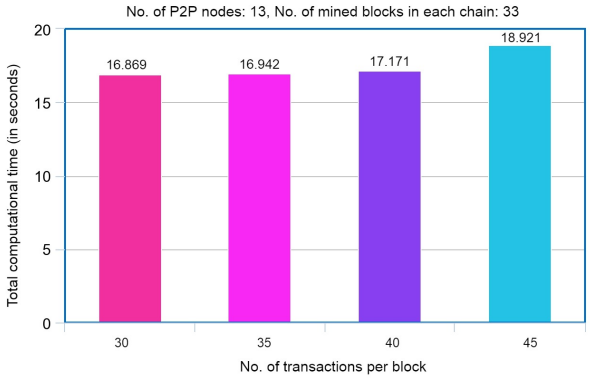

Fig. 8. The simulation results of blockchain implementation for Scenario 2

The simulation has been carried out over two scenarios as follows:

- **Scenario 1:** In this case, we measure the blockchain implementation performance in a distributed manner by considering the fixed number of transactions in each block as 37. The number of blocks mined at a different time in the blockchain is varied. It can be verified that when the number of mined blocks increases, the total computation time (in seconds) also linearly increases, which is evident from Fig. 7.
- **Scenario 2:** In this case, we fix the mined blocks for each chain as 33 and compute the execution time for various

transactions. The simulation results are provided in Fig. 8. The experimental results signify that the number of transactions is stored in a block for fixed chain length varies with the total computational time (in seconds) or execution time (in seconds) linearly.

Remark 3. *In the blockchain simulation study, we have considered two types of computational time (specifically, the execution time or the CPU time), that is, (i) Scenario 1 - the execution time requires for mining a varied (i.e., not fixed) number of blocks (i.e., 17, 22, 27, 32, and 37, respectively) into the blockchain wherein each block consists of a fixed number of transactions that is 37), and (ii) Scenario 2- the execution time requires for mining a fixed number of blocks (i.e., 33) wherein each block consists of a varied number of transactions (i.e., 30, 35, 40, and 45) into the blockchain. According to the aforesaid consensus protocol (i.e., PBFT), the estimated (average) computation time considering both the scenarios could be expressed as the summation of the following factors: (i) the execution time required to build a socket connection between two peer-to-peer server nodes, (ii) time utilization for different types of message's (i.e., "TRANSACTION", "PREPARE", "PRE-PREPARE", "COMMIT", "ROUND CHANGE") generation and broadcasting, (iii) the construction time required to build transaction pool, wallet (public and private key generation), block pool, prepare pool, commit pool and message pools, (iv) the verification time is needed to check the redundant messages, transactions, and blocks, and (v) the time spent for adding the message and the block into the respective message pool and block pool, and updation of new blocks into the blockchain, respectively. Presently, the blockchain experiments and simulations have been carried out in this study under the "Node Js" and "Python" programming languages with VSCODE 2019, but in the practical scenario with a sophisticated computing and memory capacity may reduce the computational time (i.e., 20-30 seconds of confirmation delay as shown in Fig. 7 and Fig. 8, respectively). It is worth noticing that we have not considered the link delay, Byzantine ratio, and other network characteristics of the simulated P2P network in the total computation time calculation. We will try to consider these factors in the near future to fine tune our proposed approach with more real-world IoT applications.*

IX. Conclusion

We proposed a new blockchain-based access control protocol (PBAC-SDN) for securing the SDN platform. In PBAC-SDN, the access control process among an application and a controller is executed with the help of existing ABE, where the access control among a controller and a switch is executed with the help of our newly designed access control mechanism. Transactions among various controllers, applications, and switches are put into the partial blocks which are created by the controller nodes. The newly created blocks derived from the partial blocks are then inserted into the blockchain by a leader selected from the available P2P cloud server (CS) nodes with the help of the designed voting-based PBFT consensus method. A detailed formal, informal, and formal security verification reveals that the proposed PBAC-SDN is robust against several potential attacks that are required to secure an SDN environment. After that testbed experiments using MIRACL and blockchain-based implementation have been carried out. Finally, a detailed comparative analysis shows the better performance of the proposed PBAC-SDN as compared to other competing schemes.

Acknowledgment

The authors would like to thank the anonymous reviewers and the associate editor for their valuable suggestions and feedbacks on the paper, which helped us to improve its technical quality and presentation.

References

[1] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2014.

[2] O. I. Abdullaziz, L. Wang, and Y. Chen, "HiAuth: Hidden Authentication for Protecting Software Defined Networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 618–631, 2019.

[3] D. Chattaraj, S. Saha, B. Bera, and A. K. Das, "On the Design of Blockchain-Based Access Control Scheme for Software Defined Networks," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, 2020, pp. 237–242.

[4] W. Jiasi, W. Jian, L. Jia-Nan, and Z. Yue, "Secure software-defined networking based on blockchain," *arXiv preprint arXiv:1906.04342*, pp. 1–19, 2019.

[5] M. Bonola, G. Bianchi, G. Picierro, S. Pontarelli, and M. Monaci, "Streamon: A data-plane programming abstraction for software-defined stream monitoring," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 664–678, 2017.

[6] A. Zaalouk, R. Khondoker, R. Marx, and K. M. Bayarou, "OrchSec: An orchestrator-based architecture for enhancing network-security using Network Monitoring and SDN Control functions," in *Proceedings of the Network Operations and Management Symposium (NOMS'14)*, Krakow, Poland, 2014, pp. 1–9.

[7] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *Proceedings of the Network Operations and Management Symposium (NOMS'14)*. Krakow, Poland: IEEE, 2014, pp. 1–4.

[8] S. Matsumoto, S. Hitz, and A. Perrig, "Fleet: Defending SDNs from Malicious Administrators," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. Chicago, Illinois, USA: ACM, 2014, pp. 103–108.

[9] R. Chaudhary, A. Jindal, G. S. Aujla, S. Aggarwal, N. Kumar, and K.-K. R. Choo, "BEST: Blockchain-based secure energy trading in SDN-enabled intelligent transportation system," *Computers & Security*, vol. 85, pp. 288 – 299, 2019.

[10] A. Jindal, G. S. Aujla, and N. Kumar, "SURVIVOR: A blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment," *Computer Networks*, vol. 153, pp. 36 – 48, 2019.

[11] M. Abdalla, P. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science*, vol. 3386, Les Diablerets, Switzerland, 2005, pp. 65–84.

[12] AVISPA, "Automated Validation of Internet Security Protocols and Applications," 2019, http://www.avispa-project.org/. Accessed on October 2019.

[13] "MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library," 2020, Accessed on January 2021. [Online]. Available: https://github.com/miracl/MIRACL

[14] W. Iqbal, H. Abbas, P. Deng, J. Wan, B. Rauf, Y. Abbas, and I. Rashid, "ALAM: Anonymous Lightweight Authentication Mechanism for SDN Enabled Smart Homes," *IEEE Internet of Things Journal*, 2020, DOI: 10.1109/JIOT.2020.3024058.

[15] Y. K. Ever, "A secure authentication scheme framework for mobile-sinks used in the internet of drones applications," *Computer Communications*, vol. 155, pp. 143 – 149, 2020.

[16] Z. Ali, S. A. Chaudhry, M. S. Ramzan, and F. Al-Turjman, "Securing Smart City Surveillance: A Lightweight Authentication Mechanism for Unmanned Vehicles," *IEEE Access*, vol. 8, pp. 43 711–43 724, 2020.

[17] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues, "TCALAS: Temporal Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones Environment," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6903–6916, 2019.

[18] E. Tantar, M. R. Palattella, T. Avanesov, M. Kantor, and T. Engel, "Cognition: A tool for reinforcing security in software defined networks," in *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*. Cham: Springer, 2014, pp. 61–78.

[19] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking (HotSDN'14)*. Chicago, Illinois, USA: ACM, 2014, pp. 1–6.

[20] S. Challa, A. K. Das, P. Gope, N. Kumar, F. Wu, and A. V. Vasilakos, "Design and analysis of authenticated key agreement scheme in cloud-assisted cyber-physical systems," *Future Generation Computer Systems*, vol. 108, pp. 1267–1286, 2020.

[21] A. K. Das, A. K. Sutrala, S. Kumari, V. Odelu, M. Wazid, and X. Li, "An efficient multi-gateway-based three-factor user authentication and key agreement scheme in hierarchical wireless sensor networks," *Security and Communication Networks*, vol. 9, no. 13, pp. 2070–2092, 2016.

[22] S. Chatterjee, A. K. Das, and J. K. Sing, "An Enhanced Access Control Scheme in Wireless Sensor Networks," *Ad Hoc Sens. Wirel. Networks*, vol. 21, no. 1-2, pp. 121–149, 2014.

[23] B. Bera, A. K. Das, W. Balzano, and C. M. Medaglia, "On the design of biometric-based user authentication protocol in smart city environment," *Pattern Recognition Letters*, vol. 138, pp. 439 – 446, 2020.

[24] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos, "Designing Blockchain-Based Access Control Protocol in IoT-Enabled Smart-Grid System," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5744–5761, 2021.

[25] A. Vangala, B. Bera, S. Saha, A. K. Das, N. Kumar, and Y. H. Park, "Blockchain-Enabled Certificate-Based Authentication for Vehicle Accident Detection and Notification in Intelligent Transportation Systems," *IEEE Sensors Journal*, 2020, DOI: 10.1109/JSEN.2020.3009382.

[26] H. Wang, L. Xu, and G. Gu, "Floodguard8: A dos attack prevention extension in software-defined networks," in *Proceedings of the forty fifth Annual International Conference on Dependable Systems and Networks*. Rio de Janeiro, Brazil: IEEE, 2015, pp. 239–250.

[27] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building Robust Firewalls for Software-defined Networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking (HotSDN'14)*. Chicago, Illinois, USA: ACM, 2014, pp. 97–102.

[28] R. Izard, "Floodlight Controller," https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview, 2016, online; accessed 25th December, 2019.

[29] M. Koerner and O. Kao, "Oftables: A distributed packet filter," in *Proceedings of the Sixth International Conference on Communication Systems and Networks (COMSNETS'14)*. Bangalore, India: IEEE, 2014, pp. 1–4.

[30] M. Suh, S. H. Park, B. Lee, and S. Yang, "Building firewall over the software-defined network controller," in *Proceedings of the sixteenth International Conference on Advanced Communication Technology*. Pyeongchang, South Korea: IEEE, 2014, pp. 744–748.

[31] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing network security through software defined networking (SDN)," in *Proceedings of the twenty fifth international conference on computer communication and networks (ICCCN'16)*. Waikoloa, HI, USA: IEEE, 2016, pp. 1–9.

[32] M. Luo, Y. Luo, Y. Wan, and Z. Wang, "Secure and efficient access control scheme for wireless sensor networks in the cross-domain context of the IoT," *Security and Communication Networks*, vol. 2018, pp. 1–10, 2018. [Online]. Available: https://doi.org/10.1155/2018/6140978

[33] F. Li, Y. Han, and C. Jin, "Practical access control for sensor networks in the context of the Internet of Things," *Computer Communications*, vol. 89-90, pp. 154–164, 2016.

[34] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[35] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[36] D. Chattaraj, M. Sarma, and A. K. Das, "A new two-server authentication and key agreement protocol for accessing secure cloud services," *Computer Networks*, vol. 131, pp. 144 – 164, 2018.

[37] R. Canetti and H. Krawczyk, "Universally Composable Notions of Key Exchange and Secure Channels," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*, Amsterdam, The Netherlands, 2002, pp. 337–351.

[38] B. Bera, S. Saha, A. K. Das, N. Kumar, P. Lorenz, and M. Alazab, "Blockchain-Envisioned Secure Data Delivery and Collection Scheme for 5G-Based IoT-Enabled Internet of Drones Environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9097–9111, 2020.

[39] B. Bera, D. Chattaraj, and A. K. Das, "Designing secure blockchain-based access control scheme in IoT-enabled Internet of Drones deployment," *Computer Communications*, vol. 153, pp. 229 – 249, 2020.

[40] S. Mandal, B. Bera, A. K. Sutrala, A. K. Das, K. R. Choo, and Y. Park, "CertificatelessSigncryptionBased ThreeFactor User Access Control Scheme for IoT Environment," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3184–3197, 2020.

[41] D. Chattaraj, M. Sarma, A. K. Das, N. Kumar, J. J. P. C. Rodrigues, and Y. Park, "HEAP: An Efficient and Fault-Tolerant Authentication and Key Exchange Protocol for Hadoop-Assisted Big Data Platform," *IEEE Access*, vol. 6, pp. 75 342–75 382, 2018.

[42] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, 2002.

[43] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, 2002.

[44] W. E. May, "Secure Hash Standard," 2015, FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995. Available at http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. Accessed on August 2019.

[45] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," in *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS'06)*, Alexandria, VA, USA, 2006, pp. 89–98.

[46] V. Odelu, A. K. Das, Y. S. Rao, S. Kumari, M. K. Khan, and K.-K. R. Choo, "Pairing-based CP-ABE with constant-size ciphertexts and secret keys for cloud environment," *Computer Standards & Interfaces*, vol. 54, pp. 3–9, 2017.

[47] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences," *Information and Computation*, vol. 146, no. 1, pp. 1–23, 1998.

[48] D. Johnson, A. Menezes, and S. A. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[49] H. Zhang, J. Wang, and Y. Ding, "Blockchain-based decentralized and secure keyless signature scheme for smart grid," *Energy*, vol. 180, pp. 955–967, 2019.

[50] S. Saha, D. Chattaraj, B. Bera, and A. Kumar Das, "Consortium blockchain-enabled access control mechanism in edge computing based generic Internet of Things environment," *Transactions on Emerging Telecommunications Technologies*, p. e3995, 2020.

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

[52] C. C. Chang and H. D. Le, "A Provably Secure, Efficient and Flexible Authentication Scheme for Ad hoc Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 357–366, 2016.

[53] S. Chatterjee, S. Roy, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos, "Secure biometric-based authentication scheme using Chebyshev chaotic map for multi-server environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 824–839, 2016.

[54] B. Blanchet, "Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif," *Foundations and Trends in Privacy and Security*, vol. 1, no. 1-2, pp. 1–135, 2016.

[55] AVISPA, "SPAN, the Security Protocol ANimator for AVISPA," 2021, http://www.avispa-project.org/. Accessed on January 2021.

[56] D. He, S. Zeadally, B. Xu, and X. Huang, "An Efficient Identity-Based Conditional Privacy-Preserving Authentication Scheme for Vehicular Ad Hoc Networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.