Collaborative and Verifiable VNF Management for Metaverse With Efficient Modular Designs

Dongxiao Liu, *Member, IEEE*, Cheng Huang, *Member, IEEE*, Liang Xue, *Member, IEEE*, Weihua Zhuang, *Fellow, IEEE*, Xuemin Shen, *Fellow, IEEE*, and Bidi Ying, *Member, IEEE*

***Abstract*— The metaverse is envisioned to create immersive and virtual worlds for people to experience interoperable 3D applications. However, the real-time, interactive, and multi-media characteristics of the metaverse applications require strict quality-of-service (QoS) on the underlying networking architecture, including high throughput, ultra-low delay, and human-centric service configurations. Network function virtualization (NFV)-enabled networking resource management can provide a promising solution to service-oriented QoS satisfaction for metaverse users. In this paper, we propose a blockchain-based collaborative and verifiable virtualized network function (VNF) management scheme for metaverse, named BVNF+. BVNF+ enables multiple network providers across different trust domains to abstract their services as VNFs and collaboratively manage end-to-end network slices for human-centric network services in metaverse. To address the design challenge of balancing the on-chain and off-chain overheads, we decouple the computations of VNF queries into modular components based on software and hardware verifiable computation (vc) approaches. Our modular strategy can achieve on/off-chain computation and communication efficiency while keeping low usage of the secure hardware. We conduct security analysis and extensive experiments based on a real-world blockchain testing network. The analysis and experimental results demonstrate that BVNF+ is both secure and efficient as compared with the existing works.**

***Index Terms*— Metaverse, verifiable pruning, human-centric networking, virtualized network function (VNF), VNF query.**

I. Introduction

METAVERSE is envisioned as the immersive Internet that creates a digital world for people to experience and live a fully virtual life [1]. Through interactive devices, such as virtual reality (VR) and augmented reality (AR) headsets, as well as wireless body sensors, people can connect to the digital metaverse to enjoy a wide range of applications [2], including 3D-gaming and online social networks. With the technological advances in computing, artificial intelligence, and networking, metaverse has become an increasingly promising paradigm and thus has attracted extensive attention from academia and industry [3]. Among the enabling technologies for metaverse, networking technology plays a vital role to provide not only seamless connectivity but also reliable transmission guarantees for metaverse services [4]. As the metaverse is required to provide users with real-time interactions and high-speed delivery of multimedia contents, it can put a huge burden on current networking architectures. Moreover, metaverse applications usually involve a large number of end users with differentiated quality-of-service (QoS) requirements [5], which requires flexible and automatic network management for efficient network resource utilization [6].

To cope with stringent QoS requirements, network function virtualization (NFV) [7] can enable abstractions of network services as virtualized network functions (VNF) to provide end-to-end network slices for diversified and human-centric metaverse applications [8]. Complied with decentralization features of metaverse, the NFV-enabled networking architecture for metaverse is required to have a multi-provider paradigm where VNFs can be provisioned by different resource providers [9]. First, the multi-provider paradigm aims at efficient access resource sharing among wireless operators [10], [11], [12] to reduce infrastructure deployment and management costs for metaverse applications. Second, the paradigm is expected to enable flexible slice configurations by integrating specialized VNFs from technological vendors, such as a packet inspection function at a cloud server [13] or an artificial-intelligent function at an edge server [14]. More specifically, a live music concert via VR headsets requires a network slice on demand that consists of content caching function from edge providers, ultra-reliable and low-latency wireless communication links from operators [15], packet routing function at the cloud, etc.

With the potential benefits, the multi-provider NFV paradigm for metaverse also raises many challenges. First, the lifecycle of a network slice is complex from slice configuration, creation, deployment to deletion, which requires extensive collaborations between different providers. Second, it is necessary to implement service-level agreements (SLAs) for slice users and multiple providers [11]. The SLAs specify the terms of provided services that must be complied with by each party. Finally, since the VNF providers usually come from different trust domains, the providers cannot

Manuscript received 15 March 2023; revised 1 August 2023; accepted 31 August 2023. Date of publication 21 December 2023; date of current version 1 March 2024. This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada and in part by Huawei Technologies Canada. *(Corresponding author: Cheng Huang.)*

Dongxiao Liu was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1 Canada. He is now with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.

Cheng Huang, Weihua Zhuang, and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: c225huan@uwaterloo.ca).

Liang Xue was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. She is now with the School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada.

Bidi Ying is with Huawei Technologies Canada, Ottawa, ON K2K 3J1, Canada.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3345422.

Digital Object Identifier 10.1109/JSAC.2023.3345422

simply agree on a single trusted manager [16] for VNF management.

Addressing the challenges requires a distributed, transparent and secure VNF management framework for the multi-provider networking architecture in metaverse. Consortium blockchain [17], [18] is a distributed database for industrial partners to collaboratively manage their shared business operations. Due to its nature of transparency and immutability, the blockchain is recently adopted for reliable and distributed management of VNF lifecycle, slice configurations, and SLAs [8], [9], [10], [11], [19]. More specifically, the blockchain can help multiple providers collaboratively record information about their VNFs on its storage within VNF lifecycles. For a specific metaverse service, the blockchain provides suitable VNFs from its VNF repository to form a network slice, and enforces an SLA (as a smart contract) for users and VNF providers. However, the above-mentioned approach poses prohibitively expensive implementation costs, as the blockchain distributes storage to each full node and uses consensus protocols to maintain storage consistency [20].

On/off-chain computation models can be constructed for practical blockchain-based VNF management [21]. An external (probably untrusted) computing and storage entity (VNF manager) can be introduced to relieve the blockchain from heavy storage and computation burdens. With the integration of verifiable computation (vc) techniques, such as succinct non-interactive argument (SNARG) [22], [23] and trusted execution environment (TEE) [24], the external entity can verifiably provide VNF query services for slice configurations [25], [26]. By doing so, the expensive on-chain computation and storage overheads for VNF management can be significantly reduced. However, the on/off-chain computation models cannot be directly applied to design blockchain-based VNF management. First, the VNF management is complicated with operations from VNF query to slice configurations [19]. A simple one-for-all vc framework cannot be efficiently adopted to various VNF operations with different computation features. For example, to support fine-grained slice configurations, VNF queries should support versatile functionalities [27], such as keyword or membership matching. Second, the vc techniques usually increase the off-chain processing cost at the VNF manager. The SNARG-based approaches usually abstract VNF operations into arithmetic-circuit computations [28], [29], which can significantly increase the computational costs for generating verifiable proofs at the manager. Third, a direct design from existing vc techniques can increase the operational cost at the manager. For TEE-based approaches, such as Intel software guard extensions (SGX) [24], [30], programs are executed within a secure hardware for achieving computation confidentiality and integrity. With the limited size of secure memory in current TEE implementations, it usually requires reducing the size of confidential programs and data for operational efficiency and execution security [31]. At the same time, trusted key provisioning for either the SNARG or TEEs in a blockchain environment needs further attention. To this end, it is not a trivial task to design off-chain verifiable VNF management solutions that strike a balance between versatile functionalities, proving/verification efficiency, and the management cost of TEEs.

In this paper, we propose a blockchain-based collaborative and verifiable VNF management scheme for metaverse with modular designs, named *BVNF+*. First, we focus on the designs of verifiable VNF queries by identifying versatile query functionalities including keyword, range, membership and prefix matchings. Second, we design succinct on-chain data structures from commitments and Merkle trees for storing VNF information with a distributed generation and aggregation mechanism. Third, we decouple the computations of the versatile VNF query and present a modular design by tailoring the SNARG and TEE. More specifically, we adopt a two-level SNARG for verifiable VNF query with dictionary pruning and propose a secure enclave for communication-efficient pruning proof verifications. Furthermore, we identify interfaces between the SNARG and TEE components for secure on-chain verifications with attestation-based key provisioning.

The main contributions of this work are summarized as follows:

- We design a blockchain-based collaborative and verifiable VNF management scheme with versatile query functionalities for effective slice configurations in metaverse.
- We design modular instantiations for verifiable VNF query that takes advantage of both SNARG and the TEE. While achieving efficient off-chain proof generations and on-chain pruning proof verifications, *BVNF+* reduces operational cost by loading limited computations into the secure enclave.
- Via thorough security analysis, we demonstrate that *BVNF+* achieves verifiable VNF query. With extensive experiments on a real-world blockchain testing network, we demonstrate that *BVNF+* is efficient for both off-chain VNF query and on-chain verifications.

The rest of this paper is organized as follows. We formulate the verifiable VNF query problem with security model and design goals in Section II. We present preliminaries for designing *BVNF+* in Section III, and present detailed designs of *BVNF+* in Section IV. In Sections V and VI, we give security analysis of *BVNF+*, and present on-chain and off-chain performance evaluations of *BVNF+*, respectively. We review the related works of *BVNF+* in Section VII, and conclude this paper in Section VIII.

II. System Model, Threat Model, and Design Goals

A. System Model

We consider an NFV-enabled architecture for metaverse. As shown in Fig. 1, the architecture consists of physical, virtual, and networking spaces. Users are equipped with interactive devices, such as augmented/virual reality (AR/VR) headsets, on-body sensors, and wearable devices, to enjoy the immersive life in the metaverse. The users can play the role of digital avatar for a wide range of applications in the virtual space [1], including 3D gaming/touring, and online social networks.

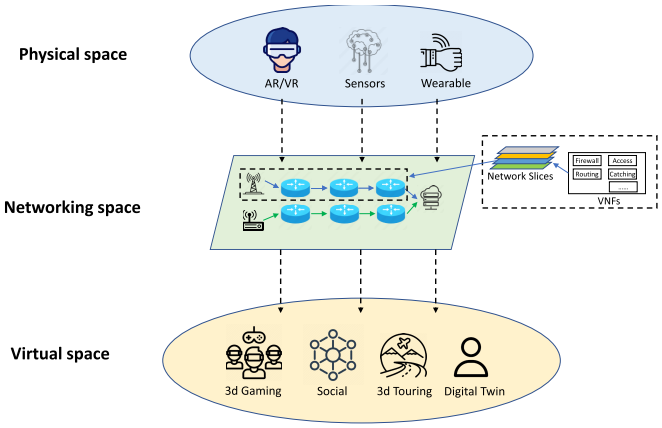


Fig. 1. System model.

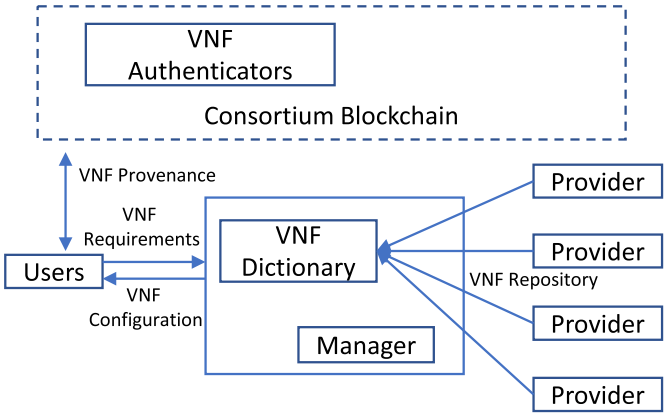


Fig. 2. Multi-provider VNF management for metaverse.

The metaverse applications, such as 3D multi-player gaming and VR touring, require real-time human-machine interaction and multi-media content delivery. This can lead to high delay and throughput requirements on the current networking architecture that connects the virtual and physical worlds [2]. To meet the requirements, *BVNF+* adopts an NFV-enabled architecture, where the network services are abstracted as VNFs, such as network catching function at the edge and reliable routing function at the network core. End-to-end slices consisting of multiple VNFs can be constructed and deployed at different network nodes, i.e., access points, switches, and the cloud, to meet the service requirements of various metaverse applications. The NFV-enabled architecture for metaverse has the following benefits:

- From the perspective of service providers, NFV enables efficient sharing and management of network resources for metaverse applications. Moreover, flexible and dynamic service provisioning can be deployed in a cost-efficient way [32].
- From the perspective of users, network slices can be configured based on user's service requirements and preferences. As a result, human-centric and differentiated metaverse services can be better provisioned with the NFV-enabled architecture.

Under the system model, we focus on VNF management issues in the networking space. The metaverse is envisioned to have an ultra-distributed architecture [2] where the network resource providers come from different trust domains. Therefore, the providers should collaboratively manage network slices for differentiated applications in metaverse, which can lead to a multi-provider VNF management paradigm. As shown in Fig. 2, there are three entities for the multi-provider VNF management paradigm: manager, users, and providers. Moreover, *BVNF+* uses the blockchain as a shared platform for providers to store and update VNF information and conduct slice configurations.

- Manager is a computing and storage unit, equipped with a secure hardware (i.e., Intel SGX), and plays the role of VNF broker in *BVNF+*. It stores VNF repositories from multiple providers as a VNF dictionary. The manager processes VNF requirements from users and replies with VNF configurations.
- Users are people or enterprise customers in metaverse. They enjoy flexible and pay-as-you-go network services for metaverse, represented by a network slice with a chain of VNFs.
- Providers are owners of VNFs. They can be a mobile operator to provision radio access functions or cloud/edge computing nodes to provision network middlebox services, such as firewall and routing.
- Blockchain is a distributed ledger maintained collaboratively by a consortium committee. It is a shared storage for the manager and providers to collaboratively manage VNF authenticators and queries.

To achieve human-centric slice configuration for metaverse applications, a key component in the multi-provider paradigm is the VNF query process. The manager collects VNF information (location, functionality, performance metrics, etc.) from providers as a VNF dictionary. With service requirements from users, the manager finds corresponding VNFs for the users to configure a network slice. Users will pay service fees to the manager and providers for provisioning VNFs and managing slice configuration, which can be specified in an SLA. The blockchain stores succinct digests (authenticators) of VNF information of different providers that can provide provenance for query processing at the VNF manager.

B. Threat Model

We focus on the threat model regarding the entities in the VNF query process. Users and VNF providers are honest in terms of VNF repository and query construction. Users faithfully provide their VNF requirements and will accept VNF configurations when the requirements are met. VNF providers faithfully supplement their VNF information to the manager and store corresponding digests on the blockchain. The manager is a rational entity that may not always follow the VNF query protocol due to the lack of management transparency and efficient regulation. For example, the manager may provide users with VNFs of higher prices for profit considerations. Blockchain is a trusted distributed ledger that can provide immutable ledger storage and secure ledger state updates.

TABLE I
ABBREVIATIONS AND NOTATIONS

CRS	Common reference string
MHT	Merkle hash tree
NFV	Network function virtualization
SGX	Software guard extensions
SNARG	Succinct non-interactive argument
VC	Verifiable computation
TEE	Trusted execution environment
D	VNF dictionary
$\mathbb{G}$	Elliptic groups
H	Hash function
Q	VNF query
V	VNF information vector
$\mathbb{Z}_P$	Integers with order p
π	Proof

C. Design Goals

Under the system model and the threat model, we have two design goals for *BVNF+*:

- Verifiable VNF query: This property ensures that VNF dictionary from providers and queries from users are authenticated, and that query process is correctly executed by the manager. At the same time, various VNF query functionalities should be supported for fine-grained slice configurations, including keyword, range, membership, and prefix matching. It should be noted that fake information provided by users or providers are out of scope of this study.
- Efficient on/off-chain overheads: The verifiable VNF query should be efficient for off-chain computation at the manager. Further, the VNF query results should be efficiently verified and the corresponding proof size should be succinct on the blockchain.

III. PRELIMINARIES

A. Cryptographic Background

We adopt the pairing-friendly elliptic groups with a prime order [33]. We denote $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ as a set of cyclic groups, where the prime order is p and a ring of integers is denoted as $\mathbb{Z}_P$; $[n]$ as a set of integers from 1 to $n \in \mathbb{Z}_P$; $H : (0,1)^* \rightarrow (0,1)^{256}$ as a collision-resistant hash function, such as SHA256, that maps an arbitrary-length string to a 256-bit string.

A digital signature is an asymmetric cryptographic scheme that proves the signer identity of a digital message [34]. It includes three algorithms:

- E.KeyGen($\mathbb{G}, \lambda$), taking a security parameter, λ, and $\mathbb{G}$ as inputs, and generating a public/private key pair, (pk, sk), as output.
- E.Sign(m, sk), takeing a message, m, and the private key as input, and generating a signature (π_s) on m as output.
- E.Verify(π_s, m, pk), taking a message, a public key and a signature as input, and deciding accept or reject as output.

A Merkle hash tree (MHT) is an authenticated data structure to generate digest and membership proof for a set of elements [35]. It includes three algorithms:

- M.Setup(M), taking a security parameter explicitly and a set of messages, $M = \{m_i\}$, as input, and generating a balanced binary Merkle tree (T) with a root (T_0) as output. Each leaf node in T is the hash of a message m_i and each non-leaf node is the hash of its two children.
- M.Prove(m_i, T), taking a message, $m_i \in M$, and the Merkle hash tree as input, identifying a path from m_i to T_0 and returning the siblings for every node along the path as a proof π_m. That is, the proof length is increasing with the height of T.
- M.Verify(π_m, m_i, T_0), re-computing the Merkle root using π_m and m_i and checking if the re-computed root equals T_0, and deciding accept if the check passes or reject otherwise.

B. Verifiable Computation

Verifiable computation enables result verifications of a general function, $F(x) \rightarrow y$. Given F and the input (x), the output (y) can be efficiently verified with a proof (π_s), without the need to redo the computation. The verifiable computation should satisfy the following security properties:

- Input authenticity: Input x should come from an authenticated source.
- Execution correctness: The function should be correctly executed with x. For $F(x) \nrightarrow y$, a computationally-bounded adversary cannot forge a valid proof that passes the verification.

Various techniques can be utilized to construct vc frameworks, such as SNARG [22], [36], zero-knowledge proof [37], and secure hardware [24]. Here, we focus on techniques with *succinct* verifications. That is, the proof size and the computing overhead for proof verification should be succinct on the blockchain regardless of the size and the complexity of the computing function. In the following, we present an overview of two vc techniques with efficient verifications: SNARG and SGX.

1) Succinct Non-Interactive Argument: The SNARG is a vc framework for general computations represented by arithmetic circuits. A toolchain of the SNARG usually consists of the following components: (1) A subset of C programs or Java programs can be written to represent a computing function with specified inputs and outputs [22], [29]; (2) A program-to-circuit compiler can convert the program to an arithmetic circuit with the same inputs and outputs; (3) The circuit can be converted to a quadratic arithmetic program (QAP) where the circuit evaluation is accordingly converted to a divisibility check with a target polynomial of QAP; (4) The divisibility check of QAP is efficiently instantiated in an elliptic group with pairing. For users with expertise in designing circuits, the first two steps in the toolchain can be omitted.

From a high-level abstraction, SNARG consists of three algorithms: S.Setup, S.Prove, and S.Verify. S.Setup takes the computing function as inputs and outputs common reference string (CRS) of the computing function; S.Prove takes computing inputs and CRS, and outputs computing results with a proof π_s; S.Verify takes CRS, the computing inputs and outputs, and π_s. It outputs either accept or reject. For

efficient on-chain verifications, the program inputs in the verification can be replaced with the corresponding succinct and trusted commitment [23]. More specifically, for input $m_i \in \mathbb{Z}_P,\ i \in [n]$, the corresponding commitment should be honestly calculated as follows [38]:

$$Com = \prod_{i=1}^{n} g_i^{m_i}, \tag{1}$$

where $\{g_i\}$ is a set of linearly independent generators from $\mathbb{G}_1 \in CRS$. At the same time, some m_i can be set to random numbers from $\mathbb{Z}_P$, which can increase the randomness of the generated commitment. The use of commitments in verifications is critical for the design of efficient on-chain verifications. For the two security properties of vc, SNARG achieves execution correctness based on its *soundness* property and the trusted setup of CRS; SNARG ensures the input authenticity of the verification with the trusted input commitment.

2) Software Guard Extensions: The SGX supports secure program executions in a protected hardware space [24], [30]. With SGX, an enclave can be created in an Intel platform for secure computations. The enclave can load program codes and the initial data into the protected hardware space, which achieves the execution integrity and confidentiality of the program. The enclave can also communicate with a host application through ECALL and OCALL to receive data at run time and to output execution results from the protected hardware to the host application. There are two important native mechanisms of SGX:

Attestation: It is important to authenticate the code and initial data in a user enclave, which is done by the attestation service [39]. Specifically, an Intel-provided quoting enclave can generate a measurement of the initial data and code of the user enclave. The quoting enclave then signs the measurement with a private group signature key. An attestation service provider (e.g., Intel) can verify the measurement and establish a secure channel with the user enclave to provision secrets.

Sealing: During the runtime of an enclave, the enclave may generate data that should be stored for use in the future. However, the protected memory cannot be used to store the data after the enclave is offline. To address the issue, SGX provides a native method called 'sealing', to encrypt and store the data on the unprotected local storage of its host machine. Later, when exactly the same enclave is loaded, the enclave can access and decrypt the stored data within its secure hardware.

While the SGX achieves efficient and secure computing, it has the limited size of protected space. As a result, the paging cost is high when the memory requirements of a program is high. At the same time, SNARG-based vc is extremely slow in non-algebraic computations, such as computing a hash. Therefore, it is important to design a vc framework with modular use of SNARG and SGX, which only operates essential operations in the enclave [31] and avoids inefficient computing instantiations with SNARG.

C. Consortium Blockchain and Smart Contract

Consortium blockchain [18] is a distributed ledger maintained by a committee of industrial partners. Since the industrial partners have a certain degree of mutual trust, more efficient consensus protocols compared with the public blockchain, such as RAFT, can be implemented. The consensus protocol helps the industrial partners to maintain storage consistency of the shared ledger. The ledger storage can be used for recording business collaborations among partners, whose terms and conditions are defined by smart contract. More specifically, a smart contract defines (1) what to store on the ledger, and (2) who and how to change the stored data. Anyone who is authorized to change the ledger data can send contract calls to blockchain nodes, which will be verified by blockchain nodes before finally being confirmed on the ledger.

We rely on Hyperledger Fabric [18] to provide the consortium blockchain service in *BVNF+*. Hyperledger Fabric provides certificate-based membership management, blockchain channel management with plug-in consensus protocols, and contract implementations from flexible library dependencies in JAVA.

IV. Proposed *BVNF+*

In this section, we present the designs of *BVNF+*. First, to support human-centric networking in metaverse, we present a design of verifiable VNF queries with various query functionalities. We elaborate on how different query functions can be efficiently instantiated for succinct on-chain verifications with the SNARG. Second, to improve the on-chain storage efficiency, we design a distributed dictionary authenticator generation and aggregation mechanism. Third, we further introduce a dictionary pruning strategy to improve the off-chain proving efficiency at the manager. We design a communication-efficient dictionary pruning method with SGX-based proof generation.

A. Versatile VNF Query

In *BVNF+*, VNF query is defined as a multi-dimensional feature vector:

$$Q = (q_1, q_2, \ldots, q_n). \tag{2}$$

In (2), q_i can be an integer in $\mathbb{Z}_P$ for a dictionary-based keyword query, a range $(r_L, r_R) \in \mathbb{Z}_P^2$ for a range query, integer $mem \in \mathbb{Z}_P$ for a membership query, or a set of integers, $(p_1, p_2, \ldots, p_{n_q}) \in \mathbb{Z}_P^{n_q}$, for a prefix query.

A single VNF dictionary, D_s, is defined as follows:

$$\begin{aligned} D_s &= (V_1, V_2, \ldots, V_{n_v}), \\ V_i &= (e_1, e_2, \ldots, e_n), \end{aligned} \tag{3}$$

where V_i is a VNF description vector that consists of n features. Similar to the query, each feature, e_i, can be an integer that represents a keyword w from the keyword dictionary, an integer that represents a numeric value for a VNF attribute, a set of integers, $(mem_1, mem_2, \ldots, mem_{n_m}) \in \mathbb{Z}_P^{n_m}$, that represents a set of n_m elements, or a set of integers $(i_1, i_2, \ldots, i_{n_f}) \in \mathbb{Z}_P^{n_f}$ that indicates sequence $i_1.i_2.,,,.i_{n_f}$.

Based on the definitions of VNF query and VNF dictionary, a VNF query execution takes Q and each $V \in D_s$ as inputs to compare each $q_i \in Q$ with corresponding $e_i \in V$. It should support the following four matching rules:

- Equality test: For a query with q_i against e_i in V, it tests if q_i equals e_i, such as VNF functionality or system version check;
- Range test: For a query with (r_L, r_R) against e_i in V, it tests if e_i lies in (r_L, r_R), such as VNF price or performance metrics;
- Membership test: For a query with mem against $(mem_1, mem_2, \ldots, mem_{n_m})$ in V, it tests if mem matches any mem_i in the set, which can simulate answering a multi-choice question;
- Prefix test: For a query with $(p_1, p_2, \ldots, p_{n_q})$ against $(i_1, i_2, \ldots, i_{n_f})$ where $n_f <= n_q$, it tests if the latter is a prefix of the former. The query can determine the affiliation relationship between two items.

We use a conjunctive matching strategy for the VNF query in *BVNF+* with the combinations of arbitrary query modules. That is, a VNF (V) is said to be matched with a query if for any query item against the corresponding item in V, the matching test passes. Moreover, a VNF query function with Q and D_s as inputs will output indexes of all matched VNFs, where VNFs can be indexed from 1 to n_v in an increasing order.

B. Distributed Dictionary Authenticator Generation

The *BVNF+* enables multi providers for VNF management in metaverse. For efficient verifications of a VNF query, the providers need to pre-compute authenticators of their VNF dictionaries. Suppose there are n_p providers in *BVNF+* and the i-th provider is denoted as P_i. For illustrative simplicity, we assume that each P_i has a VNF dictionary D_i of m VNFs, given by

$$D_i = (V_{i,1}, V_{i,2}, \ldots, V_{i,m}). \tag{4}$$

The providers can work with the manger to upload their VNF authenticators on the blockchain as follows. We assume all communications between the entities in this phase are secure and authenticated, and thus omit the descriptions of message signatures.

1) VNF Registration: Each provider registers its VNF dictionary with the manager. The manager verifies the identity and service capability of the providers, and constructs an overall dictionary, $D = (D_1, D_2, \ldots, D_{n_p})$. Given the size of the dictionary and the matching rules, a trusted entity or a distributed committee sets up CRS of the VNF query function $F(D, Q) \rightarrow R$ by running the S.Setup algorithm. After the setup, the CRS is published onto the blockchain. In the CRS, there are $n * m * n_p$ commitment keys for the overall VNF dictionary. The manager then publicly assigns provider P_i with a set of indexes for their VNFs and P_i can retrieve corresponding commitment keys in the CRS from the blockchain, denoted as $CK_i \in \mathbb{G}_1^{m*n}$.

2) Authenticator Construction: Upon retrieving the commitment keys CK_i from the blockchain, P_i computes an authenticator, $A_{V_{i,j}}$, for each of its VNF, denoted as $V_{i,j}$:

$$A_{V_{i,j}} = \prod g_{i,j,k}{}^{e_{j,k}}, e_{j,k} \in V_{i,j}, g_{i,j,k} \in CK_i. \tag{5}$$

Here, P_i computes an aggregated authenticator $A_i = \prod_{j=1}^{m} A_{V_{i,j}}$ and a Merkle root (R_i) for all $\{A_{V_{i,j}}||i||j\}_{j\in[m]}$ using M.Setup algorithm. It uploads A_i and R_i to the blockchain, and sends all $A_{V_{i,j}}$ to the manager.

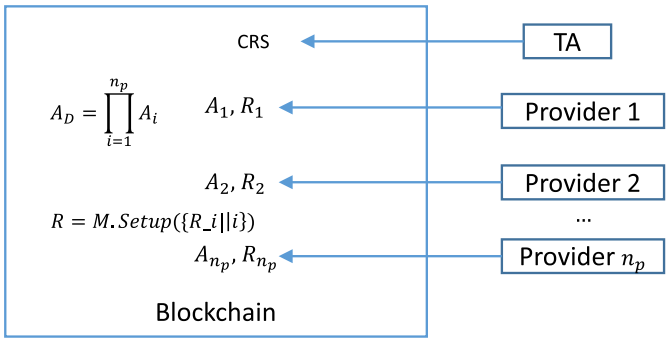


Fig. 3. Authenticator generation.

3) Authenticator Aggregation: Upon receiving A_i and R_i on the blockchain, an aggregated authenticator of all VNF dictionaries from all providers can be calculated on chain given by

$$A_D = \prod_{i=1}^{n_p} A_i. \tag{6}$$

A Merkle root (R) of all R_i can be computed using M.Setup, where $\{R_i||i\}_{i\in[n_p]}$ denotes all individual roots with corresponding indexes of their providers. The on-chain data structure is shown in Fig. 3. To this end, the verifications of a VNF query can directly take the dictionary authenticator (A_D) instead of the original dictionary, which can significantly save the on-chain storage space. Note that *BVNF+* requires all providers honestly compute their authenticators and Merkle roots. The authenticator generation phase can be re-conducted when VNF information is changed. For example, a VNF provider can change its VNF information or can be removed from the system.

C. SGX-Based Dictionary Pruning

The traditional SNARG-based verifiable VNF query suffers from the random access memory (RAM) issue [29]. The circuit-based representations cannot efficiently support dynamic loop control or array access at the program running time. As a result, for the SNARG-based verifiable VNF query, a linear scan of all VNFs is required, which can incur prohibitively expensive proving overheads at the VNF manager. To address the issue, a dictionary pruning strategy based on Merkle tree was proposed [26]. A key query item is used to reduce the number of potentially matched VNFs by a dictionary pruning function. Then, the results of the pruning function are taken into a second function that executes the full query. Both functions are implemented using different SNARGs.

The key design for the dictionary pruning is to generate a verifiable authenticator for the second SNARG system. By precomputing all VNF authenticators for the second SNARG with a Merkle digest, the required authenticator for the second SNARG can be efficiently generated [26] and verified by Merkle proofs. To further increase verification efficiency of the Merkle proofs on the blockchain, a succinct proof

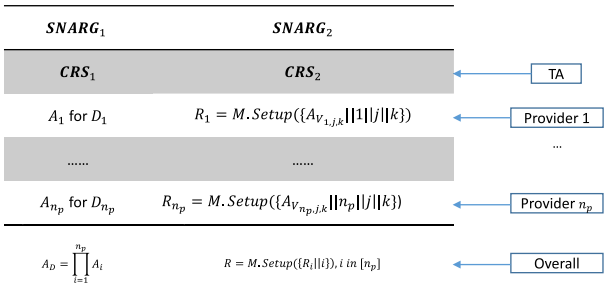


Fig. 4. On-chain data structure.

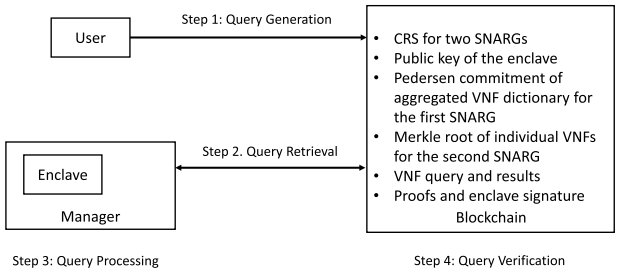


Fig. 5. Query workflow.

of the Merkle verifications can be generated using a well-designed SNARG, which however significantly increases the prover computation overhead and will be discussed in the performance evaluation.

On addressing the challenges, we present an SGX-based dictionary pruning mechanism. The mechanism uses the two-level SNARG networks for the key item query and the full VNF query. For efficient verifications of an aggregated authenticator for the second SNARG, we design an SGX enclave that faithfully outputs the aggregated authenticator for the second SNARG. In the following, we present the detailed designs of our SGX-based dictionary pruning, which consists of CRS and authenticator setup, enclave attestation and key provisioning, and VNF query on blockchain.

1) CRS and Authenticator Setup: The two-level VNF query function is given by

$$F_1(q^*, D) \rightarrow I_1,$$
$$F_2(I_1, Q) \rightarrow I_2. \tag{7}$$

For the two functions, all providers and the manager agree on two SNARGs S_1 and S_2 for a dictionary D similar to (4) and a query Q similar to (2). Here, S_1 takes a key query item (q^*) in Q and D to output indexes of VNFs in D that match the key query item. The results are denoted as I_1. The second SNARG (S_2) takes VNF description vectors of VNFs in I_1 and Q to generate the final query results (I_2) that consist of all indexes for VNFs in I_1 that match Q. Accordingly, CRS_1 and CRS_2 are faithfully generated for S_1 and S_2 using S.Setup algorithm.

Following the procedures in Subsection IV-B, providers can generate and store their authenticators of VNF dictionaries on the blockchain for S_1 using commitment keys from CRS_1. However, for S_2, providers do not know the exact order that their VNFs will appear in I_1. As a result, for each VNF, they need to pre-compute authenticators for every possible index in I_1 using CRS_2. The authenticators for the two-level SNARG system on the blockchain is shown in Fig. 4, where A_i is the dictionary authenticator of D_i using CRS_1, A_D is the aggregated authenticator, $A_{V_{i,j,k}}$ denotes the authenticator of j-th VNF of the i-th provider P_i, and k indicates that the VNF is the k-th VNF in the pruning results, I_1. Authenticators of VNF dictionaries are stored on the blockchain to preserve on-chain VNF privacy. On-chain query privacy could be enhanced by only uploading signed query commitments onto the blockchain.

2) Enclave Attestation and Key Provisioning: The manager needs to run an SGX enclave locally to produce a succinct proof of the dictionary pruning operation. The goal of the enclave is to verify the correctness of authenticators for VNFs in I_1. The enclave takes the results of S_1, associated authenticators and Merkle proofs, and decides to either accept or reject the dictionary pruning results.

To preserve the functional correctness and authenticity of the enclave, an attestation of the enclave and a signing key provisioning are necessary.[1] We briefly discuss a method as follows: First, the manager launches the remote attestation process with a trusted attestation entity. Second, the attestation entity verifies the attestation report, and establishes a secure channel with the enclave. Note that the codes and initial data of the enclave can be checked by code-review community to ensure its functionality. Third, the enclave can generate a pair of ECDSA signing keys (sk_e, pk_e), securely seal the private key on the local storage, and securely provision the public key to the attestation entity. Then, the attestation entity can upload pk_e onto the blockchain. We note that the detailed designs of attestation and key provisioning is out of scope of this paper. Moreover, frequent signing key updates of the enclave can be designed and the signing key can also be provisioned by the attestation entity only at run time without being sealed on local storage.

3) VNF Query on Blockchain: (Fig. 5) First, a user generates a VNF query Q and uploads the query Q to the blockchain via a secure and authenticated channel. The blockchain verifies the user identity and the well-formedness of the query, and stores the query on its storage. To further preserve query privacy, the user can send the query to the manager via a secure channel and store a commitment of the query on the chain in case of future disputes. Second, upon seeing the query on the blockchain, the manager retrieves Q to its local storage.

Third, the manager runs S_1 over the key query item $(q^* \in Q)$ and D to get I_1 and a proof (π_1). The manager then runs S_2 with VNF information vectors of VNFs in I_1 and Q to output I_2 and a proof (π_2). For each VNF in I_1, the manager generates a Merkle proof that its corresponding authenticator $V_{i,j,k}$ is digested in the Merkle root R. The manager sends I_1, all VNF authenticators $(V_{i,j,k})$ for VNFs in I_1 with Merkle proofs, the Merkle root (R) and Q to the enclave. From I_1, the enclave learns the indexes and order of each VNF in I_1. Then, the enclave verifies all Merkle proofs for corresponding

[1]Key Provisioning, Secure Signing, and Verifiable Remote Attestation using Intel® SGX. https://github.com/initc3/sgx-iot-gateway.

authenticators. If all verifications pass, the enclave computes an aggregated authenticator, given by

$$A_e = \prod A_{V_{i,j,k}},\ V_{i,j}\ is\ k_{th}\ in\ I_1. \tag{8}$$

The enclave sets $m_e = (Q, I_1, A_e, R)$ and generates signature $\pi_e = E.Sign(m_e, sk_e)$ using its locally sealed key sk_e. The enclave outputs (m_e, π_e) to the manager. The manager sends $(Q, I_1, I_2, \pi_1, \pi_2, A_e, \pi_e)$ to the blockchain via a secure and authenticated channel.

Finally, upon receiving the message from the manager, the blockchain verifies the following statements:

- Q and R in m_e match the stored MHT root and the query on the blockchain, to ensure that correct Merkle root and query are taken into the enclave;
- (I_1, q^*, A_D) is a valid instance for S_1 using S.Verify algorithm with CRS_1 and π_1, where A_D is the dictionary authenticator stored on the blockchain;
- (I_2, Q, A_e) is a valid instance for S_2 using S.Verify algorithm with CRS_2 and π_2;
- π_e is a valid signature using E.Verify algorithm on m_e and the stored public key pk_e, to ensure that the dictionary pruning message (m_e) comes from the attested enclave.

If all the statements are true, the blockchain confirms that the query results are correct; If any statement is not true, the query results are not correct and according actions can be taken to the misbehaving manager.

V. Security Analysis

In this section, we first discuss the security properties of SGX and SNARG. Then, we present the security analysis of the verifiable VNF query scheme in *BVNF+*.

A. SGX Security

BVNF+ requires the SGX to provide the following security features. First, SGX can provide the attestation service to verify the code and initial data of an enclave and to establish a secure channel for secret provisioning [40]. The attestation service can be provided by an trusted entity or a distributed attestation service [41]. Second, SGX can ensure that the data confidentiality and computation integrity of the enclave within its secure memory [42]. Third, SGX should have a data sealing mechanism to securely store data on local storage and only the same enclave can unseal the stored data later. Note that sophisticated attacks on SGX, such as rollback attacks and side-channel attacks, and their countermeasures are out of scope of this paper.

B. SNARG Security

SNARG security ensures that, given the inputs and outputs of a function, CRS and a proof, the verification algorithm outputs accept if and only if the inputs and outputs are valid for the function [22], [36]. The *correct execution* of the computing comes from the following two aspects: First, the CRS is securely set up by a trusted entity or a distributed committee, which ensures that the CRS is used for the desired function and the trapdoor secret used in generating the CRS is securely destroyed. Second, with the secure CRS, a computationally-bounded adversary cannot break the *soundness* of the SNARG under knowledge-based assumptions to forge an invalid statement that passes the verification [22], [36].

For the *input authenticity* of the commit-and-prove SNARG [23], part of the inputs (i.e., VNF dictionary in *BVNF+*) is digested as an authenticator to be used in the verifications. This additionally requires that the commitment keys for the authenticator should be linearly independent generators, and that the authenticator is honestly computed with the inputs (i.e., by VNF providers in *BVNF+*). At the same time, more randomness can be added in generating the authenticator, which should not affect the function execution.

C. Verifiable VNF Query

From the perspective of vc, the VNF query should achieve input authentication of the VNF dictionary, and correct execution of the query algorithm. *BVNF+* adopts a SNARG system for the key item query and an SGX enclave for verifying Merkle proofs and generating an aggregated authenticator.

For the first SNARG, dictionary authenticators are generated faithfully by VNF providers, and are uploaded and aggregated on the blockchain's immutable storage. Then, any VNF query generated by users is uploaded onto the blockchain with an authenticated channel. This ensures the authenticity of VNF dictionaries and queries. Later, only valid query results can pass the verification algorithm of the first SNARG where trusted CRS is also available on the blockchain's immutable storage.

For the pruning enclave, VNF authenticators for the second SNARG (S_2) are generated faithfully by VNF providers. At the same time, an MHT is constructed with the leaf node as the hash of each authenticator, the VNF index and the order in I_1. Both VNF authenticators and the MHT roots are stored and aggregated on the secure blockchain storage. For a VNF query Q with authenticators for the second SNARG, the enclave identifies the indexes of each VNF and its positions in I_1. With the Merkle proofs of VNFs in I_1 and Merkle root R, the enclave can verify if each VNF authenticator is in the right position of the MHT. A computationally-bounded adversary cannot forge valid Merkle proofs unless it can break the collision-resistance property of the hash function. Moreover, the enclave is attested and a public key is securely sent to the attestation entity. Since the corresponding private key is sealed at the local storage, only the same enclave can unseal the private signing key to sign on $m_e = (Q, I_1, A_e, R)$ and obtain signature π_e.

In the verification phase, the blockchain checks the results of Q and I_1 with CRS_1. Due to the trustworthiness or CRS_1 and the *soundness* of the SNARG, I_1 is secure and cannot be forged. Then, the blockchain checks π_e with the public key provided by the attestation entity and R, Q in m_e are consistent with its storage. This ensures that the enclave is attested and uses a valid MHT root for verifying the exact proofs

for I_1. Assuming the security of the SGX discussed above, authenticator A_e from m_e is correctly computed by a trusted enclave that runs the pruning function. However, additional implementation considerations may be required, such as the signing key update with an additional key derivation function and key provisioning at enclave runtime. The blockchain can also provide trusted state information to increase the SGX security [43]. With the authenticity of Q, A_e and the trusted CRS_2 on the blockchain, the final query results I_2 can be verified with the verification algorithm of the second SNARG, S_2. The security of S_2 also comes from the *soundness* of the SNARG.

In summary, *BVNF+* achieves *verifiable VNF query* defined in Subsection II-C, under the condition that the security properties of SGX and SNARG hold.

VI. Performance Evaluation

In this section, we present the performance evaluation of *BVNF+*. The experimental results provide on/off-chain execution benchmarks. Moreover, we compare the computation and communication overheads between the designed SGX-based and the non-SGX-based dictionary pruning mechanisms to demonstrate the efficiency of *BVNF+*.

A. Off-Chain Performance

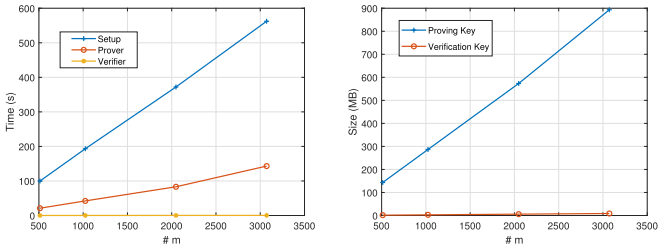
Our testing environment is a laptop with 2.3GHz processor and 8GB memory. The system is 64-bit Ubuntu 16.04. For the SGX, we implement SGX release 2.8.[2] For hash function, we implement the sgx_sha256_msg function from the default SGX security library: $sgx\_tcrypto$. For the SNARG, we implement the circuit builder[3] from xjsnark that translates JAVA codes into circuits. Moreover, we implement the circuit-to-snark interface from libsnark library[4] with QAP-based instantiations [28], [36] using bn128 curve.

We test the verifiable VNF query presented in Subsection IV-A. In our experiments, we set the query item of Q as 30, which includes 10 keyword queries, 10 range queries, 5 membership queries, and 5 prefix queries. Each membership item includes 5 words and each prefix query item also includes 5 words. We artificially set the item values in the dictionary and query as integers. We run the query against each VNF in the dictionary and report the computation and storage overhead. As shown in Fig. 6a, the computational overheads for setup and prover linearly increase with the size of the dictionary while the verifications remain very efficient, around a few milliseconds. As shown in Fig. 6b, the size of the proving key is larger compared with the verification key. At the same time, the size of the verification key increases since we output matching results for each VNF in the dictionary in our experiments. The high prover overhead is caused by many unnecessary accesses of the original dictionary. To address the issue, a two-level SNARG system and the pruning strategy can be adopted to reduce the unnecessary memory access [26].

[2]https://01.org/intel-softwareguard-extensions/downloads/intel-sgx-linux-2.8-release

[3]https://github.com/akosba/xjsnark

[4]https://github.com/akosba/libsnark



(a) Computation Overhead (b) CRS Size

Fig. 6. Complexity of versatile VNF query.

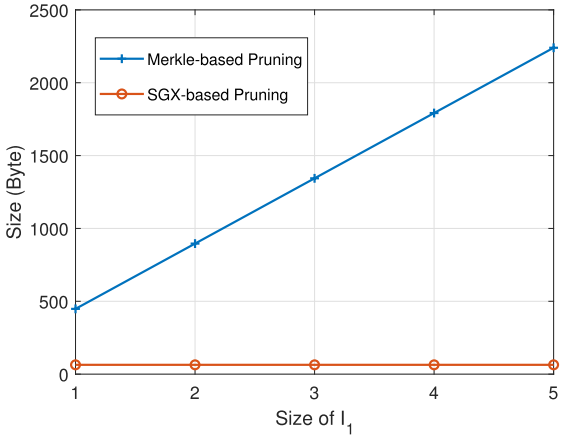


Fig. 7. Communication overhead.

TABLE II
SNARG-Based Hash Verification

Setup	Prover	Verifier
13.4s	3.12s	0.24s

B. Performance Gain With SGX-Based Pruning

For the communication overheads, the SGX-based pruning verifies the Merkle proofs with the enclave, and signs the verification result with an ECDSA signature. Therefore, the SGX-based approach reduces the verification of multiple Merkle proofs into a single verification of the ECDSA signature [26]. We set the height of MHT as 14 and the size of a SHA256 hash is 32 byte. As shown in Fig. 7, the theoretical communication cost of Merkle proofs increases with the number of VNFs in I_1, while the SGX-based approach achieves a constant 64-byte signature.

For the computation overhead, we report the performance gain when generating pruning proof with the SGX-based method. To achieve the similar efficient verifications of the pruning results, our baseline is to use the SNARG for verifying Merkle proofs. As shown in Table II, proving the verification of a single SHA256 furcation with small input size (64 byte) requires more than 3 seconds, while verifying pruning results can consist of hundreds of verifications of SHA256 hash functions. For the SGX-based pruning, we simulate the verifications of Merkle proofs by repeatedly running hash functions within the enclave. The input of each hash function is a 256-bit integer. The height of Merkle tree is set at 14 or 15. As shown in Fig. 8, when the number of Merkle proofs increases, the

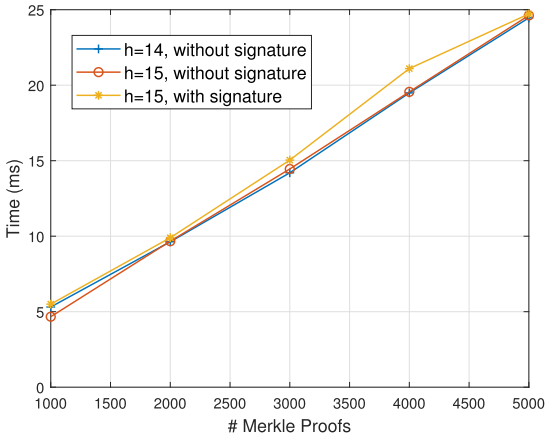


Fig. 8. Computation overhead of SGX-based pruning.

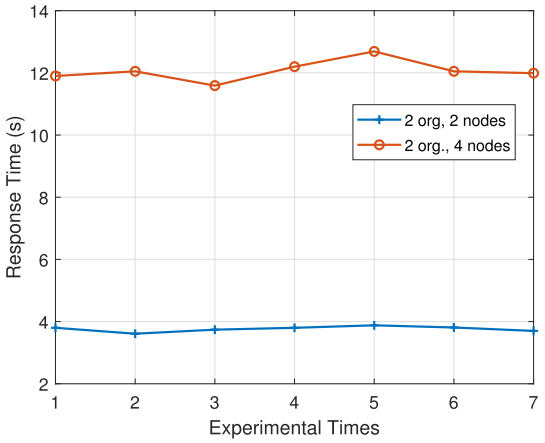


Fig. 9. On-chain response time.

TABLE III
ON-CHAIN STORAGE OVERHEAD

Authenticators	Proof
$n_p(\lvert\mathbb{G}_1\rvert + \lvert R_i\rvert)$	$\lvert\pi_1\rvert + \lvert\pi_2\rvert + \lvert\pi_e\rvert$

computational overhead for verifying the proofs increases linearly, but is limited to a few milliseconds. We separately test the computation cost of generating an ECDSA signature within the enclave, which takes roughly 0.523 milliseconds. Therefore, compared with the SNARG-based pruning verification, the designed SGX-based pruning achieves the same succinct proof size but a notable improvement in computation overheads.

C. On-Chain Performance

1) Storage Overheads: The *BVNF+* adopts Pedersen commitment and Merkle tree to store digests of VNF dictionary and VNF authenticators. As shown in Table III, it requires only $|\mathbb{G}_1| + |R_i|$ on-chain storage overheads for each VNF provider. Compared with directly storing VNF dictionary on the blockchain, *BVNF+* has constant storage overhead for individual provider. For the CRS, only verification keys are required to be stored on the blockchain, which incurs a few bytes as shown in Fig. 6. For the proof of a single VNF query, it consists of two SNARG proofs and one SGX proof (an ECDSA signature), which is also constant regardless of the size of the VNF dictionary. Each SNARG proof is 286 bytes and the ECDSA signature is 64 bytes.

2) Computation Overheads: We implement the test network[5] of Hyperledger Fabric (Release 2.1) [18] with Java native security package for ECDSA signature. The test network consists of two organizations and one ordering service. We simulate the verifications of an SGX report by verification of an ECDSA signature with a short message. More specifically, we implement the verification of a message with an ECDSA signature as a chaincode function. Each organization consists of either 1 or 2 peer nodes that install the same chaincode package. The public key of the signature is written in the chaincode, while the message and the signature are transmitted through function calls. We present the response time of such a function call, using the 'peer chaincode invoke' function in Hyperledger Fabric. We test the response time in two settings: two organizations with two peer nodes and two organizations with 4 peer nodes. As shown in Fig. 9, the response time is significantly affected by the number of nodes in the network compared with the computation cost of verifying an ECDSA signature. It should be noted that our design is independent of the underlying blockchain architecture, and thus can be adapted to any blockchain architecture with different consensus protocols.

In summary, *BVNF+* achieves efficient on-chain storage and computation overheads regardless of the size of VNF dictionary. At the same time, our SGX-based dictionary pruning mechanism significantly reduces the off-chain proving overheads and proof size, in comparison with the state-of-the-art works.

VII. RELATED WORKS

In this section, we first discuss the recent advances on emerging networking technologies for metaverse. Then, we present related works on blockchain-based VNF management to highlight their design differences compared with *BVNF+*.

A. Emerging Networking Technologies for Metaverse

A comprehensive survey of metaverse fundamentals was presented in [1]. Multiple enabling technologies for metaverse were discussed including ubiquitous computing, blockchain, AI, and networking. Emerging communication and networking technologies for metaverse were studied in [2] and [4]. More specifically, communication technologies, such as space-air-ground integrated access network and NB-IoT, can provide ubiquitous, seamless and reliable access services in metaverse. Future networking architectures will further integrate intelligent computing unit, such as cloud or edge servers, for AI-based network control. For example, a prototype of cloud-edge networking architecture was developed and verified for multi-player interactions in metaverse [44]. NFV-enabled networking architecture can provide efficient

[5]https://github.com/hyperledger/fabric-samples

and flexible management of network resources [32], and provision human-centric service configurations for differentiated metaverse applications. The NFV architecture is also promising to enhance the QoE management for human-centric multi-media services [6], including VR/AR, and to provide tree-like service function chains for multi-player transmissions in metaverse [5].

Exiting works have shed light on the design of human-centric networking architecture, especially NFV-enabled architecture, for metaverse applications. On the other hand, the existing proposals mainly rely on a centralized controller for NFV management. In practice, it is not an easy task for all providers from different trust domains to agree on a single trusted entity to manage their service information. Therefore, a distributed VNF management architecture with multiple providers is urgently desired.

B. Blockchain-Based VNF Management

In a distributed environment with multiple network resource providers, e.g., metaverse, a decentralized architecture is preferred for transparent and reliable VNF-based network management [45]. In this regard, the blockchain was proposed to act as a secure VNF broker for 5G services [9] to help construct end-to-end network slices for various users. A blockchain-based approach can avoid the risk of single-point failure to build trust among VNF providers for transparent and fair VNF management. Smart contract technology was adopted for automatic slice configuration and marketing in [19], where a verification mechanism for checking VNF image integrity was designed. The optimal network resource pricing and demand management on the blockchain were studied in [8]. The blockchain serves as a trusted platform for publishing network management information. Furthermore, a two-stage pricing game was formulated and a deep reinforcement learning-based mechanism was proposed for dynamic pricing and demand responses. On the accountability of SLA management, a blockchain-based architecture, named BEAT was proposed in [10], where the TEE was adopted for securely recording management status on network operators. The TEEs were also utilized for privacy-preserving resource management on the blockchain with a newly proposed consensus protocol [11].

Due to the expensive on-chain resources, the on-chain computation and storage efficiency of blockchain-based approaches can be further improved. Therefore, *BVNF+* focuses on the on-chain efficiency challenge of VNF management, and has a hybrid vc framework for versatile and efficient VNF query.

Verifiable data queries on the blockchain were designed in [27], [46], and [47]. In [46], an MHT-based authenticated data structure was proposed for cloud query services. In [27], verifiable range queries were constructed for the blockchain storage based on cryptographic accumulators with inter-block and intra-block authenticated data structures. Pedersen commitments were constructed as authenticated digests of data, which can be used for verifiable general computations [23]. Succinct non-interactive argument (SNARG) [22], [28], [29] is the vc framework compatible with the Pedersen commitments to support circuit-based computations. One essential property of SNARG is the succinct result verification, which makes it suitable for on-chain verifications. However, the on-chain efficiency of SNARG comes at the cost of significantly increased off-chain proving overheads. In [26], a blockchain-based prunable and authenticated dictionary was proposed for verifiable VNF management. The proposal reduces the prover overheads with efficient on-chain computations. Recently, secure hardware is driving extensive attentions for building vc frameworks. Intel Software Guard Extensions was used to build secure database systems to protect the sensitive data indexes and logs in secure memories [24], [25], [30]. By doing so, data queries can be executed in a verifiable and secure manner. Authenticated key-value stores based on SGX were proposed in [48] for data query authentication. The SGX provides the secure environment for smart contract execution that can save computation overheads for re-doing contract verifications by multiple blockchain miners. An SGX-based computing framework for the blockchain was designed in [43] considering many attacks and defenses, such as side-channel, timer fails, etc. The SGX-based solution has limited size of protected memory, which can increase the management cost for conducting verifiable VNF queries with a large VNF dictionary.

In *BVNF+*, we design a versatile VNF query scheme for collaborative VNF management in metaverse. By exploiting the strength of the cryptographic-based and SGX-based vc mechanisms, we design a two-level VNF query framework with modular operation decompositions and efficient interplays between SNARG and SGX. Our designs support the multi-provider dictionary generation and aggregation, achieve efficient on-chain verifications, and significantly reduce the communication overhead of the Merkle-tree based pruning approach with limited use of the SGX memory.

VIII. Conclusion

In this paper, we have proposed a blockchain-based collaborative and verifiable VNF management scheme for metaverse, which can support human-centric VNF queries and slice configurations for diversified and differentiated metaverse applications. We have addressed the on-chain efficiency challenges by designing an on/off-chain computation model for blockchain-based VNF management. Moreover, our modular designs with SNARG- and SGX-based VNF queries have significantly reduced the off-chain computation cost of the verifiable dictionary pruning operations. The designs, implementations, and evaluations of *BVNF+* shed light on the reliable and efficient management for multi-provider slice configurations, promoting trustworthy and reliable collaborations among distributed metaverse stakeholders. In the future, we will further investigate the fairness issue of VNF management in metaverse. Efficient discovery and regulation enforcement against users or providers supplementing fake information should be investigated in the distributed metaverse environment.

REFERENCES

[1] Y. Wang et al., "A survey on Metaverse: Fundamentals, security, and privacy," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 319–352, 1st Quart., 2023.

[2] F. Tang, X. Chen, M. Zhao, and N. Kato, "The roadmap of communication and networking in 6G for the Metaverse," *IEEE Wireless Commun. Mag.*, vol. 30, no. 4, pp. 72–81, Aug. 2023.

[3] L.-H. Lee et al., "All one needs to know about Metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," 2021, *arXiv:2110.05352*.

[4] Y. Fu, C. Li, F. R. Yu, T. H. Luan, P. Zhao, and S. Liu, "A survey of blockchain and intelligent networking for the Metaverse," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3587–3610, Feb. 2023.

[5] T. Li and Z. Zhu, "QoS-aware management reconfiguration of vNF service trees with heterogeneous NFV platforms," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 2, pp. 2013–2024, 2023.

[6] A. A. Barakabitze and R. Walshe, "SDN and NFV for QoE-driven multimedia services delivery: The road towards 6G and beyond networks," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109133.

[7] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[8] G. He, W. Su, S. Gao, N. Liu, and S. K. Das, "NetChain: A blockchain-enabled privacy-preserving multi-domain network slice orchestration architecture," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 1, pp. 188–202, Mar. 2022.

[9] B. Nour, A. Ksentini, N. Herbaut, P. A. Frangoudis, and H. Moungla, "A blockchain-based network slice broker for 5G services," *IEEE Netw. Lett.*, vol. 1, no. 3, pp. 99–102, Sep. 2019.

[10] T. Faisal, M. Dohler, S. Mangiante, and D. R. Lopez, "BEAT: Blockchain-enabled accountable and transparent network sharing in 6G," *IEEE Commun. Mag.*, vol. 60, no. 4, pp. 52–56, Apr. 2022.

[11] G. O. Boateng, D. Ayepah-Mensah, D. M. Doe, A. Mohammed, G. Sun, and G. Liu, "Blockchain-enabled resource trading and deep reinforcement learning-based autonomous RAN slicing in 5G," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 1, pp. 216–227, Mar. 2022.

[12] C. Xu et al., "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 870–882, Apr. 2019.

[13] H. Ren, H. Li, D. Liu, G. Xu, N. Cheng, and X. Shen, "Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 1052–1064, Apr. 2022.

[14] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[15] P. Li, S. Guo, and I. Stojmenovic, "A truthful double auction for device-to-device communications in cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 71–81, Jan. 2016.

[16] D. Liu, C. Huang, J. Ni, X. Lin, and X. S. Shen, "Blockchain-cloud transparent data marketing: Consortium management and fairness," *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3322–3335, Dec. 2022.

[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, 2008.

[18] E. Androulaki et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–15.

[19] E. J. Scheid, M. Keller, M. F. Franco, and B. Stiller, "BUNKER: A blockchain-based trusted VNF package repository," in *Proc. Int. Conf. Econ. Grids, Clouds, Syst., Services*. Cham, Switzerland: Springer, 2019, pp. 188–196.

[20] Z. Hong, S. Guo, and P. Li, "Scaling blockchain via layered sharding," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3575–3588, Dec. 2022.

[21] J. Eberhardt and S. Tai, "ZoKrates-scalable privacy-preserving off-chain computations," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1084–1091.

[22] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, Jun. 2013, pp. 238–252.

[23] M. Campanelli, D. Fiore, and A. Querol, "LegoSNARK: Modular design and composition of succinct zero-knowledge proofs," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2075–2092.

[24] V. Costan and S. Devadas, "Intel SGX explained," Cryptol. ePrint Arch., Paper 2016/086, 2016.

[25] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A secure database using SGX," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 264–278.

[26] D. Liu et al., "Authenticated and prunable dictionary for blockchain-based VNF management," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9312–9324, Nov. 2022.

[27] C. Xu, C. Zhang, and J. Xu, "VChain: Enabling verifiable Boolean range queries over blockchain databases," in *Proc. Int. Conf. Manag. Data (SIGMOD)*, Jun. 2019, pp. 141–158.

[28] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von Neumann architecture," in *Proc. USENIX Secur.*, 2014, pp. 781–796.

[29] A. Kosba, C. Papamanthou, and E. Shi, "xJsnark: A framework for efficient verifiable computation," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 944–961.

[30] F. McKeen et al., "Innovative instructions and software model for isolated execution," in *Proc. Hasp@ ISCA*, 2013, vol. 10, no. 1, pp. 1–8.

[31] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security vulnerabilities of SGX and countermeasures: A survey," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–36, Jul. 2021.

[32] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[33] P. S. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. Int. Workshop Sel. Areas Cryptogr.* Cham, Switzerland: Springer, 2005, pp. 319–331.

[34] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001.

[35] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. CRYPTO*. Cham, Switzerland: Springer, 1987, pp. 369–378.

[36] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. EUROCRYPT*, 2016, pp. 305–326.

[37] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE S&P*, 2018.

[38] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 1991, pp. 129–140.

[39] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel software guard extensions: EPID provisioning and attestation services," Intel Corp., Santa Clara, CA, USA, White Paper 1, 2016, pp. 110–119.

[40] Y. Nakatsuka, E. Ozturk, A. Paverd, and G. Tsudik, "Cacti: Captcha avoidance via client-side tee integration," in *Proc. USENIX Secur.*, 2021, pp. 2561–2578.

[41] G. Chen, Y. Zhang, and T.-H. Lai, "OPERA: Open remote attestation for Intel's secure enclaves," in *Proc. of ACM CCS*, 2019, pp. 2317–2331.

[42] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, and C. Wang, "Aggregation service for federated learning: An efficient, secure, and more resilient realization," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 988–1001, Apr. 2022.

[43] R. Cheng et al., "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, 2019, pp. 185–200.

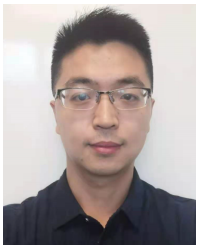
[44] Y. Huang, X. Qiao, H. Wang, X. Su, S. Dustdar, and P. Zhang, "Multiplayer immersive communications and interactions in Metaverse: Challenges, architecture, and future directions," 2022, *arXiv:2210.06802*.

[45] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[46] H. Wu, Z. Peng, S. Guo, Y. Yang, and B. Xiao, "VQL: Efficient and verifiable cloud query services for blockchain systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1393–1406, Jun. 2022.

[47] C. Xu, C. Zhang, J. Xu, and J. Pei, "SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing," *Proc. VLDB Endowment*, vol. 14, no. 11, pp. 2314–2326, Jul. 2021.

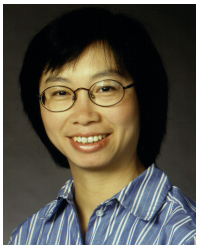
[48] K. Li, Y. Tang, Q. Zhang, J. Xu, and J. Chen, "Authenticated key-value stores with hardware enclaves," in *Proc. 22nd Int. Middleware Conf., Ind. Track*, Dec. 2021, pp. 1–8.



Dongxiao Liu (Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2020. From 2020 to 2023, he was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo. He is currently with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include security and privacy in intelligent transportation systems, blockchain, and mobile networks.

Cheng Huang (Member, IEEE) received the B.Eng. and M.Eng. degrees in information security from Xidian University, China, in 2013 and 2016, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2020. He is currently a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests include applied cryptography, cyber security, and privacy in mobile networks.

Liang Xue (Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2022. She is currently a Post-Doctoral Research Fellow with the School of Computer Science, University of Guelph. Her research interests include applied cryptography, secure cloud computing, and security and privacy in blockchain-based applications and machine learning.



Weihua Zhuang (Fellow, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from Dalian Marine University, China, and the Ph.D. degree in electrical engineering from the University of New Brunswick, Canada. She is currently a University Professor and the Tier I Canada Research Chair of wireless communication networks with the University of Waterloo, Canada. Her research interests include network architecture, algorithms and protocols, and service provisioning in future communication systems. She is an elected member of the Board of Governors and the President of the IEEE Vehicular Technology Society. She is a fellow of the Royal Society of Canada, the Canadian Academy of Engineering, and the Engineering Institute of Canada. She was a recipient of the 2021 Women's Distinguished Career Award from the IEEE Vehicular Technology Society, the 2021 Technical Contribution Award in Cognitive Networks from the IEEE Communications Society, the 2021 R. A. Fessenden Award from IEEE Canada, and the 2021 Award of Merit from the Federation of Chinese Canadian Professionals in Ontario. She was the Editor-in-Chief of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY from 2007 to 2013, the General Co-Chair of the 2021 IEEE/CIC International Conference on Communications in China (ICCC), the Technical Program Chair/Co-Chair of 2017/2016 IEEE VTC Fall, the Technical Program Symposia Chair of the 2011 IEEE GLOBECOM, and a IEEE Communications Society Distinguished Lecturer from 2008 to 2011.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include network resource management, wireless network security, the Internet of Things, 5G and beyond, and vehicular networks. He is an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Chinese Academy of Engineering Foreign Member. He received the "West Lake Friendship Award" from Zhejiang Province in 2023; the President's Excellence in Research from the University of Waterloo in 2022; the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021; the R. A. Fessenden Award in 2019 from IEEE Canada; the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019; the James Evans Avant Garde Award from the IEEE Vehicular Technology Society in 2018; the Joseph LoCicero Award and Education Award from the IEEE Communications Society (ComSoc) in 2015 and 2017, respectively; and the Technical Recognition Award from the Wireless Communications Technical Committee in 2019 and the AHSN Technical Committee in 2013. He also received the Excellent Graduate Supervision Award from the University of Waterloo in 2006 and the Premier's Research Excellence Award (PREA) from the Province of Ontario, Canada, in 2003. He serves/served as the General Chair for the 6G Global Conference 2023 and ACM MobiHoc 2015; the Technical Program Committee Chair/Co-Chair for IEEE GLOBECOM 2024, IEEE GLOBECOM 2016, IEEE GLOBECOM 2007, IEEE Infocom 2014, and IEEE VTC 2010 Fall; and the Chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the President of the IEEE ComSoc. He was the Vice President of technical and educational activities, the Vice President of publications, the Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a member of the IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief for IEEE INTERNET OF THINGS JOURNAL, *IEEE Network*, and *IET Communications*. He is a registered Professional Engineer of Ontario, Canada. He is a Distinguished Lecturer of the IEEE Vehicular Technology Society and the Communications Society.

Bidi Ying (Member, IEEE) received the Ph.D. degree. She is currently a Senior Network Architecture Engineer with Huawei Technologies Canada Company Ltd. Before that, she was with the University of Ottawa. During the past 15 years, she has published more than 200 papers in top conferences and reputable journals. Her research interests include security and privacy in wireless networks.