DAGSec: A hybrid distributed ledger architecture for the secure management of the Internet of Things

Igor D. Alvarenga, Gustavo F. Camilo, Lucas A. C. de Souza and Otto Carlos M. B. Duarte
Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro – GTA/COPPE/UFRJ
Rio de Janeiro, Brazil
Email: alvarenga@gta.ufrj.br

***Abstract*—The rise of 5G mobile broadband networks creates new possibilities for the Internet of Things. Billions of devices will provide comprehensive services, including e-health applications, smart grids, and industry 4.0. Distributed ledger technologies solve most security and privacy threats of current IoT systems connected to a cloud or multi-access edge communication (MEC). Unfortunately, the volume of transactions imposed by 5G networks prevents blockchain-based solutions due to scalability issues. Nonetheless, emerging solutions based on directed acyclic graphs (DAG), still require some form of centralization or global view. This article proposes DAGSec, a hybrid distributed ledger architecture that provides a secure Internet of Things environment with high throughput and low latency. Our proposal uses directed acyclic graphs and local transaction validation instead of global transaction validation to attain a high transaction rate. Furthermore, we propose a blockchain-based witness system to approximate chronological order of independent transactions.**

***Index Terms*—Internet of Things (IoT); blockchain; edge and cloud computing; next generation networks;**

I. Introduction

The emergence of 5G mobile broadband networks significantly improves the connectivity, latency, and bandwidth of the Internet of Things (IoT), enabling billions of IoT devices to deliver pervasive services. These services include smart cities, industry 4.0, environmental monitoring, autonomous agriculture, e-health, and smart transport systems [1]. IoT systems enable high-quality decision-making support and asset management. In 2020, the number of connected IoT devices had reached about 20 billion, and more than 40 petabytes of daily exchanged data [2]. However, this massive amount of data and applications raises new security challenges for the virtualized 5G architectural communication model.

Multiple IoT devices from different manufacturers collect data and provide services at the edge of the network, from IoT service providers to IoT service users. These IoT devices are connected, either directly or by a gateway, to a multi-access edge computing (MEC) node of a 5G communication network [3]. MEC nodes connect to the cloud and infrastructure providers at the Internet's core, either by dedicated private connections or the Internet. Furthermore, services are provided, and data is aggregated using network functions deployed and managed by the IoT service providers. Those network functions are organized with service function chains [4] that span from the MEC nodes at the edge of the network to cloud data centers at the network core. In addition, service functions are typically virtualized either as virtual machines or containers at MEC nodes and data centers through network function virtualization (NFV) platforms [3], [5].

IoT systems begin to collect, analyze, control, monitor, and manage various aspects of everyday life. It is imperative to provide basic security features for all available systems, such as authentication, auditability, authenticity, non-repudiation, and confidentiality and privacy. Blockchain-based systems can provide these security and privacy capabilities in the cloud environment [6] but face scalability challenges due to the consensus mechanism when applied to the edge of the network and IoT devices [1], [7], [8].

Directed Acyclic Graphs (DAGs) allow high transaction throughput while preserving the security properties offered by blockchain-based systems [9]–[11]. Like blockchains, DAGs are immutable path data structures chained by hash, but each transaction can be independently chained in a DAG. The main difference between a blockchain and a DAG is that a new transaction can reference any predecessor transaction, not just the last one. Directed acyclic graphs validate and process transactions without relying on a blocking consensus mechanism [10], [12]. Hence, DAG-based systems allow different views of the network transaction graph and maintain the transaction history consistency needed to issue a new transaction immediately [9], [10].

This paper proposes DAGSec, a hybrid distributed ledger architecture that provides security and auditability to crucial aspects of the IoT, such as service orchestration, configuration management, and storage. We propose a DAG-based ledger to ensure high-throughput transaction validation, which scales proportionally to the number of devices. Furthermore, we combine a blockchain-based ledger with a witness system to establish the chronology and ordering of DAG transactions. Our proposed architecture maintains the necessary premises for IoT security without the need for a global view of the ledger nor the need for consensus for the validation and publication of transactions.

The main contributions of this paper is a new architecture that provides: i) security and auditability to network function virtualization orchestration commands, IoT devices configuration, and data storage; ii) a high transaction rate by replacing global view decisions by local view decisions in a direct acyclic graph structure, and iii) the definition of an upper bound unit time approximation expression that enables absolute transaction ordering used by a witness system, which guarantees the correct system auditability.

We organize this paper as follows. Section II presents the considered attacker model. Section III presents DAGSec assumptions and requirements. Section IV presents the proposed architecture and transaction schemes, detailing proposed mechanisms. Section V presents the related works and state of the art. Finally, Section VI concludes the paper and presents future directions.

II. Attacker model and security analysis

In this work, we consider the traditional Dolev *et al.* attacker model [13], in which the attacker is able to *send*, *discard* or *read* any packet of the network, hence, an attacker monitor and interfere with the propagation of ledger transactions. An attacker may be either passive or active. An active attacker may connect to the network at any access point and capture all transmitted packets. An active attacker may actively interfere with message exchange by means of filtering, injecting, repeating, or creating packets. Any system participant, henceforth known as a node, the network, or the ledger, may be attacked at any time. Attackers may perform attacks on their own or coordinated with other attackers, with whom they may share information freely.

Attacks against DLT-based systems are attempts to prevent a legitimate transaction from being published on the ledger, publish a fraudulent transaction on the ledger, or stop the ledger operation. Using a blockchain data structure coupled with the asymmetric key signature, hashing, and encryption schemes can effectively hamper the attack surface area and the attacker success rate [6], [14]. For instance, the use of signed hashing impedes message modification or corruption, as well as personification attacks. Furthermore, public key pseudo-anonymization of user identity and data encryption severely limits sensitive information available to an attacker eavesdropping on the network. We assume that nodes have redundant connectivity to the ledger network through the Internet or redundant access points and that transaction-knowledge message exchange use gossip protocols [15]. These assumptions limit an attacker's ability to block a transaction from being widespread effectively [6].

In DAGSec architecture, while attackers may delay access to transaction information, they have no means to collude to decide the global valid transaction graph. However, the proposed architecture requires a Byzantine Fault Tolerant (BFT) consensus algorithm [16], explained afterward in Section IV-B, to decide on the first time the network saw a transaction, hence a percentage of well-behaved nodes, according to the chosen algorithm, is necessary for this purpose. This work does not address the case where a private key is compromised due to a node security fault but offers tools to eliminate listening network services besides the ones that are needed to interact with the proposed architecture, mitigating this attack vector [17]. Meanwhile, all active actions performed with a compromised key will be logged on the ledger, allowing auditing of the security incident.

III. Assumptions and requirements

We base our proposed architecture on the ETSI NFV [5] and MEC [18] architectural standards for 5G IoT networks. In this architecture, network function virtualization (NFV) is used to build IoT service function chains from Cloud/IoT platforms at the core of the network to the MEC nodes. A MEC node is a mirror of Cloud/IoT platform services with less computational resources at the edge of the network used to provide low latency and high bandwidth to near devices. Multiple IoT devices from different manufacturers and IoT service providers are connected to IoT service users and to the MEC nodes through edge networks [3], [19].

Our proposed DAGSec architecture makes the usual assumption that data center and MEC node owners are not malicious. Otherwise, the proposal would not provide security in an environment where all systems act in coordination against a specific tenant. Datacenter and MEC node owners are assumed to act in their best capacity to provide fairness and isolation to their hosted services. Nevertheless, DAGSec allows attackers to compromise systems and services under a data center or MEC node provider. Furthermore, the ability to audit the legitimate interaction between an IoT service provider and data center or MEC providers is maintained by the proposed architecture in either case.

We consider that DAGSec does not require additional investment in infrastructure because configuration/provision requests and logs are already required to be provided by cloud and IoT platforms. Therefore, DAGSec does not rely on economic or token incentives. Our proposal adds auditability, privacy, and non-repudiation to configuration/provision requests and logs that are already present in the current IoT/Cloud platforms. Then, we advocate a paradigm shift in orchestrating target systems providing security primitives.

IV. The Proposed architecture

This paper proposes DAGSec, an architecture that enables the IoT ecosystem to provide the required security primitives without sacrificing the scalability requirements inherent to IoT. Our proposal combines a DAG-based ledger to guarantee scalability with a blockchain-based witness system to guarantee chronology and ordering of transactions.

The proposed architecture, depicted in Figure 1, is based on the European Telecommunications Standards Institute (ETSI) standard for Multi-access Edge Computing architecture [18]. The MEC functional structure comprises Cloud/IoT platforms at the network core connected to MEC Nodes through dedicated networks. Cloud/IoT platforms host the IoT services through NFV technologies. MEC nodes are small deployments that mirror selected cloud infrastructure and applications but using fewer resources and are used to provide services to the devices connected to edge networks [19]. We propose to extend the ETSI MEC architecture for IoT with five modules: (i) the DAG module, that hosts the known DAG

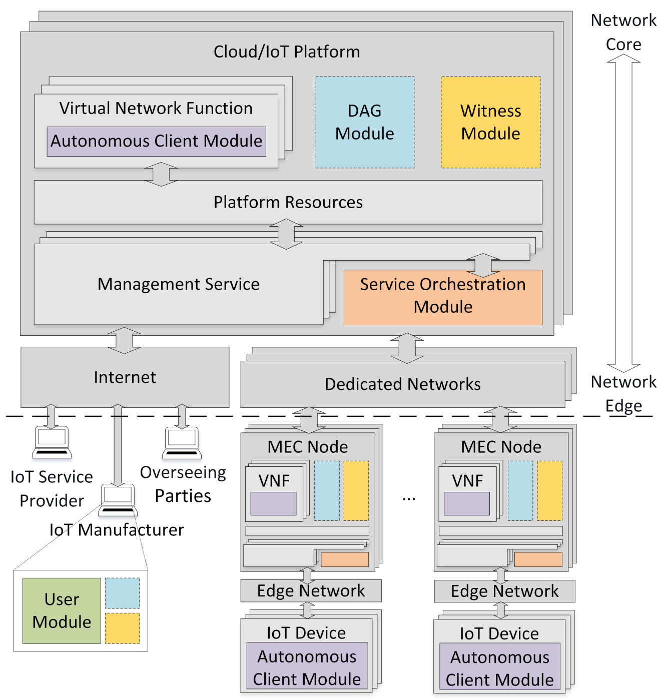

Fig. 1: Our DAGSec architecture considers 5G architectural model with five additional modules: a DAG module, a witness module, a service orchestration module, an autonomous client module, and a user module. VNF chains spanning from the core to the edge of the network provide IoT services. MEC nodes are small scale Cloud/IoT platforms used to accelerate service performance on network edge.

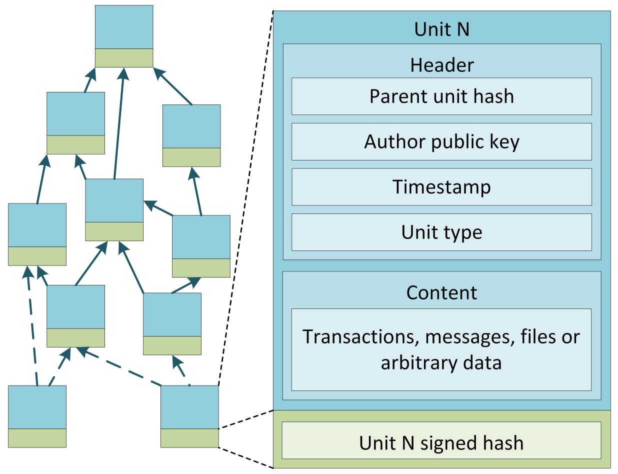

Fig. 2: The proposed DAG is composed of a set of vertices, called units. A unit is composed of an arbitrary content field and a header field. The header field contains one or more parent units hashes, the author's public key, a timestamp, and a unit type that signals the evaluation of the unit content. Finally, a signed hash of the complete unit is appended to the end of the unit.

view and is responsible for validating, forwarding, encoding, and decoding requests into DAG-units; (ii) the witness module, responsible for hosting blockchain used for DAG-unit ordering and observing new units; (iii) the service orchestration module, that is responsible for authenticating and forward service provisioning requests; (iv) the autonomous client module, that is responsible for managing configuration, applying software updates and publishing data for VNFs and IoT devices. The proposed modules exchange messages directly using the available networks or indirectly through gossip protocols.

A. Database structure

The proposed DAG ledger structure is depicted in Figure 2. The DAG is composed of hash-linked units. Each unit references previous units based on their hash and can likewise be referenced by one or more units. A unit is signed by its author and may contain one or more ordered transactions from the same author. The unit author is identified by an auto-generated public key, which makes its identity pseudo-anonymous. A transaction is a structured tuple that denotes an operation like performing a request to other systems or publishing data. The `unit type` field dictates how the transaction field is to be interpreted and is customizable by the IoT service provider in any way that suits its services. All units are timestamped. References to parents validate previous units and establish partial ordering of the transaction on the DAG, which means we can tell the absolute order of transactions in any path from the DAG origin to an arbitrary unit, like in a blockchain. Still, we are unable to tell the absolute ordering of units in disjointed paths without consulting a witness federation.

Nodes interact with the DAG using the DAG module. This module can be either standalone or a part of a role-specific architectural module. The module features a local database of known DAG-units, which is not strictly synchronized with other DAG modules. Thus, it is referred to as a local view of the DAG. All rules for unit validation are local and based on known unit types. These rules encompass field-specific validation for signatures, timestamps, and transactions. A validated unit is kept in the database and shared with known peers using a gossip protocol [15], an invalid unit is discarded. When the module requires knowledge of transaction order to perform specific validation, a witness module is consulted through a local-based witness federation list. Only prior knowledge of a partial unit set and observance of transaction rules are necessary to create a new unit. The local view validation and confirmation set the proposed system apart from similar DAG-based ledgers and allow unrestricted throughput and validation similar to a ledger-less approach. Created units are shared through a gossip protocol and directly to the target recipient when adequate. These module settings, such as unit retention, communication settings, and trusted witness list, are tailored to the hosting device, whether a small IoT device or a large VNF hosting system. It is also important to note that this rationale avoids any gate-keeping in transaction propagation and validation by a third party like it is by design in blockchain-based systems [9].

B. Witnesses: Towards a hybrid distributed ledger system

The current state of the art on DAG-based ledgers allows the improvement of only two out of three of scalability, throughput, and confirmation time characteristics over blockchain-based ledgers [20]. DAGSec opts to forego the absolute ordering of transactions in the DAG and lifts the usual global view requirement of distributed ledger systems, which implies a relaxation of strict confirmation time. Those decisions do not impact the ability to provide the intended architectural security and performance features but may impact the auditing of independent events on the DAG. Therefore, DAGSec employs an auxiliary blockchain-based witness system to approximate the absolute ordering of transactions to cope with this limitation. This is necessary because the `timestamp` field of a unit cannot be trusted, because any network participant may have an imprecise clock or intentionally lie.

A witness role is to keep track of existing units and announce when the unit was first seen to its peers. Witness nodes employ the witness module to perform no validation of a seen unit, except checking for a valid signature, integrity and if its `unit type` falls within the witnessing scope. The witness module contains a blockchain-based data repository to store data for the proposed witness system. The proposed blockchain fields are slightly modified to better cope with deterministic consensus protocols like the Practical Byzantine Fault Tolerant (pBFT) protocol [21]. These modifications include the signing of the hash field by the block proposer and the inclusion of an `acceptance proof` field that allows the transparent auditing of witness behavior by storing exchanged consensus messages.

Witnesses are organized in federations, and their member identities and network addresses are expected to be known to other federation members. There is no limit on the number of existing witness federations on the proposed architecture. Each federation is expected to have a limited and definite scope for witnessing units inside the envisioned IoT environment, for example, units belonging to a specific type of service or data center provider. As IoT systems interact, different witness federations will have intersecting witnessing roles. The ability to have multiple witness federations looking into different sections of the DAG allows for the witness blockchains to cope with the unit throughput of the underlying DAG section they observe. Witnessing does not impact unit generation nor validation. Any stakeholder can establish an independent witness federation, and this possibility can further improve the transparency and reliance on the system. A malicious or biased witness federation can be unilaterally be replaced by users with a more reputable one. System users decide explicitly in which witness federations they trust and for what services they trust those witnesses.

Witnesses reach consensus on each block to be appended to their underlying blockchain using an asynchronous Byzantine fault-tolerant protocol, such as pBFT, which can offer a high transaction throughput in a trustless federation environment if two-thirds of participants are honest [16]. Furthermore, it leverages the public key signature scheme already present in blockchains [22].

The role of a witness federation is to establish agreed-upon bounds for the time a DAG-unit is first observed. We consider the traditional timestamp and DAG-unit ordering problem at a fixed moment in time [23]. When determining the unit ordering between independent units, it is necessary to determine the lower bound $\mathbf{t}^{-}(u_i)$ and upper bound $\mathbf{t}^{+}(u_i)$ creation times for all independent units $u_i \in \mathcal{I}(w)$, where $\mathcal{I}$ represents the set of all independent units in relation to unit w. Note that the real creation time $\mathbf{t}(u)$ cannot be determined with certainty, therefore determining absolute ordering of independent units is only possible when

$$]\mathbf{t}^{-}(u_i), \mathbf{t}^{+}(u_i)[\cap]\mathbf{t}^{-}(u_j), \mathbf{t}^{+}(u_j)[= \varnothing, \ \forall \{u_i, u_j\}|_{i \neq j} \subseteq \mathcal{I}.$$

Hence, the interval between lower and upper bounds for perceived unit creation time should be as small as possible. However, given the set $\mathcal{A}(u)$ of parent units approved by u, the lower bound of this interval for a unit u is defined as $max(\{\mathbf{t}^{+}(w_i), \ \forall w_i \in \mathcal{A}(u)\})$, which results in long conflicting intervals. Therefore, we adopt the relaxation $\mathbf{t} \equiv \mathbf{t}^{+}$, which means the unit creation time will be approximated by its upper bound.

The witnessing time of a DAG-unit is announced by a witness to its federation members by means of a witnessing transaction. All transactions are signed. A signed transaction t from a node i is denoted by $\langle t \rangle_{\sigma_i}$, which implies that the transaction cryptographic hash value is signed node i and appended to to the network message containing transaction t. A witnessing transaction for a unit u by node i is a tuple in the form $\langle \mathcal{U}(u), \mathbf{t}_i^{\mathrm{w}}(u) \rangle_{\sigma_i}$, in which $\mathcal{U}(u) = \{\mathbf{u}_{\mathrm{id}}, \mathbf{u}_{\mathrm{ow}}\}$ is the identifying set $\mathcal{U}$ for a unit u composed by its signed hash value $\mathbf{u}_{\mathrm{id}}$ and owner public key $\mathbf{u}_{\mathrm{ow}}$, and $\mathbf{t}_i^{\mathrm{w}}(u)$ is the witnessing time of unit u by witness node i. Once $\mathcal{U}(u)$ for a given unit u is featured inside a block of the witnessing blockchain in transactions by one or more witnessing nodes, $\mathbf{t}^{+}(u)$ is given by

$$\mathbf{t}^{+}(u) = \mathbf{t}_i^{\mathrm{w}}(u_{\lceil (n/2)+1 \rceil}) \in ascend(\{\mathbf{t}_i^{\mathrm{w}}(u_j), \forall j = 1, \ldots, n\}),$$

where n is the number of witnesses who answered during the prepare phase of pBFT protocol or authored a witnessing transaction for unit u for the current block, and $ascend$ is the ascending sort function. If a given DAG-unit is not contained in a witness w transaction for $\mathcal{U}(u)$ of a block, we compute $\mathbf{t}_i^{\mathrm{w}}(u)$ for this witness as the time of block proposal. We can do this because at the prepare phase of pBFT protocol, if a unit u was previously unknown by some of the witnesses, we guarantee that u is now known by all witnesses who answered. This means that $\mathbf{t}^{+}(u)$ equals the witnessing time agreed by the majority of witnesses who reached consensus for the block that features DAG-unit u. Accordingly, any unit $w \in \mathcal{P}(u)$ not witnessed prior to u is witnessed with $\mathbf{t}^{+}(w) = \mathbf{t}^{+}(u)$, because $\mathbf{t}^{+}(w) \leq \mathbf{t}(u) \leq \mathbf{t}^{+}(u)$. Hence, for absolute ordering of independent DAG-units, we propose that for two DAG-units v and u, where $v \in \mathcal{I}(u)$, u precedes v if $\mathbf{t}^{+}(u) < \mathbf{t}^{+}(v)$. In case of $\mathbf{t}^{+}(u) = \mathbf{t}^{+}(v)$, we propose to select u as the preceding unit if it has more total witnessing transactions than v. If the tie still persists, then we propose that for blocks b and c, containing the witnessing transactions for units u and v, respectively, u precedes v if $(u_{id} \oplus b_{id}) < (v_{id} \oplus c_{id})$ where k_{id} denotes the hash value of element k. Nevertheless, the absolute ordering for independent DAG-units is ultimately an application or service decision, and other criteria may be considered based on a witnessing blockchain information and other trusted sources.

The `proof of acceptance` field of the block header stores all witness opinion information on block transactions and becomes immutable by the addition of a new block. Every honest node is able to compute this field locally without a new message exchange because honest nodes remain honest. Otherwise, they were malicious to begin with [16]. The `proof of acceptance` enables the identification of all malicious nodes and registers that information in the blockchain.

C. Secure IoT orchestration, configuration and storage

To provide secure orchestration, configuration, and storage facilities for IoT in a 5G environment [3], DAGSec employs three modules: (i) The user module that allows IoT service providers and IoT device manufacturers to publish data, make configuration requests and make provision requests; (ii) the service orchestration module, that receives, authenticate and forward service provisioning requests for NFV/Container/IoT management platforms; (iii) the autonomous client module, that receives and applies configuration requests and software updates to virtual network functions and IoT devices. Those three modules may exchange messages directly through the network address of the receiver and indirectly through gossip protocols. In either case, messages are always contained within a DAG-unit transaction on `content` field. Transactions do not need to be individually signed nor timestamped. All transactions on a DAG-unit always belong to the unit author and follow the presentation order.

Let $\mathcal{S}(e,r) = \{T_e^{t_{req_1}}{}_1, \ldots, T_e^{t_{req_i}}{}_n\} \cup \{T_r^{t_{resp_1}}{}_1, \ldots, T_r^{t_{resp_j}}{}_m\}$ be a session of private transactions exchanged between an emissor e and a recipient r. Transactions types in are grouped either as request t_{req} or response t_{resp} types. Response typed transactions are appended with a reference to corresponding request transaction in the form $\{u_{id}, T_n\}$, in which u_{id} is a DAG-unit identifier and T_n is the nth transaction in that unit. We denote the first request transaction from e in $\mathcal{S}$ with a type t_{req1} as $T_e^{t_{req_1}} = \langle message \rangle_{\varphi_r}$, which implies that: (i) the transaction is encrypted with a simmetric key k, then k is encripted with the public key r_{pubk} of r; and (ii) $\{r_{pubk}, k\}$ is prepended to the the transaction. Then, we denote the other transactions from $\mathcal{S}$ as $T_p^t = \langle message \rangle_{\rho_S} | p \in \{e, r\}$, which implies that the transaction is encrypted with the simmetric key established in the first transaction of $\mathcal{S}$. For a given session (S), except the first DAG-unit, each unit from a party must reference the previous unit from that party, and any unit containing a response transaction must reference the unit containing the corresponding request transaction.

IoT orchestration is performed by provision request $T^{P_{req}}$ and provision response $T^{P_{resp}}$ transactions exchanged between IoT service providers using the user module and Virtual Machine/Container/IoT platforms using the service orchestration module. Each provision request transaction demands a single response transaction. We denote a provision request transaction from a user i to a platform p as $T_i^{P_{req}} = \langle R \rangle_{\varphi_p}$, and the corresponding response transaction as $T_p^{P_{resp}} = \langle A, E \rangle_{\rho_S}$, in which R is the provision request command, A in the response to that request and E is the possible error output.

IoT data storage is performed by data T^d transaction from an IoT service provider using the user module and VNFs or IoT devices using the autonomous client module. We denote a data transaction by a user/device/VNF e as $T_e^d = \langle D, M, H \rangle$, in which D is binary data, M is the associated metadata, and H is the data hash value. Metadata is application-dependent. When the data to be stored is larger than the allowed unit size, D shall point to the address of a distributed data repository[1]. Data stored or referenced on the DAG is protected from future modifications, conferring trust to sensor-generated data repositories and software update binaries.

IoT device and VNF configuration or software updates are performed by set-configuration $T^{C_{set}}$ and get-configuration $T^{C_{get}}$ transactions. These transactions are exchanged between IoT service providers using the user module and VNFs or IoT devices using the autonomous client module. A set-configuration transaction from a user i to a device/VNF d is denoted by $T_i^{C_{set}} = \langle C, K \rangle_{\varphi_d}$, in which C is either a binary configuration file or a reference to a data transaction on the DAG, and K is a symmetric key. If the K field is not blank, it is the key to encrypt the referenced data transaction D field. A set-configuration transaction does not demand a response. A get-configuration transaction from a user i to a device/VNF v is denoted by $T_i^{C_{get}} = \langle R, D \rangle_{\varphi_v}$, in which R is the target recipient public key and D is a configuration description. If $T_i^{C_{get}}$ has a blank recipient, it will be answered by a data storage transaction T_v^d with D field set as configuration data and M field containing description D. If $T_i^{C_{get}}$ has a set recipient, it will be answered by a data storage transaction T_v^d in which D field is encrypted with the symmetric key for the session, and M field contains D as an identifier.

V. Related work

Boudguiga *et al.* proposed that blockchain based technologies enable cloud platforms to fulfill security requirements [6]. Hence, several blockchain-based architectures are proposed to manage orchestration [24]–[26], configuration [14] and slicing [27] in cloud platforms. However, blockchain-based ledgers require a global consensus to process pending transactions. Li *et al.* shows that consensus poses a severe challenge to service throughput and availability of blockchain-based systems in IoT scenarios [28], facilitating denial of service attacks as discussed by Zhao and Yu [12].

From several DAG-based ledgers are proposed in the literature [20], the most prominent for IoT usage are IOTA, Hashgraph, and Byteball. Serguei Popov proposes IOTA, a scalable DAG-based cryptocurrency architecture for the Internet of things industry [10] that centralizes transaction confirmation using a coordinator, which severely undermines transaction confirmation time and trust [20]. Churyumov proposes Byteball, a DAG system in which transaction ordering converge by

[1]For large data, we recommend IPFS (http://ipfs.io) or Storj (http://storj.io).

using special nodes with reputation-based voting power [9]. Baird and Luykx take a BFT-style approach to DAG system decentralization in Hashgraph proposal [11]. Nevertheless, in Byteball and Hashgraph, participants require the same graph view to perceive the same transaction ordering. Unlike the previously cited work, DAGSec avoids the global consensus lock for unit confirmation because our application is not financial. Thus, there is no double-spending issue. To the best of our knowledge, DAGSec architecture is the first DAG-based proposal that allows transaction creation, validation, and ordering with a restricted local view of the DAG.

VI. Conclusion

This paper proposes an architecture called DAGSec that combines a DAG-based ledger with a blockchain. The DAG-based ledger guarantees the high scalability and low latency required by the IoT environment, and the blockchain guarantees through a witness system the chronology and correct ordering of transactions. Furthermore, DAGSec provides auditability, non-repudiation, and traceability of all IoT provisioning and configuration requests.

Our proposed architecture decouples the creation and validation of transactions from the consensus mechanism, allowing DAGSec to reach the required transaction rate and scalability that are the basis of modern IoT systems.

We propose an upper bound approximation expression that enables absolute transaction ordering used by a witness system, which guarantees the system auditability of a set of unrelated events stored on the DAG. Furthermore, DAGSec enables independent auditing bodies through separate witness federations for further increase of decentralization.

Future directions envision further developing of the witnessing system to allow additional facilities like smart contracts, tokens, and payment channels.

Acknowledgment

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9. This research is also part of the Advanced and Collaborative Research Infrastructure for the Secure Internet of the Future (ACCRUE-SFI) funded by FAPESP proc. 18/23292-0.

References

[1] A. Hakiri and B. Dezfouli, "Towards a blockchain-SDN architecture for secure and trustworthy 5G massive IoT networks," in *Proceedings of the 2021 ACM International Workshop on Software Defined Networks & Network Function Virtualization Security*, ser. SDN-NFV Sec'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 11–18.

[2] R. Yugha and S. Chithra, "A survey on technologies and security protocols: Reference for future generation IoT," *Journal of Network and Computer Applications*, vol. 169, p. 102763, 2020.

[3] 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 Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[4] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, no. C, pp. 138–155, Nov. 2016.

[5] European Telecommunications Standards Institute (ETSI), "Network functions virtualisation (NFV)," Oct. 2013, https://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[6] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards better availability and accountability for IoT updates by means of a blockchain," in *IEEE EuroS PW*, April 2017, pp. 50–58.

[7] L. Chettri and R. Bera, "A comprehensive survey on Internet of things (IoT) toward 5G wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[8] X. Fu, H. Wang, and P. Shi, "A survey of blockchain consensus algorithms: Mechanism, design and applications," *Science China Information Sciences*, vol. 64, no. 2, p. 121101, Nov 2020.

[9] A. Churyumov, "A decentralized system for storage and transfer of value," 2016, "https://obyte.org/Byteball.pdf".

[10] S. Popov, "The tangle," 2018, "http://www.descryptions.com/Iota.pdf".

[11] L. Baird and A. Luykx, "The hashgraph protocol: Efficient asynchronous BFT for high-throughput distributed ledgers," in *2020 International Conference on Omni-layer Intelligent Systems (COINS)*, 2020, pp. 1–7.

[12] L. Zhao and J. Yu, "Evaluating DAG-based blockchains for IoT," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2019, pp. 507–513.

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

[14] I. D. Alvarenga, G. A. F. Rebello, and O. C. M. B. Duarte, "Securing configuration management and migration of virtual network functions using blockchain," in *IEEE/IFIP NOMS*, 2018, pp. 1–9.

[15] A.-M. Kermarrec and M. van Steen, "Gossiping in distributed systems," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 5, p. 2–7, Oct. 2007.

[16] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, p. 228–234, Apr. 1980.

[17] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker, "Off by default," in *Proceedings of the Fourth Workshop on Hot Topics in Networks (Hotnets-IV)*, 2005. [Online]. Available: http://www.icsi.berkeley.edu/pubs/networking/offbydefault05.pdf

[18] European Telecommunications Standards Institute (ETSI), "Multi-access edge computing (MEC) framework and reference architecture," Dec. 2020, https://www.etsi.org/deliver/etsi_gs/mec/001_099/003/.

[19] B. Ali, M. A. Gregory, and S. Li, "Multi-access edge computing architecture, data security and privacy: A review," *IEEE Access*, vol. 9, pp. 18 706–18 721, 2021.

[20] Q. Wang, J. Yu, S. Chen, and Y. Xiang, "SoK: Diving into DAG-based blockchain systems," 2020, https://arxiv.org/abs/2012.06128v2.

[21] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. USENIX, 1999, pp. 173–186.

[22] M. Vukolić, *The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication*. Springer, 2016, pp. 112–125.

[23] S. Popov, "On the timestamps in the tangle," 2018.

[24] N. Bozic, G. Pujolle, and S. Secci, "A tutorial on blockchain and applications to secure network control-planes," in *3rd Smart Cloud Networks Systems*, Dec 2016, pp. 1–8.

[25] R. V. Rosa and C. E. Rothenberg, "Blockchain-based decentralized applications for multiple administrative domain networking," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 29–37, 2018.

[26] S. Gu, Z. Li, C. Wu, and C. Huang, "An efficient auction mechanism for service chains in the NFV market," in *IEEE INFOCOM*, 2016, pp. 1–9.

[27] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Pérez, "NS-Bchain: A Secure Blockchain Framework for Network Slicing Brokerage," in *IEEE ICC*, 2020, pp. 1–7.

[28] Y. Li, B. Cao, M. Peng, L. Zhang, L. Zhang, D. Feng, and J. Yu, "Direct acyclic graph-based ledger for Internet of things: Performance and security analysis," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1643–1656, 2020.