Secure Resource Provisioning Across Multiple Domains

Toru Mano
NTT Network Innovation Laboratories
mano.toru@lab.ntt.co.jp

Kimihiro Mizutani
NTT Network Innovation Laboratories
mizutani.kimihiro@lab.ntt.co.jp

Osamu Akashi
NTT Network Innovation Laboratories
akashi.osamu@lab.ntt.co.jp

***Abstract*—Network resource provisioning is an important technique for infrastructure providers (infra-providers) because it enables them to utilize their facilities with high efficiency. However, to fully satisfy user requests it is probably necessary to use facilities across multiple domains, for which the conventional resource provisioning methods are unsuitable for the multiple domains because they require unrevealed information from infra-providers. The competitive relationships among infra-providers make it difficult for them to reveal their information to the infra-providers. In this paper, we propose a framework and method for resource provisioning across multiple domains that uses infra-providers' confidential information without exposing it to other infra-providers. To preserve the confidentiality of the infra-providers' information, we propose a cooperative framework using multiparty computation (MPC). However, although MPC provides confidentiality it also brings about nearly intractable computational overhead. Therefore, we pick out values that are locally commutable in each domain and essential for resource provisioning. By using MPC only for these values, our proposed method achieves both tractable MPC overhead and good quality provisioning while preserving information secrecy. Evaluation results show that the computational overhead is tractable and that the average utility fee is at least on the same level as that of the conventional methods.**

I. Introduction

Network resource provisioning is an important technique for infrastructure providers (infra-providers). In resource provisioning for user requests such as those for a virtual network that contains network and computational resources, the infra-provider determines which nodes and links should be used and how to connect them to satisfy the request. By combining this technique with network virtualization [1], infra-providers can utilize their facilities effectively and satisfy user requests flexibly. However, it is probably necessary to use facilities across multiple domains to fully satisfy user requests. For example, some users may want to use a geographically widespread network in which some nodes are located in the U.S., other nodes are located in the EU, and the other nodes are located in Asia. In such cases, the provisioning requires multiple infra-providers' facilities because one infra-provider can hardly cover the whole world. Although many existing research studies deal with resource provisioning [2]–[5], they focus on single domain provisioning.

However, using these methods for multiple domains is not suitable since in such cases the methods requires unrevealed information about the infra-providers such as network topology, facility locations, and available resources. Without this information the result will probably be provisioning failures or poor provisioning quality, but it is hard for the infra-providers to expose such information to other infra-providers due to the competitive environment that exists among them [6]. For example, it has been found that free-riding on a competitor's experiences is quite effective in general [7]. In other words, imitation costs are much lower than inventive costs.

In this paper, we address the problem of secure resource provisioning across multiple domains, that is, the problem of how to provision resources across multiple domains while keeping infra-providers' information secret from other infra-providers. We propose a framework and method to deal with this problem. At first glance, it seems impossible to calculate on the basis of infra-providers' information without revealing it, but a cooperative framework known as multiparty computation (MPC) [8]–[10] makes the calculation theoretically possible. Hence, our proposed framework uses MPC to enable secure resource provisioning. However, although MPC provides confidentiality, it also brings about nearly intractable computational overhead. Therefore, we pick out values that are locally commutable in each domain and essential for resource provisioning. By using MPC only for these values, our proposed method aims to achieve both tractable MPC overhead and good quality provisioning while preserving information secrecy. In particular, to reduce the MPC overhead with our proposed method to the tractable level, we split the task into two parts: one executed among the domains via MPC and one executed in each domain locally. We obtained evaluation results showing that the proposed method's computational overhead is tractable and that its average utility fee is at least on the same level as that of the conventional methods.

The rest of this paper is organized as follows. Section III presents the network model and formulates the secure resource provisioning problem. Section III describes the framework and method we propose. Section IV evaluates the computational overhead and the average utility fee. Section V summarizes related work. Section VI briefly concludes the paper with a summary of key points.

II. Resource provisioning across multiple domains

In this section, we first define network model, user request, and resource provisioning. Then, we formulate the resource provisioning problem for a single domain. Finally, we describe resource provisioning across multiple domains.

A. Network model

We define an infra-provider's network as comprising an undirected graph $G = (V, E)$, node attributes, and link attributes, where V is the set of nodes and E is the set of links.

978-3-901882-50-0

In this paper, we consider available CPU capacity, geographic location, and utility fee for node attributes, and denote them respectively as mappings a_V, l_V, and u_V. We also consider available bandwidth capacity and utility fee for link attributes, and denote them respectively as mappings $a_E$and u_E. Therefore, we denote the network as a tuple of an undirected graph and attribute mappings $N = (G, a_V, l_V, u_V, a_E, u_E)$.

B. User request

We define a user network request as comprising an undirected graph $H = (W, F)$, node constraints, and link constrains, where W refers to the set of nodes and F refers to the set of links. In this paper, we consider CPU capacity and geographic location for node constraints, and bandwidth capacity for link constraints. We use a mapping c_W to represent CPU capacity constraints, a mapping l_W to represent preferred geographic locations, and a mapping d_W to represent acceptable distances from preferred locations. In other words, a node $w \in W$ requires CPU resource $c_W(w)$ and requires locating in position within a radius of $d_W(w)$ from $l_W(w)$. For representing bandwidth constraints of the links, we use a mapping c_F. Thus, we denote a user network request as a tuple of an undirected graph and constraint mappings $R = (H, c_W, l_W, d_W, c_F)$.

C. Resource provisioning

We define a resource provisioning P as a combination of a node mapping M_N and a link mapping M_L. In resource provisioning, we have to determine which infra-provider's resources should be used for satisfying the user request. In other words, we have to determine which infra-provider's nodes should be used for each user requested node and how to connect them by using the infra-provider's links. Here, the node mapping M_N is a mapping from the set of request nodes W to the set of infra-provider's nodes V and $M_N(w) = v$ if and only if the infra-provider's node v is used for the request node w. The link mapping M_L is a mapping from the set of request links F to the set of 0-1 integer vector with respect to the set of infra-provider's links E. In particular, $(M_L(f))_e = 1$ if and only if the infra-provider's link e is used for the request link f. When path splitting [2] is allowed in provisioning, M_L is a mapping to the set of 0-1 real vectors and $(M_L(f))_e > 0$ means that the link e is used for the request link f in the proportion of $(M_L(f))_e$. In this paper, we assume that path splitting is allowed because it enables the efficient resource utilization.

We seek to achieve low price provisioning while satisfying the user request because users main interest, other than the availability of the requested network, is how much the network utility fee will be. Thus, we search a resource provisioning (M_N, M_L) that minimizes the network utility fee:

$$\sum_{w \in W} u_V\left(M_N(w)\right) c_W(w) + \sum_{e \in E, f \in F} u_E(e) c_F(f) M_L(f)_e \tag{1}$$

while satisfying the following conditions:

$$\begin{cases} c_W(w) \le a_V(M_N(w)) & \forall w \in W \\ \| l_V\left(M_N(w)\right) - l_W(w) \| \, f \le d_W(w) & \forall w \in W \\ \sum_{f : e \in M_L(f)} c_F(f) \le a_E(e) & \forall e \in E \\ M_L(f) \text{ are paths between } M_N(x) \text{ and } M_N(y) & \forall f = (x, y) \in F \end{cases} \tag{2}$$

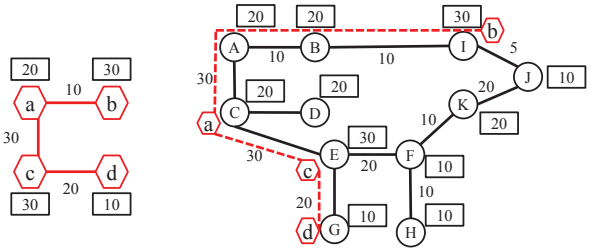

Figure 1. A user request (left) and the infra-providers' network (right) and the resource provisioning (right).

Note that the second term of the network utility fee represents the fee that comes from link utilization because for each request link $f \in F$ and each infra-provider's link $e \in E$, we have to reserve bandwidth of the link e for the link f by $c_F(f)\left(M_L(f)\right)_e$. The above four conditions signify a user request. The first condition signifies that the infra-provider's node $M_N(w)$ has enough capacity for the request node w. The second condition signifies that the infra-provider's node $M_N(w)$ is close enough to the preferred location $l_W(w)$. The third one signify that the infra-provider's link e has enough capacity to contain all the request links f such that they are mapped to the path $M_L(f)$ which contains the link e. In addition, the resource provisioning problem is one of the $\mathcal{NP}$-hard problems [11]. This means that unless $\mathcal{P} = \mathcal{NP}$, no algorithm can solve the problem in polynomial time.

D. Multiple domains

In this paper, we assume that network N consists of multiple infra-providers and the number of infra-providers is n. Here, we denote the$i$-th infra-provider's network as $N^{(i)} = (G^{(i)}, a_V^{(i)}, l_V^{(i)}, u_V^{(i)}, a_E^{(i)}, u_E^{(i)})$ where $G^{(i)} = \left(V^{(i)}, E^{(i)}\right)$, and assume that the inter-domain links, which lie in different infra-providers, belong to both infra-providers. As a result, we can express the entire set of nodes V as $\bigcup_{i=1}^{n} V^{(i)}$ and the entire set of links E as $\bigcup_{i=1}^{n} E^{(i)}$. In secure resource provisioning across multiple domains, the infra-providers seek the provisioning that minimizes the utility fee (1) and satisfies the user request(2). "Secure" means that, during this procedures, the infra-providers do not reveal their information: topology, node attributes, and link attributes, i.e., $G^{(i)}$, $a_V^{(i)}$, $l_V^{(i)}$, $u_V^{(i)}$, $a_E^{(i)}$, and $u_E^{(i)}$. However, we assume that the physical connection relationships among the infra-providers are known [12]. Additionally, we assume that the infra-providers are semi-honest or honest-but-curious, that is, they follow the protocol but keep intermediate computation results and can use them for inferring the others' information [9]. In other words, they do not cheat or not lie but merely gather information and use it for inferring.

III. Proposed Secure Resource Provisioning Across Multiple Domains

In this section, we first describe problems that occur when using conventional single domain methods. Next, we describe our cooperative framework and its reasoning behind its development. Then, we describe our proposed method by splitting in into three procedures: node-domain assignment selection, inner provisioning at each infra-provider, and inner provisioning integration.

A. Problems with the conventional single domain methods

Although several research studies have been made on resource provisioning [2]–[5], when we try to apply these conventional single domain methods to multiple domains, we encounter three problems: the domain information security, the need for a trusted and neutral third party, and MPC overhead. In consequence, applying conventional methods to the multiple domains is impractical.

Using the conventional methods does not keep the infra-providers' information secret because the information must be gathered into one place, e.g., one of the infra-providers, to execute the methods. Since, this violates the information's confidentiality, we need mechanisms to perform the provisioning while keeping the information secret. We think there are two prospective frameworks for achieving this: the trusted and neutral third party framework and the MPC framework.

In the trusted and neutral third party framework, the infra-providers ask for a third party to find the provisioning instead of them. In other words, infra-providers give their secret information to the trusted third party and the party computes the provisioning by using one of the conventional methods after aggregating the information. However, we need a third party that is not only trusted but also neutral, i.e., the party should not reveal the information or cheat in computation for a certain infra-provider's benefit. From a practical application viewpoint, it is very hard to find or establish a third party independent from any infra-providers. Hence, the trusted and neutral third party framework is impractical.

In the MPC framework, the infra-providers execute conventional methods via MPC to maintain the information security. Although this framework theoretically enables us to provision resources securely across multiple domains, it is impractical due to MPC's huge computational overhead. Directly applying MPC to an algorithm means that every algorithm operation (addition, multiplication, comparison, etc.) is executed via MPC. Since MPC usually needs communication to be established among the parties, and since the communication overhead is much larger than the ordinary computational (addition, multiplication, comparison, etc.) overhead, combining these methods results in impractically large overhead. Detailed analytic evaluations of these overheads are described in Section IV.

B. Cooperative framework via MPC

We employ a cooperative framework using MPC for two reasons. First, it is hard to find or establish a trusted and neutral third party, as described before. Second, although MPC is operationally "heavy", there is room for reducing the number of MPC operations. In the framework we propose, the user first sends a request to one of the infra-providers. The recipient infra-provider then shares it with other infra-providers in the framework. Next, the infra-providers start executing cooperative computations among them, some computations via MPC for confidentiality and the others in each infra-provider locally without MPC to reduce the MPC overhead. Finally, after the computations are finished, the recipient provider tells the user whether the requested network is available and if so what the provisioning utility fee is. Note that the infra-providers do not use any third parties' help and all infra-providers in the framework join in performing the computations.

The MPC framework is a cooperative way of conducting general calculation on the basis of the parties' input while keeping the input secret without a third party's help [8]–[10]. Using MPC enables almost all of the calculation to be done while preserving the input secrecy among multiple parties. However, MPC's computational costs are extremely high because every elemental operation (e.g. multiplication, comparison) in it requires communication established among the parties. In addition, its millisecond-order communication delay is much larger than the usual nanosecond-order computational delay. We therefore use MPC only partially in our method, for purposes of preserving secrecy.

C. Overview of the proposed method

We use MPC to carry out the cooperative computation needed for resource provisioning without revealing the infra-providers' information, but no for the cooperative computation as a whole because of its high communication overhead. Therefore, we divide the task into two parts: one executed among the infra-providers via MPC, the other executed in each infra-provider locally without MPC. As a result, we can decrease the overhead by reducing the number of MPC operations. However, this means reducing the information shared among infra-providers via MPC, which could worsen the provisioning quality. We evaluate this effect in Section IV by calculating the average utility fee.

In our proposed method, roughly speaking, the infra-providers repeat three procedures: choosing a node-domain assignment, finding the inner provisioning in each infra-provider according to the node-domain assignment, and integrating the inner provisioning to the entire provisioning (Algorithm 1). We overview our method in the following paragraph and describe these three procedures in detail in the following subsection.

Algorithm 1 Overview of the proposed method

```
1: t ← 0, x ← 0, y ← M                ▷ M is a sufficiently large number
2: for all node-domain assignments A_i in 𝒜 do
3:     find inner provisioning {P_j} in each infra-provider
4:     x', y' ← inner provisioning integration    ▷ via MPC
5:     if y' < y then                              ▷ compare via MPC
6:         t ← i, x ← x', y ← y'                   ▷ update via MPC
7:     end if
8: end for
9: reveal t, x, y and recalculate the provisioning
```

First, the infra-providers initialize the variables t to 0, x to 0, and y to M. Here, the variable t represents a temporal assignment index, x represents a temporal integration index (Subsection III-F), and$y$ represents a temporal utility fee. The constant M is a sufficiently large positive integer. We use MPC to keep these variables t, x, y secret during this procedure. Next, the infra-providers choose one of the node-domain assignments A_i, i.e., they determine which infra-provider provides resources for a certain request node. Then, each infra-provider independently finds an inner provisioning (a resource provisioning for assigned nodes and links that contains them) and calculates the utility fee for the provisioning. Next, the infra-providers share the results of the local provisioning via MPC, integrate them into one consistent resource provisioning across multiple domains, and store its

information to x' and y'. They then update the variables t, x, y if the current provisioning has a lower utility fee. Next, after searching all of the node-domain assignment candidates, they reveal the variables t, x, y and calculate the provisioning. Finally, the recipient infra-provider tells user the results. Note that there is a possibility the inner provisioning will fail; when this happens the infra-providers simply skip the node-domain assignment. However, for the sake of simplicity, in the rest of this paper we assume that no infra-provider will fail to perform its inner provisioning assignment.

D. Node-domain assignment selection

We define a node-domain assignment as a division of the request nodes W: $A = (A_1, A_2, \ldots, A_n)$, where $\bigcup_{i=1}^{n} A_i = W$ and $A_i \cap A_j = \emptyset$ for all $i, j \in \{1, \ldots, n\}$. In other words, when the i-th infra-provider's assignment A_i contains a node $w \in W$, then the i-th infra-provider provides a resource for the node w. We denote the set of node-domain assignment candidates as $\mathcal{A}$ and its size $|\mathcal{A}|$ is at most n^W. In this paper, we set the candidate $\mathcal{A}$ for all node-domain assignments and do not use any heuristics. This is because this paper focuses on the effectiveness of our cooperative framework and our proposed method. The most significant aspect of the proposed method is that it separates the computations into those locally executed at each infra-provider and those executed among infra-providers. Hence, we first evaluate that aspect and leave other aspects untouched, although reducing the search space (i.e., reducing the number of node-domain assignment candidates) is also an important issue.

E. Inner provisioning at each domain

Once the node-domain assignment is chosen, each infra-provider adds auxiliary nodes to achieve consistency and finds inner provisioning using the linear programming relaxation method [4]. After finding the inner provisioning, they share the results via MPC.

When the node-domain assignment is selected, each infra-provider (e.g., the i-th infra-provider) starts finding the local provisioning for assigned nodes A_i and links contains them. The infra-provider add auxiliary nodes to use the conventional resource provisioning method because one of the end points of the request link may not be assigned to the infra-provider. Accordingly, the i-th infra-provider adds new nodes to both infra-provider's nodes V_i and request nodes A_i. We define a boundary of the nodes V as the set of nodes that are connected to the nodes V but do not belong to them, and denote it as $B(V)$. We add the boundary infra-provider's nodes $B(V_i)$ to the infra-provider's nodes V_i, and also add the boundary request nodes $B(A_i)$ to the request nodes A_i. Additionally, we assume that any of the boundary request nodes $B(A_i)$ can be mapped to any of the boundary infra-provider's nodes $B(V_i)$ but can't be mapped to any other nodes. We also assume that there is no utility fee for the infra-provider's boundary nodes $B(V_i)$. These procedures ensure that all the request link's end points lie in the domain. Hence, the infra-providers can solve the inner provisioning problem by using one of the conventional methods. However, the provisioning is likely to be inconsistent with another infra-provider's inner provisioning due to adding the boundary nodes. These inconsistencies are resolved during the integration procedure (Subsection III-F).

We use the linear programming relaxation method to achieve inner provisioning.We chose this method for three reason. First, the method is computationally tractable and will terminate in practical time. More precisely, this method is polynomial time algorithm and the computational cost is $O(m^{3.5})$ where m is the number of variables in the linear programming [13]. Second, the method tends to output solution that are nearer to being optimal than the solutions output by other approximation methods such as greedy ones [4]. Third, the method is flexible owing to the linear programming formulation, i.e., we can easily add new constraints or change the utility fee function.

After finding the inner provisioning, each infra-provider emits one bit that signifies whether the infra-provider can afford to provide resources for the assigned nodes via MPC. In addition, each infra-provider emits the utility fee for the resources. This information is used for the next integration procedure.

F. Integration of the inner provisioning

In the integration procedure, we integrate n pieces of inner provisioning into one whole provisioning which satisfies two conditions: (a) it is consistent with boundary conditions and (b) it has a low utility fee. Here, the boundary condition represents the mapping relationships between inter-domain request links and inter-domain infra-provider links. To achieve the former condition (a), we adjust the pieces of inner provisioning to be consistent with the boundary conditions and integrate them one by one (Algorithm 2). To achieve the latter condition (b), for each piece of inner provisioning, we start integration from the inner provisioning. Accordingly, we obtain n kinds of whole provisioning. From them, we choose the best one i.e., the minimum utility fee provisioning (Algorithm 2). In the following, we provide the details of the procedure.

First, the infra-providers initialize the variables x to 0 and y to M. Here, the variable x represents a temporal integration index and y represents a temporal utility fee. The constant M is a sufficiently large positive integer. We use MPC to keep these variables x and y secret during this integration procedure. Next, the infra-providers choose one of them (e.g., the i-th infra-provider) and let T be $\{i\}$. Here, T represents the infra-providers whose inner provisioning is fixed. Then, the i-th infra-provider tells the neighbors $N(\{i\})$ its boundary conditions. We define a neighbors of the i_1-th, i_2-th, ..., i_k-th infra-providers as the infra-providers that are connected to at least one of the i_1-th, i_2-th, ..., i_k-th providers by the inter-domain link. We also denote them as $N(\{i_1, \ldots, i_k\})$. Next, the infra-providers find the neighbor infra-provider from $N(T)$ that has the highest utility fee. Here, we assume that the infra-provider is the j-th infra-provider. Note that this search is executed by using MPC. Then, the j-th infra-provider recalculates the inner provisioning in order to accommodate it with the boundary conditions. Next, the j-th infra-provider tells the neighbors $N(\{j\})$ its boundary conditions and the infra-providers add T to j. The infra-providers repeat these procedures until T becomes full, i.e.,$T = \{1, \ldots, n\}$. When T is full, then the infra-providers have the whole and consistent provisioning. Accordingly, they calculate the utility fee y' via MPC. Then they update the temporal variables x and y as follows via MPC: if $y' < y$ then they set x to i and y to y', otherwise they do nothing. Next, they choose another index i' and let T be $\{i'\}$ and repeat the above procedure. Finally, after searching all the indices, the variable y contains the minimum

utility fee found in this integration and the variable x contains the corresponding infra-provider index.

Algorithm 2 Inner provisioning integration

$x \leftarrow 0, y \leftarrow M$ ▷ M is a sufficiently large number
for $i \leftarrow 1, \ldots, n$ **do**
  $T \leftarrow \{i\}$
  The i-the infra-provider tells the neighbors $N(\{i\})$ the boundary conditions
  while $T \neq \{1, \ldots, n\}$ **do**
    find $j \in N(T)$ such that the j-th infra-provider has the highest inner utility fee ▷ find via MPC
    the j-th infra-provider recalculates the provisioning to accommodate the boundary conditions
    the j-th infra-provider tells the neighbors $N(\{j\})$ the new boundary conditions
    $T \leftarrow T \cup \{j\}$
  end while
  calculate entire provisioning fee y' with MPC
  if $y' < y$ **then** ▷ compare via MPC
    $x \leftarrow i, y \leftarrow y'$ ▷ update via MPC
  end if
end for

IV. Evaluation

We evaluate the effectiveness of our proposed method by comparing its computational overhead and average utility fee with the those of conventional methods. We evaluate the overhead analytically and the average fee numerically. The results show that the proposed method's overhead is one hundredth that of the conventional methods and that the utility fee is at least on the same level as that of the conventional methods when the problem size is relatively small.

A. Methods for comparison

We use combinations of a conventional method and MPC as methods for comparison. For the node mapping algorithm, we use three conventional methods: full search, greedy algorithm, and linear programming relaxation [4]. For link mapping, we use the linear programming method for solving the minimum-cost max-flow problem [14] that represents link mapping. We execute all these methods via MPC to keep the infra-providers' information secret. Accordingly, we denote the combination of the full node mapping search and the link mapping method as *full search*, the combination of the greedy node search and the link mapping method as *greedy*, and the combination of the linear programming relaxation and the link mapping method as *LP relax*.

B. Analytic evaluation of the overheads

The number of MPC operations in full search is $O(|V|^{|W|}|E|^{3.5}|F|^{3.5})$, in greedy is $O(|E||F|^{3.5})$, in LP relax is $O\left((|V||W| + |E||F|)^{3.5}\right)$, and in the proposed method is $O(n^{|W|+2})$. This is because the interior point method [15] can solve linear programming problem by $O(m^{3.5})$ operations, where m is the number of variables in the linear programming. Furthermore, the number of variables in the minimum-cost max-flow problem for the link mapping is $|E| \cdot |F|$. Note that, in the proposed method, each infra-provider needs $O(n^{|W|}(|E|/n)^{3.5}|F|^{3.5})$ ordinary operations (i.e., operations without MPC) on average.

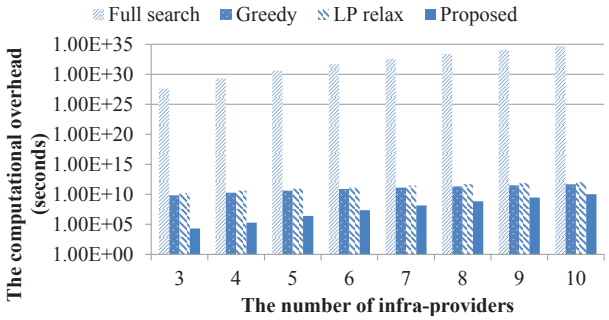

Figure 2. The estimated computational overhead of full search, greedy, LP relax, and proposed methods.

To estimate the practical computation time, we substitute the parameter values into the computation cost equations. The values were set as follows: 20 nodes in each infra-provider, 40 links in each infra-provider, a user request contains10 request nodes and 20 request links. We also assume that each MPC operation takes 10 milliseconds due to communication delay and that each ordinary operation takes one nanosecond. We vary the number of infra-providers between 3 and 10 and evaluate the computational overhead for each case (Fig. 2).

The evaluation results show that for a fairly small number of infra-providers, in particular eight at most, the proposed method's computational overhead is less than one-hundredth that of the conventional methods.

1) Numeric evaluation of the provisioning: We implemented the four methods (full search, greedy, LP relax, proposed) for the numerical simulation. In the implementation, we replaced each MPC operation with an ordinary operation because this does not change the output provisioning and reduces the computational cost. We used the GNU Linear Programming Kit (GLPK) [16] to solve the linear programming in these methods.

In order to evaluate the quality of the methods, we defined the normalized utility fee for the provisioning as the value obtained by dividing the utility fee for the provisioning (1) by the minimum utility fee. We randomly generated the problems and solved them by using the four methods and calculate the average normalized utility fee of the greedy method, the LP relax method, and the proposed method (Fig. 3). Note that the normalized utility fee of the full search method is always 1 because this method always outputs optimal provisioning.

We used the following parameters: 40 nodes in the infra-providers, a uniform distribution of 40 to 120 links in the infra-providers, and two infra-providers. We varied the number of request nodes $|W|$ between 3 and 9 and, in accordance with this, randomly chose between $|W|$ and $3 \cdot |W|$ request links. For each number of request nodes, we generated 100 problems and used them to calculate the average utility fee. We set a small number of infra-providers n to hold down the computational cost. This is because calculating the optimal utility fee necessitates executing the full search method, which has exponential computational cost.

The results show that the proposed method's average utility fee is at least on the same level as that of the conventional methods (Fig. 2). Overall, the results we obtained confirm that the proposed method has small computational overhead and

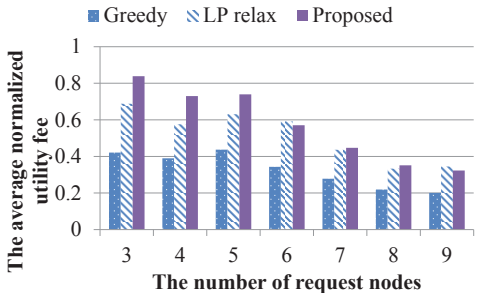

Figure 3. The average normalized utility fee of the greedy, LP relax, and proposed methods.

also outputs good resource provisioning when the problem size is relatively small.

V. Related Work

In addressing the resource provisioning problem, we find a combination of network and computational resource and determine how to connect them to each other to satisfy given conditions. Much research has been done on resource provisioning, especially in virtual network contexts such as allowing path splitting [2], using a graph isomorphism technique [3], using linear programming and random rounding techniques [4], and using a page-rank like technique [5]. However, we cannot apply these single domain methods directly to multiple domains because infra-providers do not reveal their network information to other infra-provider.

There are several ways to construct a path between two network nodes lying in different domains, including MPLS/GMPLS and RSVP [17], [18]. Although these protocols are useful, they are unsuitable for solving the resource provisioning problem across multiple domains because we first have to determine which nodes and links should be used.

Some methods developed through research handle network operations with complete confidentiality. For instance, one deals with finding the shortest path across multiple domains [6], while another copes with verifiable routing between ASs [19]. However, since specific techniques are used for each of these methods, we cannot apply the same technique to the resource provisioning problem across multiple domains.

Secrecy preserving computation methods such as MPC have been developed for general purposes [8]–[10]. In some research these techniques have been applied to practical problems [20]. Nevertheless, using these methods directly to solve the resource provisioning problem across multiple domains is impractical because they impose too much computational cost on infra-providers in compensation for providing information security.

VI. Conclusion

In this paper, we have proposed a cooperative framework and method to address the secure resource provisioning problem across multiple domains. Using our proposed method, the domains can find resource provisioning across multiple domains while keeping their information secret. In our framework, the infra-providers' information is kept secret due to the use of multiparty computation (MPC). However, the use of MPC brings about huge computational overhead. In the proposed method, we therefore reduce the MPC overhead by splitting the task into two parts: one executed among the infra-providers via MPC and the other executed in each infra-provider locally. We also evaluated the computation overhead and average utility fee of our proposed method and conventional methods. The results show that the proposed method's overhead is one hundredth that of the conventional methods and its utility fee is at least on the same level as that of the conventional methods when the problem size is relatively small.

References

[1] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, Apr. 2010.

[2] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008.

[3] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. ACM VISA*, 2009, pp. 81–88.

[4] M. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206 –219, feb. 2012.

[5] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[6] M. Fukushima, T. Hasegawa, T. Hasegawa, and A. Nakao, "Minimum disclosure routing for network virtualization," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, april 2011, pp. 858 –863.

[7] M. B. Lieberman and D. B. Montgomery, "First-mover advantages," *Strategic Management Journal*, vol. 9, no. S1, pp. 41–58, 1988.

[8] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. ACM STOC*, 1987, pp. 218–229.

[9] O. Goldreich, *Foundations of Cryptography, volume II, Basic Applications*. Cambridge University Press, 2004.

[10] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proc. ACM STOC*, 1988, pp. 1–10.

[11] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, unpublished Manuscript.

[12] The cooperative association for internet data analysis (CAIDA). [Online]. Available: http://www.caida.org/data/

[13] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. ACM STOC*, 1984, pp. 302–311.

[14] J. B. Orlin, "A polynomial time primal network simplex algorithm for minimum cost flows," in *Proc. ACM SODA*, 1996, pp. 474–481.

[15] Y. Ye, *Interior-Point Algorithms: Theory and Analysis*. New York, USA: John Wiley & Sons, 1997.

[16] GNU linear programming kit (GLPK). [Online]. Available: http://www.gnu.org/software/glpk/

[17] S. Dasgupta, J. de Oliveira, and J.-P. Vasseur, "Path-computation-element-based architecture for interdomain MPLS/GMPLS traffic engineering: Overview and performance," *Network, IEEE*, vol. 21, no. 4, pp. 38 –45, july-august 2007.

[18] J. Vasseur, A. Ayyangar, and A. Farrel, "Inter-domain MPLS and GMPLS traffic engineering–resource reservation protocol-traffic engineering (RSVP-TE) extensions," RFC 5151, Feb. 2008.

[19] M. Zhao, W. Zhou, A. J. Gurney, A. Haeberlen, M. Sherr, and B. T. Loo, "Private and verifiable interdomain routing decisions," in *Proc. ACM SIGCOMM*, 2012, pp. 383–394.

[20] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft, "Secure multiparty computation goes live," in *Financial Cryptography*, 2009, pp. 325–343.