A Privacy-Preserving Reinforcement Learning Algorithm for Multi-Domain Virtual Network Embedding

Davide Andreoletti[1, 3], Tanya Velichkova[2], Giacomo Verticale[3], Massimo Tornatore[3], and Silvia Giordano[1]

[1]Networking Laboratory, University of Applied Sciences of Southern Switzerland, Manno, Switzerland, Email: {name.surname}@supsi.ch
[2]Networking Laboratory, University of Applied Sciences of Southern Switzerland, Manno, Switzerland, Email: {name.surname}@student.supsi.ch
[3]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy, Email: {name.surname}@polimi.it

***Abstract*—The problem of optimally deploying a virtual network onto a substrate physical network is referred to as Virtual Network Embedding (VNE). In general, this embedding is requested by a customer to an Internet Service Provider (ISP), which performs the VNE over its physical telecom network. In several situations, the physical substrate infrastructure is composed of multiple independent ISPs. In this scenario, ISPs are concerned about exposing to a third-party entity (e.g., the customer) sensitive infrastructural details that are needed to perform an effective embedding. Following a common privacy-preserving approach, known as Limited Information Disclosure (LID), the embedding may be performed by the customer based on a limited and abstracted view of the multi-domain infrastructure that ISPs accept to expose. With this approach, embedding is sub-optimal (e.g., embedding cost is not minimized) in comparison with the case where all information is available, i.e., Full Information Disclosure (FID). In this work, we propose a Reinforcement-Learning-based algorithm able to process data that the customer and ISPs cipher under the Shamir Secret Sharing (SSS) scheme. This approach guarantees total privacy to both the customer and the ISPs (e.g., details about a virtual function are only revealed to the ISP in charge of hosting it) and achieves comparable embedding cost of an existing FID heuristic, as observed from extensive simulations. The main drawback of our algorithm is the high overhead of data that ISPs and the customer need to exchange with each other to execute it. Hence, we also explore the trade-off between embedding cost and data overhead resulting from the reduction of operations done by the RL. In general, intermediate embedding costs between the FID and LID heuristics can be obtained at a significant reduction of data overhead, while not sacrificing any privacy guarantees.**

***Index Terms*—Reinforcement Learning, Virtual Network Embedding, Privacy-Preserving Strategies**

I. Introduction

The decoupling of the software implementation of a service from its underlying hardware, known as Network Function Virtualization (NFV) [1], brings several advantages, such as increased service flexibility and scalability, and reduced CapEx and OPEX expenses. A virtualized service can be represented as a Virtual Graph (VG), i.e., a set of Virtual Nodes (VNs) and relative Virtual Paths (VPs), that a third-party entity (e.g., a *customer*) provisions to its users exploiting the physical infrastructure of an Internet Service Provider (ISP).

D. Andreoletti & T. Velichkova contributed equally to this work.

The problem of embedding this graph into the physical network towards some optimization objective (e.g., minimization of the deployment cost) is referred to as Virtual Network Embedding (VNE) and has received considerable attention in literature [1]. In particular, the general VNE problem was proved to be NP-hard [2]; hence, many heuristics have been proposed to efficiently solve it [3]. VNE is a well-studied problem when a single network operator is considered, but, to the extent of our knowledge, the case where the underlying infrastructure is composed of several independent ISPs' networks has only been scarcely explored so far. In such case, the customer may benefit from an extended covered geographical area, an increased heterogeneity of the available infrastructures and, eventually, also from reduced embedding costs.

However, the multi-domain scenario introduces new challenges that increase the complexity to effectively solve the VNE. In fact, ISPs may be concerned about the exposure of business-critical and privacy-sensitive details of their networks, which are required for the execution of embedding algorithms. A possible approach proposed in literature [4] to guarantee ISPs' privacy requirements while not preventing embedding consists of limiting the information available to the entity performing the optimization. Following this approach, referred to as Limited Information Disclosure (LID), the customer optimizes the assignment of portions of the VG to each involved ISP based on an abstracted view of the multi-domain infrastructure (e.g., typically, only the peering links and the cost of traversing them are visible to the customer) and, based on this assignment, each ISP embeds its sub-graph on the physical infrastructure. The main drawback of the LID approach is a sub-optimal embedding with respect to the Full Information Disclosure (FID) approach, in which the customer performs the optimization based on a complete view of the substrate multi-domain infrastructure.

In this work, we aim to guarantee a privacy level typical of LID approaches, while also achieving costs comparable to a FID strategy. Towards this objective, we initially propose a RL algorithm that a customer and the ISPs can execute in a distributed manner to solve the VNE problem in a cost-effective way. We observe, however, that this approach is

still vulnerable in means of privacy. To address this problem, we then propose the privacy-preserving version of this RL algorithm.

More specifically, we subdivide the VNE problem into several main sub-tasks, and we define a RL environment for each of them. Customer and ISPs perform operations on their environments and, based on that, receive rewards through which they learn how to efficiently solve the associated sub-task. To set a common optimization objective, rewards are properly designed and exchanged among participants. Specifically, rewards contain information about the embedding of VNs and VPs inside the single ISPs' domains, such as their embedding cost and constraints violation (e.g., nodes' capacity). We compare our RL-based algorithm with the LID and FID heuristics proposed in [4], considering the overall embedding cost. Our approach generally achieves a cost that is slightly lower than the cost obtained with FID, and significantly lower than the cost obtained with LID. However, to converge to an optimal solution of the VNE, participants need to exchange with each other a high amount of rewards, from which sensitive information can be leaked.

To solve this issue, we then propose a privacy-preserving version of the RL algorithm that is based on the Shamir Secret Sharing (SSS) scheme. By using this approach, the customer and the ISPs only learn information relative to the final embedding, and no sensitive information is leaked during the optimization process (e.g., the computational demand of a VN is only disclosed to the ISP selected to host it). The main drawback is the high volume of data that participants must exchange with each other to execute the privacy-preserving algorithm. Hence, we show how to reduce the introduced data overhead by limiting the number of expensive operations, and we evaluate the corresponding increase of embedding cost for several levels of this limitation. Results show that, even when this limitation is significant, the privacy-preserving RL generally outperforms the LID approach while achieving a reduction of the overhead of at least two orders of magnitudes.

The rest of the paper is structured as follows: in Section II we review several existing works, mainly relative to multi-domain VNE and privacy-preserving cooperation with ISPs; Section III provides the technical background to understand the proposed solution; in Section IV we formally define the problem and describe the privacy requirements of customer and ISPs; Section V presents the proposed RL algorithm; in Section VI we describe the building blocks of the privacy-preserving version of this algorithm, which is then presented in Section VII. Illustrative numerical results that validate our approach are shown in Section VIII. Finally, Section IX concludes the paper.

II. Related Work

The problem of VNE has been extensively studied [1], [5]. In particular, several heuristics have been proposed to solve it efficiently, as reviewed in [2]. To our knowledge, less attention has been devoted to VNE in multi-domain scenarios, in which ISPs do not expose sensitive information needed for the optimization. Two main classes of approaches have been proposed to solve the multi-domain VNE problem, i.e., distributed and centralized.

Examples of the former category are the works presented in [6], [7]. In [6], ISPs address privacy issues by exposing information only to other network operators they have a mutual agreement with. The main drawback of a distributed approach is that the optimization is not performed based on a global view of the overall network. On the other hand, existing centralized approaches generally divide the VNE problem into two sub-tasks: in the first, a VG is partitioned over the participants ISPs; in the second, each ISP performs the VNE of the received portion of the graph. The first sub-task is executed by a centralized entity, e.g., a customer or a broker acting on behalf of it [8], [9]. In these approaches, privacy issues are addressed as the first sub-task is executed based on the limited information about network infrastructures that the ISPs provide to the centralized entity. For example, in [4], only the peering links and the cost of embedding a given VN on a physical peering node are exposed. The main drawback of this approach is that this reduction of available information leads to a sub-optimal VNE solution. In our work, we propose a RL-based method that achieves better embedding performance with respect to such limited-information approaches, while guaranteeing total privacy.

In existing literature, a RL algorithm has already been proposed in the context of VNE over a multi-ISP network [10]. In such work, however, RL is executed by the ISPs to find the optimal embedding costs and not, as we do, to actually perform the VNE task. In general, RL has been extensively used as a tool to perform optimization in telecom networks, e.g., for QoS-driven network slicing [11], resource allocation in cloud [12] and traffic prediction [13]. A general-purpose multi-agent RL algorithm has then be proposed in [14]. This solution, however, does not fulfill the privacy requirements considered in our work and, for this reason, we do not employ it. Instead, we design a privacy-preserving RL algorithm able to process data encrypted under the SSS scheme, and we employ it to perform VNE over a multi-domain infrastructure. To the best of our knowledge, our work is the first attempt to solve the VNE problem over encrypted data and contributes to the literature on privacy-preserving strategies for cooperative service delivery (e.g., other secure multiple-party computations and SSS-based approaches have been used in the context of cooperative video content delivery in [15]–[17]).

III. Background

a) Reinforcement Learning: RL is a type of machine learning technique employed to learn a model of an initially-unknown environment $\mathcal{E}$, which describes the solution space of the problem that the RL aims to solve. The solution space is represented as a set of *states* $\mathcal{S}$ which is explored by an entity referred to as *agent*. The agent moves within the environment by performing *actions* and, based on that, it receives feedbacks (i.e., *rewards*). In the case of multiple environments

and/or multiple agents (as in our work), the approach is often referred to as *multi-agent* RL. The objective of a RL algorithm is to learn the best action to perform according to the state in which the agent is, i.e., the action that maximizes the overall received rewards.

b) Q - learning: Q-learning is a type of RL algorithm that models an environment as a matrix referred to as *Q-table*, whose sa-th entry represents to goodness of performing action a from state s. The model of the environment is learned by iteratively updating the Q-table as follows:

$$Q(s,a) \leftarrow Q(s,a) + lr \cdot \left(r + \gamma \cdot \max_{\hat{a}} Q(s,\hat{a}) - Q(s,a) \right) \tag{1}$$

where lr is the learning rate, r is the reward that the agent receives based on having performed action a from state s, and γ is the discount factor.

c) Shamir Secret Sharing: The SSS scheme [18] allows several parties to hold portions of a secret in such a way that secret reconstruction is made possible only by the cooperation of a sufficiently-large subset of them. Specifically, in a (σ, ψ) SSS scheme, the secret is divided into σ *shares* and can be reconstructed only if such subset is composed of at least ψ parties. In SSS, secret s and the corresponding set of shares $[\![s]\!]$ are defined in $\mathbb{Z}_q$, where q is a prime number greater than all the possible secrets.

d) Heuristics for VNE: We consider two existing heuristics approaches to solve the multi-domain VNE problem, namely the Limited Information Disclosure (LID) and the Full Information Disclosure (FID). These heuristics have been proposed in [4] and are based on a relaxed linear programming formulation. LID is executed in two main phases: in the first, portions of a virtual graph are assigned to the ISPs by a centralized entity (e.g., a customer) that has a limited view of the multi-domain infrastructure. In the second, each ISP performs the optimal deployment of the received sub-graph within its network. FID is more privacy intrusive than LID, as the centralized entity performs the optimization based on a full view of the underlying infrastructure. A deeper description of both heuristics is provided in [4].

IV. Problem Statement

A. Problem Statement and Motivation

The formal statement of the VNE problem is the following:

$$\min \sum_{u \in \mathcal{M}} \sum_{i \in \mathcal{VN}} w_u^i d_i c_u^i x_u^i + \sum_{\substack{(i,j) \\ i \neq j}} \sum_{(u,v) \in \mathcal{L}} y_{uv}^{ij} d_{ij} c_{uv} \tag{2}$$

subject to:

$$\sum_{u \in \mathcal{M}} x_u^i = 1 \quad , \forall i \in \mathcal{VN} \tag{3}$$

$$\sum_{v \in \mathcal{M}} y_{uv}^{ij} - \sum_{v \in \mathcal{M}} y_{vu}^{ij} = x_u^i - x_u^j, \forall (i,j) \in \mathcal{VP}, \forall (u,v) \in \mathcal{L} \tag{4}$$

$$\sum_{i \in \mathcal{VN}} d_i x_u^i \leq \zeta_u^{(nodes)} \quad , \forall u \in \mathcal{M} \tag{5}$$

$$\sum_{(i,j) \in \mathcal{VP}} y_{uv}^{ij} d_{ij} \leq \zeta_{uv}^{(links)} \quad , \forall (u,v) \in \mathcal{L} \tag{6}$$

where $\mathcal{M}$ and $\mathcal{L}$ are the sets of physical nodes and links, respectively. u and v are the indexes of generic substrate nodes $\in \mathcal{M}$, while (u,v) indicates the link $\in \mathcal{L}$ having u and v as end-points. During the rest of the paper, we may indicate a generic link also as l. $\mathcal{VN}$ is the set of virtual nodes, d_i is the computational requirement of the generic VN_i, $w_u^i \in \{1, \infty\}$ is a variable indicating the feasibility of embedding VN_i into node u, c_u^i is the cost of embedding a computational unit of VN_i in node u and $x_u^i \in \{0,1\}$ is the corresponding decision variable. $\mathcal{VP}$ is the set of virtual paths, d_{ij} is the bandwidth requirement of VP_{ij}, c_{uv} is the cost of embedding a unit of bandwidth on the link connecting nodes u and v and $y_{uv}^{ij} \in \{0,1\}$ is the decision variable corresponding to the embedding of VP_{ij} in link uv. Eqs. 3, 4, 5, 6 prescribe that each VN is embedded onto exactly one physical node, the requirement of flow consistency, the fulfillment of node capacity and link capacity constraints, respectively (being $\zeta_u^{(nodes)}$ the capacity of node u and $\zeta_{uv}^{(links)}$ the capacity of link uv).

The main motivation of this work is to minimize the embedding cost while guaranteeing the fulfilment of stringent privacy requirements to both customer and ISPs. Whilst the minimization of the cost is a plausible objective for the customer, the same cannot be said, in general, for the ISPs, whose goal is to increase their revenues. As explained in [4], however, in a multi-ISP domain the customer only pays the cost of the embedding performed based on the information that ISPs expose. Indeed, a limited exposure of information induces an *extra cost* that ISPs pay to actually embed the received portion of virtual graph in their networks. Hence, we assume that both customer and ISPs are interested to minimize the overall embedding cost.

B. Privacy Requirements and Security Models

We consider a customer and K ISPs, whose objectives and privacy requirements are discussed in the following.

1) Customer: The customer aims to deploy a VG over the multi-domain infrastructure. The computational demands of the N VNs and the bandwidth requirements of the relative VPs are represented as a vector $\vec{d}$ and a matrix $\mathbf{D}$, respectively. Moreover, the types of VNs are represented as a binary matrix Δ with N rows and a number of columns equal to the number of available VNs' types (e.g., a virtual firewall).

*a) **Privacy Requirements**:* computational demand of VN_i (i.e., d_i) and its type (i.e., δ_i) can only be disclosed to the ISP that hosts $VN_i, \forall i$ and the bandwidth requirement d_{ij} only to the ISPs that are traversed by $VP_{ij}, \forall ij$.

2) Internet Service Providers: The generic ISP_k owns a physical infrastructure composed of a set of $\mathcal{M}_k$ nodes which are interconnected by $\mathcal{L}_k$ links. Each node (resp., link) has a computational (resp., bandwidth) capacity, which are encoded

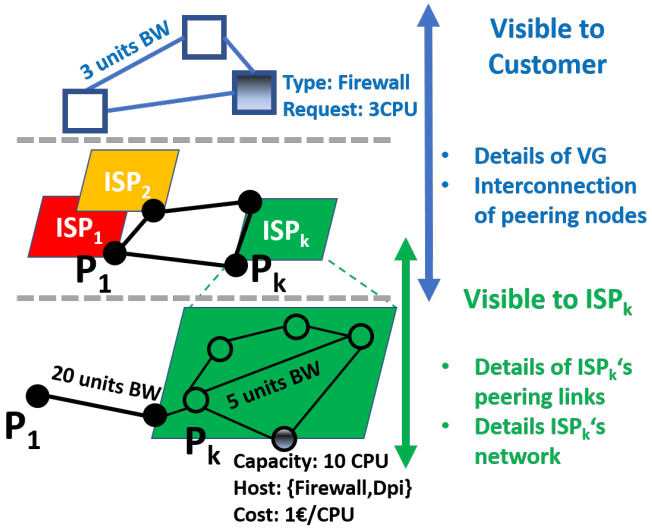


Fig. 1: Overview of the information visible to ISPs and Customer

in vectors $\zeta_k^{(nodes)}$ and $\zeta_k^{(links)}$, respectively. Moreover, the uf-th entries of matrices $\mathbf{F}_k$ and $\boldsymbol{\eta}_k$ indicate if node u can host a VN of type f and the cost of hosting it, respectively. Vector $\mathbf{C}^{(links)}$ indicates the cost of embedding a unit of bandwidth in each link.

*a) **Privacy Requirements**:* the following information can only be known to the owner ISP: (i) capacity $\vec{\zeta}^{(nodes)}$, embedding costs $\boldsymbol{\eta}$ and feasibility $\mathbf{F}$ of the physical nodes; (ii) capacity $\vec{\zeta}^{(links)}$ and cost $\mathbf{C}^{(links)}$ of the physical links; (iii) interconnection of the internal nodes, i.e., the information if two generic nodes u, v are connected by a link. On the other hand, the interconnection of the peering nodes is assumed to be known to all the participants.

In Fig. 1, we show an overview of the information visible to the considered entities. In this figure, it is possible to identify three main layers. Going bottom-up, the first layer shows the information visible only to the owner ISPs (e.g., the topology of their networks and the cost of embedding a type of VN into their nodes); the intermediate layer represents the peering interconnection, which is visible to all the participants; the top layer represents the information about the virtual graph, which are known to the customer only.

Both the customer and the ISPs are modeled as *honest-but-curious* entities, who do not deviate from the licit execution of the protocol, but try to obtain as much information as possible from it (e.g., the ISPs may be interested in the infrastructure details of their competitors). We justify this assumption in subsection VII-E, where we also elaborate on how the proposed approach fulfills the aforementioned privacy requirements.

V. The RL Algorithm for multi-domain VNE

In this Section we describe our RL-based approach to solve the VNE problem in a multi-ISPs scenario. Initially, we identify four sub-tasks in which the problem can be divided, i.e., selection of (i) the ISPs that host the VNs, (ii) the peering links that the VPs traverse, (iii) the physical nodes that embed the VNs and (iv) the intra-ISP links that embed the VPs. Note that the decisions taken in sub-task (iii) are conditioned by the output of sub-task (i). For instance, a VN can be embedded into a physical node only if that node belongs to the infrastructure of the ISP to which the VN has been assigned in task (i).

A. Environments

In this subsection, we define four types of RL environments that model the execution of the aforementioned sub-tasks. Each environment $\mathcal{E}$ is characterized by its state vector $\mathcal{S}$ and its $\mathcal{Q}$ matrix. $\mathcal{S}$ is a binary vector with a number of components equal to the number of states (where the only component equal to 1 is the state currently occupied by the agent), while $\mathcal{Q}$ has a row for each state and 3 columns, corresponding to the actions that the agent can perform, i.e., *left*, *stay* and *right*. As an example, if an agent is in state $[0,1,0,0]$ and it performs the action right, the new state becomes $[0,0,1,0]$. An action is represented as a binary vector $\vec{\alpha}$ (e.g., $\vec{\alpha} = [0,0,1]$ for action right). The proposed environments are the following:

- An environment $\mathcal{E}_{CUST}^{(i)}$ that models the selection of the ISP in which VN_i has to be embedded, $\forall i$. $\mathcal{S}_{CUST}^{(i)}$ is a vector with K components, one for each ISP to which VN_i can be assigned.
- An environment $\mathcal{E}_{ISP_k}^{(i)}$ that models the selection of the physical node in which VN_i has to be embedded, $\forall i, k$. $\mathcal{S}_{ISP_k}^{(i)}$ is a vector with $|\mathcal{M}_k|$ components, where $\mathcal{M}_k$ is the set of physical nodes $\in ISP_k$.
- An environment $\mathcal{E}_{CUST}^{(kk\prime,ij)}$ that models the selection of the path of peering nodes connecting ISP_k and $ISP_{k\prime}$ in which VP_{ij} has to be embedded, $\forall k, k\prime, i, j$. $\mathcal{S}_{CUST}^{(kk\prime,ij)}$ is a vector with $|\mathcal{P}_{kk\prime}|$ components, where $|\mathcal{P}_{kk\prime}|$ is the number of peering paths between ISP_k and $ISP_{k\prime}$.
- An environment $\mathcal{E}_{ISP_k}^{(uv,ij)}$ that models the selection of the path (between nodes u and $v \in ISP_k$) to embed VP_{ij}, $\forall u, v, i, j$. $\mathcal{S}_{ISP_k}^{(uv,ij)}$ is a vector with $|\mathcal{P}_{uv}|$ components, one for each path that interconnects nodes u and v. As this number may be very high, practically we can consider the $|\mathcal{P}_{uv}|$ shortest paths between u and v.

We propose in Tab. I an overview of the notations used in this paper. Then, in the following two subsections we describe the main operations performed in the proposed RL approach. Notice that such operations are carried out in a distributed fashion by Customer and ISPs. We use the subscripts $Cust$ and $ISP_k, \forall k$ to indicate the participant that executes a given operation (e.g., operations in environment $\mathcal{E}_{ISP_k}^{(i)}, \forall i$ are executed by ISP_k). For the ease of explanation, we provide also a pseudo-code in Algorithm 1, in which the operations

TABLE I: Table of Notations

Variable	Description	Variable	Description
N	Number of VNs	K	Number of ISPs
$\vec{\mathbf{d}}, \mathbf{D}$	Computational (resp., bandwidth) demand of the VNs (resp., VPs)	$\mathbf{P}_k^{uv}$, $\mathcal{P}_k^{uv}$	Matrix representing the (resp., Set of) paths connecting nodes u and $v \in ISP_k$
$\mathbf{W}$	Feasibility matrix (ui-th entry is 1 if node u can embed $\in VN_i$ and ∞ otherwise)	$\mathbf{F}_k$	Matrix whose uf-th entry is 1 if node $u \in ISP_k$ can embed VN of type f (and 0 otherwise)
$\mathbf{C}_k^{(nodes)}$	Matrix of Nodes' costs (the ui-th entry is the cost of embedding a computational unit of VN_i in node $u \in ISP_k$)	$\boldsymbol{\eta}_k$	Matrix whose uf-th entry is the cost of embedding a computational unit of VN of type f in node $u \in ISP_k$
$\vec{\zeta}_k^{(nodes)}$	Computational Capacity of nodes $\in ISP_k$	$\boldsymbol{\Delta}$	VNs' types Matrix (th if-th entry is 1 if VN_i is of type f, and 0 otherwise)
$\mathcal{M}_k, \mathcal{L}_k$	Set of physical nodes (resp, links) $\in ISP_k$	$\vec{\zeta}_k^{(links)}, \vec{c}_k^{(links)}$	Link capacity (resp., cost) vector indicating the capacity (resp., cost of embedding a unit of bandwidth) for links $\in ISP_k$
l	Index of the generic link l	u, U	Index of the generic internal (resp., peering) node
$\mathcal{E}_{CUST}^{(i)}, \mathcal{S}_{CUST}^{(i)}$ $\vec{\alpha}_{CUST}^{(i)}, r_{CUST}^{(i)}$	Environment, state vector, action vector and reward associated with the selection of the ISP that embeds $VN_i, \forall i$	$\mathcal{E}_{CUST}^{(kk\prime,ij)}, \mathcal{S}_{CUST}^{(kk\prime,ij)}$ $\vec{\alpha}_{CUST}^{(kk\prime,ij)}, r^{(kk\prime,ij)}$	Environment, state vector, action vector and reward associated with the selection of the peering path between ISP_k and $ISP_{k\prime}$ that embeds $VP_{ij}, \forall kk\prime, ij$
$\mathcal{E}_{ISP_k}^{(i)}, \mathcal{S}_{ISP_k}^{(i)}$ $\vec{\alpha}_{ISP_k}^{(i)}, r_{ISP_k}^{(i)}$	Environment, state vector, action vector and reward associated with the selection of the ISP that embeds $VN_i, \forall i$	$\mathcal{E}_{ISP_k}^{(uv,ij)}, \mathcal{S}_{CUST}^{(uv,ij)}$ $\vec{\alpha}_{ISP_k}^{(uv,ij)}, r_{CUST}^{(uv,ij)}$	Environment, state vector, action vector and reward associated with the selection of the path between nodes u and $v \in ISP_k$ that embeds $VP_{ij}, \forall uv, ij$

performed by each ISP are further highlighted by means of curly brackets. All the remaining operations presented in Algorithm 1 are executed by the Customer.

B. Action Selection and State Updating

After the initialization of the main variables (e.g., vectors states and $\mathcal{Q}$-tables), an action is performed in environment $\mathcal{E}_{CUST}^{(i)}$ and state vector $\mathcal{S}_{CUST}^{(i)}$ is changed accordingly. This operation is aimed to find the ISP in which VN_i has to be embedded, and it is repeated $\forall i$. Let us assume that VN_i has been assigned to ISP_k. An action is then performed in the environment $\mathcal{E}_{ISP_k}^{(i)}$ and the corresponding state is changed accordingly to select the physical node in which VN_i has to be embedded.

We now consider the operations relative to the selection of the peering paths on which VP_{ij} has to be embedded. Assuming that ISP_k and $ISP_{k\prime}$ are the ISPs that, at the current iteration of the RL algorithm, are required to embed VN_i and VN_j, respectively, the considered environment is $\mathcal{E}_{CUST}^{(kk\prime,ij)}$. An action selection and successive state updating are performed in this environment to select the peering path traversed by $VP_{ij}, \forall i, j$. The selected peering path is composed of a set of peering nodes (e.g., $U_k, ..., U_{k\prime}$). As mentioned in subsection IV-B2, all the involved participants are aware of the peering nodes that interconnect the ISPs (while the cost of traversing them is only known to the owner ISPs).

Once VN_i and VN_j have been assigned to their physical nodes and the peering links traversed by VP_{ij} has been chosen, operations are executed to embed VP_{ij} within the intra-ISP physical links consistently with the decisions that have been previously taken. Specifically, the flow consistency constraint of Eq. 4 must be fulfilled. To this end, four different scenarios must be considered: (i) ISP_k is assigned both VN_i (which is embedded in node u) and VN_j (which is embedded in node v). In this case, VP_{ij} has to be embedded within a path between nodes $u, v \in ISP_k$. To do so, action selection and state update are performed within the environment $\mathcal{E}_{ISP_k}^{(uv,ij)}$; (ii) ISP_k is assigned only VN_i (which is embedded in node u) and operations are executed in $\mathcal{E}_{ISP_k}^{(uU_k,ij)}$, where U_k is assumed to be the first peering node of the peering path that was selected in environment $\mathcal{E}_{CUST}^{(kk\prime,ij)}$; (iii) ISP_k is assigned only VN_j (which is embedded in node v) and operations are performed in the environment $\mathcal{E}_{ISP_{k\prime}}^{(U_{k\prime}v,ij)}$, where $U_{k\prime}$ is the last node of such peering path; (iv) ISP_k is assigned neither one of the two VNs and operations are performed in the environment $\mathcal{E}_{ISP_{k*}}^{(U_xU_{x+1},ij)}$, where U_x and U_{x+1} are two adjacent peering nodes of the considered peering path, which are assumed to belong to the infrastructure of ISP_{k*}.

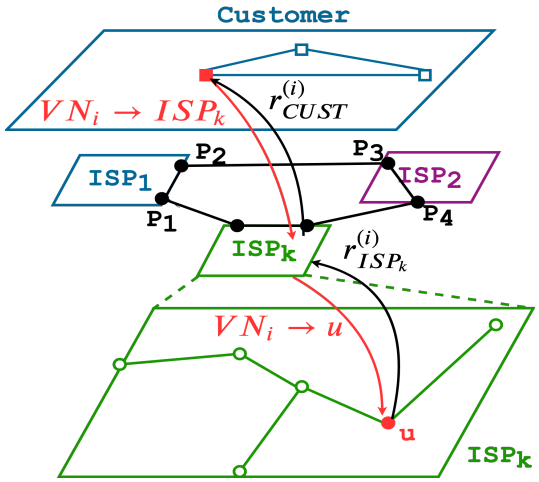


Fig. 2: High-Level representation of the reward exchange process performed by the proposed RL algorithm (being $r_{ISP_k}^{(i)}$ the reward associated with the embedding of VN_i into ISP_k)

To efficiently explore the solution space, each agent receives a reward providing a feedback on the goodness of the performed action. In our approach, actions are executed in

Algorithm 1 Reinforcement Learning Algorithm for Multi-Domain Virtual Network Embedding

Input: $Patience, \quad \mathcal{Q}^{(i)}_{CUST}, \mathcal{S}^{(i)}_{CUST}, \mathcal{Q}^{kk\prime,ij}_{CUST}, \mathcal{S}^{kk\prime,ij}_{CUST}, \mathcal{Q}^{(i)}_{ISP_k}, \mathcal{S}^{(i)}_{CUST}, \mathcal{Q}^{uv,ij}_{ISP_k}, \mathcal{S}^{uv,ij}_{ISP_k}, \forall i, \forall uv, ij, i \neq j, \forall k, \forall k\prime$

Output: $\mathcal{S}^{(i)}_{CUST}, \mathcal{S}^{kk\prime,ij}_{CUST}, \mathcal{S}^{(i)}_{CUST}, \mathcal{S}^{uv,ij}_{ISP_k}, \forall i, \forall uv, ij, i \neq j, \forall k, \forall k\prime$

Variable Initialization $Cost_{final} = \infty, epoch = 0, \quad N^{unimproved}_{epochs} = 0$
while $N^{unimproved}_{epochs} \leq Patience$ **do**
  if $epoch \equiv 0 \mod \mathcal{T}^{VF}_{CUST}$ **then**
    for $1 \leq i \leq N$ **do**
      $\vec{\alpha}^{(i)}_{CUST} \leftarrow Action\left(\mathcal{S}^{(i)}_{CUST}, \mathcal{Q}^{(i)}_{CUST}\right)$
      $\mathcal{S}^{(i)}_{CUST} \leftarrow UpdateState\left(\mathcal{S}^{(i)}_{CUST}, \vec{\alpha}^{(i)}_{CUST}\right)$
      $r^{(i)}_{CUST} \leftarrow GetReward\left(\mathcal{S}^{(i)}_{CUST}, \vec{\alpha}^{(i)}_{CUST}\right)$
    end for
  end if
  if $epoch \equiv 0 \mod \mathcal{T}^{VP}_{CUST}$ **then**
    for $1 \leq i, j \leq N, i \neq j$ **do**
      $k \leftarrow GetState\left(\mathcal{S}^{(i)}_{CUST}\right)$
      $k\prime \leftarrow GetState\left(\mathcal{S}^{(j)}_{CUST}\right)$
      $\vec{\alpha}^{VP_{kk\prime,ij}}_{CUST} \leftarrow Action\left(\mathcal{S}^{kk\prime,ij}_{CUST}, \mathcal{Q}^{(kk\prime,ij)}_{CUST}\right)$
      $\mathcal{S}^{(kk\prime,ij)}_{CUST} \leftarrow UpdateState\left(\mathcal{S}^{(kk\prime,ij)}_{CUST}, \vec{\alpha}^{(kk\prime,ij)}_{CUST}\right)$
      $r^{(kk\prime,ij)}_{CUST} \leftarrow GetReward\left(\mathcal{S}^{(kk\prime,ij)}_{CUST}, \vec{\alpha}^{(kk\prime,ij)}_{CUST}\right)$
    end for
  end if
  if $epoch \equiv 0 \mod \mathcal{T}^{VF}_{ISP}$ **then**
    for $1 \leq i \leq N$ **do**
      $k \leftarrow GetState\left(\mathcal{S}^{(i)}_{CUST}\right)$
      $\vec{\alpha}^{(i)}_{ISP_k} \leftarrow Action\left(\mathcal{S}^{(i)}_{ISP_k}, \mathcal{Q}^{(i)}_{ISP_k}\right)$
      $\mathcal{S}^{(i)}_{ISP_k} \leftarrow UpdateState\left(\mathcal{S}^{(i)}_{ISP_k}, \vec{\alpha}^{(i)}_{ISP_k}\right)$ } ISP_k
      $r^{(i)}_{ISP_k} \leftarrow GetReward\left(\mathcal{S}^{(i)}_{ISP_k}, \vec{\alpha}^{(i)}_{ISP_k}\right)$
    end for
  end if
  for $1 \leq i, j \leq N, i \neq j$ **do**
    $k \leftarrow GetState\left(\mathcal{S}^{(i)}_{CUST}\right)$
    $k\prime \leftarrow GetState\left(\mathcal{S}^{(j)}_{CUST}\right)$
    $Path^{(kk\prime,ij)}_{peering} \leftarrow GetPeeringPath\left(\mathcal{S}^{(kk\prime,ij)}_{ISP_{CUST}}\right)$
    for $Link_{peering} \in Path_{peering}$ **do**
      $u, v \leftarrow EndPoints\left(Link_{peering}\right)$
      $k \leftarrow OwnerISP\left(u, v\right)$
      $\vec{\alpha}^{(uv,ij)}_{ISP_k} \leftarrow Action\left(\mathcal{S}^{(uv,ij)}_{ISP_k}, \mathcal{Q}^{(uv,ij)}_{CUST}\right)$
      $S^{(uv,ij)}_{ISP_k} \leftarrow UpdateState\left(\mathcal{S}^{(uv,ij)}_{ISP_k}, \vec{\alpha}^{(uv,ij)}_{ISP_k}\right)$ } ISP_k
      $r^{(uv,ij)}_{ISP_k} \leftarrow GetReward\left(\mathcal{S}^{(uv,ij)}_{ISP_k}, \vec{alpha}^{(uv,ij)}_{ISP_k}\right)$ } ISP_k
      $\mathcal{Q}^{(uv,ij)}_{ISP_k} \leftarrow UpdateQ\left(\mathcal{Q}^{(uv,ij)}_{ISP_k}, \vec{alpha}^{(uv,ij)}_{ISP_k}\right)$
    end for
  end for
  if $epoch \geq 1 \ \& \ epoch \equiv 0 \mod \mathcal{T}^{VF}_{ISP} - 1$ **then**
    for $1 \leq i \leq N$ **do**
      $k \leftarrow GetState\left(\mathcal{S}^{(i)}_{CUST}\right)$
      $r^{(i)}_{ISP_k} \leftarrow r^{(i)}_{ISP_k} + \sum_{u,v} r^{(uv,ij)}_{ISP_k}$
      $\mathcal{Q}^{(i)}_{ISP_k} \leftarrow UpdateQ\left(\mathcal{Q}^{(i)}_{ISP_k}, \vec{alpha}^{(i)}_{ISP_k}\right)$ } ISP_k
    end for
  end if
  if $epoch \geq 1 \ \& \ epoch \equiv 0 \mod \mathcal{T}^{VP}_{CUST} - 1$ **then**
    for $1 \leq i, j \leq N, i \neq j$ **do**
      $k \leftarrow GetState\left(\mathcal{S}^{(i)}_{CUST}\right)$
      $k\prime \leftarrow GetState\left(\mathcal{S}^{(j)}_{CUST}\right)$
      $r^{(kk\prime,ij)}_{CUST} \leftarrow r^{(kk\prime,ij)}_{CUST} + \sum_k \sum_{u,v} r^{(uv,ij)}_{ISP_k}$
      $\mathcal{Q}^{(kk\prime,ij)}_{CUST} \leftarrow UpdateQ\left(\mathcal{Q}^{(kk\prime,ij)}_{CUST}, r^{(kk\prime,ij)}_{CUST}\right)$
    end for
  end if
  if $epoch \geq 1 \ \& \ epoch \equiv 0 \mod \mathcal{T}^{VN}_{CUST} - 1$ **then**
    for $1 \leq i \leq N$ **do**
      $\mathcal{Q}^{(i)}_{CUST} \leftarrow UpdateQ\left(\mathcal{Q}^{(i)}_{CUST}, r^{(i)}_{CUST}\right)$
    end for
  end if
  $Current_{cost} \leftarrow ComputeCost\left(\mathcal{S}_{CUST}, \mathcal{S}_{ISP_k} \forall k\right)$
  if $Current_{cost} \leq Final_{cost}$ **then**
    $Final_{Cost} = Current_{cost}$
    $N^{unimproved}_{epochs} = 0$
  else
    $N^{unimproved}_{epochs} \leftarrow N^{unimproved}_{epochs} + 1$
  end if
  $epoch \leftarrow epoch + 1$
  if $N^{unimproved}_{epochs} \geq Patience$ **then**
    break
  end if
end while

return $\mathcal{S}^{(i)}_{CUST}, \mathcal{S}^{kk\prime,ij}_{CUST}, \mathcal{S}^{(i)}_{CUST}, \mathcal{S}^{uv,ij}_{ISP_k}, \forall i, \forall uv, ij, i \neq j, \forall k, \forall k\prime$

isolated environments. To make all the agents behave towards a common optimization objective, it is crucial to carefully craft the rewards and to properly exchange them between different environments. In the next subsection, we describe the proposed reward signals, and we illustrate a high-level representation of their exchange in Fig. 2.

C. Rewards Computation and Q-table Updating

1) Embedding a VN into a physical node: The reward associated with $\mathcal{E}^{(i)}_{ISP_k}$ is defined as $r^{(i)}_{ISP_k} = -d_i \cdot c^i_u - v_u - w^i_u$, where d_i is the computational demand of VN_i and c^i_u is the cost of embedding VN_i in node u; $v_u \in \{0, \infty\}$ indicates if the node capacity constraint is fulfilled; $w^i_u \in \{1, \infty\}$ indicates if node u is eligible to host VN_i.

2) Embedding a VP into an intra-ISP physical path: The reward associated with $\mathcal{E}^{(uv,ij)}_{ISP_k}$ is defined as $r^{(uv,ij)}_{ISP_k} = -\sum_{l \in \mathcal{P}^{ij}_{uv}} (d_{ij} c_l + v_l) + r^{(i)}_{ISP_k} + r^{(j)}_{ISP_k}$, where l is the generic link belonging to the path $\mathcal{P}^{ij}_{uv}$ connecting nodes u and v and traversed by VP_{ij}; c_l is the cost of embedding a unit of bandwidth in link l and $v_l \in \{0, \infty\}$ is a penalty value that describes the fulfillment of the link capacity constraint. $r^{(i)}_{ISP_k}$ and $r^{(j)}_{ISP_k}$ are the rewards associated with the embedding of VN_i and VN_j (i.e., the end-points of the considered VP), which have been described in the previous subsection.

3) Assigning a VN to an ISP: The reward associated with $\mathcal{E}^{(i)}_{CUST}$ is defined as $r^{(i)}_{CUST} = \sum_{k=1}^{K} \sum_{j=1}^{N} r^{(j)}_{ISP_k}$. This reward is the same for all the VNs, i.e., $\forall i$, to provide a feedback that considers the current embedding of all the VNs.

4) Assigning a VP to a peering path: The reward associated with $\mathcal{E}^{(kk\prime,ij)}_{CUST}$ is defined as $r^{(kk\prime,ij)}_{CUST} = \sum_{(u,v)} \sum_{k=1}^{K} r^{uv,ij}_{ISP_k}$. This reward is the summation of the single rewards relative to the embedding of the VP_{ij} within all the K ISPs.

Once the reward relative to an environment has been computed, the corresponding $\mathcal{Q}$-table is updated according to Eq. 1. Since the considered sub-tasks can be solved at different time scales (e.g., the selection of the ISP in which a VN is embedded can be performed less frequently than the selection of the embedding node), we define the periods of execution of such sub-tasks (measured in number of RL iteration) as $\mathcal{T}^{VN}_{CUST}, \mathcal{T}^{VP}_{CUST}, \mathcal{T}^{VN}_{ISP}, \mathcal{T}^{VP}_{ISP}$. At every iteration of the RL algorithm, the current embedding cost is computed based on Eq. 2 and all the described operations are repeated until no improvement to the embedding cost is observed for a number of iterations equal to $patience$.

As it may be noticed, we design rewards that are strictly related to the embedding costs. Therefore, from their exchange, the participants obtain information about other parties' data that, if properly analyzed, can be used to violate the privacy requirements in subsection VII-E. To address this issue, we propose a privacy-preserving version of the RL approach, which is built on the elements that we describe in the following Section.

TABLE II: Data Overhead

Operation	Bits exchanged between each pair of parties	
Name	Online	Offline
Random	0	B
Mult	B	$4B$
MultDec	$(m+1) \cdot B$	$m \cdot B$
EQ	B^2	$6B^2$
GE	$9B^2$	0

VI. Building Blocks for Privacy-Preserving RL

A. Representation of Data Suitable for Secure Computation

As presented in subsection VII-E, data owned by the customer and the ISPs are arranged in vector/matrix form (e.g., vector $\vec{d}$ to represent computational requirements). Each element of these vectors and matrices can be distributed among the participants as a set of shares, thus allowing secure computation on them. In addition, we represent the paths connecting two generic nodes u and v as a matrix $\mathbf{P}_{uv}$, with a number of rows equal to the number of paths connecting nodes u and v (i.e., $|\mathcal{P}^{uv}_k|$) and a number of columns equal to the total number of links of the infrastructure (i.e., $|\mathcal{L}_k|$). Each path can be represented as a binary vector, whose l-th element is 1 if the path contains the l-th link of the ISP's infrastructure. An example of this representation is shown in the following:

$$\begin{array}{c} \\ Path_1 \\ \cdots \\ Path_{|\mathcal{P}^{uv}_k|} \end{array} \begin{array}{c} \begin{array}{ccccc} Link_1 & \cdots & Link_l & \cdots & Link_{|\mathcal{L}_k|} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

and it is defined for each possible pair of nodes, i.e., $\forall u, v, u \neq v$.

B. Existing Privacy-Preserving Primitives

In this subsection, we describe several existing operators and we show in Table II the amount of data that each pair of participants need to exchange to execute them (being B the bit-length of the shares). By *on-line*, we refer to a data exchange done contextually to the execution of the operation and cannot be performed in advance (as in the *off-line* case).

1) Secure generation of the shares of a random number: By executing **Random**, a share $[\![rv]\!]$ of a random variable is learnt by each participant (none of which knows the secret rv). We employ the implementation described in [19].

2) Secure multiplication: **Mult** takes in input the shares $[\![x]\!], [\![y]\!]$ and returns $[\![z]\!]$, where $z = x \cdot y$. We employ the protocol presented in [20].

3) Secure multiplication with a decimal number: **MultDec** takes in input a share $[\![x]\!]$ and a plain decimal value λ, and returns the share of their product $[\![\lambda \cdot x]\!]$. We realize this sub-routine based on the protocol presented in [21].

4) Secure equality test and greater-or-equal: **EQ** (resp., **GE**) takes in input the shares $[\![x]\!]$ and $[\![y]\!]$ and returns the share $[\![b_{eq}]\!]$ (resp., $[\![b_{ge}]\!]$), where $b_{eq} = 1$ (resp., $b_{ge} = 1$) iff $x = y$ (resp., $x \geq y$) and 0 otherwise. In this study, we employ the implementations of EQ and GE described in [19].

In the following subsection, we describe the new operators that we built based on the aforementioned existing ones and that are used to develop our privacy-preserving RL algorithm.

C. New Privacy-Preserving Operators

1) Secure computation of the maximum element of a vector and corresponding index: **Max** (resp., **ArgMax**) takes as input a vector of shares $\vec{x} = [[\![x_1]\!], [\![x_2]\!], ..., [\![x_\Phi]\!]]$ and returns $[\![\max(\vec{x})]\!]$ (resp., $[\![i^*]\!] = [\![\arg\max_i(\vec{x})]\!]$). This module is based on the recursive application of the GE operator. Details are omitted due to space constraints.

2) Secure update of the states vector: **UpdateState** takes as input the current vector state $\vec{s} = [[\![s_1]\!], [\![s_2]\!], ..., [\![s_\Phi]\!]]$ and outputs the new state $\vec{s\prime}$, according to the action executed by the agent (e.g., $[\![left]\!]$ = $[\![1]\!]$, $[\![stay]\!]$ = $[\![0]\!]$ and $[\![right]\!]$ = $[\![0]\!]$ if the agent chooses the action left). The β-th component of the updated state is derived as $[\![s'_\beta]\!] = [\![s_{\beta-1}]\!] \cdot [\![right]\!] + [\![s_\beta]\!] \cdot [\![stay]\!] + [\![s_{\beta+1}]\!] \cdot [\![left]\!]$.

a) Masked Secure Update of the states vector: The **MaskedUpdateState** is employed in case the participants are not aware of the vector state to modify and only know $[\![mask]\!]$, where the binary value $mask$ is 1 if the considered state has to be updated. This subroutine updates the output of UpdateState as $[\![s'_\beta]\!] \leftarrow [\![mask]\!] \cdot [\![s'_\beta]\!] + (1 - [\![mask]\!]) \cdot [\![s_\beta]\!]$.

3) Secure Selection of the row of a matrix: **RowSelection** takes as input a matrix $\vec{\mathcal{M}} \in \mathbb{Z}_q^{\Phi X \Omega}$ and a vector $\vec{x} = [[\![x_1]\!], [\![x_2]\!], ..., [\![x_\Phi]\!]]$. All the components of $\vec{x}$ are $[\![0]\!]$, except the one corresponding to the row to select that is $[\![1]\!]$ (say $[\![x_{\beta*}]\!] = [\![1]\!]$). RowSelection returns a vector $\vec{\mathcal{M}}_{\beta*}$ corresponding to the selected row. The χ-th component of this vector is given by $[\![\mathcal{M}_{\beta*}(\chi)]\!] = \sum_{\beta=1}^{\Phi} [\![\vec{\mathcal{M}}(\beta, \chi)]\!] \cdot [\![x_\beta]\!]$.

4) Secure Action Selection: **SelectAction** takes as input a $\mathcal{Q}$-table matrix $\mathcal{Q} \in \mathbb{Z}_q^{\Phi X 3}$, a decimal number $\upsilon \in [0, 1]$ and a vector representing the current state of an agent (i.e., $\vec{s} = [[\![s_1]\!], [\![s_2]\!], ..., [\![s_\Phi]\!]]$). This subroutine returns the action α that the agent should perform as $[[\![\textit{left}]\!], [\![\textit{stay}]\!], [\![\textit{right}]\!]]$ (where only one component is $[\![1]\!]$ and the others are $[\![0]\!]$). Notice that υ (resp., $1 - \upsilon$) is the probability that the action is chosen to perform *exploitation* (resp., *exploration*). The two approaches require different operations, that we omit due to space limitation.

5) Secure Computation of VN Embedding Cost: **CostEmbeddingVN** takes as input the state vector $S_{ISP_k}^{(i)}$, the cost vector $\vec{c}^{(nodes)}$ and the computational demand of VN_i, i.e., $[\![d_i]\!]$ and returns $[\![d_i \cdot c_u^{(nodes)}]\!]$ corresponding to the u-th node in which VN_i is embedded. An element-wise secure multiplication of vectors $S_{ISP_k}^{(i)}$ and $\vec{c}^{(nodes)}$ is executed. The obtained products are then summed up, and the result is securely multiplied with $[\![d_i]\!]$ to get $[\![d_i \cdot c_u^{(nodes)}]\!]$.

6) Secure Computation of VP Embedding Cost: **CostEmbeddingVP** takes as input the state vector $\mathcal{S}_{ISP_k}^{uv,ij}$, which encodes the physical path connecting nodes u and v (belonging to ISP_k) that is currently traversed by VP_{ij}, the matrix $\mathbf{P}_k^{uv}$, which encodes all the paths connecting nodes u and v, as described in Subsection VI-A, the cost of traversing the links of ISP_k, i.e., $\vec{c}_k^{(links)}$ and the amount of traffic exchaged between VN_i and VN_j, i.e., $[\![d_{ij}]\!]$. The subroutine returns $\sum_{l \in \mathcal{P}_{uv}^{ij}} [\![d_{ij} \cdot c_l^{links}]\!]$, i.e., the cost of embedding VP_{ij} in its current physical path. The RowSelection operator is applied to matrix $\mathbf{P}_k^{uv}$ and to state vector $\mathcal{S}_{ISP_k}^{uv,ij}$ to select the path belonging to ISP_k traversed by VP_{ij}. Then, Mult is used to perform a secure element-wise multiplication of $\vec{c}^{(nodes)}$ and the selected row. All the elements of the obtained vector are summed up and the result is securely multiplied by $[\![d_{ij}]\!]$ using the Mult subroutine to obtain $\sum_{l \in \mathcal{P}_{uv}^{ij}} [\![d_{ij} \cdot c_l^{links}]\!]$.

7) Secure Node's Embedding Feasibility and Cost: **NodeFeasibility** takes as input the matrices $\vec{F}$ and $\vec{\Delta}$. The uf-th element of $\vec{F}$ and the if-th element of $\vec{\Delta}$ are $[\![1]\!]$ if a VN of type f can be hosted in node u and if VN_i is of type f, (and $[\![0]\!]$ otherwise). This subroutine returns a matrix $\vec{W}$, whose ui-th element is $[\![0]\!]$ in case VN_i can be hosted in node u and $[\![\infty]\!]$[1] otherwise. The u-th row of matrix $\vec{F}$ and the i-th row of matrix $\vec{\Delta}$ are multiplied element-wise using the Mult operator. Resulting products are then summed up to obtain $[\![w_u^i]\!]$, which is $[\![1]\!]$ in case node u is eligible to host VN_i (and $[\![0]\!]$ otherwise). $[\![w_u^i]\!]$ is then updated as $[\![w_u^i]\!] \leftarrow \left(1 - [\![w_u^i]\!]\right) \cdot \infty + [\![w_u^i]\!]$

Similarly, the subroutine **NodeCost** takes as input the matrices $\boldsymbol{\eta}$ and $\boldsymbol{\Delta}$, where the uf-th element of $\boldsymbol{\eta}$ is the cost of embedding the VN of type f in u. This subroutine returns a matrix $\mathbf{C}$, whose ui-th element is the cost of embedding a computational unit of VN_i in node u. The u-th row of matrix $\boldsymbol{\eta}$ and the i-th row of matrix $\boldsymbol{\Delta}$ are multiplied element-wise using the Mult operator and the resulting products are then summed up to provide the cost of embedding VN_i in u.

8) Secure Node's Capacity Constraint Verification: **NodeCapacity** takes as input the states vector $\mathcal{S}_{CUST}^{(i)}$ and $\mathcal{S}_{ISP_k}^{(i)}, \forall i$, the VNs' computational demand vector $\vec{d}$, the nodes' capacity vector $\vec{\zeta}^{(nodes)}$ and the index u of a physical node $\in ISP_k$. The output $v_u \in \{0, \infty\}$ indicates the fulfillment of the capacity constraint for node u. The k-th element of $\mathcal{S}_{CUST}^{(i)}$ (which is $[\![1]\!]$ iff VN_i has been assigned to ISP_k) is securely multiplied with $[\![d_i]\!]$ and $[\![\mathcal{S}_{ISP_k}^{(i)}]\!]_u$ (which is $[\![1]\!]$ iff VN_i is embedded in node u) by recursively applying the Mult operator. This operation is repeated $\forall i$, and the results are summed up to obtain the current amount of computational demand embedded in node u. This value is then securely compared with the capacity of the node $[\![\zeta_u^{(nodes)}]\!]$ using the GE operator. The result of this comparison is successively multiplied by ∞ to obtain v_u (which is equal to 0 if node's capacity is not exceeded, and ∞ otherwise).

9) Secure Links' Capacity Constraint Verification: **LinkCapacity** takes as input the states vector $\mathcal{S}_{ISP_k}^{(uv,ij)}, \forall i, j, i \neq j, \forall u, v$ and a matrix $\mathbf{P}_k^{uv}$ representing the paths connecting nodes u and v (as described in Subsection VI-A), the value $[\![d_{ij}]\!]$ and the index l. This subroutine returns $[\![v_l]\!]$, where $v_l \in \{0, \infty\}$ indicates the fulfillment of the capacity constraint for link l. To select the path connecting nodes u, v that is traversed by VP_{ij}, RowSelection is applied

[1] ∞ is encoded with the value 1000 in the performed experiments

on $\mathbf{P}_k^{uv}$. Then, Mult is executed to perform the secure multiplication between $[\![d_{ij}]\!]$ and the l-th element of the selected vector. This operation is repeated $\forall u, v, \forall i, j, i \neq j$ and the results are summed up to obtain the share of the amount of bandwidth that are currently deployed on the l-th link. Finally, the GE operator is applied to securely compare this value and $\vec{\zeta}_l^{(links)}$. The result of this operation is then multiplied by ∞ to obtain $[\![v_l]\!]$, which is ($[\![0]\!]$ in case the capacity of the l-th link is not exceeded).

10) Secure Updating of the Q-table: **UpdateQ** inputs the following data: a state vector $\vec{s} = [[\![s_1]\!], ..., [\![s_\Phi]\!]]$, an action vector $\vec{\alpha}$, a $\mathcal{Q}$-table $\in \mathbb{Z}^{\Phi X 3}$, a reward $[\![r]\!]$ and two decimal values, i.e., the learning rate lr and the discount factor γ. The output of this subroutine is the updated $\mathcal{Q}$-table, which is equal to the matrix $\mathcal{Q}$ in input, except for the s, χ-th entry (which corresponds to the selected action in the current state of the agent), which is modified according to Eq. 1, that we repropose for clarity of exposition: $\mathcal{Q}(s,\chi) \leftarrow \mathcal{Q}(s,\chi) + lr \cdot (r + \gamma \cdot \max_{\hat{\chi}} \mathcal{Q}(s,\hat{\chi}) - \mathcal{Q}(s,\chi))$.

Initially, RowSelection is employed on $\mathcal{Q}$ and $\vec{s}$ to obtain the row corresponding to the current state. Max is then used to compute the maximum value of this row, i.e., $[\![\max_{\hat{\chi}} Q(s,\hat{\chi})]\!]$, which is then multiplied by the discount factor γ using the MultDec subroutine. The values within parenthesis are successively summed up and multiplied by lr, also using the MultDec subroutine. At this point, the obtained value need to be summed to $\mathcal{Q}(s,\chi)$ only, while all the other values of the matrix must remain unchanged. As participants are not aware of s and χ, all the elements of the matrix $\mathcal{Q}$ must be summed to the value $lr \cdot (r + \gamma \cdot \max_{\hat{\chi}} \mathcal{Q}(s,\hat{\chi}) - \mathcal{Q}(s,\chi))$ multiplied by a properly selected $[\![mask]\!]$, which can be obtained as follows: the Mult operator is applied to $[\![\vec{s}(\phi)]\!]$ and $[\![\vec{\alpha}(\chi)]\!], \forall \phi, \chi$, in such a way that $[\![mask]\!]$ is $[\![1]\!]$ only for the $\hat{s}\hat{\chi}$-th entry (i.e., that corresponding of the current state and selected action), and $[\![0]\!]$ otherwise.

11) Masked Secure Updating of the Q-table: **MaskedUpdateQ** differs from UpdateQ as the value $(r + \gamma \cdot \max_{\hat{\chi}} \mathcal{Q}(s,\hat{\chi}) - \mathcal{Q}(s,\chi))$ is securely multiplied by $[\![mask]\!]$ before being multiplied by lr using MultDec.

VII. Privacy-Preserving RL for VNE

In this Section, we describe the proposed privacy-preserving RL algorithm, in which operations are performed homomorphically on the shares that customer and ISPs exchange under a $(K+1, K+1)$ SSS. To yield meaningful results, these operations must be executed by all the $K+1$ participants. Notice that this is a crucial difference with respect to the RL described in Section V, in which each participant only performs the operations required to accomplish his specific sub-tasks. Notice also that results are reconstructed at the end of the optimization process, and are obtained only by the legitimate party.

A. Initial Data Sharing

1) Secret Sharing of the Data between customers and ISPs: Initially, the participants exchange with each other the following data:

a) Shares Distributed by the customer: The customer distributes to the ISPs, in secret shared form, the data described in subsection IV-B1, i.e., the vector of computational demands $\vec{d}$, the feasibility matrix $\vec{\Delta}$ and the bandwidth demand matrix $\vec{D}$. Moreover, it also distributes $\mathcal{Q}_{CUST}^{(i)}$ and $\mathcal{S}_{CUST}^{(i)}, \forall i$ and $\mathcal{Q}_{CUST}^{(kk\prime,ij)}$ and $\mathcal{S}_{CUST}^{(kk\prime,ij)}, \forall k, k\prime, \forall i, j, i \neq j$.

b) Shares Distributed by the ISPs: Each ISP distributes, in secret-shared form, the following data: (i) the nodes' computational capacity vector $\vec{\zeta}^{(nodes)}$, (ii) a feasibility matrix $\vec{\mathbf{F}}$ indicating the types of VNs that can be hosted in its physical nodes, (iii) the nodes' embedding cost matrix $\boldsymbol{\eta}$, the link capacity vector $\vec{\zeta}^{(links)}$, (iv) $\mathcal{Q}_{ISP_k}^{(i)}$ and $\mathcal{S}_{ISP_k}^{(i)}, \forall i$; the (iv) $\mathcal{Q}_{ISP_k}^{(uv,ij)}$ and $\mathcal{S}_{ISP_k}^{(uv,ij)}, \forall i, j, i \neq j, \forall u, v$.

Initially, the NodeFeasibility and NodeCost subroutines are applied to $\boldsymbol{\Delta}$,$\mathbf{F}$ and $\boldsymbol{\Delta}$,$\boldsymbol{\eta}$, respectively, to obtain the information on the feasibility and cost of embedding the VNs on the physical nodes, i.e., $\mathbf{W}$ and $\mathbf{C}$. With these data in hand, participants can perform the privacy-preserving counterparts of the operations described in Algorithm 1, which are described in the following subsection. Note that, as explained in Section V, the operations performed within an environment may depend on the operations performed in another one (e.g., the placement of a VN within the network of an ISP is consequent to the selection of ISP hosting it). When performing operations on secret shares, however, the participants are not aware of the decisions taken and, consequently, they do not know the environments they should act in (i.e., they do not know which state vector and relative $\mathcal{Q}$-table has to be updated). To address this issue, we associate a $[\![mask]\!]$ with each considered environment, where $mask = 1$ if operations must take place in that environment, and 0 otherwise. Whenever needed, we will explain the procedures followed by the participants to obtain these masks.

B. Privacy-Preserving Operations on the Environments

1) Operations on $\mathcal{E}_{ISP_k}^{(i)}$: These operations are executed to select the physical node to embeds VN_i (privacy-preserving counterparts of lines $19:25$ and $41:45$ of Algorithm 1). Firstly, the action $\vec{\alpha}_{ISP_k}^{(i)}$ is obtained using the SelectAction subroutine. The state $\mathcal{S}_{ISP_k}^{(i)}$ is then updated accordingly by means of the MaskedUpdateState subroutine, where the employed mask is the k-th component of $\mathcal{S}_{CUST}^{(i)}$, which is $[\![1]\!]$ iff VN_i has been assigned to ISP_k. The corresponding reward is $r_{ISP_k}^{(i)} = [\![-d_i \cdot c_u^{(i)} - v_u - w_u^i]\!]$, which is obtained using the CostEmbeddingVN, NodeCapacity and NodeFeasibility operators. Then, $\mathcal{Q}_{ISP_k}^{(i)}$ is updated using the MaskedQUpdate subroutine. These operations are repeated $\forall i, k$.

2) Operations on $\mathcal{E}_{ISP_k}^{uv,ij}$: These operations are executed to select the path connecting nodes u and v to embed VP_{ij} (privacy-preserving counterpart of lines $27:38$ of Algorithm 1). We remind that the selection of the physical path connecting nodes u and v belonging to ISP_k on which VP_{ij} should pass is dependent on the peering path that such VP traverses. Since customer and ISPs are not aware of the peering path in which VP_{ij} is embedded, they compute

$[\![mask]\!] = [\![(VN_i \in ISP_k) \cdot (VN_j \in ISP_{k\prime}) \cdot \mathcal{S}^{(ij,kk\prime)}_{CUST}[\beta]]\!]$, i.e., $mask = 1$ iff VN_i and VN_j have been assigned to ISP_k and $ISP_{k\prime}$ and VP_{ij} is embedded in the β-th peering path connecting them ($mask = 0$ otherwise).

Then, the participants consider the environment $\mathcal{E}^{uU_k,ij}_{ISP_k}$, where U_k is assumed to be the first peering node of the β-th peering path. SelectAction is executed to obtain $\vec{\alpha}^{uU_k,ij}_{ISP_k}$, which is successively used to update the corresponding state vector by means of the MaskedUpdateState subroutine. At this point, the reward $[\![r^{(uv,ij)}_{ISP_k}]\!] = -\sum_{l\in\mathcal{P}^{ij}_{uv}}([\![c_l \cdot d_{ij}]\!] + [\![v_l]\!]) + [\![r^{(i)}_{ISP_k}]\!] + [\![r^{(j)}_{ISP_k}]\!]$ is computed using the CostEmbeddingVP and the LinkCapacity operator to obtain $\sum_{l\in\mathcal{P}^{ij}_{uv}}[\![c_l \cdot d_{ij} + v_l]\!]$, and by summing the shares obtained as described in Subsection VII-B1 to obtain $[\![r^{(i)}_{ISP_k} + r^{(j)}_{ISP_k}]\!]$. With these values in hand, $\mathcal{Q}^{uv,ij}_{ISP_k}$ is updated with the MaskedQUpdate operator. The same process in performed considering the last node of the peering path (say $U_{k\prime}$) and the environment $\mathcal{S}^{U_{k\prime}v,ij}_{ISP_{k\prime}}, \forall v$ and the intermediate peering nodes (say U_{x_i} and $U_{x_{i+1}}$), for which the same operations are performed on the environment $\mathcal{E}^{U_{x_i}U_{x_{i+1}},ij}_{ISP_{k\prime}}$. All these operations are repeated $\forall u, v, i, j, k$.

3) Operations on $\mathcal{E}^{i}_{CUST}$: These operations are executed to select the ISP to embed VN_i (privacy-preserving counterpart of the operations presented in lines $4:8$ and $56:58$ of Algorithm 1). Firstly, action vector $\vec{\alpha}^{(i)}_{CUST}$ is obtained by means of the SelectAction subroutine and used to update the state vector $\mathcal{S}^{(i)}_{CUST}$ employing the UpdateState subroutine. Then, the reward is computed as $r^{(i)}_{CUST} = \sum_{k=1}^{K}\sum_{j=1}^{N}[\![r^{(j)}_{ISP_k}]\!]$, where $[\![r^{(j)}_{ISP_k}]\!]$ is obtained as described in Subsection VII-B1. With these data, $\mathcal{Q}^{i}_{CUST}$ is updated using the UpdateQ operator. These operations are repeated $\forall i$.

4) Operations on $\mathcal{E}^{(kk\prime,ij)}_{CUST}$: These operations are executed to select the peering path to embed VP_{ij} (privacy-preserving counterpart of the operations presented in lines $11:17$ and $48:52$ of Algorithm 1). Action $\vec{\alpha}^{(kk\prime,ij)}_{CUST}$ is obtained using the SelectAction subroutine and used to update the vector state $\mathcal{S}^{kk\prime,ij}_{CUST}$ using the MaskedUpdateState subroutine. The considered mask is obtained applying the Mult operator on $[\![VN_i \in ISP_k]\!]$ and $[\![VN_j \in ISP_{k\prime}]\!]$. Then, the reward $r^{kk\prime,ij}_{CUST} = \sum_{(u,v)}\sum_{k=1}^{K}[\![r^{uv,ij}_{ISP_k}]\!]$ is obtained by summing the single rewards computed as explained in Subsection VII-B2. Finally, $\mathcal{Q}^{kk\prime,ij}_{CUST}$ is updated using the MaskedQUpdate operator. These operations are repeated $\forall kk\prime, ij$.

C. Computation of the Embedding Cost

At each iteration of the RL algorithm the participants know $[\![d_i \cdot c^i_u]\!], \forall u, i, [\![v_u]\!], \forall u$, $[\![w^i_u]\!], \forall u, i$, which are obtained during the computation of the rewards relative to the embedding of the VNs into the physical nodes, as explained in subsection VII-B1. Similarly, the participants also obtain $[\![\sum_{l\in\mathcal{P}^{ij}_{uv}} d_{ij}c^{links}_l]\!], \forall ij, l$ and $[\![v_l]\!], \forall l$ from the computation of the rewards corresponding to the embedding of the VPs into the physical links, as explained in subsection VII-B2.

By computing $\sum_i[\![d_i \cdot c^i_u]\!] + \sum_{ij}[\![d_{ij}c^{links}_l]\!]$ participants obtain the embedding cost at the current RL iteration, in secret shared form. This cost is then summed up with $\sum_i[\![w^i_u]\!] + \sum_u[\![v_u]\!] + \sum_l[\![v_l]\!]$, which are penalty values corresponding to the feasibility of the solution. We remind that $v_u = 0$ (resp. $v_l = 0$) if node u's (resp., link l's) capacity is not exceeded (and ∞ otherwise) and $w^i_u = 1$ if node u is eligible to host VN_i (and ∞ otherwise).

The obtained value is then securely compared with the previous minimum cost (which is supposed to be initialized at $[\![\infty]\!]$) by means of the GE operator, whose output is recovered by the participants. If the current cost is less then the previous minimum (i.e., GE outputs 0), this cost is taken as the new minimum. The execution of the RL stops when the GE outputs 1 (i.e., current cost greater or equal to the previous minimum one) for a consecutive number of epochs equal to $patience$. Notice that the last iteration in which GE outputs 0 corresponds to the computation of the best embedding and the number of this iteration is known to all the participants.

D. Recovery of the Final Secrets

At the end of the execution of the RL algorithm, the K ISPs deliver to the customer their shares of the minimum cost, obtained as described in the previous subsection. In this way, only the customer can recover the final embedding cost. Then, participants identify the iteration in which the best embedding has been obtained and exchange with each other the following data relative to that iteration:

a) Data Received by the customer: the customer receives from all the K ISPs the shares relative to the states vector $\mathcal{S}^{(i)}_{CUST}, \forall i$ and $\mathcal{S}^{(kk\prime,ij)}_{CUST}, \forall kk\prime, ij$. From these data, customer discovers the ISPs in charge of embedding the VNs and the peering links traversed by the VPs.

b) Data Received by the ISPs: The generic ISP_k receives from all the other participants the k-th component of vector state $\mathcal{S}^{(i)}_{CUST}, \forall i$, from which it can recover the information if VN_i has been assigned to it, and the vector state $\mathcal{S}^{(i)}_{ISP_k}, \forall i$, from which it discovers the physical node in which it has to embed VN_i. Then, the customer delivers to ISP_k the information on the peering nodes belonging to its infrastructure that should be traversed by $VP_{ij}, \forall ij$. Finally, ISP_k receives from all the participants the vector state $\mathcal{S}^{(uv,ij)}_{ISP_k}, \forall uv, ij$. Knowing the physical nodes that host the VNs assigned to it and the peering nodes traversed by the VPs, ISP_k is able to identify the pair of nodes uv and, from $\mathcal{S}^{(uv,ij)}_{ISP_k}$, discover the links in which VP_{ij} has to be embedded.

E. Fulfillment of Privacy Requirements

As described in subsection VII-E, customer and ISPs are modeled as honest-but-curious entities. We justify this assumption showing that two common malicious behaviors are either prevented by our scheme or hard to realize in practice. i) A possible malicious attack consists in a subset of participants that collude to violate another party's privacy. This attack is prevented by the use of a $(K+1, K+1)$ SSS, in which no data can be recovered without the involvement of all the $K+1$ participants (e.g., several colluding ISPs cannot discover confidential details of a competitor's infrastructure without its

explicit consent). ii) In another type of attack, participants can execute the RL using data that they specifically craft to pursue some illicit objective. For instance, the customer might handcraft the VGs to discover specific patterns of deployment (e.g., which ISP has the lowest cost of deployment of a given type of VN). Alternatively, an ISP might advertise lower embedding costs to attract more requests from the customer, and based on how their number increases, it may also guess the embedding costs of its competitors. Since the customer (resp., ISP) has to actually pay for (resp., be paid for) the requested VGs, we argue that there are not have sufficient economic incentives to perform such an attack, mostly because the latter is likely to be effective only after a significant number of requests. We now elaborate on the fulfillment of privacy requirements under the honest-but-curious security model:

1) Customer's Privacy Requirements: All the ISPs perform operations on data relative to customer's environments $\mathcal{E}^{(i)}_{CUST}, \forall i$ and $\mathcal{E}^{kk\prime,ij}_{CUST}, \forall kk\prime, ij$. These operations, described in subsection VII-B, are based on secure primitives. Hence, no information about the VG is leaked beyond the number of VNs N. At every iteration of the RL algorithm, the participants discover if the current embedding cost is greater or equal to the previous minimum one, as described in subsection VII-C, which does not provide additional information about the VG. Finally, during the secret recovery phase described in subsection VII-D, each ISP only receives data related to the portion of VG that it has to embed, from which it derives the computational capacity d_i and type δ_i only of the generic VN_i assigned to it. Similarly, each ISP discovers the bandwidth requirement d_{ij} for every VP_{ij} that it has to embed, but no information about other VPs. Hence, the customer's privacy requirements are fulfilled.

2) ISPs' Privacy Requirements: Each participant performs operations over the shares hiding infrastructural details of the generic ISP_k and from environments $\mathcal{E}^{(i)}_{ISP_k}, \forall i$ and $\mathcal{E}^{(uv,ij)}_{ISP_k}, \forall uv, ij$. From them, it is possible to discover the number of nodes $|\mathcal{M}_k|$ and links $|\mathcal{L}_k|$ of ISP_k, as well as $|\mathbf{P}^{uv}_k|$, i.e., the number of paths connecting any two generic nodes u and v. However, since the operations performed on the shares are proved secure, no additional information about costs and capacities of ISP_k's nodes and links are exposed, as well as if there is or not a link between two generic nodes. Hence, also ISPs' privacy requirements are satisfied.

VIII. Numerical Results

A. Simulation Settings

We compare the RL-based approach with the LID and FID heuristics [4] considering the final embedding cost. We perform the experiments on a simulation platform that we realized in Python and allows us to i) define the networks (i.e., both the physical and virtual one), ii) reproduce the results of the benchmarks and iii) execute the proposed RL algorithms. Physical infrastructure and virtual graph are randomly generated at each simulation, according to the parameters presented in Table III[2]. The total number of peering nodes is the integer part of $1.5 \cdot K$, K being the number of ISPs. Each ISP is then assigned ≥ 1 peering nodes and its network is generated following the Waxman model (as also done in [4]). Peering nodes are then randomly interconnected, in such a way that no ISP is disconnected from the multi-infrastructure network and has, on average, 4 outgoing peering links.

TABLE III: Simulation Settings

Variable	Value
d_i	$\mathcal{U} \sim [0, 10]$
Probability that VN_i and VN_j exchange traffic	0.5
d_{ij}	$\mathcal{U} \sim [1, 10]$
Number of VNs Types	10
Probability that an ISP can host a certain type of VN	0.5
Probability that a physical node can host a certain type of VN	0.5
	4
Cost of embedding VN_i in node u	$\mathcal{U} \sim [1, 10]$
Computational capacity of node u	$\mathcal{U} \sim \{30, 40, 50\}$
Cost of embedding VP_{ij} in an internal link	$\mathcal{U} \sim \{6, 7, 8, 9\}$
Cost of embedding VP_{ij} in a peering link	$\mathcal{U} \sim \{11, 12, 13, 14, 15\}$
Capacity of an internal link	$\mathcal{U} \sim \{50, 100\}$
Capacity of a peering link	$\mathcal{U} \sim \{200, 400\}$
Learning rate lr and Discount Factor γ	0.125

We show the results obtained by averaging the embedding costs of 50 experiments for each type of simulated scenario. Unless stated otherwise, we consider $K = 5$ ISPs with an average number of $M_k = 15$ nodes. The training of the RL algorithm is stopped when no improvements to the final costs are observed for a number of iteration $patience = 50000$.

B. Evaluation of the RL approach

In this subsection, we evaluate the effectiveness of the RL-based approach presented in Section V (i.e., the non privacy-preserving one). Experiments are performed setting $\mathcal{T}^{VN}_{CUST} = 20, \mathcal{T}^{VP}_{CUST} = 50, \mathcal{T}^{VN}_{ISP} = 1, \mathcal{T}^{VP}_{ISP} = 1$. In Fig. 3 we show $\frac{Cost_{RL}}{Cost_{FID}}$ and $\frac{Cost_{LID}}{Cost_{FID}}$ for $N \in \{2, ..., 8\}$. We observe that the $Cost_{RL}$ is always lower than $Cost_{LID}$ and, with $N < 6$, also then $Cost_{FID}$. On average, $\frac{Cost_{LID}}{Cost_{FID}} = 1.14$ and $\frac{Cost_{RL}}{Cost_{FID}} = 1$. Hence, the RL approach is a valid alternative to both the considered heuristics.

Then, Fig. 4a shows the minimum embedding cost as a function of the number of executed iterations of the RL algorithm for $N \in \{2, ..., 8\}$. As expected, a longer exploration of the solution space is needed to find the best embedding with increasing N. More specifically, the largest decrease of embedding cost is obtained after 25000 iterations for large instances of the VG (i.e., $N \geq 5$), while much fewer epochs are needed for small VGs (e.g., the minimum embedding cost of $N = 2$ is generally achieved after as few as 5000 epochs).

We now discuss the data overhead introduced by the privacy-preserving RL presented in Section VII.

[2]In this Table, $\mathcal{U}$ stands for Uniformly-Distributed Random Variable

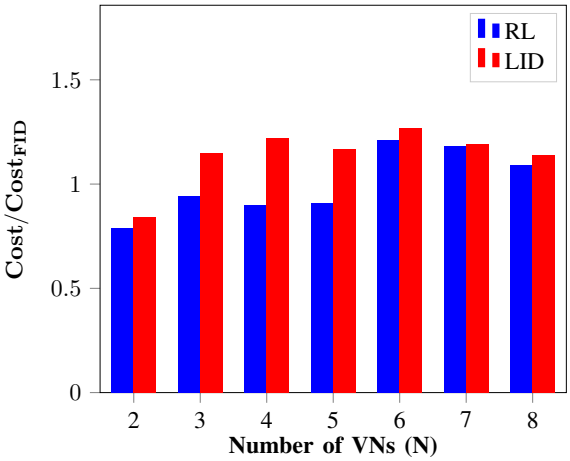
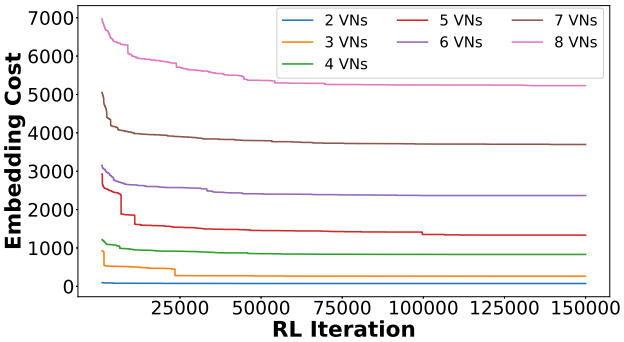
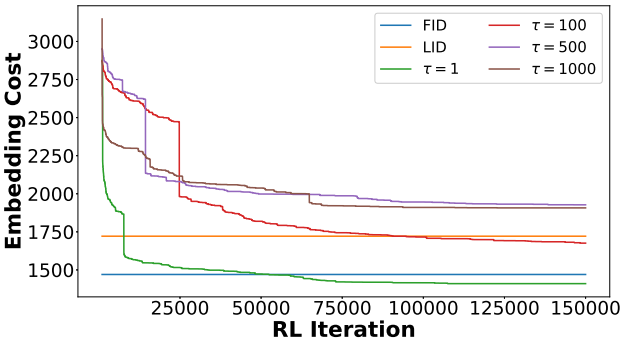


Fig. 3: Comparison of costs achieved with the non-private RL, LID and FID approaches



(a) *Non-Privacy-Preserving RL*

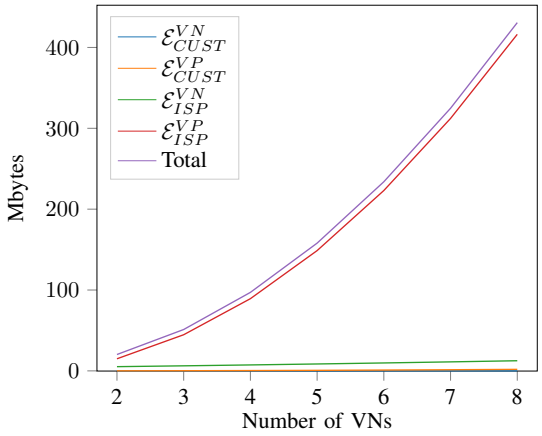


(b) *Privacy-Preserving RL ($N = 5$ VNs)*

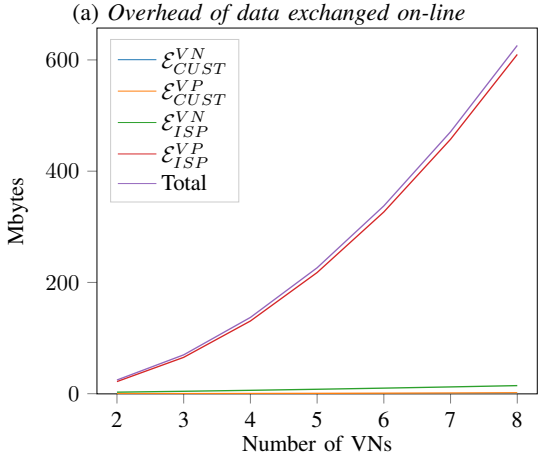
Fig. 4: Minimum Cost of the RL algorithm as a function of the number of iteration

C. Data Overhead of the Privacy-Preserving RL

In Fig. 5 we show the volume of data that each pair of participants exchange with each other at every iteration of the privacy-preserving RL algorithm (where secret shares are assumed to be represented using 20 bits). In particular, we show the overhead generated by performing operations in each



(a) *Overhead of data exchanged on-line*



(b) *Overhead of data exchanged off-line*

Fig. 5: Cumulative Overhead in each type of Environment

one of the four considered types of environments (described in subsection V-A) with increasing N, both on-line (in Fig. 5a) and off-line (in Fig. 5b).

First of all, we observe that the overhead increases with increasing N in all the environments, and this increase is much more voluminous in the environments related to the embedding of a VP into the physical links, i.e., $\mathcal{E}^{VP}_{ISP}$. As shown in Fig. 5a, the operations performed in this environment and described in Subsection VII-B2 introduce, at every iteration of the RL algorithm, a cumulative on-line overhead of up $400Mbytes$. On the other hand, operations in the other environments are much less expensive (e.g., operations in $\mathcal{E}^{VN}_{CUST}$ introduce $\sim 10^{-3}Mbytes$ per iteration). A similar trend can be observed for data exchanged off-line. Notice that these high values are mainly due to the fact that, as data is ciphered, participants are not aware of the specific environment in which they have to act and operations are repeated in all environments. However, we remind that operations are effective only in the environment associated with a $[\![mask]\!] = [\![1]\!]$, as explained in Section VII-A. A strategy to reduce this high overhead is to limit the number of operations performed in environments of type $\mathcal{E}^{VP}_{ISP}$, as

discussed in the next subsection.

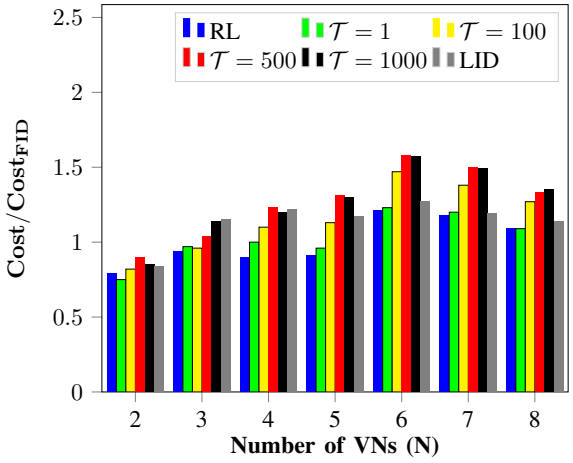
D. Comparison between Privacy-Preserving RL and the baselines

Here we evaluate the embedding cost achieved by performing operations on $\mathcal{E}_{ISP_k}^{uv,ij}, \forall k, uv, ij$ every $\mathcal{T}_{ISP}^{VP}$ iterations of the privacy-preserving RL algorithm, with $\mathcal{T}_{ISP}^{VP} \in \{1, 100, 500, 1000\}$, i.e., by varying the frequency of operations within the type of environments responsible for the greatest portion of the overhead. Obtained results are shown in Fig. 6a.

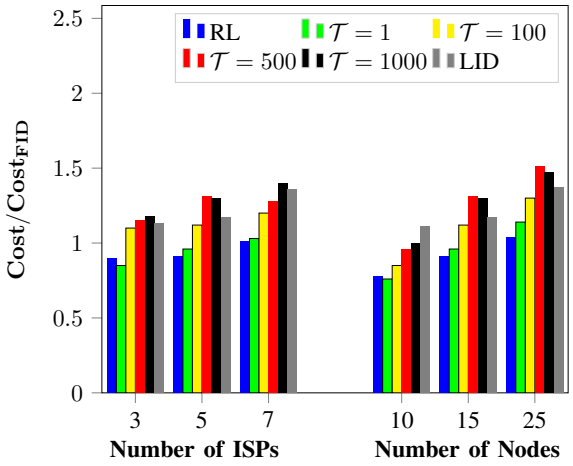
Firstly, we observe that the privacy-preserving RL yields to an average increase of the embedding cost of 3% with respect to FID heuristic, while the average costs of the non-privacy-preserving RL and FID are the same when $\mathcal{T}_{ISP}^{VP} = 1$. This result can be explained considering that operations on ciphered data introduce several approximations (e.g., multiplications by decimal numbers are truncated to integer values). Then, we notice that the cost generally increases with increasing $\mathcal{T}_{ISP}^{VP}$, as the number of operations performed to find the physical paths that embed the virtual ones are reduced. In particular, there is an average increase of cost with respect to FID of 16%, 27% and 27%, achieved for $\mathcal{T}_{ISP}^{VP}$ = 100,500 and 1000 iterations. For $N < 6$, the cost obtained with $\mathcal{T}_{ISP}^{VP} = 100$ is still lower than $Cost_{LID}$, which implies an advantage over to the privacy-preserving baseline (i.e., LID) at a significant reduction of data overhead. As an example, for $N = 5$, $\frac{Cost_{RL}}{Cost_{FID}}$ goes from 0.96 to 1.13 when $\mathcal{T}_{ISP}^{VP}$ goes from 1 to 100, which is an acceptable increase as the overhead at every iteration of the RL drops from 152.42 to 1.61 Mbytes (exchanged on-line) and from 223.12 to 2.31 Mbytes (exchanged off-line). In general, the total overhead decreases of a factor $\sim \mathcal{T}_{ISP}^{VP}$. On the other hand, for $N \geq 6$ the reduction of overhead achieved with $\mathcal{T}_{ISP}^{VP} = 100$ leads to an embedding cost that is higher than the LID heuristic. As a future study, we will evaluate such trade-off when $1 < \mathcal{T}_{ISP}^{VP} < 100$, which seems to be a crucial range to evaluate the ability of the privacy-preserving RL to effectively embed large VGs. We then show in Fig. 4b the comparison of the minimum embedding cost as a function of the RL iteration, for several T_{ISP}^{VP}, considering $N = 5$. We observe that increasing T_{ISP}^{VP} does not significantly affect the number of iterations needed by the algorithm to converge but, as expected, reduces its ability to minimize the embedding cost.

Finally, we show in Fig. 6b the embedding cost considering $N = 5$ for several number of ISPs $K \in \{3, 5, 7\}$ (and fixed number of physical nodes $M_k = 15$) and for several number of average physical nodes in every ISP, i.e., $M_k \in \{10, 15, 25\}$ (and fixed number of ISPs $K = 5$). These results show that, when $T_{ISP}^{VP} \in \{1, 100\}$, the RL yields lower costs than the LID baseline if the number of ISPs (resp., of physical nodes) is increased from 5 to 7 (resp., from 10 to 25), suggesting that the proposed RL approach is robust with respect to the variation of several conditions.

From these results, we observe that the proposed privacy-preserving RL can guarantee embedding costs similar to those



(a) Comparison for several values of N (number of ISPs $K = 5$ and average number of internal nodes $|\mathcal{M}_k| = 15$)



(b) Comparison for several number of ISPs and number of nodes (number of VNs $N = 5$)

Fig. 6: Comparison between LID, FID and privacy-preserving RL for several values of $\mathcal{T}_{ISP}^{VP}$

of the FID scenario, while fulfilling even more stringent privacy requirements than existing LID approaches. However, these results are achieved at the cost of a higher complexity, as further discussed in the next subsection.

E. Complexity of the RL algorithms

In [4] the VNE problem is formulated as a mixed integer multi-commodity flow, and sub-optimal solutions are found in polynomial time for both LID and FID scenarios. Differently from [4], we solve this problem by employing RL-based approaches that find the solution in an iterative fashion. The number of iterations needed by our algorithms to converge is a crucial factor of their complexity, but it cannot be expressed in closed form. In Table IV, we show the complexity of the operations performed at each iteration. From this table, we observe that both our approaches find a solution in polynomial time, but that the privacy-preserving RL is significantly more complex than the non-privacy-preserving one.

TABLE IV: Complexity at each iteration of the RL algorithms

Non-Privacy-Preserving RL	$\mathcal{O}(N^2 + N)$
Privacy-Preserving RL	$\mathcal{O}\left((K+1)\cdot[N^2\cdot(K^2+M_k^2K) + N(1-K^2+K-M_k^2K)]\right)$

The main reasons of the increased complexity are that i) all the $K+1$ participants execute the same operations on their set of shares (indeed, the term $K+1$ in the expression of the complexity accounts for such repeated operations) and ii) operations are performed over encrypted data. As described in Section VII, encryption makes participants not aware of the effects of their actions, and forces them to repeat the same operation on multiple environments to make it effective. However, we notice that operations executed in different environments can be performed independently, which renders the considered problem embarrassingly parallel [22]. Also, this complexity can be significantly reduced if few expensive operations are performed less frequently, as discussed in the previous subsection. Finally, note that this additional complexity is the price to be paid to ensure a privacy-preserving operation of the VNE algorithm.

IX. Conclusions

In this paper, we proposed a privacy-preserving RL algorithm to perform VNE over a multi-domain infrastructure composed of several independent and mutually-distrustful ISPs. In this context, ISPs and customer are not willing to expose details that are required for an effective embedding (e.g., cost of traversing a link). By performing operations on secrets ciphered under the SSS scheme, our algorithm allows both customer and ISPs to retain total privacy. We performed extensive simulations to evaluate the embedding cost achieved with our algorithm compared to two existing heuristics, i.e., the Limited Information Disclosure (LID) and the Full Information Disclosure (FID). The RL algorithm proved to be a valid alternative to such heuristics but introduces a high data overhead. We reduced the number of operations responsible for the highest portion of data overhead and we evaluated the resulting trade-off between overhead and embedding costs. Results have shown that a considerable reduction of data overhead can be obtained at an acceptable increase in the final cost.

Acknowledgment

Davide Andreoletti and Silvia Giordano are funded by the Swiss National Science Foundation (SNSF) via the CHIST-ERA project UPRISE-IoT. Giacomo Verticale's work has been supported by the project ATMOSPHERE, funded by the Brazilian Ministry of Science, Technology and Innovation and by the European Commission.

References

[1] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[2] H. Cao, H. Hu, Z. Qu, and L. Yang, "Heuristic solutions of virtual network embedding: A survey," *China Communications*, vol. 15, no. 3, pp. 186–219, 2018.

[3] S. M. Araújo, F. S. de Souza, and G. R. Mateus, "Virtual network embedding in multi-domain environments with energy efficiency concepts," in *2018 International Conference on Information Networking (ICOIN)*. IEEE, 2018, pp. 205–210.

[4] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-provider virtual network embedding with limited information disclosure," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 188–201, 2015.

[5] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Transactions on Network and Service Management*, vol. 10, no. 4, pp. 356–368, 2013.

[6] F. Samuel, M. Chowdhury, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 6, 2013.

[7] F.-E. Zaheer, J. Xiao, and R. Boutaba, "Multi-provider service negotiation and contracting in network virtualization," in *2010 IEEE Network Operations and Management Symposium-NOMS 2010*. IEEE, 2010, pp. 471–478.

[8] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011–1023, 2011.

[9] K. Guo, Y. Wang, X. Qiu, W. Li, and A. Xiao, "Particle swarm optimization based multi-domain virtual network embedding," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 798–801.

[10] Q. T. A. Pham, A. Bradai, K. D. Singh, and Y. Hadjadj-Aoul, "Multi-domain non-cooperative vnf-fg embedding: A deep reinforcement learning approach," 2019.

[11] X. Chen, Z. Li, Y. Zhang, R. Long, H. Yu, X. Du, and M. Guizani, "Reinforcement learning–based qos/qoe-aware service function chaining in software-driven 5g slices," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3477, 2018.

[12] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow," in *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, 2011, pp. 67–74.

[13] R. Alvizu, S. Troia, G. Maier, and A. Pattavina, "Machine-learning-based prediction and optimization of mobile metro-core networks," in *2018 IEEE Photonics Society Summer Topical Meeting Series (SUM)*. IEEE, 2018, pp. 155–156.

[14] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[15] D. Andreoletti, O. Ayoub, S. Giordano, G. Verticale, and M. Tornatore, "Privacy-preserving caching in isp networks," in *2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2019, pp. 1–6.

[16] D. Andreoletti, O. Ayoub, C. Rottondi, S. Giordano, G. Verticale, and M. Tornatore, "A privacy-preserving protocol for network-neutral caching in isp networks," *IEEE Access*, 2019.

[17] D. Andreoletti, S. Giordano, G. Verticale, and M. Tornatore, "Discovering the geographic distribution of live videos' users: A privacy-preserving approach," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[18] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[19] T. Turban, "A secure multi-party computation protocol suite inspired by shamir's secret sharing scheme," Master's thesis, Institutt for telematikk, 2014.

[20] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Annual International Cryptology Conference*. Springer, 1991, pp. 420–432.

[21] O. Catrina and S. De Hoogh, "Improved primitives for secure multiparty integer computation," in *International Conference on Security and Cryptography for Networks*. Springer, 2010, pp. 182–199.

[22] J.-C. Régin, M. Rezgui, and A. Malapert, "Embarrassingly parallel search," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2013, pp. 596–610.