Blockchain for Securing Custom/User-Defined Protocols in P4 Programmable Switches

[1]Temidayo Akinyemi,[2]Oluwaseyi Ajayi, [1]Ihab Darwish and [1]Tarek Saadawi

[1] Department of Engineering, City University of New York, City College, NY, USA
[2] Engineering Department, Vaughn College of Aeronautics and Technology, NY, USA

takinye000@citymail.cuny.edu, oluwaseyi.ajayi@vaughn.edu, idarwish@ccny.cuny.edu, saadawi@ccny.cuny.edu

***Abstract*—P4 (Programming Protocol-Independent Packet Processors) represents a paradigm shift in network programmability by providing a high-level language to define packet processing behavior in network switches/devices. The importance of P4 lies in its ability to overcome the limitations of OpenFlow, the previous de facto standard for software-defined networking (SDN). Unlike OpenFlow, which operates on fixed match-action tables, P4 offers an approach where network operators can define packet processing behaviors at various protocol layers. P4 provides a programmable platform to create and implement custom network switches/devices protocols. However, this opens a new attack surface for threat actors who can access P4-enabled switches/devices and manipulate custom protocols for malicious purposes. Attackers can craft malicious packets to exploit protocol-specific vulnerabilities in these network devices. This ongoing research work proposes a blockchain-based model to secure P4 custom protocols. The model leverages the blockchain's immutability, tamperproof ability, distributed consensus for protocol governance, and auditing to guarantee the transparency, security, and integrity of custom protocols defined in P4 programmable switches. The protocols are recorded as transactions and stored on the blockchain network. The model's performance will be evaluated using execution time in overhead computation, false positive rate, and network scalability.**

Keywords—Programming Protocol-Independent Packet Processors (P4) Switches, SDN, OpenFlow, Blockchain, Programmable Networks

I. Introduction

In recent years, network programmability has emerged as a transformative concept in networking, enabling unprecedented flexibility and control over network behavior. The emergence of the P4 (Programming Protocol-Independent Packet Processors) language has revolutionized the way network engineers design and deploy custom protocols in network switches/devices, offering significant innovations and opportunities for network programmability. Essentially, P4 programmable switches have removed the entry barrier to network design, previously reserved for network vendors [1].

P4 brings a paradigm shift by allowing network engineers to define the forwarding behavior of packets in a highly customizable and protocol-independent manner. P4 is currently the most widespread abstraction, programming language, and concept for data plane programming; that was first published as a research paper in 2014 and is now developed and standardized in the P4 Language Consortium [2]. With P4, engineers can tailor network protocols to meet specific requirements, optimize performance, and introduce new functionalities. These custom protocols are specifically designed to address unique use cases or scenarios that cannot be fully achieved with traditional, standardized protocols.

Creating custom protocols in P4-enabled switches/devices involves a multi-step process. This process begins by defining the desired protocol, specifying the packet format and the intended behavior. The P4 codes utilize the high-level language constructs offered by P4 to describe the packet parsing, header modifications, forwarding tables, and actions. The compiled P4 program is installed onto the P4-enabled switch/device, enabling the custom protocol's execution and behavior. An example of a custom protocol is the P4-KBR (P4 Key-Based Routing) which defines a network-level routing protocol where endpoints are identified by virtual identifiers (keys) instead of traditional IP addresses, and P4 network elements are configured to route packets adequately [3].

The custom protocols are created to optimize network performance by tailoring products to specific traffic patterns or application requirements or to achieve load balancing among switches [4]. Additionally, specialized functionalities or modifications can be introduced to the existing protocols, enhancing network capabilities, such as security, Quality of Service (QoS), or multicast support. Custom protocols allow the designing of highly adaptable networks to dynamic environments and specific use cases.

However, the openness and programmability offered by custom protocols also raise concerns regarding security and potential misuse by threat actors. Malicious actors may exploit vulnerabilities within the custom protocol implementations if they gain unauthorized access to P4-enabled switches/devices. Exploitation could lead to disruptions in network operations, unauthorized access to sensitive information, or manipulation of network traffic flows.

To address these security challenges, we propose a blockchain-based solution to ensure the integrity and availability of the custom protocols. The proposed solution relies on the tamperproof ability, decentralization nature, and data immutability capability to ensure transparency, integrity, and resistance against protocol tampering. Decentralized consensus mechanisms provide governance and auditing capabilities, ensure protocol changes' authenticity, and mitigate unauthorized modifications.

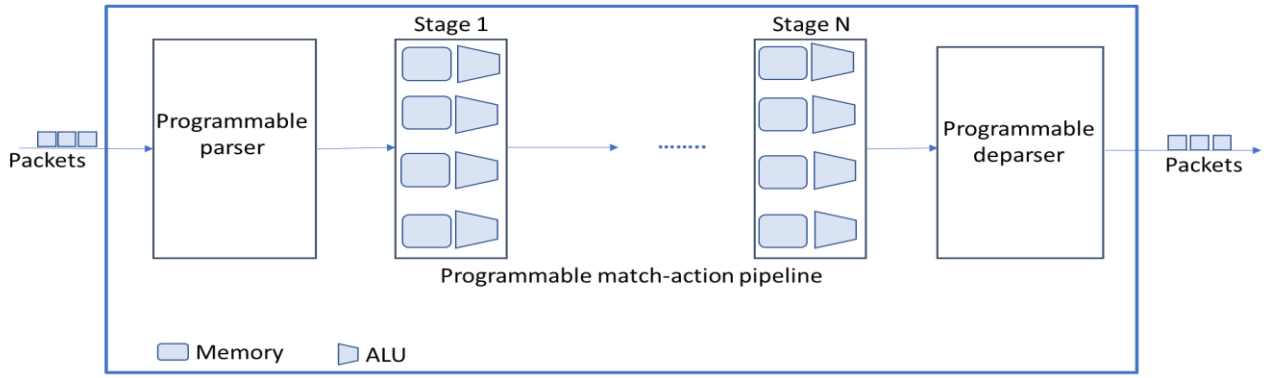

Fig. 1: Data Plane with PISA model

Therefore, the objectives of the research can be summarized as follows:

- To propose a blockchain-based architecture for custom protocols security in P4 switches
- The proposed solution will ensure the integrity, transparency, and resistance to tampering of the custom protocols.
- The proposed model can promptly detect malicious activities on the custom protocol by invoking the smart contract verification module at regular intervals.
- The solution will not introduce extra overhead on the P4 switches as it performs its verification in the blockchain network at the control plane.

The rest of this paper is structured as follows: Section II describes the background and related works. In section III, we describe the methodology and the proposed architecture. Section IV presents the performance metrics to evaluate our proposed solution. Finally, in section V, we present our conclusions and further work to be done.

II. BACKGROUND AND RELATED WORK

A. Programming Protocol-Independent Packet Processors (P4)

P4 is a high-level programming language for programming data plane network devices like switches, routers, and network interface cards (NICs). P4 represents a similar approach to how central processing unit (CPU), graphics processing unit (GPU), or digital signal processor (DSP) work, where they execute code written in a specific programming language (e.g., C++ for CPU, OpenCL for GPU, or MATLAB for DSP); In this way, P4 can be compiled against different execution machines like field-programmable gate array (FPGA), application-specific integrated circuit (ASICs), or network processors [5]. It provides a high-level abstraction that allows network operators and researchers to define the packet processing behavior in network switches and other network devices.

The P4 language allows network operators to define their packet processing logic independently of the specific networking protocols used. Hence, programmers can create target-independent programs that a compiler can map to various forwarding devices [6]. The programmer can define and parse new protocols, customize packet-processing functions, measure events occurring in the data plane, and inspect and analyze each packet. The idea of P4 was initially born in 2013, and the first formal specification of the P4 language was released in 2014, called $P4_{14}$. After that, an updated specification, $P4_{16}$, was released in 2016 [7].

B. P4 Architectures and Programmable Switches

P4 architectures refer to how the target device is designed so that the compiler can properly compile the program to the device. The concept of various "architectures" types came up with the introduction of $P4_{16}$. This is because when the previous version, $P4_{14}$, was designed, it assumed it would be used only for switches based on the PISA (Protocol Independent Switching Architecture) model. Hence, $P4_{16}$ overcomes this limitation by embracing architectural heterogeneity [8].

PISA presented the first answer to the need for data plane programmability. It is an architecture for high-speed programmable packet forwarding switches [8].

Fig. 1 presents the components that make up PISA. These include a programmable parser, match-action pipeline, and programmable deparser.

- ***Programmable Parser***: packet processing in the programmable data plane starts with a programmable parser responsible for extracting relevant header fields from the incoming packets to set them as input for the match-action units. The parser allows the programmer to define custom or standard header protocols. After the packet is parsed into individual headers, it can be used for the algorithm execution in the match-action pipeline [9].
- ***Programmable Match-Action Pipeline***: This sequence of several match-action stages consists of memory and arithmetic logic units. Here, some fields in the incoming parsed packet headers are compared with values in the memory table to see if there is a match and then perform some action based on the matching condition. Some of the activities performed include modifying packet headers or dropping the packet. The match-action pipeline can have 10 to 15 stages, containing multiple match-action units in both the ingress and egress stages. It is also possible to run part of an

application's logic here [9], such as analyzing and dropping packets for attack signatures.

- ***Programmable Deparser***: The final output of the match-action pipeline is fed to the deparser, which reassembles the packet, including new or modified headers, and serializes them for transmission.

As more programmable devices were built, it became clear that $P4_{16}$ had outgrown the PISA model. So the P4 community introduced the Portable Switch Architecture (PSA), which was created to abstract the hardware pipeline so that P4 developers need not worry about the specifics of the target device when programming the data plane, but allow the compiler to execute and map the program to the specific target device [10].

As shown in Fig. 2, PSA consists of six (6) P4 programmable blocks and two (2) fixed-function blocks. The behavior of the programmable blocks is specified using P4. The Packet Buffer and Replication Engine (PRE) and the Buffer Queueing Engine (BQE) are target-dependent functional blocks that may be configured for a fixed set of operations [11]. As an analogy, the PSA is to the $P4_{16}$ language as the C standard library is to the C programming language [11].

From Fig. 1 and 2, it can be observed that the six (6) programmable blocks in the PSA pipeline are analogous to two PISA models, with the first three (3) representing an "ingress PISA" and the last three (3), an "egress PISA." This means that when the packets reach the switch, they get processed via the ingress parser, ingress pipeline, and ingress deparser before they are sent to the PRE. After that, they get processed again via the egress parser, egress pipeline, and egress deparser and then forwarded to the BQE.

C. Blockchain

Blockchain is a decentralized ledger that records transactions or activity between participants permanently with verification. This verification comes in the form of reviewing cryptographic functions and timestamps. The transactions can be verified on multiple computers, which are called nodes. A good example is BitTorrent [12], which is a peer-to-peer file-sharing protocol that does not rely on any one server, company, or entity to work.

Blockchains have been extensively used as a security solution in many areas. The authors in [13] leveraged blockchain's distributive technology, tamper-proof ability, and immutability to detect and prevent malicious activities and solve data consistency problems facing cooperative intrusion detection. In healthcare, [14] used blockchain technology as a decentralized record management system to handle Electronic Health Records (EHRs), giving patients a comprehensive immutable log and easy access to their medical information across providers and treatment sites. The authors in [15] proposed blockchain as a decentralized application for secure authentication in fog and IoT environments. [16] proposed a security architecture that integrates blockchain and multi-controller software-defined networks to deal with network attacks like false data injection.

Cryptography and hash functions are the most integral things securing the blockchain. Public key cryptography is the foundation for how digital wallets work, how tokens are traded, how identity is verified, and is used to create verifiable historical records of transactional data. Generally, the public key generates the wallet address, while the private key signs transactions to confirm it is genuinely from the address. The hash function creates data that is recorded to the blockchain, making any change to a single piece of data easily identifiable. Some of the important properties of these hash functions are irreversibility and the impossibility of collision – it is extremely unlikely that two different inputs to a hash function produce the same output. Each block in the chain contains the previous block's hash, meaning every block is linearly connected back to the original Genesis block. The difficulty in changing a single block and finding valid hashes for all subsequent blocks makes blockchain nearly immutable [16].

D. Related Work

There are various security concerns in programmable networks generally because each plane can serve as an attack surface. For example, when considering the control plane of a programmable network, the centralized controller can present a potential single point of failure, making it attractive to denial-of-service (DoS) attacks [17]. With respect to the application layer, applications can be malicious if an adversary takes control of it and then injects malicious application flows into the control plane via the northbound API calls.

However, the security focus for this paper is on the programmable data plane. P4 provides a programmable data plane that allows the creation of custom protocols and implement them in network switches. However, this also opens a new attack surface for threat actors who can gain access to P4-enabled switches and hijack custom protocol implementations for malicious purposes. Attackers can easily trigger injection attacks by creating specially crafted packets or exploiting other protocol-specific vulnerabilities in these network devices. Several research efforts have focused on addressing the security challenges in P4 data plane networks.

In [18], the authors proposed using formal verification techniques, such as model checking and theorem proving, to ensure the correctness and security of P4 programs. Their solution is internal and considers the P4 program internally before execution. A security-aware compiler design is proposed to improve the security of P4 programmable networks [19]. This could involve incorporating security checks and constraints into the compilation process to ensure that P4 programs comply with security policies. Although the solution proposed in [19] is similar to ours, the difference is that their solution focused on the compilation process. While ours is centered on the integrity of the user-defined protocols. Researchers have also proposed some network-level security measures, such as intrusion detection and prevention systems, firewalls, and secure communication protocols to protect P4 programs from external threats [20]. Many of these tools can generate high false positives, which may affect their effectiveness.

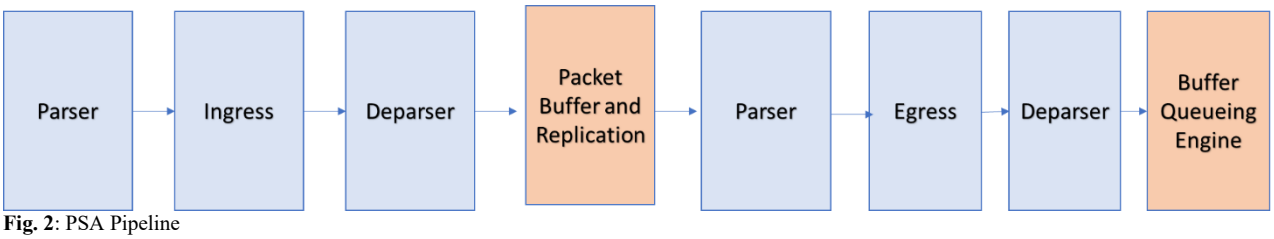

Fig. 2: PSA Pipeline

Our proposed solution is specifically aimed at overcoming these security problems by using blockchain for verifying and guaranteeing the integrity of P4 protocols.

III. METHODOLOGY

The proposed solution focuses on securing the custom protocols defined in the programmable data plane P4 switches. These user-defined protocols are created either as new protocols or by modifying existing standard protocols like ethernet, IP, MPLS, etc. These user-defined protocols can be modified or updated from the control plane of the P4 switches. Any update on the custom protocol directly affects how the P4 data process packets. Our solution introduces a blockchain network to the control plane, as shown in Fig. 3. Due to the computational overhead the blockchain network may introduce, the setup is run outside the control plane of the P4 switches so that many controllers can be part of the network. The solution is built on the Ethereum blockchain platform. Ethereum blockchain handles many concurrent transactions; the latest version runs PoS as against the PoW consensus algorithm run by the older version. This makes it faster and more scalable [21]. It is an open-source blockchain-based distributed computing featuring smart contracts. The smart contract is an agreement among consortium members that is stored on the chain and run by all participants[22]. Although the Ethereum platform blockchain platform is considered a public blockchain, in this work, we configured it to exhibit the characteristic of a private blockchain network as done in our previous work [21]. The solution is considered a private blockchain because only permissioned controllers can join and participate in the network.

The architecture consists mainly of two operations: *Protocol Verification* and *Protocol Governance*, as shown in Fig. 3.

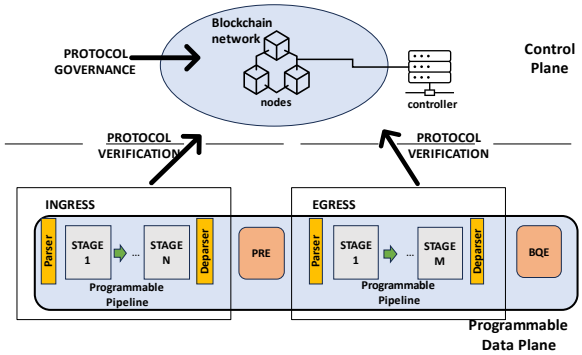

Fig. 3: Architecture of the Proposed Blockchain-based System for Securing P4 Custom Protocols

A. Protocol Verification

The smart contract handles the protocol verification process. The protocol verification step ensures that all malicious activities on user-defined protocols running in the P4 switches are identified and thwarted, thereby ensuring the integrity of the P4 protocols. The verification also ensures that any submitted transaction's integrity and consistency are verified before attaching it to the blockchain network. In this architecture, the smart contract guarantees that malicious intentions or errors do not manipulate the P4 protocols by constantly comparing a copy with what is stored in the blockchain. We implement a transaction access control policy in the private network. This means the architecture verifies both the node submitting the transaction and the transaction submitted. Authenticating a transaction requires that the smart contract invokes a code that compares transaction accompanying information with stored information.

Here, a copy of the custom protocols defined in the P4 switch is recorded as transactions and stored in the blockchain network. At a predefined regular interval, a blockchain node (controller) extracts a copy of the running protocol, which we denote as a "challenge request". The challenge request is prepared as a transaction and submitted

```
1          Algorithm1: Protocol Verification
2
3   Procedure: Protocol verification
4   Inputs: Protocol Information
5
6   If hash(transaction) == hash(stored information)
7
8          Return Success
9          Discard transaction
10         Return to 6
11  else:
12         push transaction for validation
13         alert administrator
14         broadcast copy to other controllers
15  end if
16  end procedure
```

to the blockchain network. On receiving the transaction, the smart contract verifies the authenticity of the running protocol by comparing the hash with the stored hash. If the hashes are the same, the transaction is discarded. However, if the hashes are different, the transaction is pushed to the transaction validation stage [23], and this triggers the smart contract alert module, which informs the administrator/Network engineer about possible manipulation of the running protocol while pushing a copy to other controllers via a new block of the blockchain. The Administrator/ network engineer reviews the alert and takes necessary action. This action can be that the administrator informs the parser in the data plane to drop the traffic, thereby enforcing security policies. The smart contract keeps monitoring the hash of the submitted transaction for prompt detection of malicious activities on the custom protocols running on the P4 switches. The pseudocode below (Algorithm 1) describes the snippet of how the smart contract handles protocol verification.

B. Protocol Governance

Besides detecting malicious activities, the proposed architecture can accept new custom protocols or modifications to existing ones via the consensus mechanism. The following steps ensure the protocol governance of the P4 custom protocols:

- Protocol definition: the P4 custom protocols are defined clearly by the developer or network operator as is or will be in the P4 programmable switch.
- Hashing: cryptographic hash of the protocol is generated, which represents the unique fingerprint of the protocol and ensures its integrity.
- Consensus mechanism: an appropriate consensus mechanism, like Proof of Stake (PoS), is chosen to govern how the nodes agree on the validity and ordering of protocol updates.
- Blockchain storage: the hash of the protocol is stored on the blockchain, and each time a protocol update is proposed and accepted, a new hash is recorded.

The protocol governance can be transparently managed by leveraging the consensus mechanism and blockchain.

IV. PERFORMANCE METRICS

The proposed architecture is implemented on an Ethereum blockchain platform. We use *Solidity v 0.8.17* implementation for smart contracts and *geth v 1.12.0* for Ethereum. For initial proof-of-concept testing, the private blockchain network will be set up in the laboratory with three computers serving as blockchain nodes (controllers) (Fig. 1) to evaluate performance. Some key performance metrics will be evaluated to implement the feasibility of the proposed solution. These metrics are utilized in the implementation of the solution. Three key performance metrics to evaluate the efficiency of the proposed solution are:

A. Rate of False positives

The rate of false positives is calculated as the ratio between the number of negative events wrongly categorized as positive (false positives) and the total number of actual negative events (regardless of classification). In this concept, we define a false positive rate as the percentage of times the architecture fails to alert the administrator when it is supposed to. If the custom protocols have been tampered with or modified for malicious intents, the blockchain node should be able to detect it based on the hash comparison with what was stored in the blockchain.

B. Detection time

We define the detection time as the time taken to decide on any transaction sent to the blockchain. This time spans from the transaction preparation time to the time the administrator receives an alert (in case of malicious activities). The following data are collected from each transaction to implement the detection time.

- *Transaction deployment time (t_1):* This is the time a transaction is submitted to the network. These data are collected directly from the sender console.
- *Execution time (t_2)*: This is the time taken before an administrator receives an alert about possible protocol manipulation. The time is retrieved by setting on current time on all node consoles.

C. Scalability

We will evaluate the change in the detection time of the proposed solution with an increasing number of P4 switches. This will enable us to know the relationship between the size of the P4 data plane and the blockchain performance. It will estimate the maximum number of switches that can tolerate an acceptable blockchain performance.

V. CONCLUSION

The advent of a programmable data plane comes with its security challenges. In this project, a blockchain-based security solution was proposed to tackle integrity issues of custom user-defined P4 protocols in the data plane. We proposed a blockchain network outside the control plane to avoid the additional overhead that may be introduced to the network by blockchain technology. With controllers as blockchain nodes, a copy of the running protocol is stored in the blockchain network. A challenge request is regularly prepared and sent to the blockchain for integrity confirmation. The smart contract verifies the transaction by comparing the hash of the submitted transaction to the stored value. The transaction is only pushed to the validation stage, and alerts are sent to the administrator, if the challenge information is different from the stored value else, discard the transaction.

Future Work

- Implement the performance metrics and evaluate the results.
- Implement and evaluate the performance of the solution using P4 switches.

VI. ACKNOWLEDGMENT

This work is supported by NSF Grant, INRC Testbed: COSMOS Interconnecting Continents.

REFERENCES

[1] E. F. Kfoury, J. Crichigno and E. Bou-Harb, "An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends," in IEEE Access, vol. 9, pp. 87094-87155, 2021, doi: 10.1109/ACCESS.2021.3086704.

[2] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, “A survey on data plane programming with P4: Fundamentals, advances, and applied research,” Journal of Network and Computer Applications, Volume 212, 2023, 103561, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2022.103561. (https://www.sciencedirect.com/science/article/pii/S1084804522002028)

[3] P. Manzanares-Lopez, J. P. Muñoz-Gea, and J. Malgosa-Sanahuja, “P4-KBR: A Key-Based Routing System for P4-Programmable Networks,” Electronics 2021, 10, 1543. https://doi.org/10.3390/electronics10131543

[4] B. Guan and S. -H. Shen, "FlowSpy: An Efficient Network Monitoring Framework Using P4 in Software-Defined Networks," 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall),

Honolulu, HI, USA, 2019, pp. 1-5, doi: 10.1109/VTCFall.2019.8891487

[5] P. Parol, “P4 Network Programming Language – what is it all about?,” [Online]. Available: https://codilime.com/blog/p4-network-programming-language-what-is-it-all-about/

[6] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, “P4: Programming protocol-independent packet processors,” SIGCOMM Comput. Commun. Rev., vol. 44, p. 87–95, July 2014.

[7] AsterFusion, “What You Should Know About P4 Programming Language and P4 Programmable Switch,” [Online]. Available: https://cloudswit.ch/blogs/what-you-should-know-about-p4-programming-language-p4-switch/

[8] C. Kim, “The Forwarding Plane: An Old New Frontier of Networking,” [Online]. Available: https://courses.cs.washington.edu/courses/cse561/21wi/slides/chang-pisa.pdf

[9] N. Gebara, A. Lerner, M. Yang, M. Yu, P. Costa, and M. Ghobadi, “Challenging the Stateless Quo of Programmable Switches,” In Proceedings of the 19th ACM Workshop on Hot Topics in Networks 2020 (HotNets ’20). ACM, New York, NY, USA, 153-159. https://doi.org/10.1145/3422604.3425928

[10] C. Cascaval and D. Daly, “P4_Architectures,” [Online]. Available :https://opennetworking.org/wpcontent/uploads/2020/12/p4-ws-2017-p4-architectures.pdf

[11] P4.org Architecture Working Group, “P416 Portable Switch Architecture (PSA),” [Online]. Available: https://p4.org/p4-spec/docs/PSA-v1.2.pdf

[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, “A Performance Study of BitTorrent-like Peer-to-Peer Systems,” IEEE Journal on Selected Areas in Communications., Jan 2007

[13] O. Ajayi, O. Igbe, and T. Saadawi, “Consortium Blockchain-Based Architecture for Cyber-attack Signatures and Features Distribution” 10th IEEE Annual ubiquitous computing, electronics & mobile communication conference 2020

[14] A. Ekblaw, A. Azaria, J. D. Halamka, and A. Lippman, "A Case Study for Blockchain in Healthcare: "MedRec" prototype for electronic health records and medical research data" (2016)

[15] O. Umoren, R. Singh, S. Awan, Z, Pervez, K. Dahal. “Blockchain-Based Secure Authentication with Improved Performance for Fog Computing,” Sensors (Basel). 2022 Nov 19; 22(22):8969. doi: 10.3390/s22228969. PMID: 36433564; PMCID: PMC9693513.

[16] S. J. Bigelow, “Blockchain: An Immutable Ledger to Replace the Database,” [Online]. Available: https://www.techtarget.com/searchitoperations/tip/Blockchain-An-immutable-ledger-to-replace-the-database

[17] J. Weng, J. N. Liu, JI. Weng and Y. Zhang, “Secure Software Defined Networking Based on Blockchain,” arXiv:1906.04342v1 [cs.CR], Jun 2019. https://doi.org/10.48550/arXiv.1906.04342

[18] H. A. Akarte and D. K. Yadav, “Packet Processing and Data Plane Program Verification: A Survey with tools, techniques and Challenges,” International Journal of Communication Systems, June 2023, https://doi.org/10.1002/dac.5554

[19] X. Jin, Y. Hu, “Towards Secure P4 Programmable Networks: A Survey,” (2019) IEEE Access, 7, 1,077,254-1,077,269. Doi:10.1109/ACCESS.2019.2945672

[20] M. Canini, M. Schapira, M. Schapira, “P4 Security: From Threats to Defenses,” 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) (pp. 1-6). Doi.10.1109/NFV-SDN.2018.8691793

[21] O. Ajayi, O. Igbe and T. Saadawi, "Consortium Blockchain-Based Architecture for Cyber-attack Signatures and Features Distribution," 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2019, pp. 0541-0549, doi: 10.1109/UEMCON47517.2019.8993036.

[22] Ingo Weber, Vincent Gramoli, Mark Staples, Alex Ponomarev, Ralph Holz, An Binh Tran, and Paul Rimba. 2017. On Availability for Blockchain-Based Systems. In SRDS’

[23] O. Ajayi and T. Saadawi, "Detecting Insider Attacks in Blockchain Networks," 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 2021, pp. 1-7, doi: 10.1109/ISNCC52172.2021.9615799.