2021 IEEE 7th International Conference on Network Softwarization (NetSoft) | 978-1-6654-0522-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/NetSoft51509.2021.9492615

Blockchain-based Secure Coordination for Distributed SDN Control Plane

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

***Abstract*—Software-defined wide-area network (SD-WAN) is an emerging and advanced networking platform extending software-defined networking (SDN) across multiple networking domains. Because SD-WAN manages the data plane in the networking domains separated by the public Internet, SD-WAN provides a distinct environment and challenges from SDN, including greater risks for the security threats injecting control plane communications from attackers residing outside of the SDN domain. We design and build blockchain-coordinating controllers (BCC) to secure control communications of the SD-WAN controller network formed by the distributed controllers spread across multiple domains. BCC provides resiliency against the security threats in the control plane where an attacker compromises controller communications to manipulate the coordination and the operations of the other controllers. More specifically, BCC provides secure control communications even when up to n controllers' networking credentials are compromised. BCC is also designed for modularity so that it applies generally across the controller implementations. We prototype BCC using Ethereum and smart contract on CloudLab to validate its effectiveness and efficiency. We experiment on geographically separate nodes on CloudLab and show that BCC achieves the distributed consensus at sub-second level for certificate/key distribution and for network-wide control communication synchronization.**

***Index Terms*—Blockchain, Smart Contract, SDN, Distributed PKI, Control Communication, SD-WAN, Ethereum**

I. Introduction

While the initial SDN designs and prototypes focused on one controller so that the intelligence and the management are centralized, more recent SDN designs introduce multiple SDN controllers especially in the following two contexts. First, to prevent a single point of failure, such as a controller being unavailable or unreliable, multiple controllers can be used for fault-tolerant replicas [1]–[3]. Second, to use a distributed network of multiple SDN controllers to manage a data plane across multiple networking domains [4], [5]. Furthermore, the most recently emerged software-defined wide area network (SD-WAN) technology [6] has attracted extensive attention, which aims at long-distance data transmission across multiple domains, and SD-WAN has been regarded as the promising architecture of next generation wide area network. Canales et al. [7] stated that more than 90% of WAN edge infrastructure refresh initiatives will be based on SD-WAN software by 2023. However, current SD-WAN architecture [8] still relies on one centralized SD-WAN controller to conduct distributed SD-WAN edges for WAN connections (see Fig. 1).

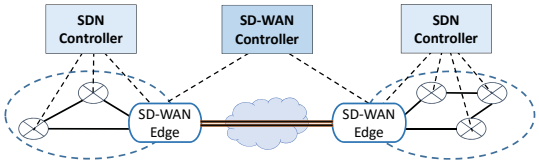

Fig. 1. A brief overview of SDN and SD-WAN model (including the overlay network virtual link between the edges)

In this paper, we are motivated to decentralize the SDN/SD-WAN control plane and build secure coordination/synchronization between the controllers to address the security issues in the existing solutions (e.g., SDNi [9]). This paper tackles two vital but unsolved security problems in the context of SDN/SD-WAN. First, a key needs to be established to provide the root of trust for authenticity to the controllers for the control communication. Public-key infrastructure (PKI) that relies on the certificate authority (CA) is a well-known technique in networking security to build the root of trust for networking. However, the reliance on the centralized authorities leaves a single point of failure, as has been demonstrated in real-world attacks, including the security breach of DigiNotar [10] and Foxconn [11]. Second, the integrity of the updates of the SDN-controller network needs to be assured. The control communication between SDN controllers should have cryptography-based integrity protection. In the advanced SD-WAN cases, the controller network resides in multiple domains and has networking packets traversing the public Internet, the attacker has greater feasibility to compromise the control-plane networking.

We use blockchain to build greater resiliency against such insider compromise and to address the aforementioned security issues. Our scheme includes two separate smart contracts for the distinct purposes. First, the authorization smart contract (Auth-SC) establishes the trust and the keys in the forms of certificates so that a certificate not only binds the keys to the controller entities (the standard use of digital certificates) but also acts as an authorization for participating in the SDN control communications. Second, the coordination smart contact (Coord-SC) coordinates between the SDN controllers and updates the controller parameters across domains, given the keys and the corresponding authorization established from

Auth-SC. Both smart contracts require a distributed voting-based consensus between the participant controllers. The blockchain are permissioned as we control and limit the participation to the control plane. The controllers can authorize their peer controllers via Auth-SC, and can reach a network-wide agreement for the control communication via Coord-SC.

The contributions of this paper are summarized as follows:

- We propose a blockchain-based secure coordination between SDN controllers (BCC) scheme addressing the insider compromise issue.
- We design an authorization smart contract (Auth-SC) based distributed PKI for trust establishment between the controllers and a coordination smart contract (Coord-SC) for updating controller parameters across multiple domains.
- We prototype our BCC on CloudLab and evaluate its performance built on the implementation.

II. BCC Design

In this section, we propose the BCC architecture and its blockchain scheme.

A. BCC Architecture

An overview of the architectural design of blockchain-enabled secure coordination amongst controllers (BCC) is presented in Fig. 2. The architecture consists of three planes: certificate plane, control plane, and data plane.

a) ***The certificate plane****:* The certificate plane is used for certificate management. Essentially, it includes the blockchain-based distributed PKI facilitated by the Auth-SC to authorize and revoke the digital certificates of each controller in order to establish trust mechanism for the controller network.

b) ***The control plane****:* The control plane is used for control communication using the key given by the certificate plane to configure the data-communication parameter setup across SDN networks. It includes the controller counterparts, which have both the SDN controller and SD-WAN controller's functionality. Every controller works for its own domain, but they are synchronized/coordinated to provide higher reliability. In other words, any controller can work for any other domain when it is needed, e.g., when the controller counterpart of the other domain tears down due to some availability problem. Regarding the synchronization/coordination, one controller counterpart need to write the parameter update on the Coord-SC, and once the update becomes viable after the distributed consensus, the other controller counterparts can read it from the blockchain ledger.

c) ***The data plane****:* The data plane is used for data exchange in terms of the data-communication parameter setup given by the control plane. It consists of both SDN switch/router and SD-WAN edge. They install the flow policies specified by the control plane. The switches and edges commonly obey the controller's commands of their domain, however, they can also follow the commands of the controller of another domain when theirs broken down.

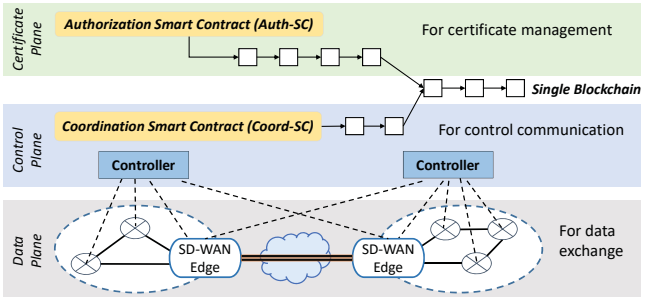

Fig. 2. An overview of BCC architecture design

B. Design Rationale: Two Smart Contracts and One Chain

In this paper, BCC focuses on the case of multiple domains using distributed control plane where the controller counterparts are used for coordination, in contrast to the one domain case where the controller replicas are used for reliability. We design our scheme to defend against the additional vulnerabilities arising from extending the SDN control management across networking domains.

Because BCC supports the logical functionalities across the two logical planes (the certificate plane and the control plane), we implement separate but dependent smart contracts for verification and processing. While the voting-based consensus protocol remains the same across the two smart contracts, they provide distinct logical operations. For example, the smart contract for the control coordination checks and depends on the authorization/membership provided from the certificate smart contract. We discuss about the two distinct smart contracts (Auth-SC and Coord-SC) in greater details when describing the blockchain processing in Section II-D.

Although we use two different smart contracts, we use one blockchain ledger because of the dependency across the events between authorization and coordination. More specifically, the certificate plane for the authorization affects the control plane operations, i.e., the dynamic control of the controller membership decides who can push and vote for the inter-controller coordination. For example, if a controller gets revoked while that controller is involved in the control-plane coordination, then our design enables which event comes first in the distributed environment.

C. Distributed Consensus Protocol

In our approach, we use practical byzantine fault tolerance (PBFT) based voting consensus to resist against upto n controllers compromise, which is defined as n-compromise resistance, where the controllers' voting can be asynchronous as opposed to crash fault tolerance (CFT) focusing on resisting against availability problem for synchronous system, and also, our distributed consensus does not rely on the trust in the leader selection like the RAFT consensus depends on [12]. Our work therefore provides greater robustness against controller compromise. Because we utilize PBFT, we need $3n + 1$ participants in total while reach $2n + 1$ as a quorum of participants to achieve the n-compromise resistance.

We define C as the participating controller in the blockchain P2P network, and use smart contract to facilitate the PBFT consensus. Every transaction pushed by one participant (i.e.

the controller in our case) using the smart contract, another $2n$ at least (plus the participant sending the transaction itself, the total number becomes $2n+1$) participants must achieve an agreement by voting for it to make that transaction viable. In other words, every transaction yielded by a smart-contract-based vote riding on one Proof-of-Authority (PoA) based broadcasting needs at least another $2n$ PoA-based broadcasting votes. PoA is largely driven by identities and registration, and thus it is applicable in permissioned environments. The authority of a PoA-based broadcasting is the controller that launches the broadcasting.

D. Block Construction and Processing

This subsection describes the construction of the block containing the payload (also called transactions or records, depending on the blockchain applications). The block contains the following parts: the hash of the previous block (in plaintext), the payload P (in plaintext), and the digital signature (processed using the aforementioned data included in the block). In other words, the m-th block $B_m = \{H(B_{m-1})||P||E(k_i, H(P))\}$.

Regarding the digital signature process, the hash output of the previous payload $H(P)$ is the input of the digital signature algorithm using the donor controller's private key k_i, so the digital signature is $E(k_i, H(P))$. The concatenation of $E(k_i, H(P))$ and P will be used for verification using the donor controller's public key K_i. So, the recipient controller takes the concatenation message and produces a hash code. The recipient controller decrypts the digital signature using the donor's K_i. If the calculated hash code matches the decrypted digital signature, the digital signature is accepted as valid. Because only the donor controller knows the k_i, only the donor controller could have produced a valid digital signature.

This block construction is applicable for both of our smart contracts described in the following subsections. Note that since in our case we use one block to contain only one transaction, so the term "block" and "transaction" are exchangeable in the context, which essentially is the payload.

1) Auth-SC for Distributed PKI : Auth-SC is used to facilitate the blockchain-based distributed PKI to replace the centralized CA and provides trust to the participating controllers. The Auth-SC enables the participating controllers self-contained to establish trust among themselves rather than rely on a 3rd party CA. In this regard, the payload is the certificate. We define that for any controller C_i, $i \in \{1, 2, ..., 3n+1\}$, $3n+1$ indicates the total number of controllers. For any time j, $j \in \{1, 2, ..., t\}$, t represents the entire number of the created public-private key pairs. Based on that, we define $K_{i,j}$ as the public key of C_i at time j, while $k_{i,j}$ is the private key of C_i at time j, and I_i is the identity of C_i. So, P=$\{I_i||K_{i,j}\}$. We define G as the function for generating digital certificate, so the digital certificate is $G(K_{i,j}, I_i, k_{i,j}) = \{I_i||K_{i,j}||E(k_i, H(I_i||K_{i,j}))\}$. Because the payload structure (i.e., P=$\{I_i||K_{i,j}\}$) for the Auth-SC is fixed, so the block can be constructed by the means described before. In addition, Auth-SC uses the PBFT-based voting consensus, which means every certificate transaction must get voted by more than 2/3 of the whole participating controllers, otherwise, the certificate will not be viable. The revocation of an effective certificate needs to be compliance with the PBFT consensus, too.

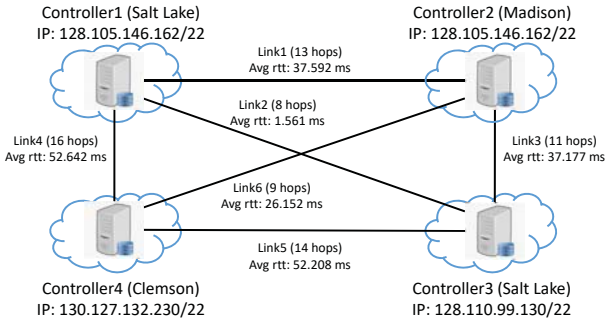

Fig. 3. Controllers deployed across 4 domains (while $n = 1$)

2) Coord-SC for Parameter Update: Regarding the Coord-SC, the smart contract needs to check if the voting controller has been authorized/registered. Also, the payload of control communication is actually the control update, e.g., update the blacklist of the malicious IP addresses. Hence, the payload of the Coord-SC is fixed according to the application, and the block can be constructed by the means described before. Further, the parameter update performed by PBFT-based voting consensus as well.

III. Prototype

In our prototype, the virtual machines with the same specification (using Ubuntu 18.10 64-bit operation system, Intel 2.20 GHz CPU, 4GB memory, and ethernet physical network adapter) are deployed on CloudLab to constitute the permissioned blockchain P2P network. Because the validation to BCC focuses on the control plane P2P network, the data plane is emulated by Mininet that supports any network topology. Figure 3 shows a deployment of distributed controllers crosses multiple CloudLab clusters. The hop distance and round trip time (RTT) between every two controllers are shown as well.

Regarding the controller software selection, for the sake of rapid prototyping, we choose Ryu SDN controller framework, since it is well-documented and Python programming-based which result in a less steep learning curve. Because of the heterogeneity requirement, other controller softwares like POX, Floodlight, etc. are applicable to the implementation too. Further, every SDN controller node installs the blockchain setup that adds the modular layer for broadcasting the control update payload. Such modular layer reduces the complexity of configuration and integration to the rest of the SDN controller operations. We use Ethereum to support smart contract programming to enable the voting-based consensus. We create one Ethereum blockchain with Proof of Authority (PoA) for doing block/transaction verification rather than using Proof of Work (PoW) for mining, and also, we develop the Auth-SC and Coord-SC built on the blockchain to process the certification management and control communication.

IV. Experimental Results

A. Networking Overhead

This subsection measures the networking overhead of using BCC scheme. We focus on the inter-domain scenario while we also consider the intra-domain scenario for reference.

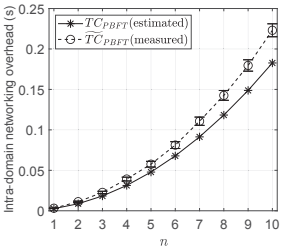

Fig. 4. Intra-dmoain networking overhead

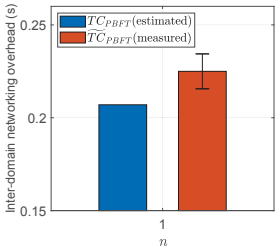

Fig. 5. Inter-domain networking overhead

TABLE I
BLOCKCHAIN TASKS COMPARISON

Payload	Size (bytes)	Gas used	Verification time (ms)
Certificate	7127	1798930	998.512
Control update	1479	252752	998.346

1) Intra-domain testing: For testing the intra-domain scenario, we deploy the BCC controller nodes on the CloudLab Emulab cluster, and every two nodes have 1 hop distance. We tested that the average RTT to this 1 hop distance is 0.58ms, so the transmission latency of each unicast is 0.29ms. Thus, we define the time cost of using voting-based PBFT as $TC_{\text{PBFT}} = 3n \cdot (1 + 2n) \cdot x$ (Equation 1), where n is the one of the n-compromise resistance, x is transmission latency for each unicast (the estimated value is 0.29ms), and the chain needs $3n \cdot (1 + 2n)$ unicasts for voting.

Based on the above setup, we estimate and measure the networking overhead in terms of the increasing n of the n-compromise resistance. Figure 4 plots the networking overhead estimated and measured respectively. The estimated values represent the baseline, while the measured values are achieved by testing the BCC scheme on the CloudLab. The measured overhead ($\widetilde{TC}$) is higher than the estimated one, since the real world implementation contains larger payload for transmission and includes extra overhead, which uses the average values (with 95% confidence interval).

2) Inter-domain testing: Figure 3 already shows our inter-domain BCC nodes deployment across four physically distinct CloudLab clusters. This is the SD-WAN related evaluation, which means the control communication across WAN networks instead of communicating within one network domain. The distance between every two nodes of different domains is often more than one hop (which denotes a gateway/router). According to the real link RTT between those nodes, when we take Controller1 as the donor, we can get the estimated overhead: $TC_{\text{PBFT}} = \sum_{i=1}^{6} x_i$, where $i \in \{1, 2, ...6\}$ (Equation 2). The transmission latency (x_i) of different pairs of nodes need to be calculated separately according to the values shown in Figure 3. Therefor, the estimated overhead is $TC_{\text{PBFT}} = 207$ms.

Figure 5 presents both the estimated and measured networking overheads when $n = 1$ while the total number of the participating controllers is 4. The measured networking overhead ($\widetilde{TC}$) has the average value 225 ms (with 95% confidence interval), which is higher than the estimated overhead 207 ms.

B. Computing Overhead

We present the system computation overhead of applying BCC scheme. Table I shows the cost measurement of blockchain resource, which includes block size (bytes), gas used, and block verification time (ms), for pushing the different payloads into the blockchain using corresponding smart contracts. The certificate payload takes the higher blockchain resource than the control update payload, since the certificate payload is much greater and more complex, e.g., using X.509 as the certificate format in our approach. Also, because we configure the Ethereum to use PoA consensus to verify the payloads, where our smart-contract-voting-based consensus builds on, and so, every broadcast is a PoA-based broadcast. So, for broadcasting each payload regardless of the scale of the blockchain P2P network, the verification time is always around 998 ms.

TABLE II
CPU, MEM AND EXECUTION TIME WHEN CARRYING OUT DIFFERENT TASKS USING BLOCKCHAIN

Task	CPU (%)	MEM (MB)	Exec. Time (ms)
Run Geth	1.18	303.35	–
Deploy smart contract	3.06	652.46	1015
Auth-SC push certificate	4.99	63.72	126.2
Coord-SC push control update	4.59	63.35	121.1

Also, Table II shows the CPU and memory usages as well as the time of executing different tasks using the Ethereum blockchain for our approach. We find that running the Ethereum blockchain only costs CPU 1.18% and deploying smart contract only costs CPU 3.06%, while they take RAM memory 303.35 MB and 652.46 MB respectively. The cost of running Geth is persistent as long as the Ethereum blockchain is running, while the cost of deploying smart contract only lasts around 1.015 second as the Table shows. The push certificate operation by Auth-SC takes more CPU, memory and time than the push control update operation by Coord-SC, which is compliance with the payload size, while the Auth-SC is used much less than the Coord-SC as the certificate management occurs much less than the control communication.

V. RELATED WORK

A. SDN Coordination Related to Coord-SC

There are a number of typical solutions have been proposed for conducting the control communication between multiple controllers. On the one hand, a number of solutions focus on intra-domain, synchronizing controller replicas, aiming to facilitate a resilient control plane to avoid a single point of failure, such as HyperFlow [13], Onix [14] and PANE [15]. On the other hand, several solutions concentrate on inter-domain coordination, managing the seamless setup of network flows across diverse SDN domains, e.g., DISCO [16] and WEBridge [17]. Furthermore, some solutions work for both intra- and inter-domain. ONOS [18] is an distributed SDN control platform, and uses a distributed key-value store, Cassandra [19], to manage and share network state across different controllers. J.H. Lam et al. [20] proposed an approach applying the TLS channel to the ONOS' native east/westbound communication. D-SDN [21] is motivated by the heterogeneous and decentralized networks, It applies Control Delegation message to carry out the coordination between the master controller and slave controllers. It uses the Identity-based Cryptography [22] to protect the control communication.

B. Distributed Trust and PKI Related to Auth-SC

Yakubov et al. presented a blockchain-based X.509 PKI management framework [23], whereby the chain of certificates is distributed. However, this approach still relies on the centralized chain of trust certificate authority model of PKI and does not have to involve any consensus protocol to vouch for certificates. The subCA is only determined by its parent CA, and the parent CA is even in charge of revocation of its subCA. So, the single point of failure was not solved. In contrast, SCPKI [24] presented an alternative PKI system based on a decentralized design, which applies a web-of-trust model and a smart contract on the Ethereum blockchain to make it easily detecting rogue certificates. IKP [25] addressed the challenge of automatic misbehaving CA detection and reaction. The IKP is instantiated by Ethereum whose financial nature incentivize the participants to monitor and report the unauthorized certificates, while our BCC focuses on block construction though incentive can be provided to drive the SDN controller operations. Also, IKP stresses on Internet networking, while our BCC focuses on SDN networking that is tailored from Internet networking.

VI. Conclusion

SDN has greater security risk for controller compromise and control-plane manipulation as there is an increasing number of SDN controllers across multiple networking domains coordinating with each other. In this paper, we propose BCC, a blockchain-based multiple SDN controllers solution to secure the coordination between different and potentially heterogeneous controllers. BCC builds one blockchain and two smart contracts, i.e., Auth-SC and Coord-SC. Auth-SC provides the secure key establishment for the controllers participating in the permissioned controller network, while Coord-SC protects the coordination between controllers from the insider compromise (up to n controller compromises). BCC also improves the networking overhead by using voting-based PBFT consensus of the permissioned blockchain, as opposed to PoW for mining in permissionless blockchain. We implement BCC using Ethereum and smart contract for proof of concept. Further, we analyze the networking overhead of BCC via both simulations and real-world CloudLab based measurement. The experimental results show that BCC has significant security and performance gains.

Acknowledgment

This research was supported in part by Colorado State Bill 18-086 and the National Science Foundation under Grant 1922410.

References

[1] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "Resilience of sdns based on active and passive replication mechanisms," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 2188–2193.

[2] K. ElDefrawy and T. Kaczmarek, "Byzantine fault tolerant software-defined networking (sdn) controllers," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, June 2016, pp. 208–213.

[3] A. J. Gonzalez, G. Nencioni, B. E. Helvik, and A. Kamisinski, "A fault-tolerant and consistent sdn controller," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[4] W. Stallings, "Software-defined networks and openflow," *The internet protocol Journal*, vol. 16, pp. 2–14, March 2013.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13, New York, USA, 2013, pp. 3–14.

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

[7] C. Canales, A. Lerner, M. Toussaint, and J. Skorupa, "Magic quadrant for wan edge infrastructure," *Gartner Research*, October 2018.

[8] R. Santitoro, "Understanding sd-wan managed services," 2017. [Online]. Available: www.mef.net/sd-wan/understanding-sd-wan

[9] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "Sdni: A message exchange protocol for software defined networks (sdns) across multiple domains," June 2012. [Online]. Available: https://tools.ietf.org/id/draft-yin-sdn-sdni-00.txt

[10] H. Hoogstraaten, "Black tulip report of the investigation into the diginotar certificate authority breach," 2012.

[11] D. Goodin, "Stuxnet spawn infected kaspersky using stolen foxconn digital certificates," June 2015. [Online]. Available: https://arstechnica.com/security/2015/06/ stuxnet-spawn-infected-kaspersky-using-stolen-foxconn-digital -certificates/

[12] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.

[13] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, ser. INM/WREN'10, 2010, pp. 3–3.

[14] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10, 2010, pp. 351–364.

[15] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: An api for application control of sdns," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 327–338, Aug. 2013.

[16] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–4.

[17] P. Lin, J. Bi, and Y. Wang, "Webridge: west-east bridge for distributed heterogeneous sdn noses peering," *Security and Communication Networks*, vol. 8, pp. 1926–1942, 2015.

[18] 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*, ser. HotSDN '14, 2014, pp. 1–6.

[19] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35–40, Apr. 2010.

[20] J. H. Lam, S.-G. Lee, H.-J. Lee, and Y. E. Oktian, "Tls channel implementation for onos's east/west-bound communication," in *Electronics, Communications and Networks V*, A. Hussain, Ed. Singapore: Springer Singapore, 2016, pp. 397–403.

[21] M. A. S. Santos, B. A. A. Nunes, K. Obraczka, T. Turletti, B. T. de Oliveira, and C. B. Margi, "Decentralizing sdn's control plane," in *39th Annual IEEE Conference on Local Computer Networks*, Sep. 2014, pp. 402–405.

[22] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in Cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 47–53.

[23] A. Yakubov, W. M. Shbair, A. Wallbom, D. Sanda, and R. State, "A blockchain-based pki management framework," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018, pp. 1–6.

[24] M. Al-Bassam, "Scpki: A smart contract-based pki and identity system," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, ser. BCC '17. New York, USA: ACM, 2017, pp. 35–40.

[25] S. Matsumoto and R. M. Reischuk, "Ikp: Turning a pki around with decentralized automated incentives," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 410–426.