Authenticated and Prunable Dictionary for Blockchain-based VNF Management

Dongxiao Liu, *Member, IEEE*, Cheng Huang, *Member, IEEE*, Liang Xue, *Student Member, IEEE*, Jiahui Hou, *Member, IEEE*, Xuemin (Sherman) Shen, *Fellow, IEEE*, Weihua Zhuang, *Fellow, IEEE*, Rob Sun, *Member, IEEE*, and Bidi Ying, *Member, IEEE*

***Abstract*—Network function virtualization is a key enabling technology in future wireless networks for flexible and efficient sharing of network resources. Due to the increasing heterogeneity of network resource providers, a blockchain-based distributed architecture is a promising solution to enable reliable and transparent virtualized network function (VNF) management. However, since on-chain storage and computation are costive, it becomes a challenging task to achieve efficient VNF management with blockchain. In this paper, we first introduce a consortium blockchain for collaborative VNF management among network resource providers. Then, we propose an authenticated VNF dictionary that can be stored as a succinct authenticator on blockchain to support rich VNF query functionalities and efficient verifications of query results. Moreover, we design a dictionary pruning strategy to securely generate a compact authenticator for a given query, which reduces unnecessary memory accesses of the original dictionary when VNF queries are represented as arithmetic circuits. Finally, we conduct extensive experiments with a consortium blockchain network. The experimental results demonstrate that our pruning strategy is efficient for both on-chain and off-chain VNF management.**

***Index Terms*—Network function virtualization (NFV), blockchain, authenticated dictionary, dictionary pruning.**

I. Introduction

Future wireless network is envisioned to have a highly dynamic and heterogeneous architecture that integrates a wide range of radio access technologies and physical network resources [1]. To achieve flexible, efficient, and cost-effective sharing of physical network resources, network function virtualization (NFV) is a key enabling technology [2]. More specifically, network resource providers (including wireless operators, edge, and cloud servers) abstract network functions into virtualized functions (VNFs) [3], such as firewall or packet inspection function at the cloud [4]. To support a particular application, a network slice can be formed that consists of a chain of VNFs, a network topology on VNF nodes, switches, and links, and networking protocol supports [5]–[7]. Service-level agreement (SLA) corresponds to Quality of Service (QoS) requirements that a user expects from a service provider. In NFV-enabled network management, an SLC for a particular service should specify a network slice with sufficient computing, storage, and transmission resources. To this end, network virtualization can achieve flexible programming of service functionalities and efficient resource management in the future wireless networks.

With the development of the NFV techniques, it is critical to have a reliable and secure NFV controller to orchestrate VNFs and manage network slices for different network applications [8]. Due to the increasing heterogeneity in wireless networks, NFV is envisioned to have a multi-provider and multi-tenant paradigm [9]. As a result, compared with a centralized controller, distributed controllers are a more practical and promising solution to provide reliable and transparent VNF management [10], [11]. Recently, extensive research efforts have been directed towards building the distributed VNF management using the emerging blockchain technology [11]–[16]. Blockchain is a distributed ledger maintained by a peer-to-peer network [17], [18]. With consensus protocols and lightweight cryptography, blockchain provides a consistent and shared view of the ledger among blockchain nodes. Moreover, smart contract technique [19] provides programing capability to securely and automatically update ledger states when conditions or terms are met. On the one hand, a blockchain-based VNF management can boost trustworthiness among network stakeholders to collaboratively manage VNFs and reduce the risk of single point failure of a centralized VNF controller [20]. On the other hand, it can enhance service fairness by implementing and monitoring service agreements on smart contracts [21].

At the heart of blockchain-based VNF management is VNF dictionary management. A VNF dictionary consists of useful information of VNFs, including VNF name, location, capabilities, version, and available resources [14]. Network resource providers can collaboratively manage a VNF dictionary on the blockchain and use the smart contract to support on-demand VNF placement, resource allocation, and slice configurations. As a result, the VNF dictionary should support rich lookup functionalities and efficient dictionary updates. Unfortunately, existing blockchain-based solutions for VNF management often directly store the VNF dictionary on the blockchain storage. First, a blockchain node must maintain local copies of the ledger and verify each ledger update. As the size of the VNF dictionary can increase dramatically, the cost of directly storing and querying the dictionary on-chain can be expensive [22]. Second, the on-chain storage is open to the blockchain nodes, while some VNF information can be sensitive and should be kept private, such as VNF location and subscription.

This work was supported by research grants from Huawei Technologies Canada and from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

D. Liu, C. Huang (corresponding author), L. Xue, J. Hou, X. Shen, and W. Zhuang are with the Department of Electrical and Computer Engineering, University of Waterloo, Canada.

R. Sun and B. Ying are with the Huawei Technologies Canada, Ottawa, Ontario, Canada.

To address the challenges, authenticated dictionary [23] on blockchain can be applied for the VNF management. Specifically, the VNF dictionary can be digested as a succinct cryptographic authenticator to be stored on the blockchain. Later, VNF lookups over the dictionary can be conducted off-chain and results can be verified efficiently on-chain with the authenticator and a succinct proof. Succinct non-interactive argument of knowledge (snark) [24]–[27] can support verifiable dictionary lookups represented by arithmetic circuits with a succinct dictionary authenticator. Moreover, verifications of snark proofs are efficient, which makes it a suitable candidate to construct blockchain-based VNF dictionary. At the same time, the snark-based authenticated dictionary may increase the computation overhead for generating a proof of correct VNF lookups. The is because the snark-based solution cannot efficiently support flexible programming of lookup functions.

In this paper, we present an authenticated and prunable dictionary for blockchain-based VNF management (*Block-VNF*). We utilize a consortium blockchain as a shared ledger between network stakeholders to conduct VNF management. We build a snark-based authenticated VNF dictionary to address the on-chain efficiency challenges. Most importantly, we identify and formulate the dictionary pruning problem in the snark-based dictionary management, and propose a highly efficient dictionary pruning solution based on Merkle tree. The main contributions of this paper are summarized as follows:

- We propose an authenticated and transparent VNF dictionary based on vector commitments and snarks on a consortium blockchain. By enabling succinct digests of the dictionary in blockchain with verifiable query proofs, our solution addresses the efficiency challenges for blockchain-based VNF management;
- We adopt a network of snark systems for verifiable VNF query. First, a pruning function can be executed with a sanrk system, and an aggregated authenticator of matched VNF functions can be generated. Second, the aggregated authenticator for a pruned dictionary can be used by another snark system for fine-grained VNF query. By doing so, we avoid a large number of unnecessary memory accesses to the original VNF dictionary;
- We design a verifiable mechanism for generating an aggregated authenticator based on the Merkle tree. Specifically, we let VNF providers pre-compute individual authenticator for each VNF function and store a succinct Merkle root for all VNF authenticators on blockchain. The aggregated authenticator of matched VNFs against the pruning operation can be verified with a Merkle proof and a snark proof. Moreover, on-chain verification overhead is further reduced by enabling efficient verifications of incorrect proofs in VNF management;
- Our security analysis demonstrates that *Block-VNF* achieves veritable VNF lookup. We conduct extensive experiments based on a real-world consortium blockchain network and snark implementations. Our experimental results demonstrate that our pruning solution obtains a significant performance gain in computational overhead when generating verifiable VNF lookup results.

The remainder of this paper is organized as follows. In Section II, we summarize the related works. In Section III, we present the building blocks of *Block-VNF*, including vector commitments and succinct non-interactive argument of knowledge (snark). We present system model and threat model, and formulate verifiable VNF lookup in Section IV. We discuss our design techniques and present detailed constructions in Section V. In Section VI, we demonstrate security properties of *Blod-VNF*. In Section VII, we present experimental results. Finally, we conclude this work in Section VIII.

II. Related Works

In this section, we summarize existing studies in blockchain-based VNF management, and blockchain-based authenticated dictionary.

A. Blockchain-based VNF Management

There were extensive studies on efficient resource management for VNFs, for vehicular networks [5], [8] and Internet of Things [7]. More specifically, abstractions of network functionalities as VNFs can help a network controller to improve the overall resource utilization efficiency. In a multi-provider multi-tenant setting with a preference of distributed VNF controlers, using blockchain as a broker was first studied in [28] to enable dynamic and automatic VNF management. Later, a framework for blockchain-based VNF auction was studied in [13]–[15]. VNF management was modeled as an auction process and blockchain was utilized to build a regulatable auction framework. In their works, VNF information of network resource providers was recorded on the blockchain, and a smart contract was utilized to manage VNF requests. A blockchain-based auditing architecture for VNF management was proposed in [16]. The blockchain was utilized as a trusted log system to record activities of VNF stakeholders.

Early attempts explored the VNF management solutions on blockchain for auction and regulatory purposes. Since the existing solutions usually adopted blockchain as a trusted and distributed database, the expensive on-chain storage and computation overhead was not fully considered.

B. Blockchain-based Authenticated Dictionary

Blockchain-based verifiable dictionary enabled transparent certificate [23], where an accumulator-based authenticator was stored on the blockchain to achieve the membership proof of certificate issuance. Existing accumulator-based solutions focused on an efficiently searchable blockchain-based dictionary [29], [30]. Specifically, an authenticated data structure based on cryptographic accumulator was constructed [30] and the searchable index based on symmetric searchable encryption (SSE) or hidden vector encryption (HVE) was adopted [29], to support keyword query and range query.

In an account-based blockchain architecture, a blockchain-based dictionary plays an important role. Compared with a blockchain architecture based on the unspent transaction output (UTOX) model, the account-based blockchain uses an account dictionary, such as a Merkle tree, to record public keys

and balances of blockchain accounts. Due to the inefficient proof size of the Merkle tree, it was proposed to utilize vector commitments [31] to construct the account-based blockchain. Specifically, the Pedersen-like commitments were used to store account information, which can be succinctly opened at a subset of vector positions [32] based on efficient polynomial evaluations. The vector commitment-based mechanism usually required verifications in plaintext. The commitment schemes can also be combined with the snark techniques for efficient proof verifications [33], [34]. For blockchain-based VNF management, the VNF lookup should be flexible and require efficient on/off-chain lookup operations.

Zk-snark can translate arithmetic circuits into a proof system with succinct and efficient verifications. Zk-snark was first proposed in [35], which utilizes a quadratic arithmetic program (QAP) to represent circuit evaluations with installation in bilinear groups. Later, a toolchain that compiled a subset of C programs to QAP was proposed in [24], [25], [36]. The proof size of QAP-based snark was further reduced in [37]. Zk-snark can be combined with Pedersen-like vector commitments to construct an authenticated dictionary. In [38], it was observed that, for the zk-snark with structured generators, an offline digest of inputs can be adopted to support general computations. Composite arguments [39] or commit-and-prove snarks [40] were studied, where either the input or the output of a snark system can be replaced by a cryptographic commitment. By doing so, the snark system can be compatible with other zero-knowledge proof systems for achieving rich functionalities in practice.

In the snark-based authenticated dictionary, a challenging issue came from the random access memory (RAM) for circuit-based computations. In a non-RAM snark system [24], [25], only static memory access was supported in a subset of C programs. As a result, dynamic array access or loop breaks cannot be achieved, which results in a linear search of all dictionary entities. RAM-enabled zk-snark was studied in [26], [27], which usually relied on a permutation proof system that may increase the circuit complexity. By contrast, *Block-VNF* addresses the inefficiency of the snark-based dictionary for VNF management via a dictionary pruning strategy.

III. Building Blocks of *Block-VNF*

In this section, we summarize the building blocks of *Block-VNF*, including vector commitments and succinct non-interactive arguments.

TABLE I: Abbreviations

NFV	Network function virtualization
VNF	Virtualized network function
SLA	Service Level Agreement
VNF-P	VNF provider
VNF-T	VNF tenant
VNF-M	VNF manager
SA	Supervising authority
SN	Supervising node

TABLE II: Notations

λ	Security parameter
$\mathbb{G}$	Multiplicative groups
$\mathbb{Z}_p$	A ring of integers with a prime order, p
$\mathbf{v}^n$	n-dimension vector
$[m,n]$	Integers from m to n
$\mathcal{R}$	A relation with instance x and witness w
$\mathcal{C}$	Arithmetic circuit
$\mathcal{Q}$	Quadratic arithmetic program
$\Omega_{\mathcal{Q}}$	A snark system for $\mathcal{Q}$
I_{io}	Input and output wires of $\mathcal{C}$
I_{im}	Intermediate wires of $\mathcal{C}$
EK/VK	Evaluation/Verification Key
V_i	n-dimension VNF information vector
$\mathcal{D}=\{V_i\}_m$	VNF dictionary of m VNF vectors
$\mathcal{F}$	VNF lookup function
$Q/\mathcal{R}$	VNF query vector and query result
$Aut_{\mathcal{D}}$	Authenticator of $\mathcal{D}$

A. Notations

Denote $\mathbb{G}=(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T)$ with a prime order p and an efficient bilinear paring $e:\mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_T$. Denote g as a generator from $\mathbb{G}$ and the tilde form $\tilde{g}$ as a generator from $\mathbb{G}_2$. We use $[m,n]$ to represent integers from m to n, and $\mathbf{v}^n=(v_1,v_2,...,v_n)\in\mathbb{Z}_P^n$ to represent an n-dimension vector from $\mathbb{Z}_P$. Key abbreviations and notations are listed in Table I and II, respectively.

B. Vector Commitments

Cryptographic commitment schemes [41] are widely used to generate commitment of secret values. Given a vector, $\mathbf{v}^n$, and a set of generators, $(g_0,g_1,...,g_n)$, a vector commitment can be constructed as a Pedersen-like commitment [32], [42]:

$$C=g_0^r\prod_{i=1}^{n}g_i^{v_i}, \tag{1}$$

where r is a random number from $\mathbb{Z}_P$. The following two properties are usually considered for vector commitment:

- Binding – If the generators are randomly chosen, each position in the commitment is bound to a specific value and cannot be efficiently opened to different values.
- Computational Hiding – The committed values cannot be derived from the commitments by a computationally-bounded adversary.

C. Succinct Non-interactive Argument

Succinct Non-interactive ARgument of knowledge (snark) [24], [40] enables a prover $(\mathcal{P})$ to convince a verifier $(\mathcal{V})$ that an instance (x,w) holds on a relation $(\mathcal{R})$. It is *succinct*, if the argument size is related only to the security parameter λ, regardless of the complexity of $\mathcal{R}$. It is *non-interactive* if the argument is a one-move proof system. In the following, we introduce the *snark* system for relations represented by arithmetic circuits.

1) Arithmetic Circuit: An arithmetic circuit ($\mathcal{C}$) consists of addition/multiplication gates interconnected as a directed acyclic graph. An arithmetic circuit can be used to evaluate a function ($\mathcal{F}$) at points $(x_1, x_2, ..., x_n)$, where $(x_1, x_2, ..., x_n)$ are the inputs of $\mathcal{C}$ and $\mathcal{F}(x_1, x_2, ..., x_n)$ is the output of $\mathcal{C}$. Given circuit $\mathcal{C}$ with assigned values of input wires, a verifier can evaluate the circuit with an increasing complexity with the number of (multiplication) gates in the circuit.

*2) Quadratic Arithmetic Program (*QAP*):* Evaluation of $\mathcal{C}$ is formulated as checking the divisibility of its equivalent *QAP* ($\mathcal{Q}$) [35]. We denote the number of multiplication gates in $\mathcal{C}$ as d, which is also the degree of $\mathcal{Q}$. We denote v as the number of wires in $\mathcal{C}$, which can be further divided into two sets: (1) $I_{io} = (c_1, c_2, ..., c_u)$, input/output wires of $\mathcal{C}$; (2) $I_{im} = (c_{u+1}, c_{u+2}, ..., c_v)$, wires of the intermediate multiplication gates. $\mathcal{Q}$ computes polynomials: $\{A_k(x)\}$, $\{B_k(x)\}$ and $\{C_k(x)\}$, $k \in [0, 1, ..., v]$, as well as a target polynomial $t(x)$ [25].

Given $\{A_k(x)\}, \{B_k(x)\}, \{C_k(x)\}$, and $t(x)$, I_{io} is a valid assignment of $\mathcal{C}$ iff I_{im} can be found, such that $t(x)$ divides $p(x)$:

$$\begin{aligned} p(x) =&(A_0(x) + \sum_{i=1}^{v} c_i A_i(x)) \times (B_0(x) + \sum_{i=1}^{v} c_i B_i(x)) \\ &- (C_0(x) + \sum_{i=1}^{v} c_i C_i(x)). \end{aligned} \tag{2}$$

It should be noted that the polynomials in $\{A_k(x)\}$ that correspond to the input/output wires should be instantiated as linearly independent [25]. By doing so, the consistency check in the later snark system can be more efficient for a verifier.

3) Succinct Non-interactive Argument of Knowledge: The *QAP* formulates evaluation of circuit $\mathcal{C}$ as checking the divisibility of $\mathcal{Q}$, which can be efficiently instantiated in bilinear groups with a snark system, $\Omega_{\mathcal{Q}}$. Recall the constructions from the Pinocchio framework [24] in asymmetric groups with an augmented QAP generation [25], [39]. Prover $\mathcal{P}$ can evaluate $\mathcal{C}$ with input/output I_{io} and generate a proof, π. Verifier $\mathcal{V}$ can efficiently check whether I_{io} is a valid assignment of $\mathcal{C}$ with π. Specifically, a snark system includes three algorithms:

- Setup$(\mathbb{G}, \mathcal{F}) \rightarrow (EK, VK)$ – The algorithm takes into bilinear group $\mathbb{G}$ and function $\mathcal{F}$ represented by arithmetic circuit $\mathcal{C}$, to output an evaluation key (EK) and a verification key (VK). Specifically, the algorithm converts $\mathcal{C}$ into QAP $\mathcal{Q}$ and encodes polynomials $A_k(x)$, $B_k(x)$, and $C_k(x)$ at a trapdoor secret (s).
- Prove$(\mathbf{x}_I, EK) \rightarrow (\mathbf{x}_O, \pi)$ – The algorithm takes a vector $(\mathbf{x}_I)$ of inputs and the evaluation key, and evaluates circuit $\mathcal{C}$ with $\mathbf{x}_I$ to obtain values of intermediate wires $\mathbf{x}_M$ and output wires $\mathbf{x}_O$. It also generates proof π;
- Verify$(\mathbf{x}_I, \mathbf{x}_O, VK, \pi) \rightarrow (0, 1)$ – The algorithm takes the input vector, output vector, the verification key, and the proof. It outputs either accept or reject.

Detailed constructions can be found in [24], [25].

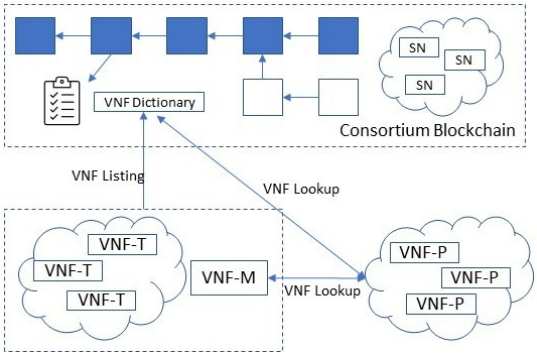


Fig. 1: System Model

IV. System Model, Threat Model and Design Goals

In this section, we present system model and threat model of *Block-VNF* with design goals.

A. System Model

In Fig. 1, there are five entities in *Block-VNF*: VNF Provider (VNF-P), VNF Tenant (VNF-T), VNF Manager (VNF-M), Supervising Authority (SA) and a consortium blockchain:

- VNF-P is an owner of virtual network resources [16], e.g. a wireless operator that controls radio spectrum resources. VNF-P lists its VNFs at the repository of VNF-M [15] to provide VNF repository services for VNF-T;
- VNF-T is a user of VNFs, and enjoys on-demand and pay-as-you-go network services from VNF-M, which may consist of a set of VNFs from different VNF-Ps. For example, an HD map service may consist of VNFs deployed at edge servers or base stations;
- VNF-M can be a third-party company, e.g., a cloud datacenter [16]. Specifically, VNF-M acts as a broker between VNF-P and VNF-T [13] for VNF repository services;
- SA is a set of supervising nodes (SN) belonging to multiple network stakeholders, e.g., wireless operators and cloud centers in future wireless networks. SA is responsible for maintaining a consortium blockchain and for setting up public parameters.
- A consortium blockchain is maintained by SA. It is responsible for listing VNFs and recording VNF query instances.

Block-VNF works with the following steps: (1) Setup – SA sets up a consortium blockchain and public system parameters; (2) VNF Listing – VNF-Ps register their VNFs at VNF-M, build a VNF dictionary with an authenticator, and upload the authenticator onto the blockchain; (3) VNF Lookup – VNF-Ts query the VNF dictionary based on VNFs' locations, functionalities, resources, prices, etc. VNF-M processes the VNF queries from VNF-Ts and returns verifiable query results. VNF-Ts can verify the query results off the blockchain and send complaints to the blockchain if an incorrect query result is identified. In the following, we define the VNF dictionary and VNF query.

Definition 1 – VNF dictionary $\mathcal{D}$ is a collection of VNF information vectors V_i:

$$\mathcal{D} = \{V_i = (attr_1, attr_2, ..., attr_n)\}_{i\in[1,m]}. \quad (3)$$

The dictionary consists of m VNF information vectors. Each vector, V_i, consists of n attributes, including id, availability, configurations, price, and so on. The attributes are represented by integer values for numeric attributes or keyword attributes. Specifically, an attribute can be a non-zero integer for the availability, or an integer encoding of a string.

Definition 2 – VNF lookup $\mathcal{F}$ is a function defined as follows:

$$\mathcal{F} : (\mathcal{D}, Q) \rightarrow \mathcal{R}. \quad (4)$$

In Equ. 4, $\mathcal{D}$ is the VNF dictionary, and $Q = (q_1, q_2, ..., q_n)$ is an n-dimension query vector that specifies VNF requirements, including functionalities, locations, computing/storage/bandwidth resources, prices, and so on. The lookup function executes Q over $\mathcal{D}$, and outputs a lookup result $\mathcal{R} = (id_1, id_2, ..., id_{m^*})$ of m^* identifiers of matched VNFs, where m^* is the number of matched VNFs.

B. Threat Model

Block-VNF focuses on security issues of the VNF query services, to increase the transparency and audibility of the VNF management. SA is honest and trustworthy in *Block-VNF*, which ensures the security of the system setup and the blockchain. VNF-Ps are audited by SA and follow pre-defined service agreements. They faithfully provide VNF information, and construct a VNF dictionary and the authenticator. VNF-Ts are rational users, who correctly construct their VNF queries and accept query results if the queries are correctly executed over the VNF dictionary. VNF-M is a third-party broker. Since VNF queries are executed locally by VNF-M, it may not always follow the pre-determined query rules for VNF lookup services [16]. For example, VNF-M can deliberatively allocate VNFs of high prices with a higher priority.

C. Design Goal

Under the system model and the threat model, we identify design goals of *Block-VNF*.

Verifiable VNF Lookup – In a distributed environment where VNF-Ps and VNF-Ts come from different domains and VNF queries are executed off-chain by VNF-M, a reliable and transparent auditing framework for VNF management is required [23]. Within this framework, VNF lookup should be verifiable with two properties: (1) Input authenticity – $\mathcal{D}$ and Q should be authenticated by VNF-Ps and VNF-Ts; (2) Execution correctness – $\mathcal{F}$ should be correctly executed over $\mathcal{D}$ and Q with a pre-determined matching rule.

Efficient VNF Management – First, on-chain costs for storing a VNF authenticator and verifying VNF queries should be low; Second, off-chain processing of VNF queries with generations of correctness proofs should be efficient.

V. The Proposed *Block-VNF* Scheme

In this section, we discuss design challenges, and present detailed constructions of *Block-VNF*.

The lookup function can be represented by an arithmetic circuit to be instantiated using the setup function of the snark algorithm by SA. Later, VNF-M can evaluate VNF query Q on VNF dictionary $\mathcal{D}$, and generate a result with a proof. The lookup result can be efficiently and securely verified using the verification function of the snark. Unfortunately, the straightforward solution can significantly decrease the prover efficiency when the size of $\mathcal{D}$ is large. First, on the VNF-T (verifier) side, the verification requires the inputs of the whole dictionary in plaintext, which can incur heavy verification cost. Second, on VNF-M (prover) side, when the lookup function is represented by an arithmetic circuit, it is difficult to have an efficient snark-based dictionary lookup as discussed in the related works. To address the challenges, we present *Block-VNF*, which consists of three phases: System Setup, VNF Listing, and VNF Lookup.

A. System Setup

To address the first challenge, we let the VNF-P precompute an authenticator, Aut_D, of $\mathcal{D}$ using the vector commitment technique. The authenticator can serve as a digest of $\mathcal{D}$ to be used in the proof verification. As a result, we obtain a modified snark [40] as follows:

- Setup$(\mathbb{G}, \mathcal{F}) \rightarrow (CK, EK, VK)$ – The algorithm additionally generates a set of commitment keys CK for the dictionary;
- Commit$(\mathcal{D}, CK) \rightarrow Aut_D$ – The algorithm generates a commitment Aut_D for the dictionary;
- Prove$(\mathbf{x}_Q, \mathcal{D}, EK) \rightarrow (\mathbf{x}_O, \pi)$ – It takes the evaluation key, the query vector, and the dictionary, and generates a result $\mathbf{x}_O$ with proof π. Note that input $\mathbf{x}_I$ is split into $\mathcal{D}$ and a query $\mathbf{x}_Q$.
- Verify$(Aut_D, \mathbf{x}_Q, \mathbf{x}_O, VK, \pi) \rightarrow (0, 1)$ – Based on the dictionary commitment, query $\mathbf{x}_Q$, matching results $\mathbf{x}_O$, the verification key, and the proof, the algorithm makes a decision either to accept or to reject.

With the modified snark system, SA can securely set up the system:

First, SA sets up a consortium blockchain based on Hyperledger Fabric. All VNF-Ps, VNF-Ts, and VNF-M can obtain an identity to communicate with the blockchain and each other from Hyperledger Fabric's membership service. In *Block-VNF*, we assume all communications between the VNF-Ps, VNF-Ts, VNF-M, and the blockchain are secure and authenticated.

Second, SA sets up a dictionary template for the modified snark. Based on Definition 1, attribute domain $(attr_1, attr_2, ..., attr_n)$ of a VNF information vector (V_i) can be divided into two parts: keyword attribute and numeric attribute. To represent keywords as integers, SA defines a dictionary $(\mathcal{W})$ that maps each keyword to an integer (w_x). v_y is an integer that represents a numeric attribute. As a result, dictionary $\mathcal{D}$ is refined as:

$$\mathcal{D} = \{V_i = (\{w_x \in \mathcal{W}\}_{x\in[1,n_1]}, \{v_y\}_{y\in[n_1+1,n]})\}_{i\in[1,m]}, \quad (5)$$

where n is the dimension of VNF information vector V_i, n_1 is the number of keyword attributes in V_i, and m is the number

of VNFs in $\mathcal{D}$. It should be mentioned that one numeric value of each V_i can be set as a unique random number, such as the VNF ID. By doing so, different VNFs can have different VNF information vectors.

A VNF query, $Q = (q_1, q_2, ..., q_n)$, can also be divided into two sets: keyword query and range query. For the keyword query, q_i is represented by an integer from $\mathcal{W}$. A range query can be represented by $[a, b]$, where a, b are two integers. Given $\mathcal{D}$ and Q, VNF lookup function $\mathcal{F}$ in Definition 2 checks each item in Q against that in each VNF of $\mathcal{D}$. We take a VNF information vector V_j to illustrate the lookup process. Specifically, for keyword query $q_x, x \in [1, n_1]$, the lookup function checks if $q_x = w_x \in V_j$; for range query $q_y = [a, b], y \in [n_1 + 1, n]$, the lookup function checks if $v_y \in V_j$ lies in $[a, b]$. VNF V_j is matched with Q if all the checks pass.

To address the second challenge for the prover inefficiency of the existing snark with/without the RAM, we propose a dictionary pruning strategy. The strategy is based on an observation that there usually exists a key query item, q^*, in query Q that can significantly prune dictionary $\mathcal{D}$ to a smaller dictionary, $\mathcal{D}'$. More specifically, SA breaks original function $\mathcal{F}$ into three functions:

$$\begin{aligned} &\mathcal{F}_1 : (\mathcal{D}, q^*) \rightarrow R_1, \mathcal{F}_2 : (\mathcal{D}', Q) \rightarrow R_2, \\ &Prune(\mathcal{D}, Aut_D, R_1) \rightarrow (\mathcal{D}', Aut_{D'}, \pi_p). \end{aligned} \tag{6}$$

In Equ. 6, $\mathcal{F}_1$ takes original dictionary $\mathcal{D}$ and key query item $q^* \in Q$. For each $V_i \in \mathcal{D}$, $\mathcal{F}_1$ checks that if q^* is matched and outputs indexes of m^* matched VNFs in R_1; $\mathcal{F}_2$ takes query Q and pruned dictionary $\mathcal{D}'$ based on q^*. It outputs the indexes of final VNFs in R_2.

For the pruning function $Prune$, based on the original dictionary $(\mathcal{D})$, its authenticator (Aut_D) and the result R_1, the function generates a pruned dictionary $(\mathcal{D}')$ with its authenticator $(Aut_{D'})$, and a correctness proof (π_p). There also exists a polynomial-time verifying function that takes the authenticators, result R_1, and the proof to check if the pruning function is correctly conducted.

SA defines two modified snark systems: Ω_1 and Ω_2 for functions $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. SA chooses system security parameter λ and bilinear groups $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with a bilinear pairing (e) and a prime order (p). SA chooses two random generators, $g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$, and a collision-resist hash function $H : (0,1)^* \rightarrow (0,1)^{512}$, such as SHA-512. SA instantiates functions $\mathcal{F}_1$ and $\mathcal{F}_2$ by running $Setup$ algorithm of the modified snark:

$$\begin{aligned} &Setup(\mathbb{G}, \mathcal{F}_1) \rightarrow (CK1, EK1, VK1) \\ &Setup(\mathbb{G}, \mathcal{F}_2) \rightarrow (CK2, EK2, VK2). \end{aligned} \tag{7}$$

For Ω_1, inputs of $Prove$ function include q^* and $\mathcal{D}$; For Ω_2, inputs of $Prove$ function include Q and $\mathcal{D}'$;

Finally, SA publishes $\lambda, \mathbb{G}, g, \tilde{g}, e, p, H, VK1, VK2$ on the blockchain, and sends $CK1$ and $CK2$ to VNF-Ps, and $EK1$ and $EK2$ to VNF-M. To support efficient verifications, $CK1$ and $CK2$ should be instantiated from an augmented QAP with linearly independent generators; Otherwise, they can be instantiated from an external commitment scheme and be linked with the modified snark system with a CP-link algorithm [40]. Details of the pruning function are presented in the next subsections.

B. VNF Listing

To design the pruning function, we have some strawman solutions from the existing works. We start with the design of the pruning function using a modified snark. The snark can check the key query item q^*, and directly copy the satisfied VNF information vectors into the new dictionary. However, such a mechanism is not efficient since this still requires RAM to the matched VNF information vectors in the original dictionary. Since Aut_D and $Aut_{D'}$ are vector commitments from bilinear groups, we may model the pruning function as a subvector commitment scheme [42]. However, the state-of-the-art subvector proposals are mainly designed for membership testing, which are difficult to encode the pruning strategy based on q^*. Moreover, verification of the subvector commitments requires the inputs of the dictionary and a large size of public parameters.

As a result, it is a challenging task to design a pruning function only from the two authenticators, which either requires an efficient snark system with RAM or an efficient vector commitment scheme. To address the challenge, we construct an auxiliary dictionary, $\mathcal{D}_{aux}$, that stores each VNF information vector V_i of $\mathcal{D}$ as an individual authenticator. The modified snark scheme Ω_1 conducts the key query on $\mathcal{D}$, and generates indexes of VNF information vectors that satisfy q^* in R_1. With the output indexes and $\mathcal{D}_{aux}$, a new authenticator, $Aut_{D'}$, can be verifiably generated based on Merkle tree [43]. By doing so, we reduce the computation cost at VNF-M for conducting the pruning operation. In the following, we present detailed constructions of $\mathcal{D}$, Aut_D and $\mathcal{D}_{aux}$.

For illustrative simplicity, we consider a single VNF-P in constructing $\mathcal{D}$, which can be easily extended to multiple VNF-Ps by allocating each of VNF-Ps a set of entries in the dictionary. The VNF-P constructs VNF dictionary $\mathcal{D} = \{V_i\}_{i \in [1,m]}$ for its VNFs. The VNF-P uses $CK1 \in \mathbb{G}_1^{m*n}$ to commit dictionary $\mathcal{D}$. We denote m^*_{max} as the maximum number of VNFs in $\mathcal{D}'$. Therefore, we have $CK2 \in \mathbb{G}_1^{m^*_{max}*n}$, since $\mathcal{D}'$ is at most $m^*_{max}*n$ dimension. The VNF-P computes:

$$\begin{aligned} &Aut_D = \prod_{i=1}^{m} \prod_{j=1}^{n} (CK1[i][j])^{V_i[j]}, \\ &Aut_{V_{i,j}} = \prod_{x=1}^{n} (CK2[j][x])^{V_i[x]}, \forall V_i \in \mathcal{D}, j \in [1, m^*_{max}], \\ &Aut_{h_{i,j}} = H(i||j||Aut_{V_{i,j}}), \forall V_i \in \mathcal{D}, j \in [1, m^*_{max}]. \end{aligned} \tag{8}$$

$CK1[i][j]$ represents a generator in $CK1$ for the j-th attribute in the i-th VNF information vector in $\mathcal{D}$. $V_i[j]$ is the value of j-th attribute in V_i. $CK2[j][x]$ represents a generator for the x-th attribute in the j-th VNF information vector in $\mathcal{D}'$. $Aut_{V_{i,j}}$ is an individual authenticator for $V_i \in \mathcal{D}$ if V_i is the j-th VNF in $\mathcal{D}'$. Since the VNF-P does not know the exact position of V_i in $\mathcal{D}'$ before the pruning function is executed, the VNF-P needs to pre-compute m^*_{max} authenticators for each $V_i \in \mathcal{D}$. As a result, there are total $m*m^*_{max}$ individual authenticators.

The VNF-P then generates a Merkle tree for all $m * m^*_{max}$ authenticators. For illustrative simplicity, assume that $m * m^*_{max} = 2^{h-1}$ is the exponentiation of 2 and h is the height of the Merkle tree. If not, we can pack items to $Aut_{h_{i,j}}$ to make it a fully balanced binary tree. The VNF-P computes a Merkle hash tree for all $Aut_{h_{i,j}}$ to obtain a Merkle tree as the auxiliary dictionary $\mathcal{D}_{aux}$ and a root $Root$. An illustrative construction of the dictionary and authenticators is shown in Fig. 2.

Finally, the VNF-P sends $\mathcal{D}$, $\mathcal{D}_{aux}$, and $Aut_{V_{i,j}}, i \in [1, m], j \in [1, m^*_{max}]$ to VNF-M. The VNF-P uploads the Aut_D and $Root$ onto the blockchain.

C. VNF Lookup

We present detailed constructions of the VNF lookup, including Query Construction, Query Processing, and Query Verification.

1) Query Construction: A VNF-T constructs a VNF query:

$$Q = (q^*, \{w_i\}_{i\in[1,n_1]}, \{[a_j, b_j]\}_{j\in[n_1+1,n]}), \tag{9}$$

where w_i is a keyword query (e.g., location, functionalities), and $[a_j, b_j]$ is a range query (e.g., computing resource, and bandwidth resource). We consider (a fixed) one of w_i as key item query q^*. The VNF-T sends query Q to a VNF lookup contract on the blockchain. The contract checks the query completeness and adds the query, the ID of the VNF-T, and a processing flag on the blockchain. The ID is used to uniquely identify the query and the flag is used to record the status of the query processing.

2) Query Processing: The Merkle-tree-based auxiliary dictionary can introduce additional communication overhead for on-chain verification of proof π_p for the pruning function. To reduce the on-chain verification overhead, we enable efficient off-chain verifications with on-chain complaints for incorrect proofs [21] in the lookup contract. Specifically, VNF-M processes the VNF query and sends the results with proofs off-chain to the VNF-T. The VNF-T verifies the results and proofs, and only makes complaints on the blockchain when any verification fails.

Upon receiving Q, VNF-M first searches dictionary $\mathcal{D}$ with q^* using $\mathcal{F}_1$. VNF-M generates a result and a proof using the snark system Ω_1 as follows:

$$\Omega_1.Prove(\mathcal{D}, q^*, EK1) \rightarrow (R_1, \pi_1). \tag{10}$$

As the definition of the modified snark system, inputs of $Prove$ of Ω_1 include $\mathcal{D}$ and q^*. Output R_1 consists of indexes of m^* VNFs (in an increasing order) that match the key query q^*. We denote $R_1 = (i_1, i_2, ...i_{m^*})$, where $V_{i_x,x}$ is the i_x-th VNF in $\mathcal{D}$ and is the x-th VNF in R_1. Then, VNF-M constructs a pruned dictionary authenticator as follows:

$$Aut_{D'} = \prod_{x=1}^{m^*} Aut_{V_{i_x,x}}. \tag{11}$$

VNF-M computes a Merkle proof π_p as in Algorithm 1. The algorithm is used to demonstrate that the authenticators of VNFs in R_1 are consistent with the committed authenticators from the VNF-P. Specifically, the algorithm finds a Merkle path for each authenticator of V_i in R_1 from $\mathcal{D}_{aux}$ and returns the siblings along the path, which is essentially a membership proof of Merkle tree.

A pruned dictionary $\mathcal{D}'$ is constructed from all VNF information vectors indicated by R_1. After that, VNF-M performs query Q over $\mathcal{D}'$ using $\mathcal{F}_2$. VNF-M generates a final result R_2 with a proof π_2 using the snark system Ω_2 as follows:

$$\Omega_2.Prove(\mathcal{D}', Q, EK2) \rightarrow (R_2, \pi_2). \tag{12}$$

R_2 consists of indexes of VNFs from R_1 (in an increasing order) that match the query Q.

Finally, VNF-M sends π_1, π_2, π_p, R_1, $\{Aut_{V_{i_x,x}}\}_{i_x\in R_1}$ and R_2 to the VNF-T via a secure and authenticated channel.

Algorithm 1: Merkle Tree Proof

Input: $\mathcal{D}_{aux}$, R_1
Output: Proof π_p
for $\forall i \in R_1$ **do**
  Identify V_i as j_{th} VNF in R_1
  Find a path from $Aut_{h_{i,j}} \in \mathcal{D}_{aux}$ to the root
  Add siblings of nodes on the path to π_p

3) Query Verification: The VNF-T retrieves Aut_D and $Root$ from the blockchain, and runs the following verifications:

$$\begin{aligned} &\Omega_1.Verify(Aut_D, q^*, R_1, VK1, \pi_1), \\ &\Omega_2.Verify(Aut_{D'}, Q, R_2, VK2, \pi_2), \\ &Aut_{D'} = \prod_{x=1}^{m^*} Aut_{V_{i_x,x}}. \end{aligned} \tag{13}$$

The VNF-T also re-constructs the Merkle root from π_p and $\{Aut_{V_{i_x,x}}$, and checks if the reconstructed root is equal to $Root$. If all the verifications pass, the VNF-T sends the query information and a correctness confirmation to the VNF lookup contract. The contract checks that the confirmation and the original query are sent from the same VNF-T. If all checks pass, the contract concludes the query is processed correctly.

If any of the proofs fails, the VNF-T can send the received proof with the corresponding result to the lookup contract. Since communications between the VNF-T and VNF-M are authenticated, the contract can check that the proof and the result are sent from VNF-M. After that, the contract runs the corresponding verification function to check the received proof. If the verification by the contract fails, the contract concludes the query is not processed correctly and enforces associated accountability against VNF-M.

Remarks. (1) All communications between the VNF-T and VNF-M are authenticated, which can thus serve as the evidence in case of any complaint; (2) VNF-M can generate commitments of R_1 and R_2 similar to Aut_D. With R_1 and R_2, VNF-T can locally verify that the commitments of R_1 and R_2 are correctly computed. By doing so, the verifications of proofs π_1 and π_2 require less on-chain communication overhead; (3) For the Merkle proof, if an incorrect proof for a VNF is detected, the VNF-T only needs to upload the

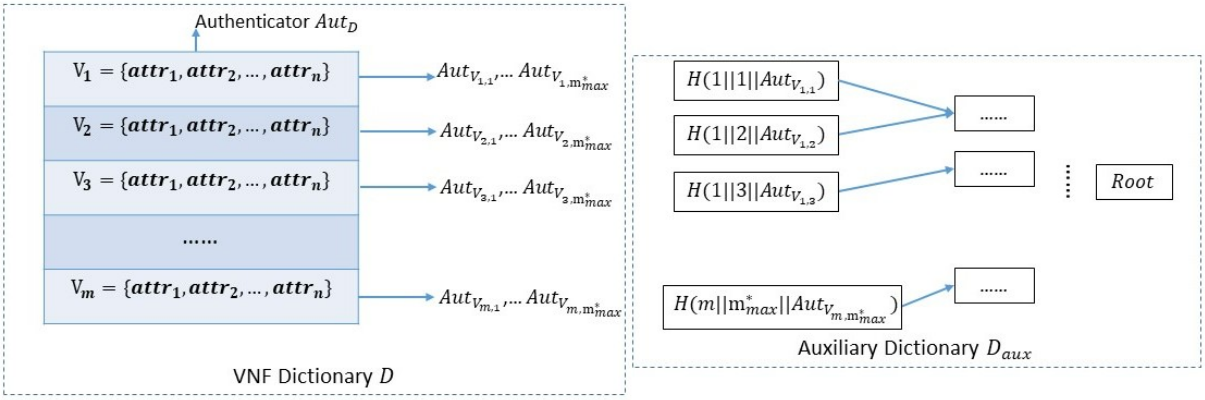


Fig. 2: An Illustration of VNF Dictionaries

corresponding proof and result (instead of the whole proof) to the blockchain. This can significantly reduce the on-chain verification overhead.

Discussions. (1) The modified snark system can be instantiated from online-offline snark systems, such as variants of Pinocchio [25] [36] or [37]; (2) To generate relation-independent dictionary authenticators, the modified snark system can also be instantiated from external generators with a CP-link scheme [40].

VI. Security Analysis

In this section, we first analyze the security of components in *Block-VNF*, including blockchain, vector/Merkle commitment, snark, and the pruning function. Then, we summarize how the security of verifiable VNF lookup is achieved.

A. Blockchain Security

Blockchain security ensures that (1) the on-chain storage cannot be maliciously modified, and (2) a valid transaction will be included in the ledger within a certain time. To achieve the blockchain security, different blockchain architectures can utilize different consensus protocols. In *Block-VNF*, we adopt a consortium blockchain architecture, i.e., Hyperledger Fabric [18]. Specifically, Hyperledger Fabric supports plug-in consensus protocols, such as Byzantine Fault Tolerant (BFT). Since *Block-VNF* relies on authenticated network stakeholders to maintain the consortium blockchain, the most of the stakeholders are honest and the blockchain security is achieved.

B. Security of Vector/Merkle Commitments

For the vector commitment, it should be computationally-infeasible for an adversary to find two different vectors that generate the same authenticator if the commitment keys are randomly constructed. In *Block-VNF*, commitment keys are from the common reference strings of the modified snark system. We can instantiate the snark system with strong binding [40], or use an augmented QAP with linearly independent polynomials for I/O wires [25]. Otherwise, we can also use external vector commitments with a CP-link algorithm for the snark system. Most importantly, the commitment keys are correctly set up by SA and VNF authenticators are honestly computed and stored on the blockchain by VNF-P. Even if the generators are not necessarily independent, it still does not affect the security of the snark system as long as the *soundness* property holds [40] and the authenticators are honestly computed and stored. For the hiding, it is not required in *Block-DM* since we do not consider zero-knowledge property of the snark system.

For Merkle commitment, given a Merkle root, an efficient adversary cannot open a specific leaf node to two different values. Otherwise, the adversary can break the collision resistance of the hash functions for computing the Merkle root.

C. Security of snark

In *Block-VNF*, we utilize the modified snark system. Its security is defined as:

$$\mathbf{Pr}\left[\begin{array}{l} Verify(Aut_D, Q, \mathcal{R}, VK, \pi) = 1 : \\ Setup(\mathbb{G}, \mathcal{F}) \rightarrow (CK, EK, VK) \wedge \\ Commit(\mathcal{D}, CK) \rightarrow Aut_D \wedge \\ \mathcal{F} : (\mathcal{D}, Q) \nrightarrow \mathcal{R} \wedge \\ \mathcal{A}_F(\mathcal{D}, Q, EK) \rightarrow (\mathcal{R}, \pi) \end{array}\right] = neg(\lambda). \tag{14}$$

$Commit$ is a function that honestly computes a dictionary authenticator. $\mathcal{A}_F$ is an adversarial function that aims to forge invalid results and proof. The above definition is similar to the *Soundness* definition of QAP-based snarks [24], [25]. Therefore, the security of the snark is guaranteed if (1) the $Setup$ algorithm is run by a trusted party or a multi-party computation protocol [44]; (2) the snark system is *sound*, where an efficient adversary cannot forge a valid tuple that passes the verification algorithm but is not an instance of the relation; (3) The Pedersen-like commitment Aut_D is honestly computed (by VNF-P). Since VNF-Ts are rational, they will accept results and proofs if they are correct. We do not require *zero knowledge* of the modified snark to be activated in *Block-VNF*.

D. Security of Pruning Function

Similar to the modified snark, the pruning function should be *verifiable*. That is, given a key query item, a Merkle root

of authenticators of individual VNFs, a result of the pruning function, and a Merkle proof, an aggregated authenticator of VNFs in the pruned dictionary can be verified.

The security of the pruning function comes from three aspects: (1) Dictionary authenticator Aut_D and a corresponding Merkle root $Root$ for individual authenticators of VNFs are securely set up by the VNF-P and stored on the immutable blockchain storage; (2) The security of modified snark Ω_1 ensures that result R_1 is correctly computed over $\mathcal{D}$ with q^*.That is, indexes of matched VNFs in R_1 are verifiable; (3) Given that R_1 and $Root$ are authentic, a computationally-bounded adversary cannot return invalid individual authenticators or forge proof π_p unless the adversary can break the security of the Merkle commitment.

E. Security of VNF Lookup

The security of VNF lookup consists of two aspects: input authenticity and execution correctness.

Input Authenticity: First, all submitted queries, Q, from VNF-Ts are authenticated since communications among VNF-Ts and VNF-M are secure and authenticated. Second, authenticator Aut_D and Merkle root $Root$ are truthfully computed and uploaded to the blockchain by VNF-P. Since the blockchain storage is immutable, the input of function $\mathcal{F}_1$ is also authenticated. Due to the security of the pruning function, inputs $Aut_{D'}$ and Q of function $\mathcal{F}_2$ are also authenticated.

Execution Correctness: First, since functions $\mathcal{F}_1$ and $\mathcal{F}_2$ are instantiated from secure snark systems Ω_1 and Ω_2, the execution correctness of the two functions is ensured. A computationally-bounded adversary cannot forge invalid results to pass verifications unless he can break the security of the snark system. Second, due to the security of the pruning function, the execution correctness of the pruning function is also ensured.

VII. Performance Evaluation

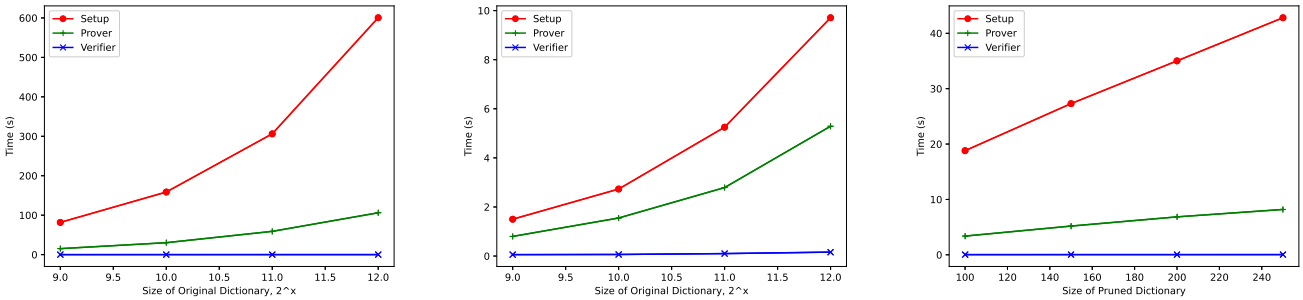
First, we give comprehensive evaluations of snark systems Ω_1 and Ω_2. The performance metrics include computational time of setup, prover, and verifier, public parameter size, and the memory usage. Second, we present the computation and communication overhead of the pruning function with Merkle proof. Third, we give the correlated analysis on the query accuracy with the snark systems. To demonstrate the efficiency of the proposed pruning strategy, we compare the performance between a VNF query without the pruning function and a VNF query with the pruning function. Finally, to demonstrate the feasibility of *Block-VNF* on the blockchain, we set up a consortium blockchain network and evaluate *Block-VNF* with different network settings.

A. Off-chain Experiments

We implement xjsnark [26] as a program-to-circuit compiler and libsanrk [45] as a circuit-to-snark compiler in a preprecessing mode [25] with alt-BN128 curve. We conduct our experiments on a laptop with 2.30GHz processor and 8GB memory. We set the dimension of VNF information vector V_i as 20, which includes 10 numeric values and 10 keyword values. Similarly, query Q is set to a vector of 10 keyword values and 10 ranges. We set q^* as VNF availability, and let $\mathcal{F}_1$ check if the corresponding attribute for $q*$ in each VNF vector is non-zero. Query function $\mathcal{F}_2$ is implemented as a one-by-one comparison between each $q_i \in Q$ and $attr_i \in V_i$. For range values, the function checks that if the item of each VNF in a pruned dictionary $\mathcal{D}'$ lies in the range of corresponding range values in Q; For keyword values, the function checks if the query has the same value compared with each VNF in $\mathcal{D}'$. The outputs of both snark systems are a vector with the same size of their input dictionaries. Verifications of both snark systems are implemented with $\mathcal{D}$ or $\mathcal{D}'$ as inputs rather than their authenticators.

1) Snark Complexity: In Fig. 3a, we plot the computation overhead of the setup, prover and verifier of the snark system for initial function $\mathcal{F}$ without implementing our dictionary pruning strategy, which changes the number of VNF functions (denoted as m) in $\mathcal{D}$ from 2^9 to 2^{12}. While the most expensive cost is the setup phase, the verifier overhead is around 100 ms. This is because the setup phase needs to compute the QAP and a large number of public parameters. It should be mentioned that the setup phase is conducted once. We present the performance of the snark systems Ω_1 and Ω_2 in Fig. 3b and Fig. 3c, respectively. In Fig. 3b, the number of VNF functions in $\mathcal{D}$ still changes from 2^9 to 2^{12}. Since the prover only needs to conduct comparisons at one out of 20 items between each VNF information vector and the query vector, the dictionary pruning incurs much fewer prover overhead as compared with that in Fig. 3a. In Fig. 3c, we test snark system Ω_2 against the number of remaining VNFs after the pruning. Since the size of the pruned dictionary is significantly smaller than the original dictionary, the snark system Ω_2 is much more efficient as compared with that in Fig. 3a.

In Table III, IV, and V, the QAP degree refers to the number of multiplication gates in the generated circuit, that can represent the complexity of associated functions. The memory usage includes both the physical and swap memory. We convert bits to KBs and MBs, where 1 KB = 8,000 bit and 1 MB = 1,000 KB. The function complexity of Ω_1 and Ω_2 is much less than the VNF lookup without the dictionary pruning, and all performance metrics increase with the number of VNFs. The PK size is much larger than VK size, which is important for achieving efficient proof verifications. The size of VK is affected by the total number of inputs and outputs. It leads to the same values in Table III and IV with the same number of VNFs, since we output results for all VNFs in the two functions. The computation complexity increases with m for Ω_1 while the complexity increases with m^*n for Ω_2, where m is the number of VNFs in the original dictionary and m^* is the number of VNFs in the pruned dictionary. n is the dimension of V_i and is set to 20. In Table IV and V, we present the performance of Ω_1 and Ω_2 with different pruning efficiency, which is the ratio between the sizes of the original dictionary and the pruned dictionary. As m^*n is larger than m in our experiments, the complexity of Ω_2 is higher than that of Ω_1. Since the memory usage increases with the computation complexity, the snark system in Table V takes more memory



(a) VNF Lookup without Pruning (b) Snark Ω_1 (c) Snark Ω_2

Fig. 3: Computation Overhead of Snark

as in Table IV.

TABLE III: VNF Lookup Without Pruning

Height	9	10	11	12
QAP Degree (10^6)	0.4	0.8	1.6	3.1
PK (MB)	113	226	452	904
VK (KB)	430	859	1717	3432
Memory (MB)	1170	2424	4305	8591

TABLE IV: Snark System Ω_1

Height	9	10	11	12
QAP Degree (10^3)	16	33	66	131
PK (KB)	1131	2262	4522	9042
VK (KB)	430	859	1717	3431
Memory (MB)	38	63	89	180

TABLE V: Snark System Ω_2

m^*	100	150	200	250
QAP Degree (10^3)	82	131	164	197
PK (MB)	22	34	44	55
VK (KB)	85	127	169	211
Memory (MB)	382	494	720	881

2) Merkle Tree Complexity: For the Merkle tree, we adopt SHA512 hash function in Java Pairing Based Cryptography (JPBC) [46] on the same laptop. We test the Merkle tree setup cost with the height of the Merkle tree (h). The computation overhead in the setup phase increases with h but is still very low compared with the computation overhead of the snark systems. More specifically, it takes a few milliseconds to generate a Merkle tree when $h = 15$. For the Merkle proof generation and the verification, the computation cost is negligible. This is because calculating hash functions are extremely efficient. In Fig. 4, we report the proof size against the number of VNFs in the pruned dictionary and the height of the Merkle tree. We take the worst-case scenario where the size is roughly $O(m^*(h-1))$, where m^* is the number of VNFs in the pruned dictionary. As shown in Fig. 4, the proof size is reasonable with a few KBs.

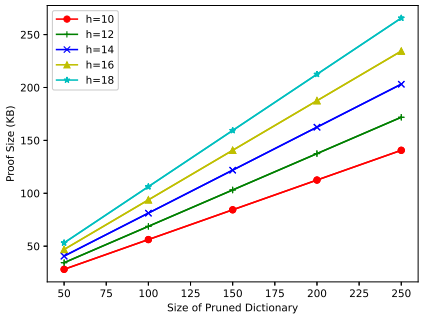


Fig. 4: Proof Size of Merkle Tree

3) Query Accuracy: Block-VNF achieves the same query accuracy compared with the non-verifiable setting. First, the VNF lookup function consists of keyword or numeric comparisons between a VNF query and the original VNF dictionary, which can be realized by a circuit without loss of accuracy. Second, the VNF lookup function is performed over the original VNF dictionary to output query results. The authenticator is a succinct digest of the original dictionary and is only used in the verification of the query results.

B. Performance Comparison

We compare the straightforward lookup strategy without the pruning and our pruning strategy at the prover and the verifier. For the verifier, both strategies have cost around a few milliseconds due to the verification efficiency of the snark and the Merkle proof. However, at the prover side, the proposed pruning strategy is much more efficient. In Fig. 5, we calculate the computational cost of our pruning strategy as the summation of prover cost in Ω_1 and Ω_2 (when the number of VNFs in the pruned dictionary is 150 and 200, respectively). The Merkle tree proof generation is extremely efficient and is thus omitted in the comparison. As we can see, our strategy has a significant efficiency gain, which essentially depends on the pruning ratio m^*/m. The reason is that the pruning strategy reduces many unnecessary but computationally-expensive comparisons between the query vector and the original dictionary.

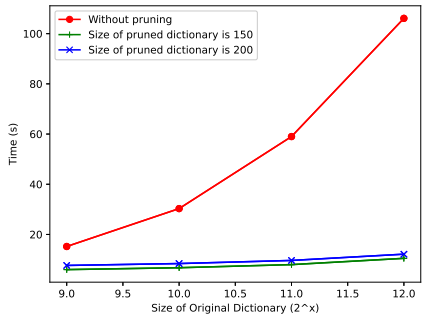
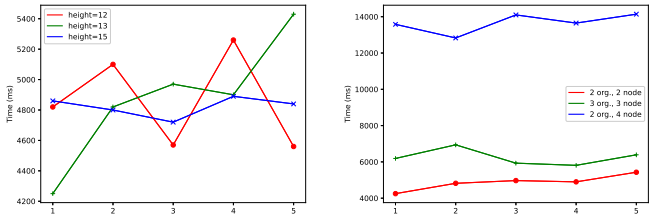


Fig. 5: Performance Gain with Pruning



(a) 2 org., two nodes (b) Height=13

Fig. 6: Response time

C. On-chain Experiments

We set up a real-world consortium blockchain network based on Hyperledger Fabric [18] on the same laptop in a Linux system. Specifically, we use the RAFT consensus protocol [18] with an ordering node. We test the verification of a single Merkle proof in different settings: two organizations, two peers; three organizations, three peers; two organizations, four peers. Specifically, we write the verification algorithm as a function in the chaincode. The function stores the Merkle root and receives Merkle proofs from a peer node. We send a function call and measure the response time of the blockchain network.

In Fig. 6a, we show the response time when there are two organizations and each organization has a peer node in the network. We change the Merkle tree height from 12 to 15, which slightly increases the proof size by one hash element. As we can see, the response time does not depend on the height but is mostly affected by the blockchain network. In Fig. 6b, we fix the tree height to 13 and plot the response time with different network settings. As the number of peer nodes increases in the network, the response time in our experiments increases due to the increasing number of required endorsements. However, the proposed mechanism does not depend on a specific blockchain architecture and thus can be tailored to the blockchain designs with other consensus protocols.

VIII. Conclusion

In this paper, we have designed an authenticated and prunable dictionary for blockchain-based VNF management in future wireless networks, which achieves rich VNF query functionalities and succinct on-chain storage and computing overhead. Our dictionary pruning strategy resolves the RAM issue and significantly reduces the computational overhead in processing VNF queries with a significant performance increase in comparison with the snark-based solution without pruning. The design and constructions of the authenticated and prunable dictionary can be of independent interests for other blockchain-based network resource managements. In the future, we will further explore blockchain-based VNF configurations and slice formations for NFV-enabled wireless networks.

IX. Acknowledgements

The authors would like to thank Yuxiang Zhang for testing JPBC in the consortium blockchain, and Jimmy Qi for testing xjsnark.

References

[1] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 45–66, 2020.

[2] S. Zhang, H. Luo, J. Li, W. Shi, and X. Shen, "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2150–2162, 2020.

[3] Z. Jia, M. Sheng, J. Li, D. Zhou, and Z. Han, "VNF-based service provision in software defined leo satellite networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 6139–6153, 2021.

[4] T. Gao, X. Li, Y. Wu, W. Zou, S. Huang, M. Tornatore, and B. Mukherjee, "Cost-efficient vnf placement and scheduling in public cloud networks," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4946–4959, 2020.

[5] Q. Ye, W. Zhuang, S. Zhang, A.-L. Jin, X. Shen, and X. Li, "Dynamic radio resource slicing for a two-tier heterogeneous wireless network," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9896–9910, 2018.

[6] H. A. Shah and L. Zhao, "Multiagent deep-reinforcement-learning-based virtual resource allocation through network function virtualization in internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3410–3421, 2020.

[7] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Dynamic service function chain embedding for NFV-enabled IoT: a deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 507–519, 2019.

[8] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future iov with enhanced communication, computing, and caching," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 274–291, 2019.

[9] G. A. F. Rebello, G. F. Camilo, L. G. Silva, L. C. Guimarães, L. A. C. de Souza, I. D. Alvarenga, and O. C. M. Duarte, "Providing a sliced, secure, and isolated software infrastructure of virtual functions through blockchain technology," in *IEEE International Conference on High Performance Switching and Routing (HPSR)*, 2019, pp. 1–6.

[10] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, "Distributed SDN controller system: A survey on design choice," *Computer Networks*, vol. 121, pp. 100–111, 2017.

[11] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1689–1703, 2019.

[12] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5G and beyond networks: A state of the art survey," *Journal of Network and Computer Applications*, p. 102693, 2020.

[13] M. F. Franco, E. J. Scheid, L. Z. Granville, and B. Stiller, "Brain: blockchain-based reverse auction for infrastructure supply in virtual network functions-as-a-service," in *IFIP Networking Conference (IFIP Networking)*, 2019, pp. 1–9.

[14] E. J. Scheid, M. Keller, M. F. Franco, and B. Stiller, "Bunker: A blockchain-based trusted VNF package repository," in *International Conference on the Economics of Grids, Clouds, Systems, and Services*. Springer, 2019, pp. 188–196.

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

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

[17] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for iiot-enabled retail marketing atop pos blockchain," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3527–3537, 2019.

[18] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proc. of the Thirteenth EuroSys Conference*, 2018, pp. 1–15.

[19] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger byzantium version," *Ethereum project yellow paper*, pp. 1–39, 2018-06-05.

[20] H. Zhang, J. Liu, H. Zhao, P. Wang, and N. Kato, "Blockchain-based trust management for internet of vehicles," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1397–1409, 2020.

[21] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proc. of ACM CCS*, 2018, pp. 967–984.

[22] D. Liu, J. Ni, C. Huang, X. Lin, and X. Shen, "Secure and efficient distributed network provenance for iot: A blockchain-based approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7564–7574, 2020.

[23] A. Tomescu, V. Bhupatiraju, D. Papadopoulos, C. Papamanthou, N. Triandopoulos, and S. Devadas, "Transparency logs via append-only authenticated dictionaries," in *Prof. of ACM CCS*, pp. 1299–1316.

[24] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. of IEEE S&P*, 2013, pp. 238–252.

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

[26] A. Kosba, C. Papamanthou, and E. Shi, "xjsnark: a framework for efficient verifiable computation," in *Proc. of IEEE S&P*, 2018, pp. 944–961.

[27] R. S. Wahby, S. T. Setty, Z. Ren, A. J. Blumberg, and M. Walfish, "Efficient ram and control flow in verifiable outsourced computation." in *Proc. of NDSS*, 2015.

[28] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä, "Blockchain network slice broker in 5G: Slice leasing in factory of the future use case," in *Internet of Things Business Models, Users, and Networks*, 2017, pp. 1–8.

[29] K. Nguyen, G. Ghinita, M. Naveed, and C. Shahabi, "A privacy-preserving, accountable and spam-resilient geo-marketplace," in *Proc. of ACM SIGSPATIAL*, 2019, pp. 299–308.

[30] C. Xu, C. Zhang, and J. Xu, "VChain: Enabling verifiable boolean range queries over blockchain databases," *Proc. of SIGMOD*, pp. 141–158, 2019.

[31] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Proc. of PKC*. Springer, 2013, pp. 55–72.

[32] A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich, "Aggregatable subvector commitments for stateless cryptocurrencies." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 527, 2020.

[33] A. Chepurnoy, C. Papamanthou, and Y. Zhang, "Edrax: A cryptocurrency with stateless transaction validation." *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 968, 2018.

[34] A. Ozdemir, R. Wahby, B. Whitehat, and D. Boneh, "Scaling verifiable computation using efficient set accumulators," in *Proc. of USENIX Security*, 2020, pp. 2075–2092.

[35] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Proc. of EUROCRYPT*. Springer, 2013, pp. 626–645.

[36] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur, "Geppetto: Versatile verifiable computation," in *Proc. of IEEE S&P*, 2015, pp. 253–270.

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

[38] D. Fiore, C. Fournet, E. Ghosh, M. Kohlweiss, O. Ohrimenko, and B. Parno, "Hash first, argue later: Adaptive verifiable computations on outsourced data," in *Proc. of ACM CCS*, 2016, pp. 1304–1316.

[39] S. Agrawal, C. Ganesh, and P. Mohassel, "Non-interactive zero-knowledge proofs for composite statements," in *Proc. of CRYPTO*, 2018, pp. 643–673.

[40] M. Campanelli, D. Fiore, and A. Querol, "Legosnark: Modular design and composition of succinct zero-knowledge proofs," in *Proc. of ACM CCS*, 2019.

[41] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Annual International Cryptology Conference*. Springer, 1991, pp. 129–140.

[42] R. W. Lai and G. Malavolta, "Subvector commitments with application to succinct arguments," in *Proc. of CRYPTO*, 2019, pp. 530–560.

[43] R. C. Merkle, "A certified digital signature," in *Conference on the Theory and Application of Cryptology*. Springer, 1989, pp. 218–238.

[44] S. Bowe, A. Gabizon, and M. D. Green, "A multi-party protocol for constructing the public parameters of the pinocchio zk-snark," in *Proc. of FC*. Springer, 2018, pp. 64–77.

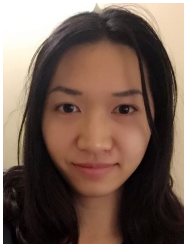
[45] libsnark: a C++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark. Accessed January 2020.

[46] A. De Caro and V. Iovino, "JPBC: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, Kerkyra, Corfu, Greece, June 28 - July 1, 2011, pp. 850–855.

Dongxiao Liu (M'20) is a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo. He received the PhD degree at the Department of Electrical and Computer Engineering, University of Waterloo, Canada in 2020. His research interests include security and privacy in intelligent transportation systems, blockchain, and mobile networks.

Cheng Huang (M'20) received his B.Eng and M.Eng in information security from Xidian University, China, in 2013 and 2016 respectively, and received the Ph.D. degree in Electrical and Computer Engineering, University of Waterloo, ON, Canada in 2020. He is currently a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests are in the areas of applied cryptography, cyber security and privacy in the mobile network.

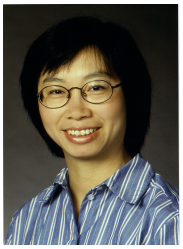
Liang Xue (S'19) received her B.S. and M.S. degree in School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), China in 2015 and 2018, respectively. Currently, She is pursuing the PhD degree at the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include applied cryptography, cloud computing, and blockchain.



Jiahui Hou (M'20) is currently a Postdoc Fellow in the Department of Electrical and Computer Engineering at the University of Waterloo. She received her Ph.D. degree in the Department of Computer Science, Illinois Institute of Technology, USA, in 2020. She received her B.E. degree in Computer Science and Technology from University of Science and Technology of China, P.R. China, in 2015. Her research interests include privacy and security issues in AI and mobile computing. Specifically, she is working on secure AI.

Xuemin (Sherman) Shen (M'97-SM'02-F'09) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks. Dr. Shen is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

Dr. Shen received 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, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom' 16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. Dr. Shen is the President of the IEEE Communications Society. He was the Vice President for Technical & Educational Activities, Vice President for Publications, Member-at-Large on the Board of Governors, Chair of the Distinguished Lecturer Selection Committee, Member of IEEE Fellow Selection Committee of the ComSoc. Dr. Shen served as the Editor-in-Chief of the IEEE IoT JOURNAL, IEEE Network, and IET Communications.



Weihua Zhuang (M'93−SM'01−F'08) received the B.Sc. and M.Sc. degrees from Dalian Marine University, China, and the Ph.D. degree from the University of New Brunswick, Canada, all in electrical engineering. She has been with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, since 1993, where she is a Professor and a Tier I Canada Research Chair of Wireless Communication Networks. She is the recipient of 2017 Technical Recognition Award from IEEE Communications Society Ad Hoc & Sensor Networks Technical Committee, and a co-recipient of several best paper awards from IEEE conferences. Dr. Zhuang was the Editor-in-Chief of IEEE Transactions on Vehicular Technology (2007−2013), Technical Program Chair/Co-Chair of IEEE VTC Fall 2017 and Fall 2016, and the Technical Program Symposia Chair of the IEEE Globecom 2011. She is a Fellow of the IEEE, the Royal Society of Canada, the Canadian Academy of Engineering, and the Engineering Institute of Canada. Dr. Zhuang is an elected member in the Board of Governors and VP Publications of the IEEE Vehicular Technology Society.

Rob Sun currently is a Principal Engineer with Huawei Technologies Canada Co. ltd. His work primarily focuses on the advancement of the NG wireless, including 5G/6G and WiFi/IoT security architecture and standardization. Rob was also vice chair of IEEE PMP (Privacy Management Protection) Task Group which was to set out the best practices for protecting personal privacy information, and support efficient, adaptable and innovative approaches for privacy governance. He was regarded as one of the core contributors to the standardizations of a series of NG Wi-Fi security protocols and certifications, including the most recent Wi-Fi WPA3 protocol suites. He also co-authored a few books on wireless security technologies.

Bidi Ying (Phd) is currently working in Huawei Technologies Canada Co., LTD as a senior network architecture engineer. Her main research is about security and privacy in wireless network. Before that, she was working in University of Ottawa. During the past 15 years, she has published more than 200 papers in top conferences and reputable journals.