Computers & Security 88 (2020) 101629

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

P4-to-blockchain: A secure blockchain-enabled packet parser for software defined networking

Abbas Yazdinejad[a], Reza M. Parizi[b], Ali Dehghantanha[a], Kim-Kwang Raymond Choo[c,*]

[a] Cyber Science Lab, School of Computer Science, University of Guelph, Ontario, Canada
[b] Department of Software Engineering and Game Development, Kennesaw State University, GA 30060, United States
[c] Department of Information Systems and Cyber Security, University of Texas at San Antonio, Texas, United States

ARTICLE INFO

Article history:
Received 28 March 2019
Revised 6 September 2019
Accepted 28 September 2019
Available online 2 October 2019

Keywords:
Software defined network
SDN
Packet parser
Blockchain
DoS
P4
FPGA

ABSTRACT

Security is one of the most challenging issues in software defined networking (SDN), and causes include the inability to detect the contents of the packets. In addition, file transferring to SDN is a potential attack vector that can be exploited at the network level. On the SDN data plane (i.e. the switch), packets are processed so that their behavior in the network can be determined. Packet parser (PP) acts as the main role of this operation on the switch. PP plays a very important role in identifying packets and supporting protocols and testing new ideas in SDN, which requires packet parsers on the data plane to provide the necessary flexibility and programmability. In recent times, cryptographic ledger technology (also referred to as Blockchain in the literature) has been applied to securely transfer transactions and files in the networks, mainly for crypto payments. In this paper, we propose a new packet parser architecture called Blockchain-enabled Packet Parser (BPP) based on the security characteristics of the blockchain and support for data processing functions with the description of Programming Protocol-Independent Packet Processors (P4) language that has the BPP-independent attribute of the protocol. In the proposed architecture, we provide a mathematical model based on a multivariate correlation approach for attack detection from the observed packet traffic. The implementation of BPP architecture is on Field Programmable Gate Array (FPGA), which is known for its higher processing speed, flexibility, and lower consumption resources. The result indicates that the proposed BPP is able to detect attacks and policy performed on the control plane in how to detect an attack from a packet structure efficiently. Furthermore, the impact of the bandwidth with and without the BPP in hardware environment is evaluated to demonstrate BPP's efficiency.

1. Introduction

There are a number of known benefits associated with the deployment of software defined network (SDN), such as the separation of the control plane from the data plane that results in better network management and reduced implementation and runtime costs with dynamism and programming capabilities (A. Yazdinejad et al., 2018). SDN also allows venders to test new ideas on the networks (e.g. define new protocol, custom packet header, change role and policy security) (A. Yazdinejad et al., 2018). One of the concerns today about the SDN is however its security, which has become a major challenge. SDN security risks are mainly due to the lack of integration with existing security technologies and the inability to examine the contents of each packet (Basnet and Shakya, 2017). Moreover, file transferring to SDN is one of the critical zones, which has been the target of malicious activities and attacks at the network level. One of the destructive attacks is the Denial of Service (DoS) attack (Rao et al., 2019; Homayoun et al., 2019). Currently, there are two ways to manage the new input streams on the SDN network when there is no match between the input streams and the to-do list switches on the data plane (Parizi et al., 2019; Parizi et al., 2018). First, a part of the packet or the entire packet is sent to the controller to resolve the problem, meaning that the full transmission will depend on the controller, which in turn may create problems in the bandwidth. Second, if a portion of the packet or header is sent to the controller, the packet stays in the memory of the device until the input of the stream table is returned, in which case the attackers may simply cripple the DoS attack and create an unknown stream of the network.

On the other hand, to enhance the security in SDN, we need to also secure the communication in the data plane. Files transfer

* Corresponding author.
E-mail addresses: abbas@cybersciencelab.org (A. Yazdinejad), rparizi1@kennesaw.edu (R.M. Parizi), ali@cybersciencelab.org (A. Dehghantanha), raymond.choo@fulbrightmail.org (K.-K.R. Choo).

https://doi.org/10.1016/j.cose.2019.101629
0167-4048/

process in the SDN can be (significantly) secured against tricky activities by the implementation of blockchain (Rao et al., 2019; Taylor et al., 2019). The blockchain technology presents one of the most recent mechanisms that we currently have to protect data from cyber attackers (e.g. hackers), anticipating possible fraud and reducing the risk of data being stolen or compromised. Innovative uses for blockchain technology can be especially useful to boost cyber security. For instance, blockchains can prevent distributed denial-of-service (DDoS) attacks by decentralized DNS (Liu et al., 2018). As advocated in (Muthanna et al., 2019; Zheng et al., 2018), when the security of network's users increases via blockchain, the reliability of network systems, whether on data plane or control plane, increases. In fact, blockchain is a secure and confidential solution to the network (Bozic et al., 2016). The motivation behind the use of blockchain technology in SDN is its security-by-design benefits suited for secure communications. The existence of an architecture to identify an attack on a packet structure, and also support blockchain structure on the data plane is necessary to increase the reliability and security of communications, which is currently lacking in the literature. Attacks happen on the data plane at first, then the users will know about them in the application layer. Therefore, the data plane is a critical bottleneck in the SDN network. In this regard, we integrate blockchain mechanism into the data plane for much better supporting of security policies in the control plane of the SDN network. Implementing blockchain technology in the data plane, especially in switches, would distribute the roles and security policies to a large number of decentralized switches and making it nearly impossible for cyber attackers to compromise and take over. Roles and policies editing rights would only be granted to the validator nodes and no other user could make changes. This would significantly reduce the risk of data being accessed or changed by unauthorized parties, resulting in increasing the reliability of the system, and achieving superior system security by preventing various cyber security attacks.

So far, hardware architectures have generally not considered to support detecting attacks and the blockchain structure in the data plane and the available architectures do not detect the attack on the packet structure according to the control plane policy. Having the ability to detect attacks and awareness of the control plane in the SDN by hardware architecture, increases the level of safety and reliability, also the SDN controller can define new patterns for detecting attacks based on packet structure because the availability of a programmable architecture in SDN switch. To support better implementation of blockchain technology at the SDN data plane, hardware with specific programmability and dynamicity in the network switches is needed to enforce policies and enhance security (Basnet and Shakya, 2017; Sharma et al., 2018). Also, on the data plane, there is a need for packet parser in switch to provide the flexibility of the security features applied by network administrators and new technologies tools such as blockchain.

This paper proposes a Blockchain-enabled Packet Parser (BPP) of the SDN, which, in addition to high programmability and flexibility, supports detecting the attacks (five category of patterns: Normal, Remote to Local (R2L), DoS, User to Root (U2R), and Probing Attack (Probe)) and blockchain protocol implementation on the data plane. The description of our proposed BPP architecture is based on P4 that is used to describe and customize packet processing functions, generating a protocol-independent property. We have used the features of P4 language to create a P4-based security in data plane. Our P4 program is able to detect attacks and announce them to the SDN controller via defined rules. With examining the packet data structure and using a mathematical model based on a multivariate correlation approach for attacks detection, we are able to detect common attacks on the network, such as the DoS attack through the mathematical model based on multivariate correlation approach to investigate from the observed packet traffic by BPP. Finally, the description of the RTL PP which is blockchain-specific, is implemented on Field Programmable Gate Array (FPGA) (A. Yazdinejad et al., 2018; A. Yazdinejad et al., 2018) hardware platform. Using platforms such as FPGA was of interest because of its high processing speed, flexibility and programmability, and the ability to describe the complex behavior of the network. In fact, FPGAs have been a good target for implementing data-plane performance (Parizi and Dehghantanha, 2018; A. Yazdinejad et al., 2019).

The rest of the paper is organized as follows: Section 2 reviews the extant literature. Section 3 describes BPP architecture and its characteristics. In Section 4, the compilation and production of BPP architecture is described via P4. In Section 5, BPP's evaluation results, including simulation and implementation as well as implications and limitations, are presented. Finally, the conclusion and suggestions for future research are presented in Section 6.

2. Related work

One of the most important components on SDN especially in the data plane is a switch. The inability of the SDN switch to examine data content and packet processing on the data plane can lead to security problems and network disruptions. SDN data plane must be programmed to respond to requests, enforce security policies for network administrators to address security deficiencies and threats (Parizi and Dehghantanha, 2018). The process of detecting the headers and extracting the packet header fields for processing on the switch or sending to the parsing controller are assumed to be done by a unit on the switch, called Packet Parser (PP). PP is based on the parsing graph to identify the packet header and extract its fields for subsequent processing on the SDN data plane and even control plane (A. Yazdinejad et al., 2018; A. Yazdinejad et al., 2019). In fact, the first packet operation on the SDN switch is on the data plane by the PP. If PP has programmability on the data plane, the SDN switch is flexible and programmable, and network administrators will be able to define new and custom protocols, and even further data-plane security. In such cases, the SDN switch, with the help of PP, can detect and extract the new packet header fields for other sections such as, Match table, Action table in SDN switch or sending to controller plane (A. Yazdinejad et al., 2019). Also, the controller should be able to inform network administrators of sudden attacks and have a controller relationship with the data plane, therefore the data plane should able to detect attack and support security policy. It is necessary to provide the programmability and flexibility in the data plane through well-defined architectures in this level.

SDN data plane must have the required flexibility and programmability to support new needs and technologies. Technological solutions need to respond to actual requests in the infrastructure, especially in security issues. As the integration of SDN and blockchain technologies concern, the data plane should support the blockchain specification and its requirements in the transmission tools. Looking at prior studies proposed in the area of packet parsers on SDN, the work in (Benácek et al., 2016) presented a high-speed programmable PP on an FPGA, generated automatically by a generator, named P4-to-VHDL. In addition, the VHDL code that can be synthesized for the PP fragment is generated from the description of P4. Although the PP focuses on high-speed processing and gives a low level of flexibility on the data plane at runtime, it simply does not support the blockchain protocol and its features for security fields. In another work (A. Yazdinejad et al., 2018), their proposed PP is independent of a protocol that automatically generates a P4 description by a specific methodology. This PP is implemented on configurable hardware such as the FPGA. It, while delivering data-driven flexibility and programmability, has a good processing capability and speed, but does not consider security aspects on the data plane and cannot support blockchain headers.

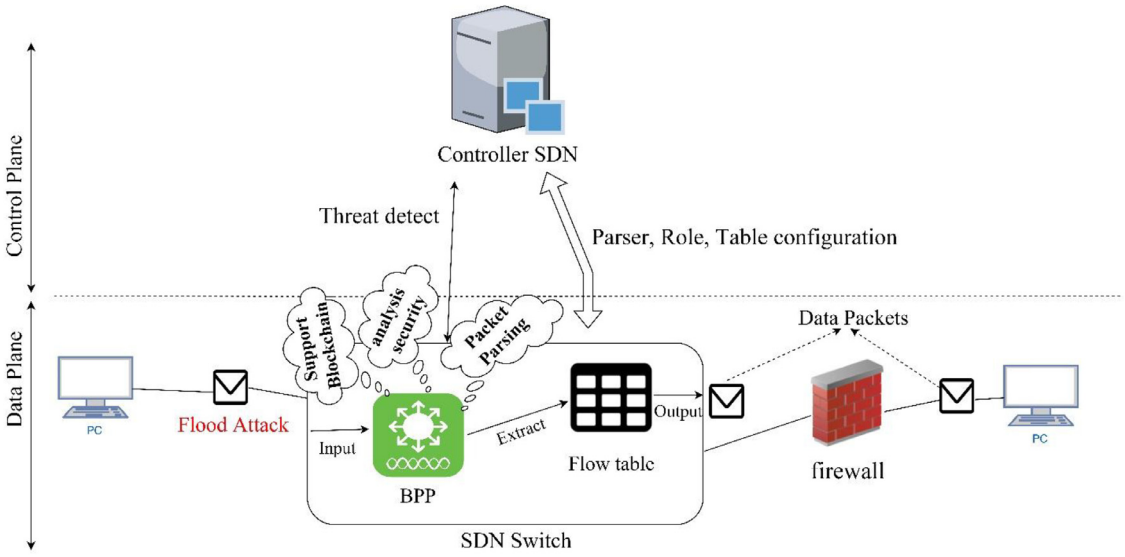

Fig. 1. The role of BPP in SDN network.

The work in (Santiago da Silva et al., 2018) showed that a base FPGA architecture for the PP in the SDN network provides low latency and high speed data rates. This architecture uses a high level *C++* template and its RTL code synthesized and implemented on the hardware. This architecture however does not take into account the security and technological aspects of the blockchain. The work in (A. Yazdinejad et al., 2018) presented an architecture for OpenFlow, whose PP, in addition to boosting rapidity and processing of the pipeline, supports 40 packet headers and only considers features of OpenFlow V1.3. This architecture was implemented on FPGA and occupies a small area of the board. As it turns out, it is not possible to support the blockchain technology and blockchain headers by this PP in this architecture. In work (Santiago da Silva et al., 2018), an architecture called RMT was presented. It is a sample of the programmability data plane in SDN implemented on ASIC. ASIC has some limitations compared to FPGAs, as the architecture is limited by pipeline steps and match tables. The PP of this architecture benefits from high-speed processing and programmability owing to the use of TCAM's memory, but it does not consider the security aspects of the app layer, and has limitations in supporting blockchain technology.

Regarding the research on blockchain in SDN, we can refer to work on blockchain security over software defined network (BBS) (Basnet and Shakya, 2017), which investigated how to secure SDN file transferring with the help of blockchain. Their proposed work uses the Ethereum platform (open blockchain) and the openday-light controller to share files as P2P among users, which is a software approach but it only applies at the app-level and does not consider the data plan in the network layer.

Among other works, one can point to Disbolcknet (Sharma et al., 2017), an architecture for the IoT network, which uses blockchain to validate the rules of flow and distribution of tables among IoT tools. This exploits the distribution feature of the SDN with the blockchain technique and provides security and comparability aspects in the IoT network. There are no aspects of hardware implementation and data-processing of the data plane such as blockchain header processing. In work by Sharma et al. (2018), a cloud distributed blockchain based architecture with SDN was introduced, which uses distributed SDN controllers at the edge of the network. Based on blockchain technology, this architecture has secure features, low cost and on-demand availability. It however does not address the challenges of the IoT and the processing aspects at SDN data plane. Similarly, work (Sharma and Park, 2018) presented a hybrid architecture for smart city by 2 emerging technologies of SDN and blockchain. This hybrid architecture is discrete into two core and edge network to achieve efficiency and eliminate IoT constraints. With this hybrid architecture, features such as centralization and distribution are used to improve security. In our view, this hybrid architecture does not consider SDN data plane and how packets are processed as a part of their security strategies.

3. Proposed blockchain-enabled packet parser (BPP) architecture for SDN switch

The proposed packet parser architecture, BPP, is based on the unique characteristics of the blockchain structure and its header for SDN data analysis. Fig. 1 shows its role in SDN network to give an overall picture where it is intended to be placed. As shown in Fig. 1, BPP is placed at the SDN switch in the data plane, and it has a dominant role here due to supporting three key functions in the network, namely: supporting blockchain, analyzing security and PP process. Packets are processed firstly is BPP, then other parts of the SDN switch can start to work, e.g. Flow Tables. In fact, BPP plays an important role in the data plane for supporting the policies and rules defined by the control plane.

In the following sub-sections, we will discuss the internal structure of BPP architecture and provide details how it supports the blockchain data structure and P4.

3.1. Describing BPP architecture

Fig. 2 presents the high-level structure of BPP. According to the figure, once packets arrive, BPP processes them in order that eventually extract field of the headers'. These fields can be included in the header of the protocol and blockchain structure, transactions, hash blocks. Other information needed to maintain security and support for the block header can be extracted by BPP. Blockchain produces SDNs for data processing units and even controllers if needed. As shown in Fig 2, based on the packet parser, the **Header-detection** unit detects FSM-based packet headers. This unit generates information from the current state and the next state for other existing units. **Data-Ext** extracts the detected fields in terms of information from the unit. In fact, this extraction is based on the type of commonly identified packet headers, such as Eth, IPV6, IPV4, UDP, TCP and the knowledge of the properties of

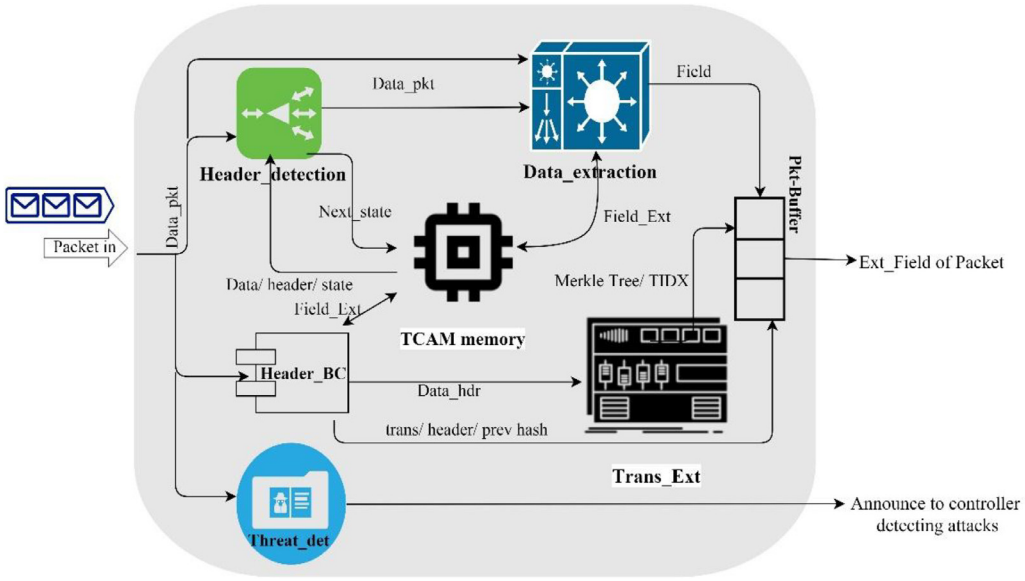

Fig. 2. BPP architecture.

those headers. If it reaches the blockchain header, it will be sent to other units to extract header fields and transactions. **Header-BC** extracts the blockchain header fields and generates outputs such as the previous hash block and timestamp. Extraction of transactions is based on the Merkle tree of the blockchain header and is carried out by the **Trans-Ex** unit. The three **Header-detection, Header-BC,** and **Trans-Ex** units are available in parallel. The TCAM memory in this architecture is a ternary memory, and in addition to zero and one, it supports a "X (don't-care)" in practice.

TCAM memory has a high speed and flexibility, creating programmability at runtime for the proposed architecture. It also serves to update the BPP state, storing headers and data. After completing the matching operation, the unit matches the fields that send the headers to the units that perform the extraction operation. The **Pkt-Buffer** stores fields extracted from other units in BPP architecture to eventually send to other SDN data processing units or to the controller for being used. The **Threat_det** unit is one of the most important modules of this architecture that is able to enforce policies to prevent and detect attacks (Sharma et al., 2017; Sharma and Park, 2018). This unit recognizes, records, and counts certain fields of a package based on the policy of the controller. For example, in this module, if an IP address observes over TCP or UDP packets over a specified limit, it informs the controller, or that repeated ICMP, TCP, and UDP packets are sent to a specified address over a specified limit. Even sometimes with a specific address across the network, packets are required to be fooled, detected, and communicated to the controller by this module. This module controls a packet stream based on the packet behavior on the data plane and works in parallel with other processing modules in BPP architecture.

3.2. Support of blockchain in BPP architecture

In this section, we describe the blockchain structure that our BPP architecture supports in the SDN network for secure communications and how the devised blockchain works as a whole.

The principle workflow of blockchain can be described as follows. SDN architecture has three layers, namely: data plane, control plane and application layer (the top layer). To support blockchain in the application layer, all the SDN switches work together at runtime in which the SDN controller supervises the BPPs. In the application layer, if user A wants to send a file or any asset to user B, then the transaction data is created as a block and it is broadcasted in the SDN network. In the (public) blockchain, any user can join the SDN network and can have their own private and public keys. Other users perform a consensus algorithm, such as Proof of Work (PoW), delegated Byzantine Fault Tolerance (dBFT) or Proof of Stake (PoS), to agree that the transaction within the block is valid. All associated users can be involved in the consensus mechanism and check to validate all transactions. At this step, the block is attached to the chain of blocks. While in the low level, in the data plane, each SDN switch has a BPP which is able to control and check incoming packets including detecting blockchain headers, type and protocol via the information that BPP receives during configuration. Also, BPP concurrently checks the pattern of received packets for detecting attacks, and use the P2P feature of blockchain at the run time. When all the SDN switches work in the network if a BPP in a given SDN switch detects a malicious behavior, it announces this to the SDN controller, and then it begins to P2P communicate with others BPP in the SDN switches for reporting it. In fact, each BPP creates a P2P communication with its adjacent BPP for reporting the detection results, then other BPPs continue this operation for the rest of the network.

Table 1
Contents of a block.

Contents	Data (bytes)	Explanation
Block header	80	Illustrated further
Transactions	Unknown	The transactions
Transaction counter	1 to 9	Quantity of transactions

In Fig 3, the blockchain structure is illustrated with header properties and its transaction list (Antonopoulos, 2014). All transactions in the same block are executed at a single moment (Zhou et al., 2016). BPP can identify the characteristics of the blockchain protocol via analyzing packet data in data plane at SDN network.

Each block in blockchain contains the same elements as header block, block size, counter, and transaction as listed in Table 1 (Antonopoulos, 2014; Sakakibara et al., 2018). Block header is first 80 bytes of the block as specified by the encoding used by "block" messages. Transaction counter (1 to 9 bytes) illustrates whole number of transaction that are contained within the block. Each

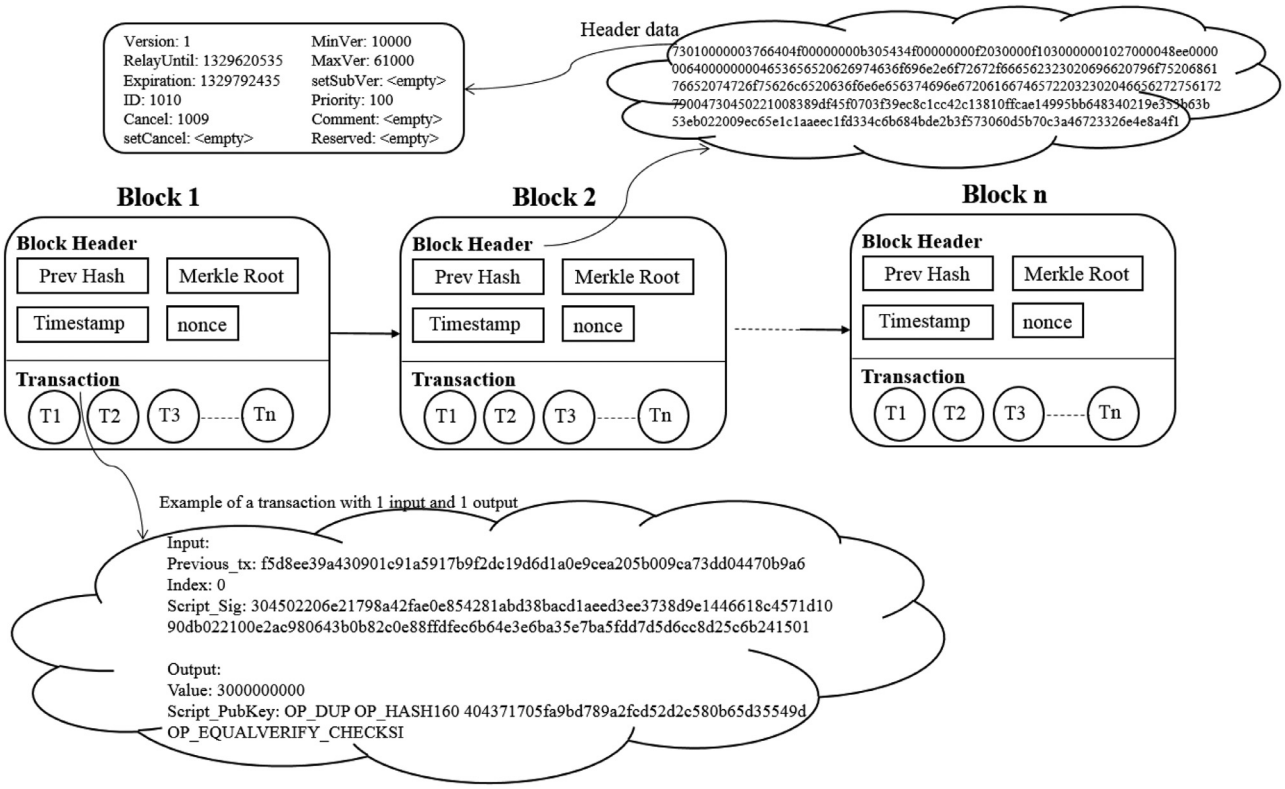

Fig. 3. Structure of blockchain supported by BPP in data plane.

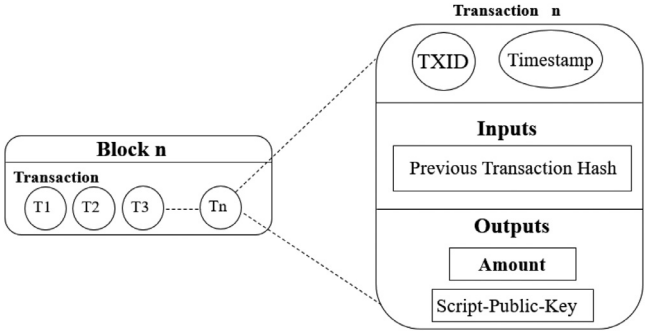

Fig. 4. Structure of transactions.

transaction is not of the identical size, and there is a variable number of transaction in each block. BPP is able to enforce control policies by detecting and extracting these fields for other switching sections such as match+action tables (Flow table) and checking a specific field in these packets for adopting control policies. We can use the programmability of this architecture to support all types of blockchain in any specific application in SDN including, public and private blockchains with customized header or data structure in each block.

The block header is the most important part of the block and includes the information needed to check block validity. Table 2 shows the contents and information in the block header (Sharma and Park, 2018; Zhou et al., 2016).

In block's structure, the previous block header is employed to create a chain link, Timestamp for process time, and Merkle tree to summarize all transactions. Generally speaking, blocks are identified by two properties: header blocks value and positions in blockchain. A transaction indicates that the digital asset is transmitted from the sender to the receiver. Fig 4 shows a structure of a transaction.

Table 2
Contents of the block header.

Field	Size (bytes)	Explanation
Prev Hash	32	Hash of previous block header
Bits	4	Demonstrates the difficulty
Timestamp	4	Current time in seconds since 1970
Merkle Root	32	Hash of Merkle root
Nonce	4	Counter value used in mining

A header in BPP is a set of metadata whose most important elements are the TXID (Transaction ID), TimeStamp (Transaction Time), INPUT (reference to the previous transaction) and OUTPUT for the transfer of the digital asset to the receiver). As the generalization of the BPP concerns, the type of transaction depends on the type of blockchain that it is considered by the control plane in the SDN network, therefore BPP would be able to support different types of transactions based on control plane policy. In this regard, BPP architecture recognizes the structure of the transaction via configuration owing to the fact that it has been designed as a programmable architecture. In the current implementation of the BPP, a transaction contains *date, type of attacks, malicious users*, and *asset*. Each BPP is able to extract this information from received packets into SDN switch. The structure of transaction is defined by P4, and BPP works based on the packet processing functions which are considered by the SDN controller. A BPP can check the network behavior at runtime and prevent the network from emerging attacks and malicious behaviors. Therefore, the extraction and detection results from transactions will be used for validating the accuracy of the attack detection by sending the extracted field of the transaction to the control plane for the attack acknowledge.

When a BPP submits a transaction to a validator in the control plane, these operations are happening (BPP ($T1$) → Control plane (*Validator*), and $T1$ = Vector info (*data, TXID, Input_Attack, type of attacks, malicious users,...*)). The control plane performs

Table 3
Contents of a transaction in BPP.

Field	Data (bytes)	Explanation
TXID	32	Transaction ID
Input_Attack	8	Show the number of attacks
Timestamp	4	Indicates when a transaction is generated
malicious users IP	32	Show the IP address
BPP_ID	8	BPP ID in SDN switch
IP packets	32	Like SRC, DES
Type of protocol	8	Like UDP, TCP
detected attacks	16	Such as TCP AYN Flood, UDP Flood
Amount	8	Transfer asset / sum of money
Outputs	4	Detection attack
Script-Public-Key	Variable Length	Transfer asset to a recipient
Output counter	1 to 9	Amount of outputs

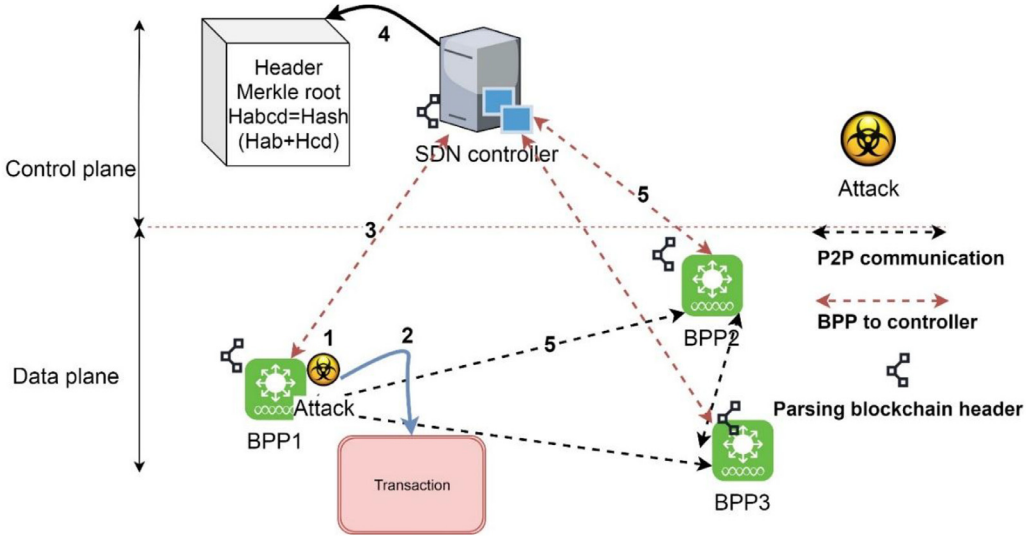

Fig. 5. The validation process flow.

preliminary checks on the transaction in order to decline abnormal transactions (validator check extracted field which sent by BPP, Validator → check *(ValidateTransaction ())*). Then the execution unit in the control plane starts to execute a transaction (Exe_unit → *ExecuteTransaction* (*T1*)). After executing the transactions, it returns the result of it without errors to BPP (Validator → BBP (*Add_T1_WithValidation* ())). When validator of control plane receives a transaction from a BPP, it verifies that the sequence number for the transaction, then it announces it to BPP. In fact, a transaction plays a vital role via BPP in the data plane for transferring important data and security affair of the network. The details of the transaction structure are shown in Table 3 (Sharma and Park, 2018; Zhou et al., 2016).

In our scenario presented in Fig 5, there exist both a control plane and a data plane in the SDN network. Everything starts off with packet processing in the SDN switch by the BPP module, which uses the intrusion detection functions in relation to attack types (e.g. five categories of patterns as mentioned in the introduction section) defined by the SDN controller. At run time, the BPP detects attacks based on functions programed by the SDN controller in Step 1. If the BPP detects a given attack, a transaction will be prepared to be validated and then added to a block in the ledger for that attack (Step 2). Generally, a block contains several transactions. The hash of each block creates a chain communication in the blockchain, and the hash never changes unless the transaction is changed by BPP. The BPP submits a transaction to a validator in the control plane and announces to the SDN controller of that attack detection at Step 3. Therefore, the SDN controller can use the Merkle Tree for the evaluation and correctness of a transaction in the proposed architecture (Step 4). A Merkle tree summarizes all transactions in a block, therefore, enabling a BPP to verify whether or not a transaction is validated in a block when it parses the block header. The SDN controller is able to manage the transactions and generate a block with a hash. In fact, the SDN controller would be aware of all transactions and can check the validity of the transactions via its validator unit. It should be noted the BPP uses insights from the Merkle Tree and the Merkle path for validating the transactions. The BPP parses the header of the blocks and extract the Merkle Tree from the root. Then, the SDN controller based on the transaction which received from the BPP starts to check the correctness of these transactions, and a Merkle path creates with related the transaction and it sends to the BPP. If the comparison is successful, the transaction will be marked as validated in the whole of the network and it is transparent for other BPPs in our architecture (Step 5). At the same time, the blockchain will be updated and if it is necessary, the BPP will be re-programmed concurrently.

Also, in the SDN switches, BPP has P2P communication to establish a connection with other BPPs to facilitate intrusion detection, and our architecture utilizes the cryptographically secure mechanism for obtaining a publicly verifiable and immutable sequence of records by blockchain. In fact, each block is identified by its cryptographic hash and also contains the hash of the previous block. In our context, the BPP is not required to reproduce the detection results since the results of attacks detection are transparently available in the blockchain, and the SDN controller can manage stored results to define new rules for re-programming the BPPs in the SDN switches based on the structure of attacks.

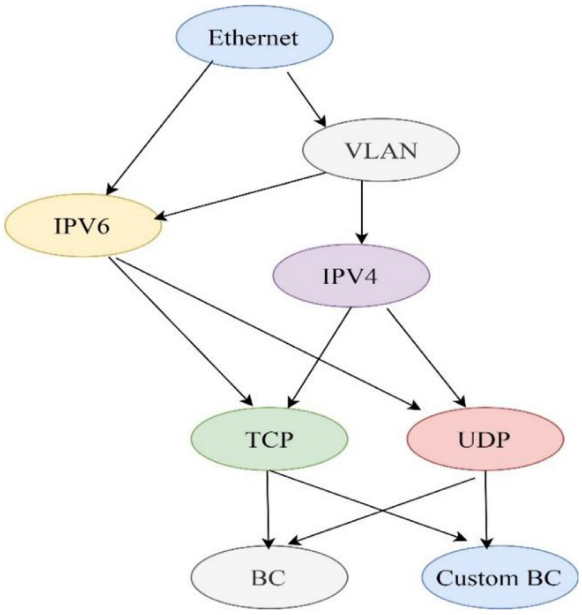

Fig. 6. Parsing graph with the blockchain header.

```
header_type  Blockchain_h  {
fields {
      Previous Hash: 32;  /* 062a2789d7e82d7b33c838352bfba*/
      Merkle Root  : 32;  /* 2ea8f4e784f68e5dd9eedde5a62866e3fadfa*/
      Time :    4 ;              /* 370602521*/
      Bits  :    4 ;              /* a011338 */
      Nonce :     4 ;            /* 332491547*/
}

    Transaction  Tx_h  {
    fields {
         Inputs  :  4;            /* 1 */
          Previous Hash  : 32  ;  /* a52c458c3a4e39b63d4a7bdcf */
          Tx Out Index   : 4  ;   /* 0 */
          Script Length  : 4 ;    /*  108*/
          Script Sig   :  16  ;   /*47ad3044220447d5ae4624357f6b136*/
          Outputs      :   4  ;       /* 2*/
          Value        :   8          /*32500000*/
          Script Len   :   8 ;        /* 25*/
          Pubkey       :    16 ;      /*bf18e9cc4c287764e29759b6*/
          Timestamp    :  4 ;         /* 0 */
     } }
```

Fig. 7. Description of the block header with P4.

3.3. Describing BPP architecture via P4

P4 is an open-source high-level language for packet processing and P4 consortium is in charge of its maintenance (Zhou et al., 2016). This language is used to collaborate and contribute to realizing ideas in SDN and open source ecosystems (Open Networking Foundation (ONF) and Linux Foundation) (Sakakibara et al., 2018). P4 is domain-specific, and we are using it to customize packet-level data processing functions to support blockchain, which has BPP independent protocol-specific attribute. This language allows us to define the required headers and fields so that the programmer can describe the function of packet processing independent of the hardware platform. In general, P4 addresses packet processing on the data plane from 5 perspectives including Packet Parser, Header Formats, Table Specification, Action Specification, and Control Program (Sakakibara et al., 2018; Bosshart et al., 2014; Bosshart et al., 2013).

The parsing graph actually shows the order of the headers known by a switch at the data plane. A parsing graph is a finite state machine (FSM) detecting packet headers, respectively. This detection is based on field values and goes into another state in the machine state. Even sometimes it can be processed multiple times in the FSM. Fig 6 shows the parsing graph in BPP. In this graph, the customized blockchain header and blockchain header are also supported.

The definition of the packet header in this language is like to that of the C language, which we defined for blockchain based on the data structure of the header by P4. Regarding the header types, descriptions and definitions of the fields are done. Fig 7 depicts the description of headers, including header type and transactions with their fields. The header blockchain includes fields such as, previous hash, Merkle tree, time, bits and nonce, which is the width of the fields by bytes.

Metadata stores in-packet information, which is not shown; for example, in the Ingress port. Metadata provides information about a data source or objects to enhance security. We use Metadata to achieve security in the BPP, which specifies the status of blocks, public and private keys.

P4 forwarding mechanism is shown in Fig 8. SDN control plane is the top of this hierarchy, which is managing the SDN network via P4 configuration. Complier P4 must map rules from SDN controller to data plane especially in BPP. First, the inbound packets are parsed as it is specified in the configuration and the BPP is checking control policy. The egress and the ingress processing are worked based on the field of headers extracted by BPP. P4 also forgives the chance to hold extra data among the stages, in fields called metadata. We can add processing specific (control program) to BPP via P4 complier.

3.4. P4 based security

We use the features of P4 language to create the P4 based security in data plane. It tries to indicate if the traffic flows are IPv4 or IPv6 or VLAN, TCP or UDP, BC or custom BC header. Hence, in our actions, such protocol header fields are applied fully. Flood attacks can be identified and, by checking the packet sending rate, it will enable us to block individual users or subnets from the network. P4 language provides meters where these meters are used to calculate the number of demands per second from a IP address or subnet.

As shown in Fig 9, after parsing incoming packets is to check whether any packet's filed exists on the forbidden list. The forbidden list depicts one example of the stateful memory, or the users (IP addresses and MAC) that break the rules, or making highly significant packet rate. The packets matched in such a list would be omitted instantly. P4 provides 3 paths to retain stateful memories. These memories are ounters, meters and registers. Counters are used to measure the number of unsuccessful connection attempts (Syn flood or portscan), the packet rate that each host makes (DoS), or the number of bytes the host has transferred. Direct counters are moreover gained in the background passively. The control plane can provide the number of packets that are sent by each host, so direct counter is used in the BPP description. The rules are elaborated in the BPP to attend some block protocols that are contained in the list of protocols that are dropped, permitted or forbidden. As our use of a list as a forbidden list is part of the security, i.e. everything is dropped by default, the list merely has the allowed protocols and the ones we want them to be forbidden. To forbid the stateful information on the users, registers must be used. In the BPP, registers are those fields that are like the ones come from parsing. When register is set to one, the packet is going to be forbidden and dropped.

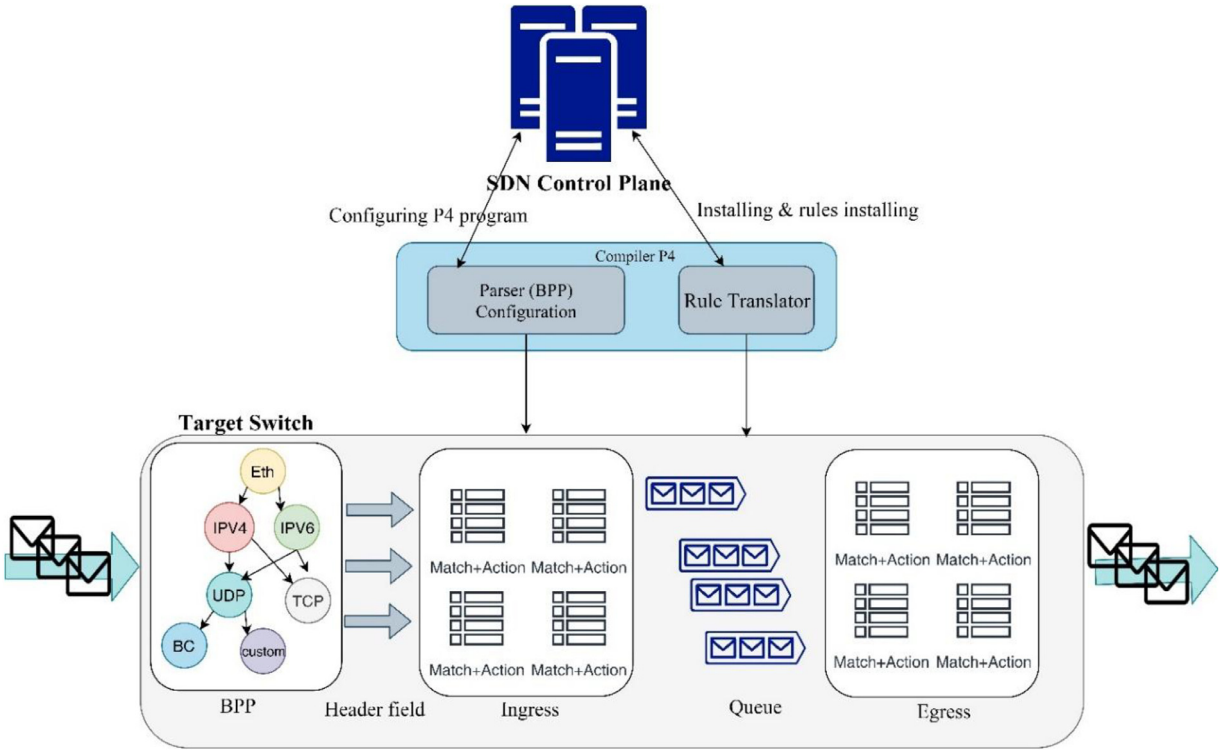

Fig. 8. P4 forwarding model in our approach.

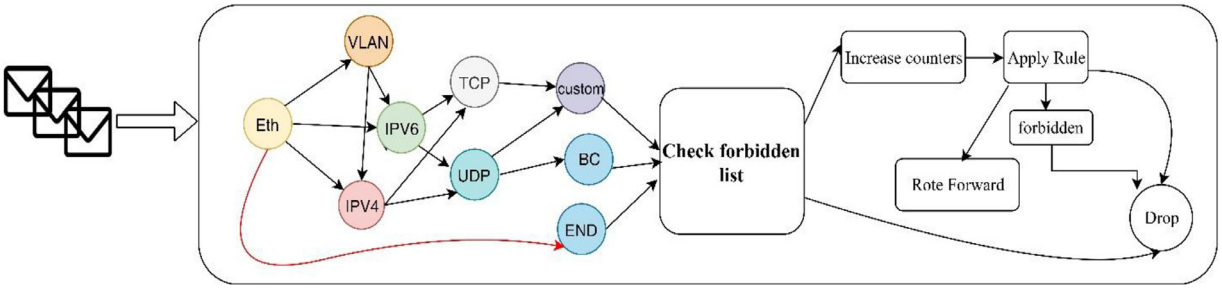

Fig. 9. The operation of BPP in control program.

BPP is a network security system. This system controls and monitors the egress and ingress network traffic. Based on various security rules, we are able to use the stateful filters, packet filters and application level attacks. In the packet filter, BPP decisions are made about different header fields, such as IP, ports and protocols. BPP works with a set of rules and it inspects each packet. It then applies the specified action for each matched rule. It mainly works with the OSI model (first 3 layers). Stateful filters operates up to layer four of the OSI. It has a stateful memory, which is necessary for recognizing certain attacks, such as DoS. Application level attacks understand the application protocols, such as the blockchain, and they can examine the deep content of a package. It can identify various application level attacks. For instance, in case a person is to SSH into a blockchain port, then BPP blocks the connection because the requests are not valid in the blockchain requests. These inspections cannot be done without checking the content of the packets

4. Compiling and generating BPP from P4 description

After defining the characteristics of BPP, it is compiled to support the blockchain data structure by P4. We used the behavioral model of the P4C compiler (Sivaraman et al., 2015; P4 Language Consortium 2015), called BMV2 (P4 Language Consortium 2014), which is modular and serves as a medium for implementing the P4 code. In Fig 10, the compilation methodology is shown for implementation which divides it into three sections: Target, P4 complication and RTL implication. The Target section describes the structure and characteristics of BPP with P4. In P4 compilation, compilation of P4 code takes place in the behavioral model of the P4C compiler, BMV2 (P4 Language Consortium 2014). By adding the PD library to BMV2, we are able to generate *C++* code from BPP. In RTL Implementation, we use the *C++* descriptor to implement hardware from (Vivado HLx 2018) vivado HLS 2018.2 to generate RTL code. After synthesizing the RTL code, we implemented it on the FPGA, and eventually the bit stream generated by vivado is used which is compatible with the Xilinx FPGA.

5. Evaluation and results

In this section, to assess BPP architecture, we simulate and implement the hardware part of its parser using vivado design suite 2018.2 and the ZedBoard Zynq FPGA (Bibi 2019). The hardware board used in the study is manufactured by Xilinx, which has the following features:

- Enjoy Xilinx Zynq-7000

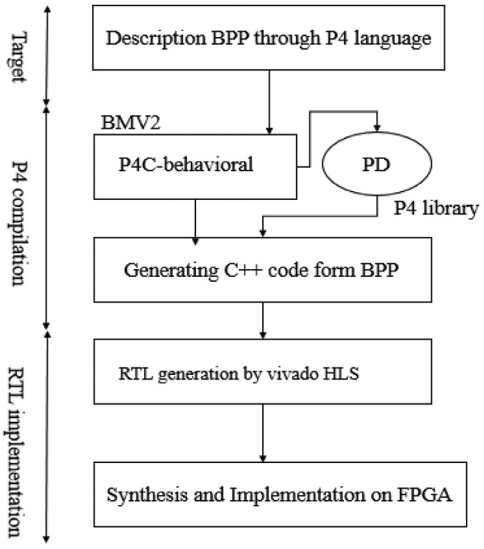

Fig. 10. Compilation methodology of BPP.

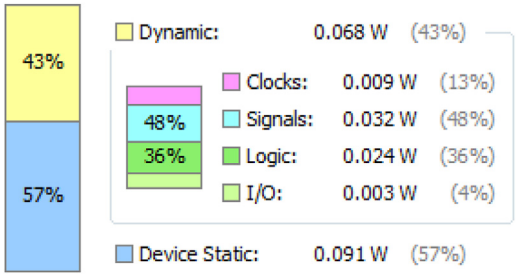

Fig. 11. Static and dynamic power consumption by BPP.

- Having Cortex™-A9 (dual-core ARM)
- 10/100/1000 Ethernet port
- Enjoy On-board USB JTAG Programming
- I/O development
- devices ADV7511 powerful 225 MHz
- USB-UART and USB OTG 2.0
- Using 512 MB DDR3
- Having 256 MB Quad-SPI Flash
- benefits of 4GB SD card
- Enjoy analog devices ADAU1761, Low Power, 96 kHz

We evaluate the proposed architecture in terms of its resource utilization, and power consumption in Section 5.1 (Implementation of BPP), Security analysis of mathematical model and attack detection performance are examined in Section 5.2, and bandwidth efficiency, probability of false alerts and probability of absence of real alerts are evaluated in Section 5.3, hardware comparison of BPP with other architectures is shown in Section 5.4.

5.1. Implementation of BPP

BPP was implemented on ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1) Xilinx in FPGA, and that has high capabilities and features to enhance BPP's performance (Bibi 2019). The board has programmable blocks and logical on the system level, and allows us to have a great of performance within the FPGA platform.

After the synthesis and implementation of BPP architecture on FPGA, the amount of material and volume of consumable hardware, clock speed and resource consumption of this board which were used are reported in Table 4. Based on the analysis graph

Table 4
Design Summary/Reports.

Resources	Estimation	Available	Utilization
Number of LUT Elements	5801	53,200	%10.9
Available IOBs	12	200	%6
Number of Flip Flops	3012	106,400	%2.8
Number of Block RAMs	16	140	%11.4
I/O Pins	256	484	%52
Clock Speed	400 MHZ		
Junction Temperature	24.8 C		
Consumption power	159 mW		

in Fig 5, P4 uses the optimized functions which we used at the time of compilation for BPP, improving performance in our work. The total BPP resources consumed to support blockchain have been around 10.9% LUT 2.8% FF and 11.4% of the memory blocks, while still large resources are available for logical implementation form ZedBoard.

Contrary to ASIC, the FPGA has no limitation to support TCAM for implementing the exact match. As can be seen in Table 5, up to a 256-bit key TCAM can be implemented, which is associative memory with minimum resource utilization. If we consider the 256-bit key for TCAM memory, then as many as 100 K entries on the ZedBoard Zynq FPGA is appropriate. In Table 5, the amount of resources consumed by 1024 entries is based on different keys in the hash-based method.

The total power consumption of BPP (as shown in Table 4), is 159 mW and the temperature of the connection point on the board is 24.8 Celsius. The architecture, based on the static and dynamic power consumption, is 57% and 43% on the chip as shown in Fig 11. Static power is used when there is no circuit acting. For instance, when the clock and the input neither have active inputs, Eq. (1) is used to calculate the power that is used by a flip-flop.

$$\text{Static Power} = \text{VCC(V)} * \text{ICC(mA)} \tag{1}$$

Dynamic power is used when there are active inputs. When there are active inputs, as a result there are capacitances charging and discharging and the power increases. Eq. (2) is used to obtain the dynamic power, including signal, clocks, logics and I/O.

$$\text{Dynamic Power} = f * C * \text{Vcc}^2 \tag{2}$$

Eq. (3) is used to obtain the total power usage of an FPGA device, which is broken down into dynamic power and static power.

$$\text{P}_{\text{TOTAL}} = \text{Dynamic Power} + \text{Static Power} \tag{3}$$

5.2. Security analysis of mathematical model

The availability and reliability of SDN services are being threatened via attacks likes DoS attack. There must be effective models for the detection of the DoS attack. Hence, a mathematical model is proposed according to the multivariate correlation approach for detection of the attack from the packets traffic inspected by BPP. Statistical properties are used to reverse the behavior of the SDN traffic. DoS attack is a type of interference which is used for exhausting the resource losses and its traffic has a different behavior from that of the normal network traffic. Hence, to announce the difference, there can be used the statistical properties, such as repetition of the packets and specific fields. To present more convenient statistical properties, a novel multivariate correlation analysis approach is presented in the BPP architecture. It hires Euclidean distance to extract correlative data. In an observed data object, the data are named inner correlation among the features. Considering the arbitrary dataset $P^T=[P_1{}^T P_2{}^T P_3{}^T ... P_n{}^T]$, where $P_i{}^T = [z_1{}^i z_2{}^i z_3{}^i ... z_m{}^i]$ $(1 <= i <= n)$ is the ith m-dimensional traffic record.

Table 5
Resource utilization TCAM memory on ZedBoard Zynq.

Resource utilization	Key size	Entries	LUTs	Flip-Flops	BRAM
TCAM	8	1024	(178/53,200)	(122/106,400)	(2/140)
Model	32	1024	(356/53,200)	(256/106,400)	(4/140)
	128	1024	(505/53,200)	(450/106,400)	(5/140)
	256	1024	(879/53,200)	(764/106,400)	(7/140)

The dataset is as follows:

$$P = \begin{bmatrix} z11 & \cdots & z1m \\ \vdots & \ddots & \vdots \\ z1n & \cdots & znm \end{bmatrix} \quad (1a)$$

In the above equation, in the *i*th traffic record, $z^i{}_1$ is the value of the *l*th feature. *l* and *i,* respectively, differ from 1 to *m* and from 1 to *n*. The inner correlations of the *i*th traffic record on a multidimensional space are explored by first transforming the record $P_i{}^T$ into a new *m*-by-*m* feature matrix $P_i{}^{'}$. It is simply done by multiplying an *m*-by-*m* identity matrix *I,* shown in (2).

$$P_i^T I = P_i^{'} = \begin{bmatrix} zi1 & 0\cdots & 0 \\ 0 & zi2 & 0 \\ 0 & 0 & zim \end{bmatrix} \quad (2a)$$

The features of the record $P_i{}^T$ denote the elements on the diagonal of the matrix $P_i{}^{'}$. The columns of the matrix $P_i{}^{'}$ each is a new *m*-dimensional feature vector specified by (3)

$$Z^{iT}{}_j = \left[Z'^i{}_{j,1} Z'^i{}_{j,2} Z'^i{}_{j,3} \ldots Z'^i{}_{j,m}\right] \quad (3a)$$

In the above equation, $Z'^i{}_{j,k} = 0$ if $j =/ k$ and $Z'^i{}_{j,k} = z^i{}_j$ if $j = k$. The parameters satisfy the conditions of $1 <= i <= n$, $1 <= j <= m$ and $1 <= k <= m$. Therefore, the following equation can be used to write the m-by-m feature matrix $P_i{}^{'}$.

$$P_i{}^{'} = \left[Z^i{}_1 Z^i{}_2 Z^i{}_3 \ldots Z^i{}_m\right] \quad (4)$$

When the transformation is ended, the Euclidean distance (ED) can be applied to extract the correlation between the feature vectors *j* and *k* in the matrix $P_i{}^{'}$. The following can be used to specify this correlation

$$ED^i{}_{j,k} = \sqrt{(Zij - Zik)T + (Zij - Zik)} \quad (5)$$

Where $1 <= i <= n$, $1 <= j <= m$ and $1 <= k <= m$. In practice, (5) can be nevertheless be simplified and rewritten as (6) to reduce the computational complexity.

$$ED^i{}_{j,k} = \begin{cases} \sqrt{(Zij - 0)2 + (0 - Zik)2}, & j = /k \\ 0, & j = k \end{cases} \quad (6)$$

As a result, the Euclidean Distance Map (EDM) is used to define the correlations between features in the traffic record $P_i{}^T$

$$EDM^i = \begin{bmatrix} EDi\ 1,1 & \cdots & EDi\ 1,m \\ \vdots & \ddots & \vdots \\ EDi\ m,1 & \cdots & EDi,mm \end{bmatrix} \quad (7)$$

EDM is a symmetric matrix (where $ED^i{}_{j,k=}ED^i{}_{K,j}$) and the distance from a feature vector to itself is zero ($ED^i{}_{j,k} = 0$, if $j = k$). to reveal the inner correlations, the upper or the lower triangle of the matrix suffices. Therefore, we can simplify and convert the EDM into a new inner correlation vector that contains only the lower triangle of the EDM, shown in (8).

$$EDM^{iT}{}_{Low} = \left[ED^i{}_{2,1} ED^i{}_{3,1} \ldots . ED^i{}_{m,1} ED^i{}_{3,2} ED^i{}_{4,2} \ldots ED^i{}_{m,2} \ldots ED^i{}_{m,m-1}\right] \quad (8)$$

The inner correlations of the dataset X can be shown by (9).

$$EDM_{Low}{}^T = \left[EDM^1{}_{Low}{}^T EDM^2{}_{Low}{}^T \cdots EDM^i{}_{Low}{}^T \cdots EDM^n{}_{Low}{}^T\right] \quad (9)$$

Table 6
Number of records of attacks.

Normal	U2R	DoS	R2L	Probe
60,410	4100	224,020	16,200	500

The variations of the network behavior as caused by DoS attack can be revealed by applying the inner correlations. Moreover, the distance measure helps us facilitate the analysis approach for tolerating the issue of linear change in all features.

The Detection Rate measurement (DRM) is used to evaluate the efficiency of the multivariate correlation approach and the False Alarm Rate (FAR) is computed, respectively, by the use of Eqs. (10) and (11).

$$DR = TP/TP + FN \quad (10)$$

In the above equation, the True Positives (TP) are attack traffic logs that are classified as attacks, and False Negatives (FN) are those attack traffic logs that are ordered as lawful.

$$FA = FP/FP + TN \quad (11)$$

Where False Positives (FP) denote the lawful traffic logs that are classified as attack, and the True Negatives ((TN) are the lawful traffic logs that are ordered as lawful.

KDD CUP 99 alternatives dataset was used to perform the security evaluation of BPP (Divekar et al., 2018). There are 5 classes of patterns for the KDD-99: DoS, Normal, U2R (User to Root), Probe (Probing Attack), and R2L (Remote to Local). The detection rate and FP rate are the two momentous metrics for the evaluation of an IDS. It is aimed to arrive a high detection rate when retaining a low FP rate. First, all records are filtered via the labels of DoS, U2R, R2L, Normal from the%20 labelled dataset. After that, they are further classified into various clusters based on their labels. Table 6 show the explanation of the filtered data.

5-fold cross-correctness and 24 continuous features were executed for evaluating the detection performance of the proposed approach. With respect to various types of traffic, such as ICMP traffic, UDP and TCP, the norm indexes were built. The Normal records were engaged in the training phase, it is while the attack records and the Normal records were involved in the test phase. Fig. 12, based on detection approach in BPP Operating Characteristic, shows the performance of multivariate correlation analysis.

As shown in Fig 12, in most of the cases, the detection based on multivariate correlation analysis approach performs well. The Normal records detection rates ascent from 97.92% to 99.13% along the increment of the records. The Dos and Probe attack records are recognized entirely. They are not affected by the change of the threshold. The figure reveals the tradeoff between the false positive rate and the detection rate. There is a common trend in this process that is when larger numbers of false positive are tolerated, the detection rates of all attacks increase. In case of requirement of high detection rates, a comparatively high false positive rate of 2.08% must be endured. 2.08% false positive rate is however in the acceptable range and they are even better than other detection approaches. Multivariate correlation analysis based detection approach has a false positive rate of 2.08% and a detection rate of

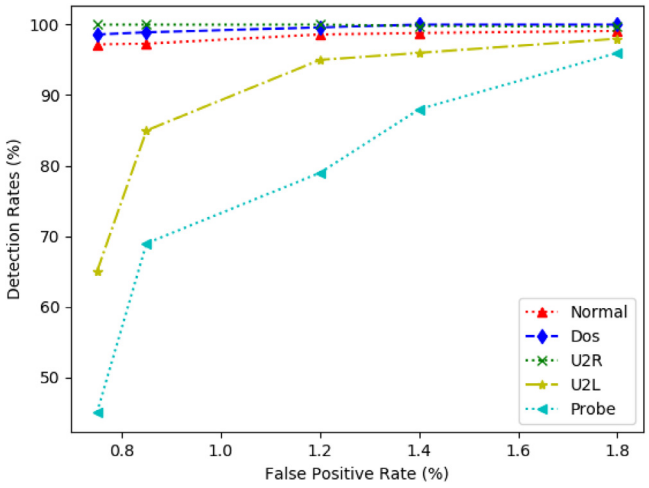

Fig. 12. BPP Operating Characteristic for the detection of attacks.

99.96%. Generally, the evaluation results have proven the multivariate correlation analysis and it can extract the statistical attributes correctly so as to represent the behavior of the network traffic. There are promising outcomes for the application of the multivariate correlation analysis in DoS discovery.

5.3. Analysis of simulation

To identify and recognize the DoS attack and to inform the control plane over the possibility or the occurrence of an attack by the BPP's parser, a test bench was designed. In this test bench, we produced one million different packets and features similar to common attacks. In simulation, packets with different variables were based on the parsing graph. The test bench was written to test the functions of the various BPP sections to identify the headers and extract the required fields, especially in the blockchain structure. Some attacks, such as UDP Flood (Singh and o. E. S. Juneja, 2010), make UDP packets flood proof on randomistic ports also, the Ping of Death attack (Singh and o. E. S. Juneja, 2010) is a situation, in which the attacker sends a lot of ping packets to a system. SYN Flood attack (Eddy, 2007; A. Yazdinejad et al., 2019)generates TCP packet SYN protocols to occupy the server's memory space.

The **Threat_det** unit can easily enforce policies to prevent and prevent these attacks by informing the controller. In Fig 13, the packets of inputs to BPP architecture are used to detect and extract its fields. After executing the simulation according to the figure, Pkt-in reads packets from the input to process headers. Information such as protocol numbers, source and destination addresses, and for IP packets (TOS, SRC, DES, Type (UDP, TCP)), ARP (Src, Des, Op), MPLS (Lable, TC, TTL), Outputs IPv6 (Src, Des), blockchain (Pre-Hash, Hash, Trans, timestamp), block header (Prev Hash, Merkle Root, Timestamp, Nonce, Bits) were extracting and checking for detecting attacks and sending to match table on SDN Switch, then the switch will be able to accept packet actions.

In Fig 14, a **Threat_det** signal is one and sent to the controller to inform the attack caused by the traffic pattern and the packets entered as an attack. It can been seen from this figure other fields that were extracted by BPP for detected attacks or protocol fields including, TCP AYN Flood, UDP Flood, IP_Protocol, IPV6_SRC, IP_Des. We used the simulation to ensure the correctness and accuracy of the results by sending packet flows that include are attacked, and understanding the running application are running secure or not by using controller policy on incoming packets.

Using the test bench, the architecture (BPP) was compared and evaluated with an available OpenFlow network in order to recognize its defense effects. Some packets were used to send-out a UDP floating attack in the data plane. The bandwidth of the network was measured with and without the flooding attacks that were generated at several speeds to the BPP on data plane. In hardware environments, the impact was evaluated on the bandwidth with and without the BPP model. The results of this test bench are shown in Fig. 15. In this test, the bandwidth initiated at 14.5 Mb/s both with and without the use of the BPP model with any attack. In this testing, it was understood that the bandwidth went up to half without using the BPP. It was occurred when the attack rate of the packet arrival touched 2000 packets/s. It then started to malfunction after the attack rate touched 6000 packets/s. using the BPP model, the bandwidth was held above 14 Mb/s up until the packet arrival touched 2600 packets/s. Then, a decrease was observed in the bandwidth due to the BPP announced to controller for detected threats.

The accuracy rate of the discovery of BPP was evaluated under two different parameters, one was in the real-time identification of the attacks and the other when various traffics with specific defects were present in the system. There is an ability with the BPP architecture which can identify every attack instantly. For the case

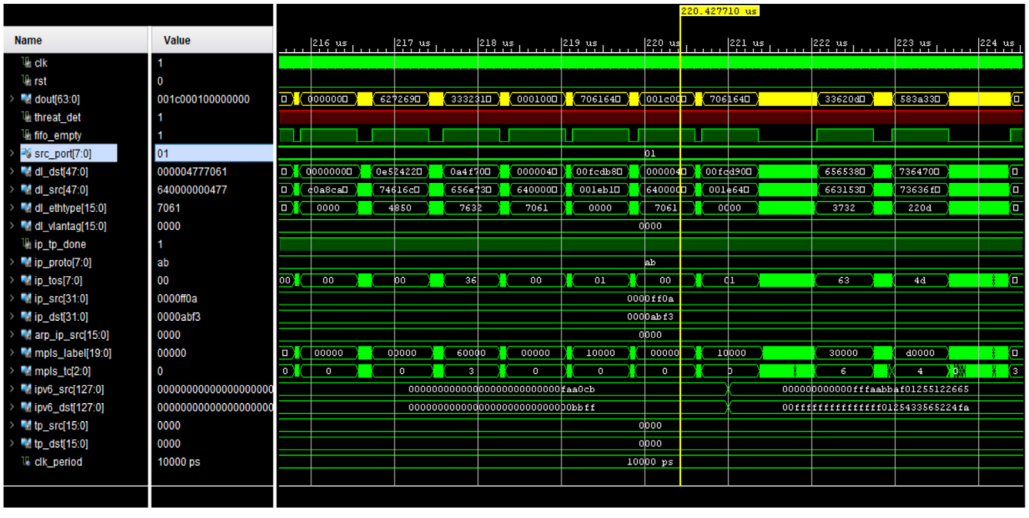

Fig. 13. Stimulation of BPP input packets processing.

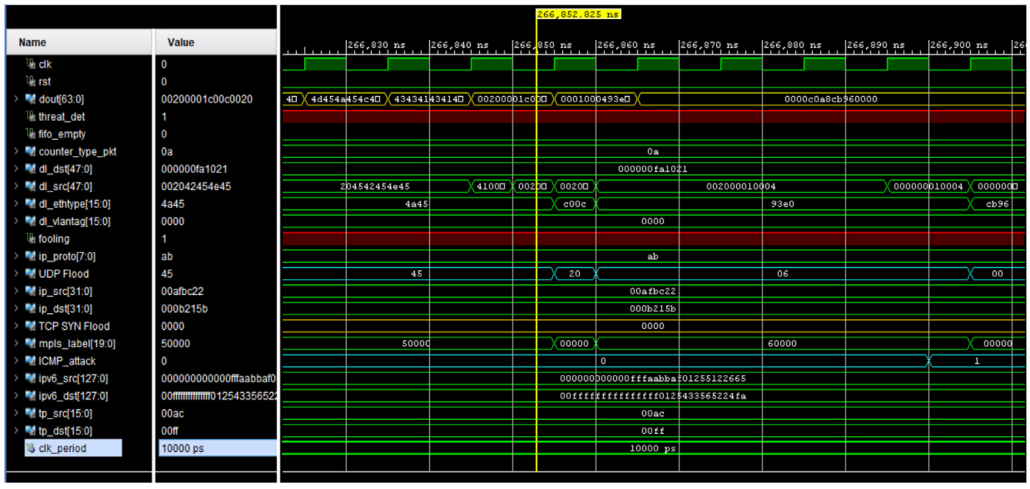

Fig. 14. Simulation of input packets for detecting the attacks.

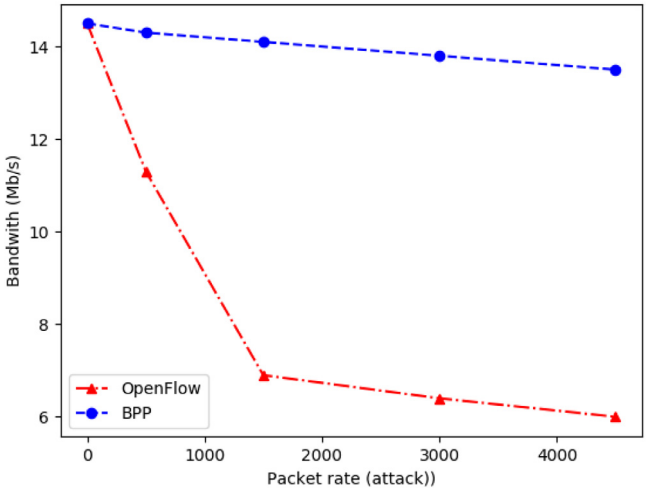

Fig. 15. Effect of bandwidth on BPP during attack.

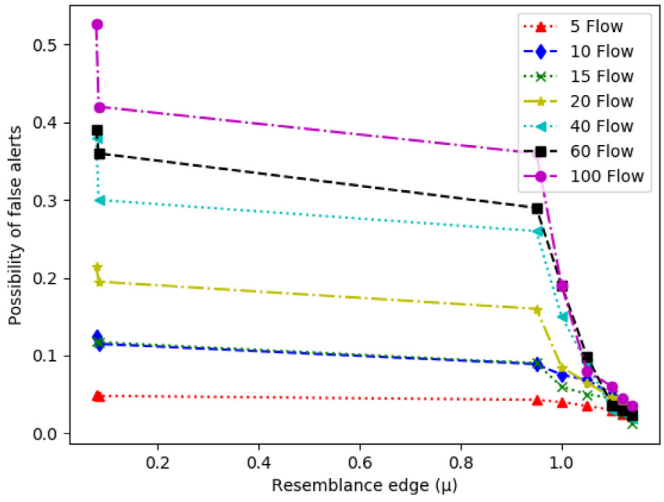

Fig. 16. False alerts probability with variation in μ and flows.

of real-time identification, there were used the synthetic faults in equal with the fit traffic. With 4 K on our work, the discovery time was shown. It was a time necessary for issuing of an alert for the moment the delinquent packet was received by the BPP architecture.

The custom traffic generator was used. This generator can generate 2500 flow. One can easily identify the ARP attacks and fake topology when the BPP process the packet messages. The detection times might undulate. It is because in order to identify the DoS/DDoS attacks, the BPP runs the validator and, therefore, there is an increase in the flow size. In another case, the number of flow entrance is incremented. Then, DDoS, ARP stabilizing, and fake topology attacks are leaded throughout the distributed blockchain network. Each detection is repeated more than 10 times and it is considered that the BPP effectively recognized each of the issues under the specific topologies. Using conflicting TCP streams, a pessimistic scenario was run on the false alerts as appointed for a given μ. Oscillations are created on the exhibition character of the TCP in the flow thus to cause changes in the switches along the flow path. This would raise some cautions. It was observed that with the increase of μ, the probability of false alarms occurrence decreases, as shown in Fig. 16.

Because of the lack of a true positive, the summon and accuracy are zero. These experiments helped us observe that there were 8 alarms out of 10 competing flows over 4 min at the default value of $\mu = 1.02$. This experiment was also performed on our test bench and comparable results were taken. In order in order to specify the absence of real alerts for a given μ, the ratio between the numbers of checks was defined that did not raise alerts to the total number of the checks that raised alerts during verification. Among the BPP for controlled flows, the above metric was measured. As it is observed in Fig 17, it was seen that the absence of the real alerts within verification would lead to the increase of the μ. For a given μ, the accuracy and recall are identical. This accuracy is equal to one minus the possibility of the absence of the real alerts at each data point.

5.4. Hardware comparison of BPP with other architectures

As discussed, so far, no work has proposed to support hardware at SDN data plane for blockchain. In this particular evaluation, we compared the work done on the data plane with having the packet processing feature. There are different hardware platforms that differ only in the amount of resources available on the board. Table 7 compares the comparison between BPP and other tasks in terms of consumption resources and performance.

Our architecture does not depend on a particular version of the data plane and protocol switches. Owing to the BPP's programma-

Table 7
Hardware comparison of Packet parser.

Work	Performance				Resources		
	Data Bus [bits]	Frequency [MHz]	Throughput [Gb/s]	Latency [ns]	LUTs	FFs	Slice Logic (LUTs+FFs)
Gibb et al. (2013)	64	172.2	11	N/A	6946	2600	9546
Benácek et al. (2016)	512	195.3	100	46.1	10,103	5537	15,640
Santiago da Silva et al. (2018)	320	312.5	100	25.6	7831	13,671	21,502
A. Yazdinejad et al. (2018)	320	330.9	105	24.8	7259	3628	10,884
BPP	320	400	118	18.3	5801	3012	8813

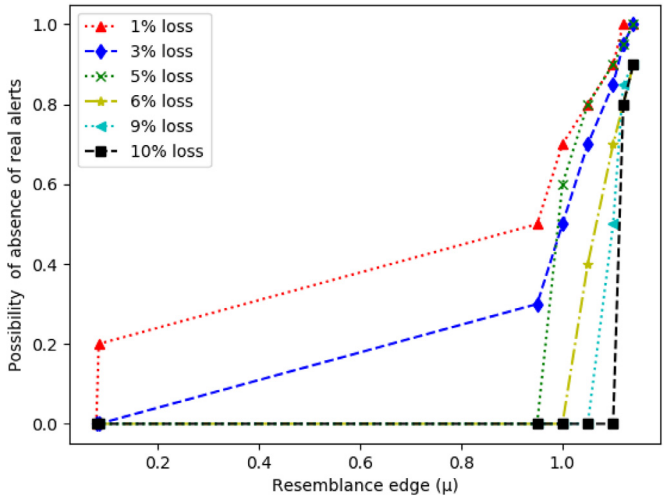

Fig. 17. Absence of real alerts with μ and loss rate.

bility, it supports different protocols of blockchain and customized blockchain -headers at runtime, which are implemented through the network control plane by network administrators. As shown in the table, BPP architecture is faster in terms of processing speed and has less hardware volume than FPGA, resulting in high performance at SDN data level.

5.5. Implications

To clarify the practical and theoretical implications of applications of the proposed architecture, we discuss some of its viable features here:

- **Programmability:** This is one of the sought-after aspects in the network architecture since the architecture of the current network (non-SDN) is static and none-programmable especially in the data plane. There are many devices which are not programmable in the data plane including switches. When the switch is programmable, it would be able to change the parse graph at run time and this operation is done by PP. Our proposed BPP is programmable and is able to support each protocol and parse graph. As a result, BPP can support each role from the control plane in the SDN network. Also, we can test and define new headers and protocol since BPP is programmable.
- **High-level description:** We designed and described BPP via P4 which is a high-level language. P4 is domain-specific, and we are using this feature to customize packet-level data processing functions to support blockchain protocol, which has BPP independent protocol-specific attribute. Also, we are able to describe any function of packet processing independent of the hardware platform that means that in addition to implementing on FPGA platform we can implement BPP on other platforms like ASIC. By providing BPP, developers can focus on network applications and packet processing patterns, but not on the detail and complexity of data plane in the SDN network.
- **Security:** There is widespread concern regarding security and secure communication in the data plane at the SDN network. Inspired by the security-by-design capability of the emerging blockchain technology, BPP architecture supports detecting attacks and the blockchain structure in the data plane for secure communications. The attack detection process is done by BPP based on the packet structure and control plane policy. BPP has not only the ability to detect attacks but also the awareness of the attacks to the control plane. High programmability and flexibility of BPP can allow running trusted applications of blockchain in the data plane. Therefore, BPP increases the level of safety and reliability of the SDN network.

By and large, we can use BPP in the infrastructure of the network for admins since it is a general propose architecture for improving and removing the limitation of the insecure and static infrastructure of the network. BPP has a direct and indirect influence investigation research via implementing it in any type of hardware platforms for supporting of customizing packet headers, protocol, and rising security. We can examine new ideas in the network as it supports new technologies like SDN, Network Functions virtualization (NFV) and blockchain protocol and provides easy definition of the packet processing functions for admins in the application level.

5.6. Limitations

In this section, the limitations or challenges of using blockchain on the proposed BPP architecture are discussed from various points of view:

- **Scalability**: As has been mentioned blockchain has a wide and distributed network which is large scale, and scalability is one of the most important and critical steps in the growth of blockchain. Also, the issue of scalability is a challenge for SDN network. As a result, controller plane in SDN network has duty for handling scalability. If we see the support of blockchain from this angle, BPP does not have any role to support scalability due to the fact that the functions and applications of it work in top-level and BPP works in the data plane. We assume that in the control plane layer, the issue of the scalability is handled (SDN network equipped with distributed controllers and switches), so BPP architecture does not focus on this feature in the data plane. In fact, BPP supports structure of blockchain at low level in the SDN network. BPP focus on blockchain protocols and structure of blockchain headers in the data plane based on the goals of administrators at the top level. Since the controller has the duty of management and monitoring in the SDN network, it must use a multi-layer structure to overcome this issue.
- **Smart contract applications:** Smart contract (Parizi et al., 2018) is one of the important features of the blockchain which can essentially run on top of a blockchain network. Smart contract contains a set of rules and self-executing statements

which is running in top-level at the SDN network. Owing to the fact that BPP architecture works on the data plane, it cannot directly focus on the blockchain applications (i.e. smart contracts) that are running at the top-level, but BPP is able to control these applications via their port numbers, protocol types and headers. As a result, if one wants to check the behavior of a smart contract by BPP, we will have to reconfigure BPP via the P4 program. Having said that, when we have a considerable number of smart contracts it could be challenging for data plane especially for BPP.

- **Consensus mechanism**: One of the issues of the blockchain that it is a well-known issue is consensus protocols (Nyaletey et al., 2019). In regard to POW, it is the original consensus algorithm in a blockchain that is used to confirm new blocks to the chain. The consensus mechanism happens at the application layer among all participations in the SDN network, therefore BPP architecture does not have any role and affects consensus due to the fact that BPP only can consider the structure and protocols of blockchains in the data plane and focus on packet processing functions and security attack. Here, the control plane is responsible for checking consensus mechanism and defining algorithm among all participations in the SDN network.
- **Storage:** Storing data on the blockchain means that the information is saved by each node in the network. It also determines that users cannot remove anything. As a result, data storage requires a cost on a decentralized network. Due to the fact that every user has to save more and more data, storage usage could be an issue for data plane. We essentially do not provide any external storage neither in SDN switches nor in BPP since we cannot consider a huge volume of storage in the switch or BPP. A simple remedy for this could be a storage on the cloud .

6. Conclusion

Security is one of the most notable and applied concepts in SDN and should be built into control and data planes. Moreover, the data plane should be able to support control-level policies to examine the behavior and patterns of the input packets, especially in the switch. In fact, the critical role of packet parser is at the data plane, which contributes to flexibility and programmability. The data plane should be able to apply security controls from the control plane to detect attacks from a packet structure. Given the widespread use of blockchain technology, the data plane should be able to identify its fields for use at data and even control planes. In addition, FPGAs are increasingly important in today's networks, making them flexible and programmable for SDN and NFV networks. A blockchain-enabled Packet Parser (BPP) architecture was designed to capitalize on both secure decentralized ledger technology and FPGA hardware which in addition to supporting detect attacks by the behavior of packets and the structure of the blockchain, has increased the speed and reduced hardware resources than other works.

Future research could include the new and optimal algorithms for blockchain data processing on the data plane, as well as the use of static processing and search engines to detect public and private blockchains. In addition, the security tests should be implemented with various attack scenarios in order to validate all the aspects of security, and compared with the proposed architecture in this paper.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors gratefully acknowledge the constructive feedback of the handling editor and the reviewers. The last (corresponding) author is partially funded by the University of Texas at San Antonio and the Tecnológico de Monterrey System (UTSA & ITESM) Seed Funding Program (Project “Cyberattack Mitigation in Software Defined Networks”.)

References

https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html.
Antonopoulos, A.M., 2014. Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Inc..
Basnet, S.R., Shakya, S., 2017. BSS: blockchain security over software defined network. In: 2017 International Conference on Computing, Communication and Automation (ICCCA). IEEE, pp. 720–725.
Benácek, P., Pu, V., Kubátová, H., 2016. P4-to-vhdl: automatic generation of 100 gbps packet parsers. In: 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, pp. 148–155.
Bosshart, P., et al., 2014. P4: programming protocol-independent packet processors, vol. 44, no. 3, pp. 87–95.
Bosshart, P., et al., 2013. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN, vol. 43, no. 4, pp. 99–110.
Bozic, N., Pujolle, G., Secci, S., 2016. A tutorial on blockchain and applications to secure network control-planes. In: 2016 3rd Smart Cloud Networks & Systems (SCNS). IEEE, pp. 1–8.
Divekar, A., Parekh, M., Savla, V., Mishra, R., Shirole, M., 2018. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. In: 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS). IEEE, pp. 1–8.
Eddy, W., 2007. TCP SYN flooding attacks and common mitigations, 2070-1721.
Gibb, G., Varghese, G., Horowitz, M., McKeown, N., 2013. Design principles for packet parsers. In: Architectures for Networking and Communications Systems. IEEE, pp. 13–24.
Homayoun, S., Dehghantanha, A., Parizi, R.M., Choo, K.K.R., 2019. A blockchain-based framework for detecting malicious mobile applications in App stores. 32nd IEEE Canadian Conference of Electrical and Computer Engineering (IEEE CCECE'19.
Liu, Jingqiang, et al., 2018. A data storage method based on blockchain for decentralization DNS. 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE.
Muthanna, A., 2019. Secure and reliable IoT networks using fog computing with software-defined networking and blockchain. J. Sensor Actuator Netw. 8 (1), 15.
Nyaletey, E., Parizi, R.M., Zhang, Q., Choo, K.-K.R., 2019. BlockIPFS - Blockchain-enabled interplanetary file system for forensic and trusted data traceability. 2nd IEEE International Conference on Blockchain (IEEE Blockchain-2019).
P4 Language Consortium. Behavioral Model (bmv2), https://github.com/p4lang/behavioral-model, 2014. [Online].
P4 Language Consortium. P4c-behavioral, https://github.com/p4lang/p4c-behavioral, 2015. [Online].
Parizi, R.M., Dehghantanha, A., 2018. Smart contract programming languages on blockchains: An empirical evaluation of usability and security. In: International Conference on Blockchain. Springer, pp. 75–91.
Parizi, R.M., Homayoun, S., Yazdinejad, A., Dehghantanha, A., Choo, K.K.R., 2019. Integrating privacy enhancing techniques into blockchains using sidechains. 32nd IEEE Canadian Conference of Electrical and Computer Engineering (IEEE CCECE'19).
Parizi, Reza M., et al., 2018. Empirical vulnerability analysis of automated smart contracts security testing on blockchains. In: Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering. IBM Corp..
Rao, N.S., Sekharaiah, K.C., Rao, A.A., 2019. A survey of distributed denial-of-service (DDoS) defense techniques in ISP domains. In: Innovations in Computer Science and Engineering. Springer, pp. 221–230.
Sakakibara, Y., Morishima, S., Nakamura, K., Matsutani, and Systems, H.J.I.T.o.I., 2018. A hardware-based caching system on FPGA NIC for blockchain, vol. 101, no. 5, pp. 1350–1360.
Santiago da Silva, J., Boyer, F.-.R., Langlois, J., 2018. P4-Compatible high-level synthesis of low latency 100 Gb/s streaming packet parsers in FPGAs. In: Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, pp. 147–152.
Sharma, P.K., Chen, M.-Y., Park J.H.J.I.A., 2018. A software defined fog node based distributed blockchain cloud architecture for IoT, vol. 6, pp. 115–124.
Sharma, P.K., Park, J.H.J.F.G.C.S., 2018. Blockchain based hybrid network architecture for the smart city, vol. 86, pp. 650–655.
Sharma, P.K., Singh, S., Jeong, Y.-S., Park, J.H.J.I.C.M., 2017. Distblocknet: a distributed blockchains-based secure SDN architecture for IoT networks, vol. 55, no. 9, pp. 78–85.
Singh, A., Juneja, and Technology, D.J.I.J.o.E.S., 2010. Agent based preventive measure for UDP flood attack in DDoS attacks, vol. 2, no. 8, pp. 3405–3411,.
Sivaraman, A., Kim, C., Krishnamoorthy, R., Dixit, A., Budiu, M., 2015. Dc. p4: programming the forwarding plane of a data-center switch. In: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research. ACM, p. 2.

Taylor, P.J., Dargahi, T., Dehghantanha, A., Parizi, R.M., Choo, K.-K.R, 2019. A systematic literature review of blockchain cyber security. Digital Commun. Netw..
Vivado HLx 2018.2 - High-Level Synthesis (C based) – Xilinx. https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0012-vivado-high-level-synthesis-hub.html [Online].
Yazdinejad, A., Bohlooli, A., Jamshidi, K., 2018. P4 to SDNet: automatic generation of an efficient protocol-independent packet parser on reconfigurable hardware. In: 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE). IEEE, pp. 159–164.
Yazdinejad, A., Bohlooli, A., Jamshidi, K., 2018. Efficient design and hardware implementation of the OpenFlow v1. 3 switch on the virtex-6 FPGA ML605. J. Supercomput. 74 (3), 1299–1320.
Yazdinejad, A., Bohlooli, A., Jamshidi, K., 2019. Performance improvement and hardware implementation of OpenFlow switch using FPGA. Presented at the 2019 5th International Conference on Knowledge-Based Engineering and Innovation (KBEI).
Yazdinejad, A., Parizi, R.M., Dehghantanha, A., Choo, K.-K.R., 2019. Blockchain-enabled authentication handover with efficient privacy protection in SDN-based 5 G networks. IEEE Trans. Netw. Sci. Eng. doi:10.1109/TNSE.2019.2937481.
Zheng, Z., Xie, S., Dai, H.-N., Chen, X., Wang, H.J.I.J.o.W., Services, G., 2018. Blockchain challenges and opportunities: a survey, vol. 14, no. 4, pp. 352–375.
Zhou, X., Wu, Q., Qin, B., Huang, X., Liu, J., 2016. Distributed bitcoin account management. In: 2016 IEEE Trustcom/BigDataSE/ISPA. IEEE, pp. 105–112.

Abbas Yazdinejad received the BSc degrees in Computer engineering from the department of Computer Engineering Shahid Bahonar University of Kerman, Iran in 2014. He majored in the MSc degree in computer system architecture at the University of Isfahan, Iran in 2016. His research interests are software defined networking (SDN), FPGA design, IoT and network modeling and he is a scientific researcher.

Reza M. Parizi is a faculty at Kennesaw State University, GA, USA. He is a consummate technologist and software security researcher with an entrepreneurial spirit. He is the member of IEEE, IEEE Blockchain Community, IEEE Computer Society and ACM. Prior to joining KSU, he was an Associate Professor at New York Institute of Technology. He received a Ph.D. in Software Engineering in 2012 and M.Sc. and B.Sc. degrees in Software Engineering and Computer Science respectively in 2008 and 2005. His research interests are R&D in blockchain, smart contracts, IoT and emerging issues in the practice of secure software-run world applications.

Ali Dehghantanha is the director of Cyber Science Lab (http://cybersciencelab.org/) in University of Guelph, Ontario, Canada. His lab is focused on building AI-powered solutions to support cyber threat attribution, cyber threat hunting and digital forensics tasks. Ali has served for more than a decade in a variety of industrial and academic positions with leading players in Cyber-Security and Artificial Intelligence. Prior to joining UofG, he has served as a Sr. Lecturer in the University of Sheffield, UK and as an EU Marie-Curie International Incoming Fellow at the University of Salford, UK. He has PhD in Security in Computing and a number of professional certifications including CISSP and CISM.

Kim-Kwang Raymond Choo holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is the recipient of the 2019 IEEE TCSC Award for Excellence in Scalable Computing (Middle Career Researcher), 2018 UTSA College of Business Endowed Research Award for Tenured Faculty, IEEE Access Outstanding Associate Editor of 2018, British Computer Society's 2019 Wilkes Award Runner-up, 2019 EURASIP JWCN Best Paper Award, Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019, IEEE Blockchain 2019 Outstanding Paper Award, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, an IEEE Senior Member, and Co-Chair of IEEE MTCT's Digital Rights Management for Multimedia Interest Group.