Securing SDN-Controlled IoT Networks Through Edge Blockchain

Jiejun Hu, Martin Reed, *Member, IEEE*, Nikolaos Thomos, *Senior Member, IEEE*, Mays F. AI-Naday, *Member, IEEE*, and Kun Yang, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) connected by software-defined networking (SDN) promises to bring great benefits to cyber–physical systems. However, the increased attack surface offered by the growing number of connected vulnerable devices and separation of SDN control and data planes could overturn the huge benefits of such a system. This article addresses the vulnerability of the trust relationship between the control and data planes. To meet this aim, we propose an edge computing-based Blockchain as a Service (BaaS), enabled by an external BaaS provider. The proposed solution provides verification of inserted flows through an efficient, edge-distributed, blockchain solution. We study two scenarios for the blockchain reward purpose: 1) information symmetry, in which the SDN operator has direct knowledge of the real effort spent by the BaaS provider and 2) information asymmetry, in which the BaaS provider controls the exposure of information regarding spent effort. The latter yields the so-called "moral hazard," where the BaaS may claim higher than actual effort. We develop a novel mathematical model of the edge BaaS solution and propose an innovative algorithm of a fair reward scheme based on game theory that takes into account moral hazard. We evaluate the viability of our solution through analytical simulations. The results demonstrate the ability of the proposed algorithm to maximize the joint profits of the BaaS and SDN operator, i.e., maximizing the *social welfare*.**

***Index Terms*—Blockchain, flow verification and validation, reward scheme, software-defined networking (SDN).**

I. Introduction

The Internet of Things (IoT) finds many applications in both industrial and domestic spheres and promises to bring great benefits through increased connectivity to cyber–physical systems. However, as IoT systems generally lack computational power, the computation tasks are moved to edge computing systems such as multiaccess edge computing (MEC). A key component of modern flexible compute systems, such as MEC, is software-defined networking (SDN), which has generally been proposed for IoT architectures to deal with highly time-varying communication demands, prolong the lifetime of energy constraint devices, provide scalability, and improve flexibility.

Manuscript received April 1, 2020; revised July 20, 2020; accepted August 4, 2020. Date of publication August 18, 2020; date of current version February 4, 2021. This work was supported by the Project SerIoT, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant 780139. *(Corresponding author: Jiejun Hu.)*

The authors are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: jiejun.hu@essex.ac.uk).

Digital Object Identifier 10.1109/JIOT.2020.3017354

While SDN is an essential component of many edge systems, supporting IoT through flexible networking [1], [2], it also offers benefits in improving the security of networked systems. This has been demonstrated by the SerIoT project [3], which has proposed a fully integrated system that combines edge computing and SDN to address the security of IoT. However, although SDN can benefit security [4], it has also been noted that the SDN subsystem itself can be a target of attacks [5], [6]. The security issues in SDN are complex [5] to go into great detail here. However, in brief, the issues arrive from: a centralized SDN controller that implements highly complex software actions from the contents of network flows, leading to software actions too complex to test every outcome; implicit trust between the centralized controller and the edge switches; the switches are essentially dumb and simply implement the flow rules sent to them; and a controller who often acts in a reactive mode and sends flow rules for any new network flows that arise without necessarily holding historical immutable state of the rules. Thus, this article provides a solution to facilitate SDN security, i.e., ensuring flow rules are verified, at the edge, against network policies before being inserted into SDN switches; and, maintaining an independent, immutable, history of SDN flow insertion (which may be used by anomaly detection systems). This is an important addition to systems such as that proposed by SerIoT to ensure SDN security. In particular, our solution maximizes the joint satisfaction of the different blockchain stakeholders (i.e., the SDN operator and the blockchain provider). Evaluating the security performance is out of the scope of this article as security is directly related to the choice of the implemented policies and is left for future work.

To provide the security solution above, this article proposes a novel solution that carries out a separate flow verification/validation using blockchain technology. As it is vital that the flow verification occurs close to the edge switches, to reduce the opportunity of attacks [7], it is required that the blockchain technology is also implemented at the edge through a mechanism, such as MEC, to form an *edge blockchain*. Blockchain technology, a subconcept of distributed ledger technology, is essentially an append-only data structure maintained by a group of *not-fully trusted* nodes, which nevertheless provide a *trusted* data structure through a suitable consensus algorithm [8]. Furthermore, blockchain technology is decentralized, immutable, transparent, and reliable, which allows it to stand independently from the SDN network and the IoT network to establish a distributed trust mechanism.

The distributed nature of edge blockchain is ideally suited to support the security of the edge SDN systems [9] as it allows the flow verification to take place directly next to the edge switches while ensuring there is distributed consensus over the whole network.

To provide a straightforward deployment of blockchain, we utilize Blockchain as a Service (BaaS) [10]. We introduce blockchain agents (BCAs) that are collocated with SDN switches at the edge, and also in the core. The BCAs are responsible for flow verification/validation by running smart contracts. Flow verification requires a group of edge BCAs (e.g., in MEC) to act as a verifier initiator (VeIn) and verifiers. The verifiers inspect the new flow information, e.g., addresses, ports, and any other required fields against the policy. After flow verification, the verification result is sent back to the VeIn. Then, the flow validation process will be conducted. Flow validation requires BCAs to act as peers to validate the new flow in the block and update the flow ledger. Flow verification/validation in an SDN application needs computation capability to execute the encryption algorithm, consensus mechanism, and other smart contracts. As mentioned earlier, it is essential that the verification process takes place as close as possible to the edge SDN switches to minimize opportunities for malicious interference. Thus, the proposed architecture adopts compute in the edge (e.g., MEC) that is co-located with the SDN switches to host the BCAs. With lightweight IoT hubs, the use of this edge compute may be vital as edge blockchain is likely to be beyond the computational capabilities of many IoT devices and hubs. Thus, the edge-assisted BCA can provide advantages, such as promoting the scalability of the IoT networks with the number of IoT devices and reducing the communication overhead compared with cloud computing.

In the proposed architecture, BCAs run the SDN *flow verification application* as an external service. Meanwhile, there could be other applications from other parties running on the BCAs. This necessitates an incentive mechanism to stimulate the BCAs to perform the SDN flow verification. When a BCA initializes the flow verification procedure (i.e., it acts as a VeIn), it *hires* a group of BCAs as verifiers and offers them a reward. The verifiers know their computational ability, channel condition, workload, and the size of the task. Hence, the verifiers can decide how much reward they want. In this scenario, we realize that there is an *information barrier* between the verification initiator and verifiers. Specifically, a verification initiator can only decide the reward by the information provided by verifiers, however, the verifiers may not always be honest. In this article, we propose two reward schemes for: 1) an information-symmetric scenario (ISS), where the verifiers report their effort honestly and 2) an information-asymmetric scenario (IAS), where the verifiers hide their effort but reveal part of their information. In IAS, the verifier may behave greedily and demand a higher price for a simple task by revealing a high performance or else they will not implement the task. This phenomenon is known as the "moral hazard" of the verifiers [11]. The underlying reason for the moral hazard is the asymmetric relationship about system information: a verifier has access to more information to make a decision than a VeIn. In this work, game theory and contract theory are used to solve this problem. To the best of our knowledge, our work is the first to propose the workflow of flow rule verification/validation in blockchain-aided SDN (BC-SDN), to do so using edge computation and also to use contract theory to study the quantified performance of blockchain in the considered use case.

The main contributions of this article are summarized as follows.

1) We propose the architecture of edge blockchain-assisted SDN (BC-SDN) for flow conformance in an IoT system. Edge-blockchain helps maintain a distributed, immutable flow ledger for SDN.
2) We design the workflow of flow verification and subsequent validation for the SDN IoT system with the assistance of edge blockchain.
3) We devise fair reward schemes based on information-symmetric and information-asymmetric scenarios to stimulate the performance of BCAs on edge servers. Moreover, we compare the information symmetric and asymmetric cases in the proposed reward scheme using numerical simulations.

Although there is a clear need for security mechanisms in SDN, as we propose, in this article, we concentrate on the edge-blockchain mechanism itself, as this is the fundamental mechanism that is required before its benefits can be deployed. Thus, now we briefly elucidate the benefits of our architecture and its intended uses. The major benefit of the mechanism described is that it provides a security framework that is completely separate from the complex SDN controller process and thus provides prevention of malicious behavior that may occur in the SDN subsystem and verifiable security auditing of SDN. By using a smart contract for flow verification, it allows verifiable, independent, code to be developed so that the safety of the flow insertion can be ensured. A smart contract also provides a development environment that allows a deployer of the technology to select a wide range of flow verification policies, for example, a simple case of checking addresses of connected devices through to complex verification of end-to-end paths for specific flows. By using the immutable ledger facility of the blockchain inserted flows, it can be audited in the knowledge they cannot be changed or tested for anomalies using machine-learning approaches.

The remainder of this article is organized as follows. We first review related literature in Section II. Then, in Section III, we introduce the proposed architecture and workflow of BC-SDN. Next, we provide the considered system model in Section IV. We then formulate the problem and present our solution for both the ISS and IAS in Section V. Our solution is evaluated extensively in Section VI to get an understanding of the influence of the various system parameters. We also compare the performance of our solutions against other state-of-the-art methods, and the results show the advantages of using the proposed method. Finally, we draw conclusions in Section VII.

II. Related Works

Blockchain is a disruptive technology that has been used in many scenarios in real life, i.e., governance [12], healthcare [13], smart grid [14], and so on. In this article, we review the related works on blockchain-based SDN, edge computing-assisted blockchain, and the pricing scheme of blockchain.

A. SDN Using Blockchain Technology

Blockchain technology uses a group of not-fully trusted nodes, which nevertheless provide a trusted data structure through a suitable consensus algorithm. Thus, it is commonly used as a solution for security challenges in IoT and SDN. In [15], an SDN-based decentralized security architecture using blockchain technology for the IoT ecosystem is presented. This work aims to mitigate the recent challenges and detect attacks more efficiently. It adopts the blockchain technology to dynamically update the attack detection model and reward the fog nodes according to the "Proof of Work." Boukria *et al.* [16] proposed a blockchain-based controller against false flow rule injection, focusing mainly on the SDN controller authentication. Yazdinejad *et al.* [17] introduced a novel authentication handover based on blockchain in the SDN-based 5G network with the aim to remove unnecessary reauthentication in repeated handover among heterogeneous cells in 5G. Qiu *et al.* [18] studied the scenario of industrial IoT with multiple SDN controllers. A blockchain-based consensus protocol is presented to collect and synchronize network-wide views among different SDN controllers. This work employs the *Q*-learning method to jointly optimize the view change, access selection, and computational resources.

B. Blockchain and Edge Computing

Blockchain technology is computationally intensive during the consensus mechanism and ledger updating procedure. Therefore, typically there is a need to offload these, relatively, computationally heavy tasks to edge devices [19], [20]. Task offloading is not new and has been researched since more than a decade ago [21]. Wang *et al.* [22] presented MEC in the form of mobile cloud and combined it with cloud radio access networks (C-RANs) with a particular focus on joint resource allocation across computation and communication domains. Hu *et al.* [23] further introduced wireless power transfer into MEC to battle the energy supply issue of battery-powered IoT devices. A blockchain-based distributed cloud architecture has been proposed in [24]–[26], with fog nodes at the edge providing the functionality of the SDN controller. This work introduces a hierarchy with a central blockchain-based cloud moving toward a blockchain-based edge, with the latter taking responsibility for updating the flow rules. Although these studies show how cloud/fog computing would support blockchain, the presented solution neglects the issue of flow conformance testing that we address in this article. The work in [27] proposes a *Trust List* that represents the distribution of trust among IoT related stakeholders and provides autonomous enforcement of IoT traffic management at the edge networks by integrating blockchains and SDN. Ethereum is used to store the information of the controller on the edge computing servers, which lead to severe delays. In [28], an optimal pricing-based edge computing resource management is presented. This solution can support blockchain applications, where the mining process can be offloaded to an edge computing server provider (ESP). The authors adopted a two-stage Stackelberg game to maximize the utility of ESPs and miners.

Edge computing not only relieves the computational demand of blockchain technology but can also resolve security issues with the edge systems. First, using blockchain, it is possible to build a distributed control mechanism over all the edge systems [29]. Second, blockchain maintains data consistency in every edge node through its consensus mechanism; for example, Yang *et al.* [30] proposed a smart edge computing oriented data exchange prototype using Hyperledger to solve the issue of data automatically maintaining. Last but not least, blockchain enables dynamic resource allocation among edge nodes [31].

C. BaaS Reward Scheme

Due to the fact that blockchain technology is a resource-consuming application, studies concentrate on optimization problems that consider various aspects, such as: the latency, throughput of transactions, finality, security, and decentralization [32]. A survey covering the use of contract theory in wireless networks is presented by Zhang *et al.* [33]. This article reviews works that focus on the design of incentive mechanisms in wireless networks to ensure participants from the third party, such as access point, small cells, and users execute tasks with a proper reward. In [34], blockchain-based block verification is used in an Internet-of-Vehicles (IoV) setting. This article considered verification latency, verifiers' reputation, and network scale to construct the contract between the block manager and the verifiers. Differently, multidimensional contract theory in mobile crowdsourcing is studied in [35]. This article designed an incentive mechanism for mobile crowdsourcing by considering participants' effort, performance, and reward. The work in [36] proposed a directed acyclic graph (DAG)-based vehicle-to-vehicle communication network to solve the limited scalability and improve blockchain's efficiency. This article also proposed a game-theoretic solution to optimize bandwidth allocation and information transmission. However, it did not provide any incentive to conduct the DAG service.

However, these solutions fail to exploit blockchain technology in a manner that is compatible with existing SDN architectures. Specifically, they do not consider how to verify a new flow using blockchain in a fine-grained manner; nor, how to enable blockchain technology without changing the foundation of SDN. Additionally, they do not take into account how to secure the communication between the SDN controller(s) and switches; nor do they consider how to use blockchain as a service that needs a pricing scheme for the business model used in practice. This article aims at solving these problems by introducing our novel solution of BC-SDN. We achieve this by designing a flow verification/validation workflow that uses smart contracts. We design two reward schemes for ISS and IAS scenarios, analyze these schemes mathematically, and evaluate them through simulations.

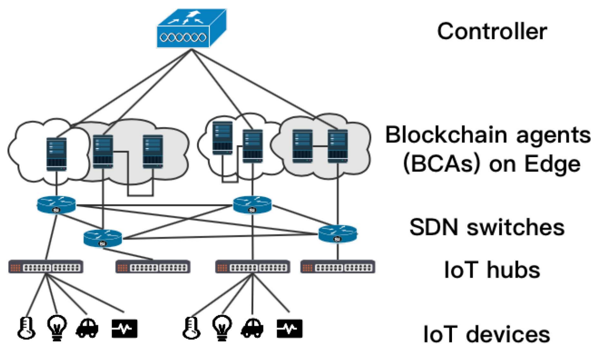

Fig. 1. System model of blockchain-based SDN.

III. Architecture and Workflow of BC-SDN

A. *Architecture of BC-SDN*

We consider an architecture in which blockchain capabilities can be offered as a service, termed BaaS to multiple customers. The architecture consists of a traditional SDN network, complemented by edge computing [37] to facilitate communication and provide applications in a generic IoT scenario. In this scenario, IoT devices are attached to edge computing systems and communicate with each other through an SDN-based network [3]. The edge computing provides virtual resources to a set of applications; one of which is a blockchain overlay, which provides blockchain services to customers both inside and outside the network. We assume the SDN operator to be one such customer, which purchases the service of the blockchain overlay to perform flow verification/validation between the control and forwarding counterparts. Fig. 1 shows an architectural view of the entities in our BC-SDN proposition, which includes the following.

1) *IoT devices* that interact with end-user environment and exchange data to influence said environment.
2) *IoT hubs* that connect the IoT devices to edge nodes through SDN switches.
3) *SDN switches* that detect new flows and execute a forwarding plan calculated by the *SDN controller(s)*.
4) *Blockchain agents* are software components (i.e., servers) provided by a blockchain service provider utilizing edge computing. BCAs are in charge of flow verification and validation via smart contracts. Furthermore, BCAs also execute basic blockchain functions, such as the consensus process, sending transactions, and maintaining the flow ledger. We consider that one or more BCAs are located in the edge servers for each SDN switch to provide computation ability. For simplicity and without loss of generality, hereafter, we assume there is only one BCA associated with each SDN switch.
5) *Edge nodes* are a selection of edge computing nodes connected through the SDN infrastructure. They provide computation and storage capabilities to the blockchain service, among others. Similar to IoT hubs, edge nodes are connected by the SDN infrastructure. Notably, in a generic scenario, the distribution of edge nodes could be different from that of the SDN switches; however, to simplify our work, we assume that one edge node is collocated with each SDN switch. To remove the effect of network-related latency, we assume that a BCA running on an edge node serves the switch collocated with that node.
6) *SDN controller* has the global view of the network and is able to calculate optimized paths in the network according to predefined objectives and policies. In practice, multiple SDN controllers are likely to be used for reliability and scalability. This does not change the solution or architecture in any way as the BCA is colocated with a switch and, in the same manner as a switch, a BCA would be configured to operate with a load-balanced controller group that sees multiple controllers as a single, virtual, controller.

We note that as with the existing SDN controller and switch interconnections [5], we assume that the controller-to-BCA and BCA-to-switch connections are protected using transport layer security (TLS). This provides basic support against malicious interventions in the control plane channels and is an important security step. However, it should be pointed out that TLS in the control plane does not provide the independent policy conformance testing that is provided by the solution presented in this article, which protects against much wider malicious activity against the SDN system. For example, malicious behavior through the manipulation of the controller or wider attempts at the injection of data plane traffic that goes against wider site policy necessitates a solution such as that presented here. Additionally, the immutable storage of the flow rules (and their changes) in the blockchain ledger allows security analysis of the behavior of SDN. This ledger can be accessed through a role-based approach so that, for example, users of the network can confirm that their relevant rules have been accepted, but without having wider access to sensitive information, while a network operator may have a full view of the rules for wider security analysis.

In this article, we adopt a permission-based consortium blockchain, such as Hyperledger [38], which means that only authorized BCAs can conduct blockchain functions. Furthermore, we adopt a BaaS infrastructure with a BCA collocated with an SDN switch. There are three main advantages of using the proposed mechanism for flow verification/validation. First, as BCAs are collocated with the SDN switches, providing a blockchain service to the SDN, the BCAs can assist with the secure communication between the controller and the corresponding switch. Second, BCAs are components provided by an external entity, which conducts flow verification/validation outside the SDN network to guarantee connection privacy. Finally, BaaS enables a straightforward deployment of blockchain in SDN without the SDN operator needing to create their own blockchain system.

B. *Smart Contracts and Workflow of BC-SDN*

In BC-SDN, the workflow includes flow verification and flow validation. We show these two parts in Figs. 2 and 3, respectively. Flow verification mainly focuses on new flow verification initiation, flow conformance policy checking, and result feedback to the VeIn. The flow validation is in charge of

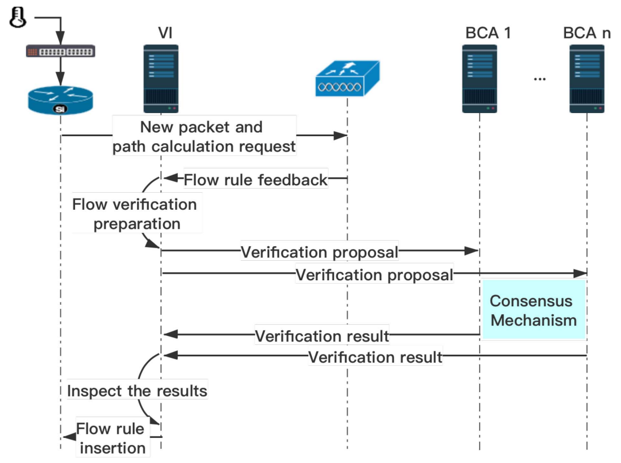

Fig. 2. Workflow of flow verification.

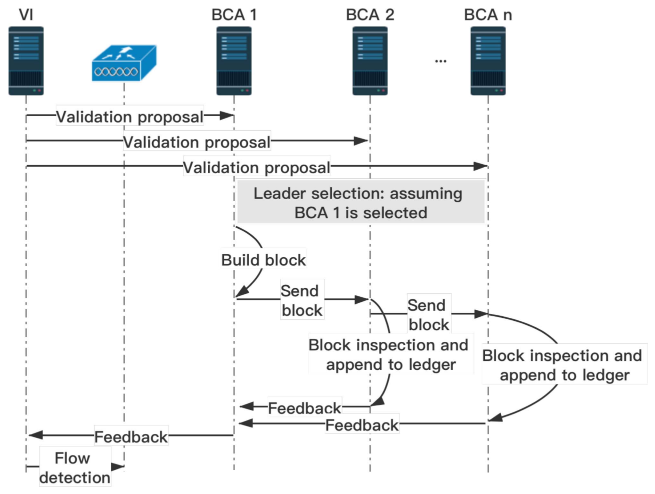

Fig. 3. Workflow of flow validation.

flow ledger updating. In Hyperledger, verification is performed by a leading verifier and the following *verifiers*. Validation is performed by a leading peer, namely, the *orderer* and the following *peers*. In BC-SDN, VeIn acts as the leading verifier and orderer. There is a group of BCAs that act as verifiers and peers in verification and validation, respectively. In this work, we adopt *Hyperledger* as edge blockchain and the *Byzantine consensus* mechanism [39]. First, we assume the following.

1) This blockchain application is using a unique Hyperledger *channel*. To allow multiple applications to use Hyperledger at the same time, each application is allocated a unique channel with an individual channel ID. In BC-SDN, it only has one SDN flow verification/validation application.
2) The IoT devices/IoT hubs that require new flow rules have been registered and enrolled with the organization's Certificate Authority and received back necessary cryptographic material, which is used to authenticate the device.
3) The BCAs have been fed with previous topology and connectivity information from the controller. Moreover, all the BCAs apply the same flow conformance policy to check the new flow rules. The conformance policy is defined as the simple policy, i.e., it verifies the source and destination IP addresses and the port numbers.
4) The controller is responsible for path calculations, which will result in a set of flow rules according to the path.

1) Smart Contracts in Edge Blockchain: We define a group of smart contracts to conduct flow verification/validation on edge blockchain. First, we have *verification initiation contract* and *validation initiation contract* to start the preparation of new flow verification/validation by obtaining the key information of the flow, essential encrypted materiel, and so on. Then, edge blockchain deploys *verification contract* to inspect the signature of the verifiers, conduct flow conformance policy, and construct response messages to peer verifiers. Last but not least, we have the Byzantine consensus mechanism deployed as *consensus contract* when the response is checked among all the peers in the verification/validation phase.

2) Flow Verification: In the traditional SDN architecture, when IoT devices initiate a new communication request, the corresponding access switch sends the new packet with source/destination IP, source/destination port ID, and protocol to the controller. It requires the controller with the global view of the network to calculate the path. In BC-SDN, the flow verification process initiates after the SDN controller sends the flow rule back with *verification initiation contract*. Instead of sending the flow rule back to the switch, the SDN controller sends it to the corresponding BCA of the switch. Then, the BCA VeIn will start flow verification by running a verification initiation contract. This contract requires preparing the proposal and sending it to the other BCAs, namely, verifiers. Here, the VeIn adopts a preset endorsement policy to employ the verifier. The endorsement policy [38] requires that the VeIn must obtain the inspection feedback of the new flow rule from a certain number of verifiers, otherwise, the endorsement is considered as failed. The endorsement policy is crucial to justify if the Byzantine consensus mechanism is reached. The actions of establishing and verifying/validating a new flow rule are embedded in consensus contract within the blockchain. Below, we explain these actions in more depth.

1) An IoT device causes an SDN switch to initiate a new flow establishment request when it sends a packet with source IP/port information, destination IP/port information, and flow conformance policy. The message is formed as $packet = <souIP, desIP, souPort, desPort, Policy>$ without an existing flow rule, and this is sent to the controller over a secure communication channel. The controller calculates the path according to the *packet* and generates new flow rules that it then forward them to the corresponding BCA (VeIn). The ID of flow is $fid = hash(packet)$. This reply from the controller targets BCA1.
   a) The VeIn constructs a new flow proposal $<PROPOSAL, tx, Consig>$ and makes sure the new flow proposal is properly formed; *tx* includes: the ID of controller, ID of the flow, packet, ID of smart contract, endorsement policy, and timestamp.

   b) The VeIn's credentials are used to produce a signature *ConSig* = *hash*(*tx*) for this proposal.
2) The VeIn starts the verification of the new flow by *verification contract*. The contract requires the VeIn to employ verifier peers. Then, verifiers inspect the signature *ConSig* and execute the new flow verification.
   a) The verifiers check whether: the proposal is well formed; it has not been submitted already in the past (i.e., it is not some form of replay); the signature is valid; and the VeIn is properly authorized to perform the proposed operation.
   b) The verifiers use the new flow proposal as input to invoke the verification contract.
   c) The verification contract checks the new flow against the flow conformance policy and asserts as TRUE or FALSE accordingly. The response values along with the verifiers' signature first are passed among the verifiers till the consensus is reached by running the Byzantine consensus contract, and then the responses of verifiers are passed back to the VeIn as a "proposal response." Byzantine consensus requires the VeIn to receive enough verifiers' "TRUE" response of the flow, i.e., the number of verifiers satisfying the endorsement policy. All the response values will be stored in blockchain's status database *readset* and *writeset*.
   d) The response values along with the verifiers' signature is passed back to the VeIn as a "proposal response" *ProRes* = <*TransactionEndorsed, fid, TranProposal, epSig*>, where *TranProposal* = *(epID, fid, chaincodeID, tx, readset, writeset)*.
   e) If this new flow is invalid, then send message <*TransactionEndorsed, fid, REJECT, epSig*> to the VeIn.

   Till here, the consensus contract completes and the VeIn receives the consensus result from the verifiers. Note that no changes are made to the flow ledger up to this point.
3) The proposal responses are then checked as follows.
   a) The VeIn inspects the verifiers' signature and confirms that the number of identical *ProRes* responses reaches the number expected by the endorsement policy.
   b) If the VeIn only enquires of flow rules from the flow ledger, then there is no need to update the blockchain database, namely, the *ordering service*.
   c) The VeIn checks if the endorsement policy has been reached before the new flow is stored in the ledger.

3) Flow Validation: In the flow validation phase, each BCA appends the block with the new *fid*. After flow validation, all the flow rules, no matter if they are legitimate or not, will be stored in the flow ledger immutably, which provides evidence to anomaly detection algorithms and applications.

1) To start validation initiation, the VeIn executes the *validation initiation contract* to broadcast the proposal <*PROPOSAL*> with <*ProRes*> in one message *broadcast = (PROPOSAL, ProRes)* to all of the BCAs.
2) When all the BCAs receive the proposal, the leader selection procedure is triggered. We assume that VeIn is the leader of the flow ledger updating. VeIn sends a message *deliver(seqno, prevhash, endorsement)*, where *seqno* is the sequence number and *prevhash* is the hash of the most recently delivered endorsement. VeIn orders all the flow rules chronologically and creates blocks of flow rules.
3) The blocks of flow rules are delivered to all BCAs. The *consensus contract* is executed by all the BCAs as it is in the verification consensus mechanism.
   a) The BCAs verify the ID of consensus contract, endorsement policy, and consistency of the status database to avoid violations. They wait until all the peer BCAs' feedback to reach consensus.
   b) If the checks pass, the flow rule is deemed valid or committed. In this case, the BCAs set the bitmask of the *flow Ledger*.
   c) If the checks fail, the new flow establishment is considered invalid and the BCA unsets the bitmask of the *flow Ledger*. This invalid flow rule is still stored until it is deleted by a periodic blockchain function.

Up to this point, flow verification/validation is completed.

IV. System Model

We use BaaS in BC-SDN, as described in Section III, the BaaS is running over virtual resources rented from the edge computing provider, independently from the SDN network. The BaaS provides services to multiple organizations, one of which is the SDN network. This means that the VeIn may lack knowledge of the edge servers' performance (and other abilities). Thus, there is an information barrier between VeIn and BaaS. In this article, we assume two scenarios, namely, an ISS and an IAS. IAS leads to what is known as a moral hazard [40]. The moral hazard is commonly solved by contract theory in the field of economics. Motivated by the above, in this article, we design two reward schemes for BaaS, which can relieve the difficulty of the information barrier between VeIn and the third-party BCAs.

In BC-SDN, flow verification/validation is provided by BCAs and a contract is designed based on the outcome of the verifiers. We consider N verifiers. Each verifier offers n different execution latencies according to the reward, workload, CPU capability, and so on. The set of possible latency values for flow verification and validation is denoted as $\mathcal{L} = \{l_1, \ldots, l_n\}, l_i \in \mathcal{L}$ and the set of blocksizes as $\mathcal{S} = \{s_1, \ldots, s_n\}, s_i \in \mathcal{S}$ with elements in $\mathcal{S}$ and $\mathcal{L}$ having one to one correspondence. When a flow verification process is initiated, the VeIn presents a contract for the verifiers offering a reward. Then, the verifiers have the option to either accept the reward or reject it. According to the reward, the verifiers exert the flow verification/validation latency. During this procedure, the verifier has to report its latency of execution to the VeIn. We represent the honesty factor of a verifier by $h_i \in \mathcal{H}$, where $\mathcal{H} = \{h_1, \ldots, h_n\}$ is the set of honesty factors. The honesty factor indicates the level of honesty of a verifier when it

TABLE I
NOTATION AND DESCRIPTIONS

Description	Parameter
Number of verifiers	N
Set of possible rewards to verifiers	$\mathcal{R} = \{r_1, ...r_n\}, r_i \in \mathcal{R}$
Honesty factor of verifiers	$\mathcal{H} = \{h_1, ...h_n\}, h_i \in \mathcal{H}$
Set of blocksizes	$\mathcal{S} = \{s_1, ...s_n\}, s_i \in \mathcal{S}$
Set of latencies	$\mathcal{L} = \{l_1, ...l_n\}, l_i \in \mathcal{L}$
Probabilities of latency	$\mathcal{P} = \{p_1, ...p_n\}, p_i \in \mathcal{P}$
Cost factor	α
Income factor	β
Honesty factor	τ_1
Latency factor	τ_2
Reservation utility	σ
Social welfare	ω
Maximum latency	$l_{\max}$
Maximum blocksize	s_{max}

reports its performance to the VeIn. In this work, we propose two reward mechanisms for the two scenarios studied here, ISS and IAS. Let t be the reward mechanism indicator, where $t \in \{t_1, t_2\}$. When $t = t_1$, we consider ISS, and *vice versa*. In ISS, VeIn can observe the effort, namely, the blocksize s_i, to make a decision regarding the reward. Thus, we consider the reward mechanism as contract $\mathcal{C}^{t_1}(r_i, s_i)$, where r_i is the reward of the ith verifier. In IAS, BCAs hide their true effort blocksize s_i, so VeIn can only make a decision of the reward by the latency l_i of BCAs. Thus, the contract is defined as $\mathcal{C}^{t_2}(r_i, s_i)$, with $t = t_2$ indicating IAS contract.

In this section, we first introduce the system model by analyzing the cost and income of the VeIn and verifiers. Then, we define the employed utility functions of both the VeIn and verifiers. Finally, we propose our solution based on contract theory. Note we list the notations and descriptions in Table I.

A. Execution Cost of the Verifier

Consider verifiers who participate in BC-SDN and make a choice of flow verification/validation latency. However, the latency may not just be the consequence of the block it actually processes but may also be influenced by other tasks that the verifiers choose to carry out. However, generally, we can say that the execution cost of the verifier is related to the blocksize. Similar to [19], the execution cost of a verifier is defined in a quadratic form, which is thus convex and provides a straightforward evaluation of the derivative. When verifiers create a blocksize s_i, the execution cost of the verifier is

$$\phi(s_i) = \frac{1}{2}\alpha s_i^2 \tag{1}$$

where $\alpha > 0$ is the cost factor for the verifiers. The cost function shows that there exists an optimal blocksize, which could lead to the optimal cost.

B. Reward Plan for Verifier

We consider the set of rewards $\mathcal{R} = \{r_1, \ldots, r_n\}, r_i \in \mathcal{R}$. We assume that verifiers with the same honesty factor have the same reward. Thus, we define the reward to verifiers with honesty factor h_i as

$$w_i = \begin{cases} r_i\chi_i(h_i, s_i), & \text{when } t = t_1 \\ r_i\chi_i(h_i, l_i), & \text{when } t = t_2 \end{cases} \tag{2}$$

where $\chi_i(\cdot)$ is a function of verifiers with honesty factor h_i's performance. The reward plan for verifiers in both scenarios should respect the following: the bigger the honesty factor is, the bigger the reward is. Moreover, for ISS, the bigger the blocksize is, the bigger the reward is. On the contrary, for IAS, the bigger the latency is, the smaller the reward is. Therefore, the following conditions should hold when designing the reward function: $(\partial\chi_i/\partial h_i) > 0$, $(\partial\chi_i/\partial l_i) < 0$, and $(\partial\chi_i/\partial s_i) > 0$.

C. Income of VeIn

The income of VeIn depends on the contribution of the verifiers, i.e., the blocksize s_i and the honesty factor h_i for ISS and the latency l_i of flow verification/validation and the honesty factor h_i of the verifier for IAS. The income function of VeIn $\mathcal{I}_i$ in terms of verifiers with honesty factor h_i, blocksize s_i, and latency l_i is given by

$$\mathcal{I}_i^t = \begin{cases} \tau_1 f(h_i)s_i, & \text{when } t = t_1 \\ \tau_1 f(h_i) - \tau_2 g(l_i, l_{\max}), & \text{when } t = t_2 \end{cases}$$

where $\tau_1 > 0$ and $\tau_2 > 0$ are the weighting of honesty and the weighting of latency, respectively. The parameter $l_{\max}$ is the maximum latency.

We should have $(\partial f(h_i)/\partial h_i) > 0$, $(\partial g(l_i, l_{\max})/\partial l_i) > 0$, $(\partial\mathcal{I}_i/\partial h_i) > 0$, and $(\partial\mathcal{I}_i/\partial l_i) < 0$. This means that the bigger is the honesty factor, the higher is the income. It also means that larger latency leads to lower income. To simplify the problem, we define the income function as

$$\mathcal{I}_i^t = \begin{cases} \tau_1 h_i s_i, & \text{when } t = t_1 \\ \tau_1 h_i - \tau_2 \frac{l_i}{l_{\max}}, & \text{when } t = t_2. \end{cases} \tag{3}$$

V. Problem Formulation

In this section, we analyze the ISS and IAS scenarios in detail. In ISS, the VeIn is informed by the verifiers about their efforts of executing a verification/validation task with respect to their honesty factor. Therefore, in ISS, the VeIn can optimize the reward according to the efforts of verifiers.

We also investigate IAS, which suffers a moral hazard. Since in IAS, the VeIn does not have full knowledge of the verifiers, it can only be informed from their performance, i.e., the execution latency of the verifiers. Thus, the IAS scenario leads to a more complex problem to solve than the ISS scenario. The rest of this section studies ISS and IAS scenarios in Sections V-A and V-B, respectively.

A. Information Symmetric Scenario

In ISS, we assume that verifier i will report the actual effort, i.e., the blocksize s_i considering a honesty factor h_i. The honesty of the verifier affects the income of the VeIn. We define the utility function of the verifier as the income of the verifier

i minus the execution cost (1). Hence, it is

$$U_i^{t_1} = r_i s_i - \frac{1}{2}\alpha s_i^2 \tag{4}$$

where the income of the verifier i is proportional to the blocksize s_i. We also formulate the utility function of VeIn as

$$U_v^{t_1} = \beta \tau_1 h_i s_i - r_i s_i \tag{5}$$

where $\beta > 0$ is the VeIn's income factor. We can then propose a two stage optimization method. In the first stage, the verifier considers the reward r_i from VeIn as known and computes the optimal blocksize s_i. In the second stage, the verification initiator uses the optimal blocksize $s_i{}^*$ and solves a second optimization problem to compute the optimal reward $r_i{}^*$. This formulation falls to the category of Stackelberg games, which also means that Stackelberg game is information symmetric between the VeIn and the verifiers.

Definition 1 (Stackelberg Equilibrium): The system reaches the Stackelberg equilibrium, if and only if the verifiers and the verification initiator reach the relationship described by the following equations:

$$U_i^{t_1}\left(r_i, s_i{}^*\right) > U_i^{t_1}(r_i, s_i) \tag{6}$$

$$U_v^{t_1}\left(r_i{}^*, s_i{}^*\right) > U_i^{t_1}\left(r_i, s_i{}^*\right). \tag{7}$$

Following the Stackelberg equilibrium, we use the backward induction algorithm [41] to determine the equilibria of the subgames (6) first. The maximization problem for verifier i is defined as

$$\max_{s_i} \; U_i^{t_1} = r_i s_i - \frac{1}{2}\alpha s_i^2 \tag{8a}$$

$$\text{s.t} \;\; s_i < s_{\max} \tag{8b}$$

where $s_{\max}$ stands for the maximum blocksize. To calculate the optimal blocksize, we set the first derivative of the maximization problem in (8a) to zero. Hence, we obtain

$$\frac{\partial U_i^{t_1}}{\partial s_i} = r_i - \alpha s_i = 0.$$

By solving this equation, we can determine the optimal blocksize $s_i{}^* = (r_i/\alpha)$. We then substitute $s_i{}^*$ in the utility function of the VeIn and solve the second stage of the optimization problem so that it respects (7)

$$\max_{r_i, s_i^*} \; U_v^{t_1} = \beta \tau_1 h_i \frac{r_i}{\alpha} - \frac{r_i{}^2}{\alpha}. \tag{9a}$$

Following the same method with the first stage, we can obtain the optimal reward by finding the derivative of the objective function with respect to the reward and setting it equal to zero

$$\frac{\partial U_v^{t_1}}{\partial r_i} = \frac{1}{\alpha}\beta \tau_1 h_i - 2\frac{r_i}{\alpha} = 0.$$

Therefore, we can calculate the optimal reward and the blocksize by the following equations:

$$r_i^* = \frac{1}{2}\beta \tau_1 h_i \tag{10}$$

$$s_i^* = \frac{1}{2\alpha}\beta \tau_1 h_i. \tag{11}$$

We can observe from (10) and (11) that the honesty factor h_i is proportional to both the reward and the blocksize.

B. Information Asymmetric Scenario

In this scenario, the VeIn considers the honesty and latency of the verifiers provide. Therefore, according to the income of VeIn in (3), we define the utility function of VeIn as the gross income minus the reward plan w to the verifier. Due to the uncertainty of the verifiers' behavior, we define that verifiers with honesty factor h_i choose latency l_i with probability p_i. We denote the discrete set of probabilities $\mathcal{P} = \{p_1, \ldots, p_n\}$, with $\sum_{i=1}^{n} p_i = 1$, where $p_i \in \mathcal{P}$ is the probability the verifier to choose a blocksize $s_i \in \mathcal{S}$ that results to a latency l_i. Similar to the ISS case, the utility of the verifier is defined as the reward plan minus the execution cost. Meanwhile, we assume that verifiers who choose the same latency l_i have the same honesty factor. Thus, the verifier's utility is defined as

$$U_i^{t_2} = r_i \chi_i(h_i, l_i) - \frac{1}{2}\alpha s_i^2. \tag{12}$$

Let us assume that the total number of verifiers is N. The VeIn's utility function is given by

$$U_v^{t_2} = \sum_{i=1}^{n} N p_i [\beta \mathcal{I}_i(h_i, l_i) - r_i \chi_i(h_i, l_i)] \tag{13}$$

recall β is the income factor of the VeIn. For IAS, the optimization problem is defined as

$$\max_{r_i, l_i} \; \sum_{i=1}^{n} N p_i [\beta \mathcal{I}_i(h_i, l_i) - r_i \chi_i(h_i, l_i)] \tag{14a}$$

$$\text{s.t} \;\; r_i \chi_i(h_i, l_i) - \frac{1}{2}\alpha s_i^2 \geq r_j \chi_i(h_i, l_i) - \frac{1}{2}\alpha s_j^2 \tag{14b}$$

$$r_i \chi_i(h_i, l_i) - \frac{1}{2}\alpha s_i^2 \geq \sigma \tag{14c}$$

$$i \neq j \tag{14d}$$

where σ is reservation utility, which is the minimum profit that must be guaranteed by the contract to make it acceptable to the verifiers. Note that without loss of generality, we set $\sigma = 0$ to simplify the problem. Equation (14b) is the individual compatibility (IC) for verifiers, which means that by exerting the optimal blocksize, it can obtain the optimal reward. Equation (14c) is the individual rational (IR) that requires the utility of the verifier is positive. For the sake of simplicity of representation, we omit the arguments of $\chi(\cdot)$ function when they do not affect the calculations.

Lemma 1 (Monotonicity): If we have for the income function $\chi_i \geq \chi_j$, $\forall i, j \in \{1, \ldots, n\}$, the reward satisfies $r_i \geq r_j$.

Proof: This lemma is a direct application of (14b). Thus, for $i, j \in \{1, \ldots, n\}$, we get

$$r_i \chi_i - \frac{1}{2}\alpha s_i^2 \geq r_j \chi_i - \frac{1}{2}\alpha s_j^2 \tag{15}$$

$$r_j \chi_j - \frac{1}{2}\alpha s_j^2 \geq r_i \chi_j - \frac{1}{2}\alpha s_i^2. \tag{16}$$

By adding inequalities (15) and (16), we find that

$$\left(r_i - r_j\right)\left(\chi_i - \chi_j\right) \geq 0. \tag{17}$$

■

Lemma 2: If the utility function of the verifier with χ_1 satisfies the IR constraint, then this holds for $\forall i \in \{1, \ldots, n\}$.

Proof: Let us assume that $\chi_1 < \cdots < \chi_i < \cdots < \chi_n$, then according to Lemma 1, we derive

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 \geq r_1\chi_i - \frac{1}{2}\alpha s_1^2 \tag{18}$$

$$r_1\chi_i - \frac{1}{2}\alpha s_1^2 \geq r_1\chi_1 - \frac{1}{2}\alpha s_1^2. \tag{19}$$

By adding (18) and (19), we get

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 \geq r_1\chi_1 - \frac{1}{2}\alpha s_1^2. \tag{20}$$

From (20), we can derive that if $r_1\chi_1 - (1/2)\alpha s_1^2 > 0$, then IR in (14c) holds.

If we consider $i \in \{1, \ldots, n\}$ and $i - 1 < i < i + 1$, then there are $n(n-1)$ constraints [these replace (14b)] of the form

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 \geq r_{i-1}\chi_i - \frac{1}{2}\alpha s_{i-1}^2 \tag{21}$$

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 \geq r_{i+1}\chi_i - \frac{1}{2}\alpha s_{i+1}^2. \tag{22}$$

■

Definition 2 (Local Upward and Downward Constraints): If verifiers with χ_i prefer contract (r_i, l_i) to (r_{i+1}, l_{i+1}), then the local upward constraint for χ_i is satisfied, as shown in (22). Similarly, if verifiers with χ_i prefer contract (r_i, l_i) to (r_{i-1}, l_{i-1}), then the local downward constraint for χ_i is satisfied, as shown in (21).

Lemma 3: Local downward compatibility is reached as follows:

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 \geq r_{i-1}\chi_{i-1} - \frac{1}{2}\alpha s_{i-1}^2. \tag{23}$$

Proof: We assume that $\chi_{i-1} < \chi_i < \chi_{i+1}$, then according to the IC constraint, we have

$$r_{i+1}\chi_{i+1} - \frac{1}{2}\alpha s_{i+1}^2 \geq r_i\chi_{i+1} - \frac{1}{2}\alpha s_i^2 \tag{24}$$

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 \geq r_{i-1}\chi_i - \frac{1}{2}\alpha s_{i-1}^2. \tag{25}$$

According to Lemma 1, we have

$$(\chi_{i+1} - \chi_i)(r_i - r_{i-1}) \geq 0$$
$$\Longrightarrow \chi_{i+1}(r_i - r_{i-1}) \geq \chi_i(r_i - r_{i-1}). \tag{26}$$

By combining (25) and (26), we get

$$r_i\chi_{i+1} - \frac{1}{2}\alpha s_i^2 \geq r_{i-1}\chi_{i+1} - \frac{1}{2}\alpha s_{i-1}^2. \tag{27}$$

Then, by combining (24) and (27), we find that

$$r_{i+1}\chi_{i+1} - \frac{1}{2}\alpha s_{i+1}^2 \geq r_{i-1}\chi_{i+1} - \frac{1}{2}\alpha s_{i-1}^2. \tag{28}$$

Finally, by expanding (28), we get

$$r_{i+1}\chi_{i+1} - \frac{1}{2}\alpha s_{i+1}^2 \geq r_{i-1}\chi_{i+1} - \frac{1}{2}\alpha s_{i-1}^2$$
$$\cdots \geq r_1\chi_1 - \frac{1}{2}\alpha s_1^2 \tag{29}$$

which completes the proof. ■

Lemma 4: Upward local incentive is satisfied in the proposed problem as follows:

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 \geq r_{i+1}\chi_i - \frac{1}{2}\alpha s_{i+1}^2. \tag{30}$$

Proof: We can prove Lemma 4 in the same way as Lemma 3. ■

In the optimization problem (14), constraint (14b) includes $n(n-1)/2$ upward local incentive and $n(n-1)/2$ downward local incentive conditions. Additionally, constraint (14c) consists n IR constraints. By applying the lemmas we proved previously, we can reduce the constraints as follows.

We know that in order to maximize the utility of VeIn, we need to provide the minimum reward to verifiers, which leads to the minimum utility of the verifier in (14c) as $r_1\chi_1 - (1/2)\alpha s_1^2 = 0$. Therefore, (14c) can be reduced. Moreover, for (14b), if rewards r_i and r_{i-1} lower the utility value by the same amount and the downward local incentive still holds, then eventually it can reach $r_i\chi_i - (1/2)\alpha s_i^2 = r_{i-1}\chi_i - (1/2)\alpha s_{i-1}^2$ because we have

$$r_i\chi_i - \frac{1}{2}\alpha s_i^2 = r_{i-1}\chi_i - \frac{1}{2}\alpha s_{i-1}^2. \tag{31}$$

Since monotonicity holds, when $\chi_i \geq \chi_{i-1}$, it is also $r_i \geq r_{i-1}$, so we have

$$\chi_i(r_i - r_{i-1}) \geq \chi_{i-1}(r_i - r_{i-1}). \tag{32}$$

By adding (31) and (32), we derive

$$\frac{1}{2}\alpha s_i^2 - \frac{1}{2}\alpha s_{i-1}^2 \geq \chi_{i-1}(r_i - r_{i-1})$$
$$r_i\chi_{i-1} - \frac{1}{2}\alpha s_i^2 \leq r_{i-1}\chi_{i-1} - \frac{1}{2}\alpha s_{i-1}^2. \tag{33}$$

By observing (33), we note that it is the same as Lemma 4. Thus, constraint in (14b) can be reduced as (31). Therefore, we can propose the new optimization problem as follows:

$$\max_{r_i, l_i} \sum_{i=1}^{n} Np_i[\beta\mathcal{I}_i(h_i, l_i) - r_i\chi_i(h_i, l_i)] \tag{34a}$$

$$\text{s.t } r_i\chi_i(h_i, l_i) - \frac{1}{2}\alpha s_i^2 = r_{i-1}\chi_i(h_i, l_i) - \frac{1}{2}\alpha s_{i-1}^2 \tag{34b}$$

$$r_1\chi_1(h_1, l_1) - \frac{1}{2}\alpha s_1^2 = 0. \tag{34c}$$

By solving (34c), we can obtain

$$r_1 = \frac{\alpha {s_1}^2}{\chi_1}.$$

Then, by solving recursively (34b), we can find

$$r_i = \frac{\alpha {s_1}^2}{\chi_1} + \sum_{q=2}^{i}\left(\frac{\alpha {s_q}^2}{\chi_q} - \frac{\alpha {s_{q-1}}^2}{\chi_q}\right).$$

As blocksize and latency have a linear relation, we can substitute blocksize with latency without losing optimality

$$r_i = \frac{\alpha {l_1}^2}{\chi_1} + \alpha\sum_{q=1}^{i-1}\left(\frac{{l_{q+1}}^2 - {l_q}^2}{\chi_{q+1}}\right). \tag{35}$$

Let us define

$$g_q = \sum_{q=1}^{i-2} \frac{{l_{q+1}}^2 - {l_q}^2}{\chi_{q+1}}. \quad (36)$$

Then, we can substitute r_i in (34a) and the objective function becomes

$$U_v^{t_2} = \sum_{i=1}^{n} Np_i\beta\mathcal{I}_i - N\alpha\sum_{i=1}^{n} p_i \times \left[\left(\frac{{l_1}^2}{\chi_1} + g_q\right)\left(\tau_1 h_i - \tau_2\frac{l_i}{l_{\max}}\right) + {l_i}^2 - {l_{i-1}}^2\right]. \quad (37)$$

We can calculate the derivatives of the utility function in (37) with respect to l_i as follows:

$$\frac{\partial U_v^{t_2}}{\partial l_i} = -Np_i\beta\frac{\tau_2}{l_{\max}} - N\alpha p_i\left[-\frac{\tau_2}{l_{\max}}\left(\frac{{l_1}^2}{\chi_1} + g_q\right) + 2l_i\right]$$

$$\frac{\partial {U_v^{t_2}}^2}{\partial {l_i}^2} = -N\alpha p_i 2 < 0. \quad (38)$$

From (38), we can see that $(\partial {U_v^{t_2}}^2/\partial {l_i}^2) < 0$, thus the objective function/ utility function of VeIn is concave. Furthermore, as the constraints are all affine, the optimal value of latency l_i and reward r_i can be computed using an optimization solver such as CVX [42]. We would like to note that by using the proposed scheme, the verifiers can be stimulated by the reward and conduct the flow verification/validation.

VI. Experimental Evaluation

In this section, we numerically evaluate the performance of the proposed reward schemes in BC-SDN. We consider that BCAs work as both verifiers and validators in the proposed architecture. In the following, we first introduce the benchmark solutions. Then, we evaluate the performance of the proposed reward schemes with respect to the cost factor of verifiers α, the income factor of VeIn β, the number of verifiers N, and the probability p of blocklength selection which is proportional to latency. In the simulation set up, we consider a group of verifiers and different numbers of χ values, in which χ_i stands for the combination of latency l_i and honesty factor h_i. The key parameters used in the evaluation are listed in Table II. In the simulations, we consider these values unless otherwise stated.

To evaluate the performance of the proposed reward scheme, we introduce the concept of "social welfare" ω of the BaaS service and defined as the profit of the verifiers and VeIn. Thus, the social welfare is

$$\omega = \sum_{i=1}^{n} U_i + U_v. \quad (39)$$

Social welfare is a rational way to analyze if the contract can maximize the total utilities, where the utility indicates the preference of VeIn and the verifiers choosing and consuming the contracts and resources, respectively, which leaves utility without unit. Social welfare also indicates the BCAs' resource utilization of the edge. For the sake of comparison, we compare the proposed reward scheme with a solution

TABLE II
SIMULATION SETTINGS

Description	Setting
Number of verifiers	$N = 10$
Honesty factor of verifiers	$\mathcal{H} = \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$
Probability of latency	$p_i = 0.1$
Cost factor	$\alpha = 0.5$
Income factor	$\beta = 10$
Honesty weighting	$\tau_1 = 1$
Latency weighting	$\tau_2 = 2$
Reservation utility	$\sigma = 0$
Maximum latency	$l_{\max} = 20$

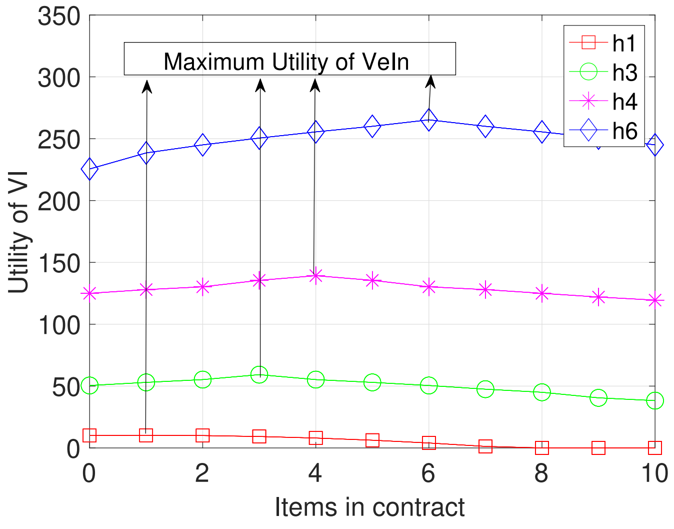

Fig. 4. Utility of VeIn with respect to honesty factor h_i.

based on the Stackelberg game. We examine the ISS scenario described in Section V-A, where VeIn knows the true effort s_i of the verifiers. We also investigate the IAS scenario, as discussed in Section V-B, where the VeIn is only aware of the performance, namely, latency l_i, of the verifier. Our evaluations attempt to capture the impact of honesty on the derived solutions. Specifically, the ISS scenario can be seen as a special subcase of the IAS scenario, whereas in ISS, VeIn and the verifiers share the same information while deciding the optimal blocksize, for example. Finally, we also compare our scheme against the fixed reward scheme proposed in [43].

1) Utility of VeIn With Respect to the Honesty Factor: We evaluate the utility of VeIn with respect to the honesty factor h_i. We assume ten verifiers (i.e., $N = 10$) with four values of honesty factors, i.e., $h_1, h_3, h_4, h_6 \in \mathcal{H}$ as defined in Table II. From Fig. 4, we can note that the utility of VeIn reaches its maximum when the corresponding contract is obtained, i.e., when h_4 with contract $\mathcal{C}^{t_2}(r_4, l_4)$ can allow the utility to be the maximum. Note that, in practice, we actually consider χ_i, which is a combination of both latency l_i and honesty h_i, for convenience here we have considered the case where $h_1 < h_3 < h_4 < h_6$. This result verifies IC in (14b), i.e., for verifiers with χ_i, there is one, and only one, optimum contract item.

2) IAS's Social Welfare With Respect to β: We examine the social welfare of IAS with respect to the VeIn's income factor

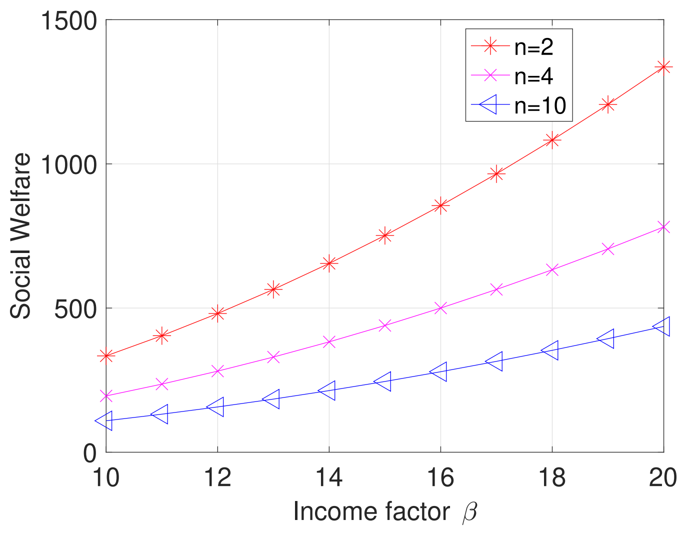

Fig. 5. Social welfare with respect to income factor β.

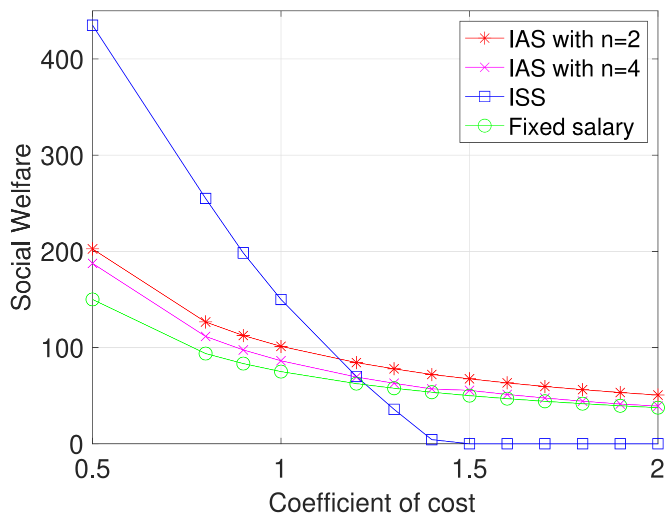

Fig. 6. Social welfare with respect to cost factor α.

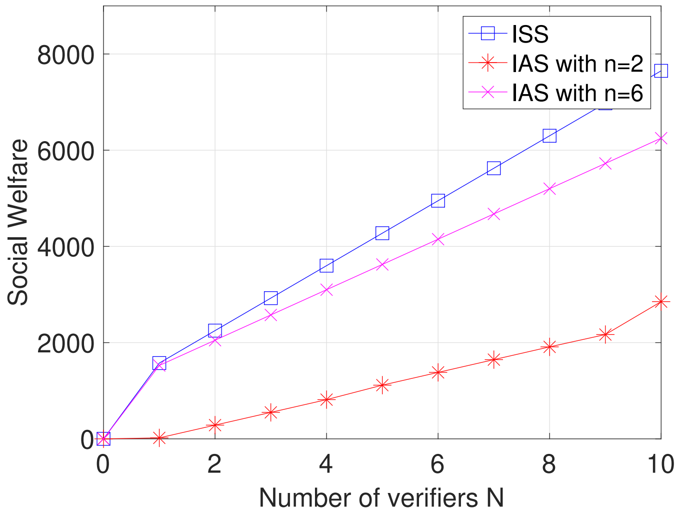

Fig. 7. Social welfare with respect to the number of verifiers N.

β. In Fig. 5, we consider different numbers of latency and honesty factors' combinations. From this evaluation, we can observe that with increasing income factor β, social welfare also increases. Second, we note that when the number n of combinations increases, the social welfare decreases. This is due to the fact that increasing the number of combinations, n, adds uncertainty to the system, which leads to a higher cost of VeIn, and causes social welfare degradation.

3) Impact of α to the Social Welfare: In Fig. 6, we compare the social welfare of the proposed IAS, ISS, and fixed salary scheme with respect to the cost factor α. We simulate IAS with two kinds of combination χ and four kinds of combination χ. Note that parameter n means that the verifiers have n different probabilities to choose from these combinations. From the simulation results, we can see that as α increases, the social welfare decreases. The reason for this behavior is that the larger the cost factor is, the larger the cost of the verifier is, which means that it is going to cost more to maintain the same blocksize. However, the IAS achieves higher social welfare when α is about 1.3. This is due to the fact that in IAS, the verifiers can choose a smaller blocksize to compensate the execution cost in order to maintain social welfare.

4) Impact of the Number of Verifiers to the Social Welfare: We also analyze the social welfare of ISS, and IAS with respect to the number of verifiers N. The results are illustrated in Fig. 7. We consider verifiers have honesty factor h_6 and latency l_6, and have different numbers of verifiers for ISS and IAS. We apply the setting of three different combinations of χ, i.e., χ_5, χ_6, and χ_7. As we can observe from the figure, the social welfare increases when the number of verifiers increases. Note that the least social welfare is observed for IAS. This is attributed to the fact that VeIn has no knowledge of the verifiers' real effort in executing the application. From this, we can conclude that the IAS costs more than the ISS to the VeIn.

5) Impact of the Probability Distributions of Latency to the Utility of VeIn: In Fig. 8, we investigate the impact of different probability distributions of latency on the utility of the VeIn under IAS. We consider five combinations of χ and ten verifiers in this simulation. For the comparisons, we assume a uniform distribution with $\mathcal{P} = \{0.2, 0.2, 0.2, 0.2, 0.2\}$, an affine distribution with $\mathcal{P} = \{0.35, 0.3, 0.2, 0.1, 0.05\}$, and also a discrete Gaussian distribution with $\mathcal{P} = \{0.1, 0.15, 0.5, 0.15, 0.1\}$. As is shown in Fig. 8, the Gaussian distribution achieves the maximum utility for χ_5 setting. In addition, we evaluate (Fig. 9) the utility of VeIn with respect to the number of verifiers for the considered probability distributions. We consider a range of verifiers [10, 14], corresponding to the deployment in a regional edge network. As we can observe from Fig. 9, the utility of VeIn increases for all distributions when the number of verifiers increases. This comparison also shows that the proposed reward scheme can cope with different scenarios in a real-world situation. In this setting, the discrete Gaussian distribution achieves the highest utility among three distributions, followed by the uniform distribution and the affine distribution. The reason for this is that the utility of VeIn increases

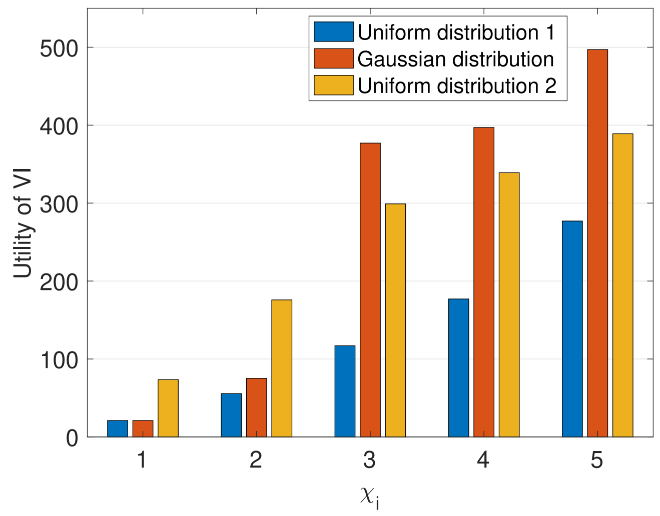

Fig. 8. Utility of VeIn with respect to different probability distribution of χ's combinations.

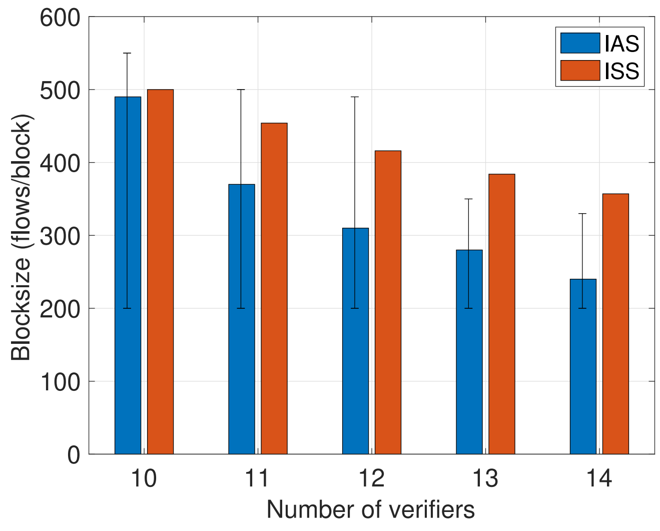

Fig. 10. Blocksize for the ISS and IAS scenarios with respect to the number of verifiers. Note that there is variance in the IAS case due to varying solutions with maximum and minimum shown on the graph.

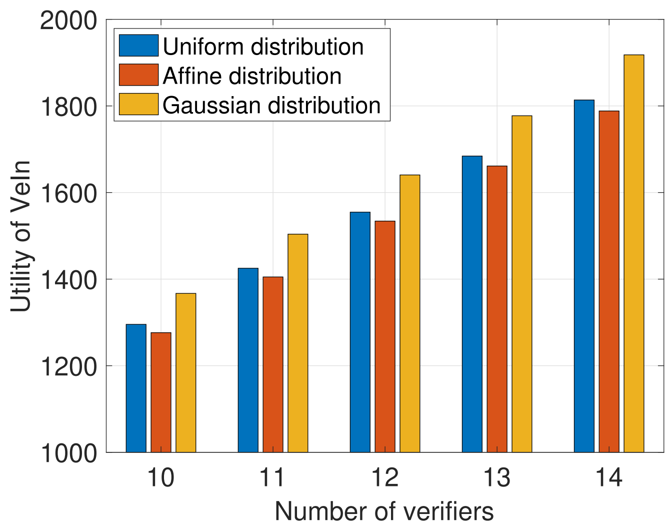

Fig. 9. Utility of VeIn with respect to the number of verifiers under different probability distribution of χ's combinations.

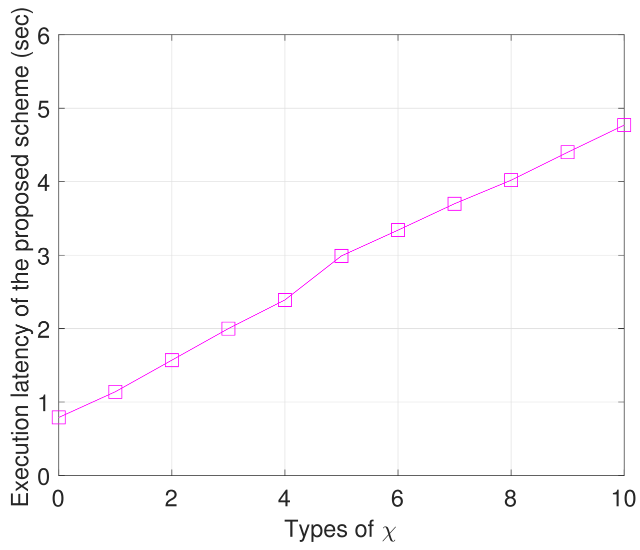

Fig. 11. Evolution of reward scheme with respect to number of χ's combinations.

when the combination χ's index i increases, which is also the case of monotonicity in Lemma 1.

6) Edge-Blockchain Performance in Terms of Blocksize: We analyze the edge-blockchain performance by considering the blocksize in the ISS and IAS scenarios in Fig. 10 with a fixed reward budget. Blocksize can represent the flow conformance task volume on the edge servers. For ISS, from (11), we know that in ISS, the optimal blocksize only relates to the reward. When more verifiers join the verification, the reward to each individual decreases, which leads to a decrease of blocksize. For IAS, we assume five combinations of χ with a limited reward budget. We observe that when the number of verifiers increases, the average blocksize in both scenarios decreases due to fixed reward. Limited reward budget leads to verifiers choosing different contracts. We also add a range indicator to the IAS case to show the biggest and largest blocksizes (the blocksize does not change in the ISS case). The smallest IAS blocksize is always the same as there are only five contracts to choose from. When there are 14 verifiers, they can only choose between two contracts due to the limited reward budget. We also observe that the ISS often has larger blocksizes than the IAS, particularly as the number of verifiers increases (with a constrained budget).

7) Impact of the Number of Combinations χ in Terms of Latency: Finally, we investigate the complexity of the proposed IAS with respect to the number of the combinations of latency l_i and honesty h_i factor values, i.e., χ_i. We should emphasize that here we are interested in the relationship of the computational complexity to the size of the problem rather than the absolute runtimes. We leave the design of a fast heuristic to future work. To investigate this relationship, we use CVX [42] to solve the proposed optimization problem shown in (14b). The results are depicted in Fig. 11. From this comparison, we can see that the proposed reward scheme is bounded by the number of χ_i's combinations and that the execution latency grows only linearly with the number of χ combinations. Furthermore, this comparison makes clear that the computational complexity of the solution will not be high.

VII. Conclusion

In this article, we have investigated a novel security solution for SDN supported by edge blockchain, which interconnects IoT networks. We proposed an architecture for blockchain-based software-defined networks. Then, we suggested the workflow of the flow verification and validation according to the architecture of BC-SDN. To support blockchain technology, we deployed BCAs with edge computing servers to reduce the computational burden on the IoT systems. Owing to the fact that we use BaaS, we have designed two reward schemes for ISS and IAS to tackle the potential moral hazard caused by BCAs hosted by a third-party edge computing provider. By using the proposed reward scheme based on contract theory, we can determine the optimal blocksize and latency of the blockchain and the corresponding reward value. Finally, we evaluated our system to demonstrate the impact of different parameters using two different incentive mechanisms. The results showed that the proposed reward scheme can achieve good social welfare. For our future work, we will consider how different flow conformance policies can be implemented within a smart contract.

References

[1] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1994–2008, Dec. 2017.

[2] J. L. Hernández-Ramos, G. Baldini, R. Neisse, M. Al-Naday, and M. J. Reed, "A policy-based framework in Fog enabled Internet of Things for cooperative ITS," in *Proc. Global IoT Summit (GIoTS)*, Jun. 2019, pp. 1–7.

[3] E. Gelenbe, J. Domanska, T. Czàchorski, A. Drosou, and D. Tzovaras, "Security for Internet of Things: The Seriot project," in *Proc. IEEE Int. Symp. Netw. Comput. Commun. (ISNCC)*, 2018, pp. 1–5.

[4] P. Amangele, M. J. Reed, M. Al-Naday, N. Thomos, and M. Nowak, "Hierarchical machine learning for IoT anomaly detection in SDN," in *Proc. IEEE Int. Conf. Inf. Technol. (InfoTech)*, 2019, pp. 1–4.

[5] A. Abdou, P. C. van Oorschot, and T. Wan, "Comparative analysis of control plane security of SDN and conventional networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3542–3559, 4th Quart., 2018.

[6] X. Wang, X. Li, S. Pack, Z. Han, and V. C. Leung, "STCS: Spatial–temporal collaborative sampling in flow-aware software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 999–1013, Apr. 2020.

[7] J. Hu, M. Reed, M. Al-Naday, and N. Thomos, "Blockchain-aided flow insertion and verification in software defined networks," in *Proc. IEEE Global Internet Things Summit (GIoTS)*, 2020, pp. 1–6.

[8] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BlockBench: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manag Data*, 2017, pp. 1085–1100.

[9] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud-edge-end in IoT: A blockchain-assisted collective *Q*-learning approach," *IEEE Internet Things J.*, early access, Jul. 7, 2020, doi: 10.1109/JIOT.2020.3007650.

[10] M. Samaniego and R. Deters, "Blockchain as a service for IoT," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber Phys. Soc. Comput. (CPSCom) IEEE Smart Data (SmartData)*, 2016, pp. 433–436.

[11] P. Bolton *et al.*, *Contract Theory*. Cambridge, MA, USA: MIT Press, 2005.

[12] M. P. Singh and A. K. Chopra, "Computational governance and violable contracts for blockchain applications," *Computer*, vol. 53, no. 1, pp. 53–62, 2020.

[13] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," *J. Amer. Med. Inf. Assoc.*, vol. 24, no. 6, pp. 1211–1220, 2017.

[14] V. Hassija, V. Chamola, S. Garg, N. G. K. Dara, G. Kaddoum, and D. N. K. Jayakody, "A blockchain-based framework for lightweight data sharing and energy trading in V2G network," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5799–5812, Jun. 2020.

[15] S. Rathore, B. W. Kwon, and J. H. Park, "BlockSecIoTNet: Blockchain-based decentralized security architecture for IoT network," *J. Netw. Comput. Appl.*, vol. 143, pp. 167–177, Oct. 2019.

[16] S. Boukria, M. Guerroumi, and I. Romdhani, "BCFR: Blockchain-based controller against false flow rule injection in SDN," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2019, pp. 1034–1039.

[17] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, and K.-K. R. Choo, "Blockchain-enabled authentication handover with efficient privacy protection in SDN-based 5G networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 28, 2019, doi: 10.1109/TNSE.2019.2937481.

[18] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu, and C. Zhao, "Blockchain-based software-defined industrial Internet of Things: A dueling deep *Q*-learning approach," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4627–4639, Sep. 2018.

[19] J. Hu, K. Yang, L. Hu, and K. Wang, "Reward-aided sensing task execution in mobile crowdsensing enabled by energy harvesting," *IEEE Access*, vol. 6, pp. 37604–37614, 2018.

[20] S. Shen, Y. Han, X. Wang, and Y. Wang, "Computation offloading with multiple agents in edge-computing–supported IoT," *ACM Trans. Sensor Netw.*, vol. 16, no. 1, pp. 1–27, 2019.

[21] K. Yang, S. Ou, and H.-H. Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile Internet applications," *IEEE Commun. Mag.*, vol. 46, no. 1, pp. 56–63, Jan. 2008.

[22] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 760–770, Jan. 2016.

[23] X. Hu, K.-K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2375–2388, Jul. 2018.

[24] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2017.

[25] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, Sep. 2017.

[26] P. K. Sharma, S. Rathore, Y.-S. Jeong, and J. H. Park, "SoftEdgeNet: SDN based energy-efficient distributed network architecture for edge computing," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 104–111, Sep. 2018.

[27] K. Kataoka, S. Gangwar, and P. Podili, "Trust list: Internet-wide and distributed IoT traffic management using blockchain and SDN," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, 2018, pp. 296–301.

[28] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[29] A. Stanciu, "Blockchain based distributed control system for edge computing," in *Proc. IEEE 21st Int. Conf. Control Syst. Comput. Sci. (CSCS)*, 2017, pp. 667–671.

[30] J. Yang, Z. Lu, and J. Wu, "Smart-toy-edge-computing-oriented data exchange based on blockchain," *J. Syst. Architect.*, vol. 87, pp. 36–48, Mar. 2018.

[31] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Dec. 2018.

[32] M. Liu, R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial Internet of Things (IIoT) systems: A deep reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3559–3570, Jun. 2019.

[33] Y. Zhang, M. Pan, L. Song, Z. Dawy, and Z. Han, "A survey of contract theory-based incentive mechanism design in wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 80–85, Feb. 2017.

[34] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled Internet of Vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2906–2920, Jan. 2019.

[35] Y. Zhang, Y. Gu, M. Pan, N. H. Tran, Z. Dawy, and Z. Han, "Multi-dimensional incentive mechanism in mobile crowdsourcing with moral hazard," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 604–616, Mar. 2018.

[36] V. Hassija, V. Chamola, G. Han, J. J. Rodrigues, and M. Guizani, "DAGIoV: A framework for vehicle to vehicle communication using directed acyclic graph and game theory," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4182–4191, Jan. 2020.

[37] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[38] C. Cachin *et al.*, "Architecture of the hyperledger blockchain fabric," in *Proc. Workshop Distrib. Cryptocurrencies Consensus Ledgers*, vol. 310, 2016, p. 4.

[39] V. Gramoli, "From blockchain consensus back to byzantine consensus," *Future Gener. Comput. Syst.*, vol. 107, pp. 760–769, Sep. 2020.

[40] B. Hölmstrom, "Moral hazard and observability," *Bell J. Econ.*, vol. 10, no. 1, pp. 74–91, 1979.

[41] M. J. Osborne *et al.*, *An Introduction to Game Theory*, vol. 3. New York, NY, USA: Oxford Univ. Press, 2004.

[42] M. Grant and S. Boyd, "CVX: MATLAB software for disciplined convex programming, version 2.1," 2014.

[43] J. Hu, K. Yang, K. Wang, and K. Zhang, "A blockchain-based reward mechanism for mobile crowdsensing," *IEEE Trans. Comput. Soc. Syst.*, vol. 7, no. 1, pp. 178–191, Feb. 2020.

Jiejun Hu received the M.Sc. and Ph.D. degrees from the School of Computer Science and Technology, Jilin University, Changchun, China, in 2015 and 2019, respectively.

She is currently a Senior Research Officer with the University of Essex, Colchester, U.K. Her research interests include mobile crowdsensing, communication networks, future network technology, blockchain, and network security.

Martin Reed (Member, IEEE) received the Ph.D. degree from the University of Essex, Colchester, U.K., in 1998.

He is currently an Associate Professor with the University of Essex. His research interests include network control planes, information centric networking, network security, and multimedia networking. He has been involved in a number of EPSRC, EU, and industrial projects in these areas, which have made contributions to IETF and 3GPP 5G standards. He has held a Research Fellowship at BT in the area of access networks. He has led a number of international research testbeds that demonstrate and evaluate novel networking protocols and architectures.

Nikolaos Thomos (Senior Member, IEEE) received the Diploma and Ph.D. degrees from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2000 and 2005, respectively.

He was a Senior Researcher with the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, and the University of Bern, Bern, Switzerland. He is currently an Associate Professor with the University of Essex, Colchester, U.K., and the Group Leader of the Communications and Networks Group. His research interests include machine learning for communications, multimedia communications, network coding, information-centric networking, source and channel coding, device-to-device communication, and signal processing.

Dr. Thomos received the highly esteemed Ambizione Career Award from the Swiss National Science Foundation. He is an Elected Member of the IEEE MMSP Technical Committee from 2019 to 2022.

Mays F. Al-Naday (Member, IEEE) received the Ph.D. degree from the University of Essex, Colchester, U.K., in 2015.

She worked as a Senior Research Officer with the Network Convergence Laboratory (NCL), University of Essex, where she is currently a Lecturer with the School of Computer Science and Electronic Engineering. Her research focuses on future network architectures, including microservice architectures, IoT communications, fog computing, next-generation content delivery networks and security, and quality of service in 5G and beyond. She has been the organizer of prestigious workshops in Sigcomm 17-18 and IFIP 17. She has actively contributed to a number of EU research projects in the area of future networking architectures.

Kun Yang (Senior Member, IEEE) received the Ph.D. degree from the Department of Electronic and Electrical Engineering, University College London (UCL), London, U.K., in 2000.

He worked at UCL on several European Union (EU) research projects for several years. He is currently a Chair Professor with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K., where he leading the Network Convergence Laboratory. His main research interests include wireless networks and communications, IoT networking, data and energy integrated networks, and mobile-edge computing. He manages research projects funded by various sources, such as U.K. EPSRC, EU FP7/H2020, and industries. He has published over 150 journal papers and filed ten patents.

Dr. Yang serves on the editorial boards of both IEEE and non-IEEE journals. He has been a Fellow of IET since 2009.