A Blockchain-based System for Secure and Distributed Virtual Network Functions Orchestration

Gustavo F. Camilo, Lucas Airam C. de Souza, Miguel Elias M. Campista,
Luís Henrique M. K. Costa, Otto Carlos M. B. Duarte

Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ

***Abstract*—Service provisioning in next-generation networks, such as 5G and 6G, relies on virtualization to carry out multi-domain and multi-tenant connections. In these scenarios, virtual network functions (VNF) orchestration becomes susceptible to security threats once trust between peers cannot be assumed. This paper[1] proposes a blockchain-based system for an agile, secure, and distributed provisioning of virtual network functions in scenarios with multiple administrative domains. Our proposal employs smart contracts to deliver all stages of a service-level-agreement management life cycle automatically. We develop, implement, and evaluate a prototype of the proposed system using smart contracts running on Hyperledger Fabric. The performance evaluation results show that the system guarantees high-rate VNF provisioning, reaching hundreds of slice requests per second in a trustful way.**

***Index Terms*—blockchain; network function virtualization; service level agreement; smart contract; network slicing.**

I. Introduction

Next-generation mobile networks aim to provide connectivity models that meet user-specific demands. Network functions virtualization (NFV) is key technology to provide these customized services, allowing the flexible chaining of virtual network functions that meet the quality of service (QoS) required by applications. Service function chaining (SFC) offers a mechanism to connect virtual network functions, allowing carriers to meet user demands. Despite allowing fast and flexible service provisioning, service function chaining presents many security challenges [1]. Service function chains may include virtual network functions (VNFs) instantiated at competing service providers. As a consequence, accountability and punishment of operational failures and malicious behavior is difficult across multiple administrative domains. Thus, it is necessary to ensure *secure* provisioning of service chains, correctly identifying failures and malicious behavior on the network. In these no-mutual-trust scenarios, blockchains can provide a reliable ledger for distributed and immutable log of operations, providing transparency to users and precise misbehavior identification.

The customer and the service provider establish a service level agreement (SLA), which defines the service provider's performance levels to tenants. This SLA is essential to guarantee that the service provider has correctly provisioned the resources and honors the contracted services. Nevertheless, tenants do not have visibility into the network management and as such are unable to verify and validate the offered service, which makes the financial reimbursement process for non-compliance difficult and creates an uneven agreement [2]. Moreover, the service agreement may also contain restrictions on the user side, therefore it is necessary to verify that the user acts as determined in the contract. Smart contracts provide the automation and transparency required for the correct and reliable verification and validation of service agreements in a distributed manner.

This paper proposes a blockchain-based system for secure network slice orchestration, through a transparent and immutable log of operations. Our contributions are three-fold:

- The conception of a secure and fast system to provide transparency, non-repudiation, and traceability of network slice-orchestration operations. Our system allows the correct identification and accountability of errors and malicious behavior. We use smart contracts to automatically punish malicious behavior in the network.
- The proposal of an efficient model that meets all the requirements of an SLA management life cycle and ensures confidentiality without losing transparency.
- The development and implementation of a system prototype through smart contracts developed using the Hyperledger Fabric platform. The performance evaluation results show that the system is agile, recording around 115 slice creation requests per second.

We organize the paper as follows. Section II details the SLA management life cycle and the attacker model. Section III describes the proposed system flow, transaction types, and message exchange. Section IV describes and evaluates the performance of a prototype of the proposed system. Section V discusses the related work. Finally, Section VI concludes the paper and presents future work.

II. SLA Management Life Cycle and Attacker Model

While SLAs are just a contract between the user and the service provider, the SLA management is much more complex because it requires the provided services monitoring. We assume that the service monitoring is performed and that the service quality verification is recorded in a blockchain. Besides, we consider the SLA management life cycle proposed

[1]A preliminary version of this paper was published in Portuguese and is available at http://www.gta.ufrj.br/ftp/gta/TechReports/CSD21.pdf.

by Sun Microsystems Internet Data Center Group [3]. The life cycle is divided into six stages [4]:

1) **Service provider discovery**: The tenant chooses the service provider responsible for providing the infrastructure to run the required services.
2) **SLA definition**: The service provider and the tenant agree on the QoS parameters and define the punishments in case of failure to comply with the agreed levels.
3) **Establish agreement**: The parties involved establish a template defining the levels discussed during the SLA definition step. The service provider and the tenant sign the agreement, validating the levels and punishments.
4) **SLA violation monitoring**: The service delivered by the service provider is tested to verify compliance with the levels defined in step 2 and agreed upon in step 3.
5) **SLA termination**: The SLA expires due to previously agreed timeout or to breaches in contract compliance.
6) **SLA punishment enforcement**: The provider is punished according to the clauses defined in the contract, if he does not comply with the agreed levels.

We consider attacks which target tenants, VNFs, the blockchain, and the network. The attacker model is similar to the one defined by Dolev *et al.*, in which an attacker can read, send and discard a transaction addressed to the blockchain or any packet on the network [5]. We assume the same attack models employed by Alvarenga *et al.* [6], that describes attacks on the blockchain, tenants and VNFs and, finally, attacks on the network, which we review in the following.

Blockchain attacks prevent a transaction or a legitimate block from being added to the blockchain. Our architecture employs a Byzantine fault tolerant consensus protocol, such as pBFT [7], that mitigates this type of attack. Transactions have a signed hash to prevent corruption and tampering attacks.

Attacks on tenants or VNFs attempting to obtain orchestration information or impersonation of the target are not possible, because all transactions are signed, and confidential information is encrypted. If an attacker tries to modify the blockchain using stolen key pairs, the attempt is logged. After an incident is discovered, the victim can replace the stolen key pairs, restoring security and preventing further damage.

Finally, we assume a redundant public network, such as the Internet, which interconnects all participants to prevent network attacks that isolate a single tenant, a group of tenants, or a group of VNFs from the network.

III. The Proposed System

The proposed architecture, shown in Figure 1, ensures security in the creation of network slices using two components: a multi-domain orchestrator (MdO) and a distributed application (DApp). The MdO provides the management life cycle of a network service composed and managed across multiple administrative domains [8]. Tenants and service providers securely demand, negotiate, and create network slices using the multi-domain orchestrator. The orchestrator uses the service providers' infrastructure to create network slices that meet tenant demands. The proposed architecture includes a reference monitor that checks key performance indicators to validate compliance with SLAs. The distributed application executes smart contracts that automatically validate compliance with established service level agreements. Thus, the distributed application records all operations in the SLA management life cycle in an immutable and distributed way in the blockchain, providing transparency, non-repudiation and traceability.

Tenant users rely on the distributed application to find service providers interested in instantiating the demanded network service. Tenants record key performance indicators publicly in the blockchain as a form of reverse auction announcement to find providers. Service providers use mapping techniques to transform key performance indicators into service chains. Thus, service providers who wish to participate in the auction can plan a network slice that meets the key indicators and bid on the reverse auction. A bid consists of the proposed SLA, the offered service price, the financial restitution from the provider, and a fine from the tenant in case of non-compliance with the service levels. The tenant checks the blockchain, selects the service provider with the best bid, and generates a symmetric key to guarantee confidentiality when exchanging messages with the selected provider.

Service providers use the distributed application to verify demands and to offer network slices. The reliable and distributed log of requests in the blockchain allows for easy and quick verification of tenants' requests. After being selected and having established the contract with the tenant, the provider uses the multi-domain orchestrator (MdO) to create the service function chain. The orchestration module records the commands in the blockchain in an encrypted way using the distributed application, guaranteeing confidentiality, non-repudiation, and transparency to the tenant user.

Our proposed architecture also defines a reference monitor that communicates with the instantiated service chains, carrying out active and passive measures. Thus, the monitor performs evaluations at random times to the key performance indicators (KPIs) of the allocated chain and records the obtained values in the blockchain. At least three instances execute the reference monitor: on the provider's side, the customer's side, and an exempt third party, to measure the KPIs and even verify a measurement tampering by any of the parties. A smart contract receives the set of measurements and uses a predetermined policy to verify if they correspond to the SLA previously agreed to decide to punish or not the service provider or the tenant. Different KPIs can be introduced over time through updates to smart contracts with more agility and with less operational (OPEX) and capital (CAPEX) expenditure impact compared with a system not using smart contracts. We assume that the monitor is secure and tamper-proof by both the provider and the tenant. The implementation of this secure monitor, will be the goal of future work exploring technologies such as Intel software guard extension (SGX)[2].

[2]Available at https://www.intel.com.br/content/www/br/pt/architecture-and-technology/software-guard-extensions.html

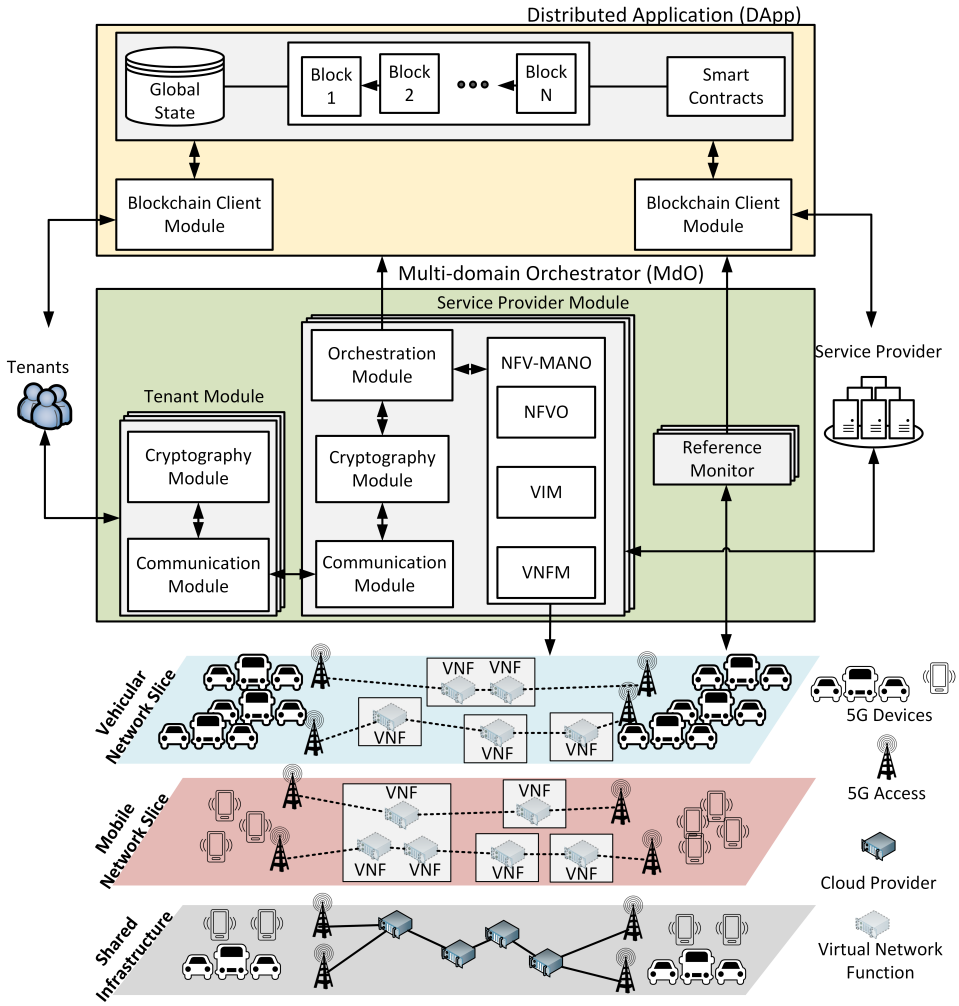


Fig. 1: The proposed architecture of the distributed network slice creation system. The multi-domain orchestrator accesses the service provider module to create specific network slices, such as vehicle networks and mobile networks. The orchestrator logs the SLA life cycle operations in the blockchain-based distributed application to provide non-repudiation and transparency.

A. *The Proposed Message and Operation Scheme*

We divide the flow of operations in the proposed system into three phases: (i) reverse auction, (ii) service functions orchestration, and (iii) SLA verification. The three parts of the system cover all stages of an SLA management life cycle. All operations generate signed transactions recorded in the blockchain, ensuring transparency and non-repudiation.

The reverse auction phase advertises a service demand and selects the service provider to create the service functions chain to serve the tenant through an end-to-end service. We adopt the message notation used by Castro and Liskov [7], where a signed transaction t from a node j is denoted by $\langle t \rangle_{\sigma_j}$. Thus, a tenant i interested in receiving a service issues a reverse auction request transaction, $T_i^{\text{A}_{\text{req}}}$, to the smart contract informing the desired KPIs and starting a reverse auction in the blockchain, defined as $T_i^{\text{A}_{\text{req}}} = \langle ID_{\text{auc}}, \text{KPI}, t_{\text{out}} \rangle_{\sigma_i}$, where the ID_{auc} field is a unique identifier for the transaction that initiated the reverse auction, the KPI field is the set of key indicators of desired performance, and τ_{out} is the reverse auction timeout.

Service providers can easily verify service requests by consulting the blockchain and can bid on the active auctions. We employ a two-phase bidding to promote a privacy-preserving auction and avoid last-minute bids from malicious providers. In the first phase, each provider generates a secret key SK_A to be used exclusively for the auction and issues an encrypted bid transaction. Providers issue a bid transaction $T_j^{\text{B}_{\text{resp}}}$, defined as $T_j^{\text{B}_{\text{resp}}} = \langle ID_{\text{auc}}, \text{Enc}_{PK_A}(V_b), t_P, C_P \rangle_{\sigma_j}$, where the ID_{auc} field identifies the reverse auction; the $\text{Enc}_{SK_A}(V_b)$ field is the encrypted bid amount using the generated secret key SK_A offered by the provider SP_j; the t_P field defines a minimum threshold for service levels, such as throughput and latency, that the service provider j (SP_j) must comply with; and the C_P field defines the financial restitution or penalty in case of breach of contract. When issuing the first bid, the smart contract blocks a value V_{stake} as a guarantee of

the real intention to proceed with $T_j^{\text{B}_{\text{resp}}}$. The value V_{stake} is agreed among the participants in the blockchain network initialization. The amount is returned to the provider SP_j after the confirmation of the orchestration transaction $T_{\sigma_j}^{\text{O}}$ or after another participant has won the auction. If the provider does not honor the promised T_j^{O}, the value V_{stake} is credited to the tenant i. In the second phase of the auction, providers reveal their generated auction key by issuing a reveal transaction $T_j^{rev} = \langle ID_{\text{auc}}, SK_A \rangle_{\sigma_j}$. Thus, every participant can easily verify the winner of the reverse auction without intermediaries.

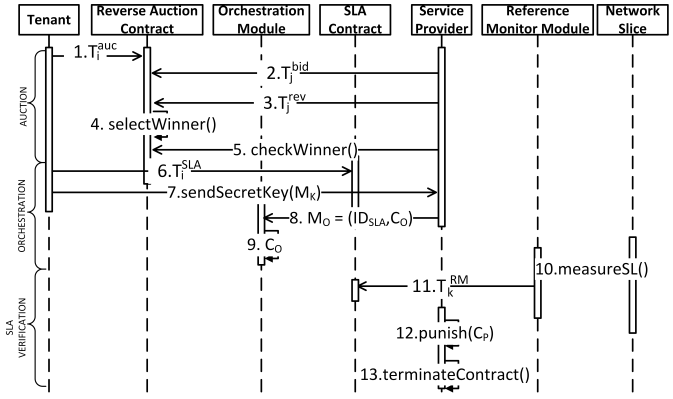


Fig. 2: Sequence diagram of the proposed system showing all the steps and proposed phases.

The service function orchestration phase records the orchestration of service chains to provide transparency to tenants and service providers. The tenant i selects the winning service provider from the bids in the reverse auction and records the agreed service levels by issuing a T_i^{SLA} transaction, defined as $T_i^{\text{SLA}} = \langle ID_{\text{bid}} \rangle_{\sigma_i}$, where the ID_{auc} field identifies the transaction that started the auction. After issuing this transaction, the service provider SP_j sends a message $M_O = (ID_{\text{SLA}}, C_O)$, containing the transaction identifier ID_{SLA} and the orchestration command C_O. The orchestration module checks the transaction ID_{SLA} in the blockchain and passes the command C_O to NFV-MANO to create the service chain. The module then encrypts the command C_O using a secret key SK_{ij} shared with the tenant i and generates the cipher $c_o = \text{Enc}_{SK_{ij}}(C_O)$. Finally, the module issues an orchestration transaction $T_{\sigma_j}^{\text{O}}$ to register the command in the blockchain. The orchestration transaction is defined as $T_j^{\text{O}} = \langle ID_{\text{SLA}}, c_o \rangle$, where the ID_{SLA} field is the identifier of the corresponding SLA transaction, and the c_o field is the orchestration command encrypted using the secret key SK_{ij} shared between the tenant and service provider. We encrypt the command to guarantee the confidentiality of the functions used by tenant i while ensuring transparency to the parties. When issuing T_j^{O}, the service provider receives the payment V_b for the service offered. The V_b value is blocked from the tenant's account when the tenant issues the transaction T_i^{SLA} to prevent malicious behavior that results in non-payment.

Service function chains in scenarios with multiple administrative domains often use functions instantiated in competing domains. Our proposed orchestration phase includes access control to prevent abuse of domain resources. Service providers register resource quotas that each of the other domains can use to orchestrate services in the blockchain when entering the network. Thus, each time a provider uses the infrastructure of another domain to host a network function, the orchestration contract automatically removes the number of resources used from the quota allocated to that domain.

The service level verification phase registers in the blockchain the performance measures of the service chain obtained at random times. A smart contract receives the measurements and compares them with the SLA previously registered with the SLA transaction T_i^{SLA} and checks if there is a measurement less than the registered threshold t_P. The contract automatically punishes the service provider in C_P if the provider fails to comply with the agreed service levels or issues a fine to the tenant when the tenant is the offender. We define the transaction T_k^{RM} sent to the smart contract by the reference monitor as $T_k^{\text{RM}} = \langle ID_{\text{SLA}}, m \rangle$, where the ID_{SLA} field is the identifier of the transaction in which the blockchain register the service levels and the m field is the set of measures for the KPIs in the service chain.

Figure 2 shows the complete sequence diagram of the proposed system, detailing the steps. The reverse auction phase covers the Service Providers discovery phase, given that the tenant T_i discovers the provider SP_j that will provide the service. The transaction T_i^{SLA} covers the definition of SLA and the agreement establishment, once the tenant and the service provider, T_i and SP_j, sign the service levels and the punishment model for breach of contract. The reference monitor verifies the service levels, covering the SLA violations monitoring. Finally, the smart contract verifies the set of measurements and applies the SLA penalties and may terminate the SLA. Thus, our proposed system secures the complete SLA lifecycle by registering all the operations in the blockchain.

IV. Prototype Development and Evaluation

We develop and implement a prototype of the proposed system[3] using the open-source platform Hyperledger Fabric v2.0 [9] for the development of permissioned blockchain using the Raft consensus [10]. The organizational aspect of Hyperledger Fabric fits the scenario of multiple administrative domains of the proposal [11]. Although the prototype uses Hyperledger Fabric, the proposed system is agnostic to a specific blockchain and can be implemented on other platforms that support smart contracts. An i7-8700 3.20 GHz CPU with 32 GB RAM and 6 cores runs the network nodes as Docker containers. A self-executing smart contract written in Go implements the proposed transaction logic. The cryptography module uses the Rivest-Shamir-Adleman (RSA) public key cryptographic system with a key size of 2048 bits. We imple-

[3]The implementation is available at https://github.com/GTA-UFRJ-team/NFVIaaS-Distributed-Orchestration.

ment the digital signature system using the public key cryptography #1 standard probabilistic signature scheme (PKCS#1-PSS) and advanced encryption standard (AES) symmetric encryption using the counter mode (CTR) as the operation mode. The results present a 95% confidence interval.

The first experiment evaluates the time added by the system in the provisioning of VNF chains. This time is the total duration of the step in which the tenant creates a secret key and sends it to the provider (sendSecretKey - #6) and the step in which the service provider encrypts the orchestration command (M_O - #7). The experiment sends a 32-byte key in the Message 6 and a 64-byte instruction in the Message 7. Figure 3a shows that the additional delay of the two proposed steps is around 0.5 second, a negligible increase in time for the participants. The overhead is insignificant compared to the high latency of orchestrating the service chain [12].

The second experiment of the cryptography module evaluates the time to encrypt a message with the message size using AES256 symmetric key cryptography. The experiment verifies the overhead caused by encryption time in the message exchange from the service provider to the tenant. As the messages exchanged do not have a fixed size, the experiment varies from 1 kB to 16 kB to consider configuration messages of complex VNF chains. Figure 3b shows that for a 16 kB message, the overhead is less than 0.35 ms, a small value when compared to the message delivery time.

The third experiment verifies the impact of the orchestration instruction size on the transaction throughput. Service providers record the orchestration instructions in the blockchain through the orchestration module. The experiment considers that more complex operations, such as instantiating functions in different domains, imply longer instructions. The scenario considers eight clients in the blockchain that issue 100 orchestration transactions each, acting as service providers registering VNF instantiation commands. Figure 3c shows that the proposed contract is agile and records hundreds of instructions even with more complex instructions, reaching 117 transactions per second up to 256 B. Our proposed system performs similarly to other proposals using the same consensus configuration [13], [11] and does not affect the performance of the consensus mechanism.

The final experiment of the distributed application assesses the growth of the blockchain by varying the instruction size. Figure 3d shows that the growth of the blockchain is stable with instructions up to 256 B when it starts to grow more significantly. Except for specific scenarios, orchestration commands on popular platforms are no more than a few dozens of bytes [14]. Therefore, blockchain growth is slow in the majority of use cases.

V. Related Work

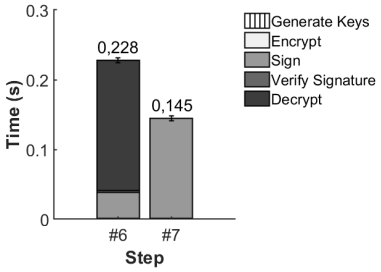
Selecting service providers to provide the necessary infrastructure to support VNFs is the first step in orchestrating service chains. An efficient way for tenants to discover service providers is through auctions and electronic markets. Gu *et al.* develop an electronic auction for the provision of service function chains (SFC) in a data center [15]. In the authors' proposal, service providers are auctioneers who sell service chains to users who act as bidders. Zhang *et al.* propose a stochastic auction mechanism to provide pricing for on-demand service chains to service providers [16]. The proposed auction mechanism is centralized, which makes auditability difficult for tenants. The blockchain provides an immutable distributed record to auctions and electronic markets [17].

Several works apply blockchain technology to ensure security in virtual network environments with multiple administrative domains. Zanzi *et al.* propose NSBchain, a blockchain-based framework that supports end-to-end network slice resource brokerage [18]. NSBchain logs the resource availability of intermediary brokers in the blockchain and employs smart contracts to manage slice requests from tenants. Nevertheless, the framework does not include the management of SLAs among participants and is restricted to distributed resource brokerage. Rebello *et al.* present BSec-NFVO, a blockchain-based system to ensure security in the orchestration of virtual networks [14]. BSec-NFVO stores the VNF orchestration commands in the blockchain, guaranteeing transparency of operations. The authors' proposal, however, does not include important steps in the provision of VNFs, such as the discovery of service providers, access control during VNF orchestration and SLA monitoring. Rosa and Rothenberg present a framework for orchestrating multi-domain services using distributed blockchain-based applications [8]. The authors present a use case in which the blockchain is used to store the access permissions of each domain. A smart contract verifies the stored access permissions to ensure access control in the VNF orchestration. The proposed access control, however, is not implemented by the authors, and the proposal does not consider all stages of an SLA management life cycle. Balachandran *et al.* [2] propose EDISON, a system for blockchain-based authentication and access control to ensure the management and orchestration of software defined networks (SDN). EDISON uses smart contracts to control tenant access to network elements, and session keys to ensure confidentiality and forward secrecy in communication between entities. The proposal guarantees security by encrypting all traffic between tenant and network element and transparency by logging packets in the blockchain, but requires a large storage space to support the complete blockchain. Furthermore, the proposal assumes that tenants and service providers know each other in advance, which does not meet the life cycle requirements of an SLA.

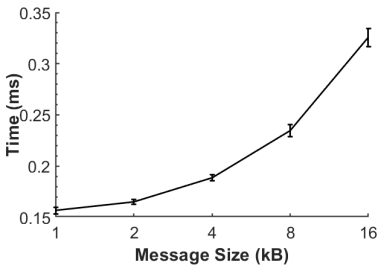
Unlike the previous work, we propose an efficient and fast system to provide orchestration of VNFs in a distributed and secure manner considering all stages of an SLA lifecycle. The system logs all provider operations in the blockchain to provide complete transparency to tenants.

VI. Conclusion

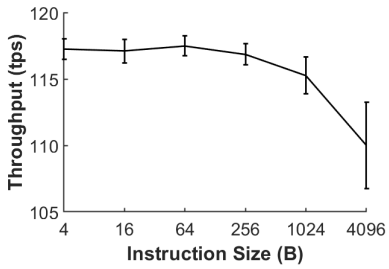
Network slicing provides customized end-to-end services to tenants by chaining virtual network functions instantiated in multiple competing domains without mutual trust. Therefore,



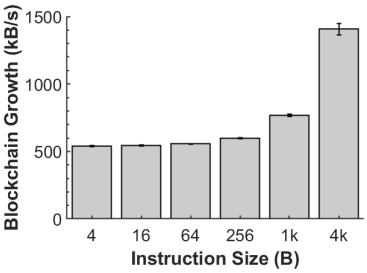
(a) Evaluation of the time increased by steps 6 and 7 of the proposed system, divided by cryptographic operations.



(b) Growth of the encryption time using AES256 in relation to the message size.



(c) Impact of the instruction size in the blockchain throughput.



(d) Blockchain growth with an increasing instruction size.

Fig. 3: Performance evaluation of the proposed system.

providers and tenants must identify and hold malicious behavior, whether instantiating the service chain or complying with the service level agreement. This paper proposes a blockchain-based system to guarantee a secure orchestration of network slices quickly and distributed in environments with multiple administrative domains. The proposed system fulfills all life cycle requirements for managing an SLA while recording the orchestration operations in the blockchain to provide transparency, traceability, and non-repudiation. The performance evaluation results of a developed prototype show that the system records orchestration operations quickly, achieving over 100 transactions per second. Moreover, the cryptographic scheme used guarantees confidentiality in an agile way without losing transparency of operations between users.

VII. Acknowledgment

This work was funded by CNPq; Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001; FAPERJ; and Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Grants 2015/24494-8, 2018/23292-0, 2015/24485-9, 2014/50937-1.

References

[1] A. M. Medhat *et al.*, "Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 216–223, 2017.

[2] C. Balachandran, P. A. C, G. Ramachandran, and B. Krishnamachari, "EDISON: A Blockchain-based Secure and Auditable Orchestration Framework for Multi-domain Software Defined Networks," in *2020 IEEE Blockchain*, 2020, pp. 144–153.

[3] L. Wu and R. Buyya, "Service Level Agreement (SLA) in Utility Computing Systems," 2010.

[4] A. Maarouf, A. Marzouk, and A. Haqiq, "Practical modeling of the SLA life cycle in Cloud Computing," in *2015 ISDA*, 2015, pp. 52–58.

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

[6] 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 *2018 IEEE/IFIP NOMS*, 2018, pp. 1–9.

[7] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.

[8] 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.

[9] E. Androulaki, *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, p. 30.

[10] D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm," in *USENIX ATC 14*. Philadelphia, PA: USENIX, Jun. 2014, pp. 305–319. [Online]. Available: https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

[11] L. A. C. de Souza, G. Antonio F. Rebello, G. F. Camilo, L. C. B. Guimarães, and O. C. M. B. Duarte, "DFedForest: Decentralized Federated Forest," in *IEEE Blockchain*, 2020, pp. 90–97.

[12] G. A. F. Rebello, I. D. Alvarenga, I. J. Sanz, and O. C. M. B. Duarte, "Bsec-nfvo: A blockchain-based security for network function virtualization orchestration," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[13] G. F. Camilo *et al.*, "AutAvailChain: Disponibilização segura, controlada e automática de dados IoT usando corrente de blocos," *SBRC*, 2020.

[14] G. A. F. Rebello, I. D. Alvarenga, I. J. Sanz, and O. C. M. B. Duarte, "BSec-NFVO: A Blockchain-Based Security for Network Function Virtualization Orchestration," in *IEEE ICC*, 2019, pp. 1–6.

[15] 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.

[16] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online Stochastic Buy-Sell Mechanism for VNF Chains in the NFV Market," *IEEE JSAC*, vol. 35, no. 2, pp. 392–406, 2017.

[17] G. F. Camilo, G. A. F. Rebello, L. A. C. de Souza, and O. C. M. B. Duarte, "AutAvailChain: Automatic and Secure Data Availability through Blockchain," in *IEEE GLOBECOM*, 2020, pp. 1–6.

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