This paper was presented as part of the 14th IEEE Global Internet Symposium (GI) 2011 at IEEE INFOCOM 2011

Minimum Disclosure Routing for Network Virtualization

Masaki Fukushima Teruyuki Hasegawa Toru Hasegawa
KDDI R&D Laboratories Inc.
Email: {fukusima,teru,hasegawa}@kddilabs.jp

Akihiro Nakao
The University of Tokyo
Email: nakao@iii.u-tokyo.ac.jp

***Abstract*—Although virtual collocation of *Service Providers (SPs)* on top of *Infrastructure Providers (InPs)* via network virtualization brings various benefits, we posit that *operational confidentiality* has not been considered in this network model. We extend and apply the Secure Multiparty Computation (SMC) protocol to solving Minimum Disclosure Routing (MDR), that is, enabling an SP to route packets without disclosing routing information to InPs. Our study reveals that MDR can be achieved securely with marginal latency overhead with regard to the convergence time in well-engineered routing algorithms. Our study sheds light on the path for network virtualization to be used to resolve the challenges for ISPs of today.**

I. Introduction

As multiple access technologies to the Internet become available to users, such as ADSL, FTTH and 3G/4G wireless, Internet Service Providers (ISPs) are facing a mixture of challenges that may seem harder than ever to fulfill concurrently, such as (1) extending *footprint* to cover large user-base and multiple access means per user, (2) reducing operational *cost*, (3) improving network *availability*, and (4) maintaining *operational confidentiality*.

An emerging concept of network virtualization (NV) recently proposed in various projects [1]–[3] is expected to help ISPs achieve some of the goals, especially (1)-(3), at the same time. For example, virtual collocation [4], [5] separates *Infrastructure Providers* (*InPs*) that provide multiple isolated slices of physical resources and *Service Providers* (*SPs*) that utilize slices to operate virtual networks on, in order for the SPs to *cost-effectively* extend their network *footprint* on top of multiple InPs without investing on physical infrastructure and to improve network *availability* by splicing multiple paths [6].

However, the last bullet (4) mentioned above, *operational confidentiality*, may be left unresolved even with such a new concept of NV to the rescue for achieving the diverse mixture of goals of ISPs. We observe that an ISP often strives to keep its competitors away from its operational practices developed to survive business tussle. It is common for the followers of market-leaders in various businesses to analyze their strategies/operations and to employ them to catch up [7], [8]. Free-riding on other ISPs' operational expertise turns out to be quite effective since ISPs often cultivate the market in similar geographical regions [9], [10]. In the German wireless telecommunication market, followers have taken such a "herding strategy" to bring severe price-cutting competition and the profit has plunged by 50% in five years [11]. In the light of these observations, one must note that NV in fact may have negative impact on *confidentiality of operational information of SPs*, since virtual collocation [4], [5] implies that an InP runs its own SP service on its top as well as on the other (competing) InPs that have access to the operational details of the SP on top of them. For example, virtual collocation may allow two competing InPs such as AT&T and Verizon to extend the footprint of their own SP services over the resources of each other, which, however, endangers the operational confidentiality of both SPs. Therefore, if NV is to be employed to satisfy all the goals (1)-(4) of ISPs mentioned previously, the challenge is to achieve secure network operations of SPs without disclosing much information to the underlying InPs.

In this paper, taking distributed routing in SP's virtual networks as an example, we focus on *Minimum Disclosure Routing (MDR)*, where an SP overlaid on top of multiple InPs minimizes disclosure of its routing information of the virtual network, such as topology and link/path cost, to the underlying InPs, while the SP's virtual routers collectively and securely perform distributed routing computation[1]. We show that the MDR problem can be solved through the extension to *Secure Multiparty Computation* (*SMC*) [14], where multiple parties cooperatively compute a function from each party's confidential input. Our feasibility study reveals that the proposed method is supposed to achieve secure routing without degrading convergence time. Accordingly, we conclude that our proposed method together with NV fulfills all the contradicting goals of ISPs concurrently.

The rest of the paper is organized as follows. Section II defines the problem and Section III proposes the solution. Section IV evaluates the proposal and Section V discusses the important issues. Section VI introduces the related work. And finally Section VII briefly concludes.

II. Problem

A. A Walk-through Scenario

We consider the network virtualization shown in Figure 1 where a service provider (SP) is operating a *slice* (i.e., a virtual network) on top of four infrastructure providers (InPs) I_e, I_f, I_g, and I_h. For the sake of discussion, we introduce a term *subslice* to denote a part of a slice on top of each InP,

[1]Note that disclosure of link and path cost leads to disclosure of further information such as link bandwidth [12], [13].

978-1-4244-9920-5/11/$26.00 ©2011 IEEE

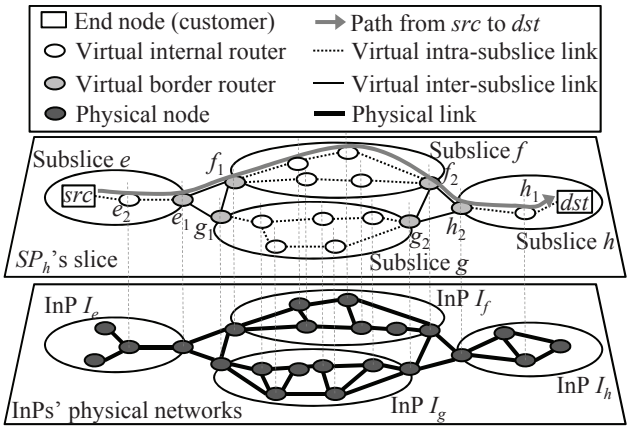

Fig. 1. An example virtualized network environment (an SP operating over four InPs)

in other words, a set of virtual routers and virtual links each InP is hosting. For example, in Figure 1, the subslice e is a part of the SP's slice on top of InP I_e. Virtual routers on the borders of the subslices are interconnected at two peering locations (e_1, f_1, g_1) and (f_2, g_2, h_2), as is often the case with the Internet of today [9], [15].

Suppose in this example that the SP is actually operated by InP I_h that is a business competitor to the rest of InPs, I_e, I_f, and I_g. The SP (SP_h) purchases virtual routers and virtual links from these three InPs, builds a slice running an arbitrary networking protocol, and provides end-to-end services to its customers src and dst.

Suppose SP_h is willing to implement the shortest-path routing, that is, to route a packet from src in the subslice e to dst in the subslice h through the path with the smallest hop count. For the sake of simplifying routing, we set the hop-count between routers within a peering location such as e_1, f_1, g_1 to zero since they are typically collocated [9], [15]. For example, the shortest path from src to dst is 7 hops (7 virtual intra-subslice links) via e_2, e_1 through the subslice f and via h_2, h_1.

Now, SP_h faces a fundamental problem that its virtual routers must calculate the shortest path while disclosing only the encrypted or fragmented topology information that cannot be reconstructed into something meaningful to the other InPs, e.g., how e_1 finds that this packet should exit from e to f rather than to g without obtaining raw hop-counts of virtual links in the other subslices f, g, h. Unless SP_h could not assure the confidentiality of its operational information, it would not utilize virtualized network resources from its competitors.

B. Minimum Disclosure Routing (MDR)

As shown in Section II-A, an SP may encounter MDR problem, where each virtual router needs to obtain its next hop information without disclosing any such *confidential routing information* to its underlying InPs.

MDR is formulated as the problem of distributed computation that (1) takes *local topology information* as an *input* from each virtual router, which is a set of distances (e.g., hop-counts, link-weights, etc.) of the virtual links to its neighbors, (2) exchanges only the encrypted or fragmented *routing information* among virtual routers as *intermediate results* of computation, which is computed from the local topology information, (3) gives *next-hop information* as an *output* to each virtual router.

As a result, an InP hosting a subslice can obtain only the local topology information and next hop information of the subslice it is hosting. It cannot obtain local topology information and next hop information of the other subslices and any routing information.

C. Threat Model

Any solution to a security problem needs clear definition of the threat model to assume. Thus, for MDR, we assume that an InP be a *curious-but-honest* adversary in security jargon, i.e., an adversary that may passively collect information but will not actively attack the system.

We assume that an InP can observe any information (including software, data and protocol messages) stored at any virtual router it is hosting. This is because, technically, an InP has access to any information stored in registers, memory, storage on its top. However, we assume the InP will not maliciously add, modify or remove any such information since such active attacks can be traced by the SP. Besides, we assume that two or more InPs may not collude to expose the SP's confidential routing information.

The threat model defined above implies that none of the existing routing algorithms such as RIP and OSPF can achieve MDR, because they require the virtual routers to send/receive confidential routing information (e.g., the distance to a destination) between different subslices and thus disclose it to underlying InPs. Most importantly, even if an SP naïvely encrypts confidential routing information, the SP needs to store the encryption key on top of InPs' memory and storage in order for a virtual router to decrypt the encrypted information. Therefore, the InP also has access to the key and can decrypt the encrypted information.

III. Proposed Solution

Our approach to achieving MDR is to extend the generic SMC by defining a new primitive to transfer a secret in addition to the standard primitives so that our extended SMC may be applied to a distributed routing problem such as MDR.

A. Secure Multiparty Computation (SMC)

SMC is defined as cooperative computation of a function where multiple parties provide inputs and cooperatively calculate outputs of the function, while keeping each party's input and output invisible from the other parties. SMC requires that any intermediate results of the computation must not be disclosed to each party, since they might contain the information that may infer the other party's input and output. Although the requirement of SMC appears infeasible to satisfy, surprisingly, there exist several generic SMC protocols [14], [16], [17] *if*

every pair of the parties has a communication channel between them (i.e., if the parties have full-mesh connectivity.)

In the generic SMC protocol [17], each party first protects its own input by using the *secret sharing scheme* [18]; a scheme to encode a secret input into multiple *shares* and distribute the shares among the parties. Any single party cannot recover the secret unless a certain subset of the other parties disclose their shares to this single party for decoding. Then, they compute the function while all the intermediate computation results are also shared among them, which requires exchanging messages between every pair of parties. Finally, they recover the final output of the function.

B. Overview of Solution

From a viewpoint of SMC, MDR is an SMC problem that multiple "virtual routers" (hereafter referred to as "routers") need to compute a function that takes local topology information as an input from each router and gives next hop information as an output to each router. However, the generic SMC protocol cannot be applied to MDR due to a chicken-or-egg problem—the generic SMC protocol requires full-mesh connectivity between the routers, while routing is a problem to logically establish such full-mesh connectivity.

Also, MDR can be viewed as a distributed routing problem, which can be often divided into the local router problems. Exercising this insight, we consider decomposing MDR into local router computations, each of which is performed in a set of fully-connected border routers located within the same peering location, namely, defined as *inter-subslice clique* (or simply referred to as *clique*). By definition, the border routers in such a clique can perform the generic full-mesh version of SMC protocol to solve the local problems. Furthermore, since each border router in a clique is from a different InP, running SMC among the border routers in the clique ensures none of the underlying InPs may obtain the inputs (i.e., routing information) from the other InPs.

In a nutshell, we solve MDR by running a distributed routing algorithm in a logical topology called *clique-level topology* shown in Figure 2. Formally, a clique-level topology is a *multigraph* of cliques, where a pair of cliques is connected by clique-level link(s) if it is connected by intra-subslice path(s). Each clique-level link between a pair of cliques represents the shortest intra-subslice path between two routers, each at the different clique. Also, the topology includes *stubs* (i.e., end nodes) and *clique-stub links* connecting these stubs to cliques. Each clique-stub link represents the shortest intra-subslice path connecting a stub to a border router in the same subslice.

C. Primitive Operations in Extended SMC

In order to run a distributed routing algorithm in the clique-level topology, we extend the SMC protocol in two-fold and define four primitives for a clique of routers to invoke.

First, we identify the following three primitives for solving local problems in a clique consisting of L parties (i.e., L border routers). In the following, $[x] = ([x]^1, \ldots, [x]^L)$ denotes L shares of a *secret* x (e.g., a distance value), generated by the secret sharing scheme [18].

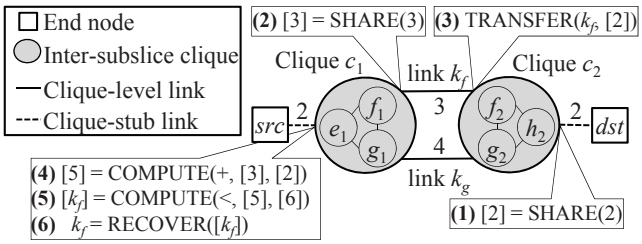

Fig. 2. The clique-level topology of the slice shown in Figure 1. The operations (1) to (6) are described in Section III-D

- $[x] = \mathrm{SHARE}(x)$ is for sharing a secret x among the clique. It takes the secret x as an input from one of the L parties, and gives the shares $[x] = ([x]^1, \ldots, [x]^L)$ of the secret x as outputs to the L parties. Each share $[x]^\ell$ is held by a different party.
- $[y] = \mathrm{COMPUTE}(F, [x])$ is for computing a secret $y = F(x)$ by the generic SMC protocol [17] from a publicly known function F and another secret x shared among the clique. It takes the shares $[x]$ of the secret x as inputs from the L parties, and gives other shares $[y]$ of the computed secret y as outputs to the L parties. Each party learns nothing regarding the secrets x, y and intermediate computation results.
- $x = \mathrm{RECOVER}([x])$ is for recovering a secret x shared among the clique. It takes the shares $[x]$ of the secret x as inputs from the L parties, and gives the secret x as an output to the L parties.

Second, we design another primitive to transfer a secret shared among a clique to its neighboring clique in the clique-level topology.

- $\mathrm{TRANSFER}(link, [x])$ is for transferring a secret x shared among this clique to a neighboring clique via $link$. It takes the shares $[x]$ of the secret x as inputs from the L parties in this clique, and gives other shares of the same secret x as outputs to the parties in the neighboring clique.

D. Walk-through Scenario Revisited

We revisit the same example in Section II-A to sketch our idea to solve MDR (i.e., how the border router e_1 obtains next hop information) by using Figure 2 and the primitives defined in Section III-C.

First, as shown in Figure 2 **(1)**, the border router h_2 shares the secret distance 2 of the clique-stub link to dst by invoking SHARE among the clique c_2. This distance 2 should be invisible from the other subslices e, f, g because it includes the distances of links in the subslice h. Also, in Figure 2 **(2)**, f_1 shares the secret distance 3 of the link k_f among the clique c_1.

Then, in Figure 2 **(3)**, these secret distances are flooded in the clique-level topology by advertising them between the

neighboring cliques. For instance, the clique c_2 advertises the secret distance 2 originated from h_2 to the neighboring clique c_1 by invoking TRANSFER to k_f. During this flooding process, the secret distances of links along the flooding path are added to the secret distance. For instance, in Figure 2 **(4)**, the clique c_1 adds the secret distance 3 of k_f to the secret distance 2 advertised via k_f by invoking COMPUTE and obtains an accumulated secret distance 5 from the clique c_1 to dst via k_f. Likewise, the clique c_1 obtains a secret distance 6 from the clique c_1 to dst via k_g (not shown in Figure 2).

Finally, in Figure 2 **(5)**, the clique c_1 is ready to compare two secret distances, 5 and 6, advertised via k_f and k_g respectively by invoking COMPUTE, and in Figure 2 **(6)**, obtain a secret next hop information k_f. By invoking RECOVER, the border routers in the clique c_1 (including e_1) find that k_f is the link to their next hop for dst.

E. Formal Solution

We describe our solution to the shortest path MDR in a generic clique-level topology consisting of N cliques and M destinations. This problem is addressed by a protocol that runs a distance vector routing algorithm between cliques, while each secret distance is invisible from underlying InPs by our primitives as follows.

A clique has K clique-level links and S ($S \leq M$) clique-stub links. We consider constructing a *clique-level routing table* for a given clique, $\boldsymbol{r} = (r_1, \ldots, r_M)$, where r_m takes the link ID $k \in \{0, \ldots, K\}$ of the link to the next hop clique for each destination $m \in \{1, \ldots, M\}$. Link ID $k \in \{1, \ldots, K\}$ is mapped to each of K clique-level links and $k = 0$ is reserved for S clique-stub links, where $r_m = 0$ indicates that the packets for the destination m should be forwarded from this clique to the direct clique-stub link (i.e., the shortest intra-subslice path) to m. Each of its clique-level link $k \in \{1, \ldots, K\}$ has length d_k, and its clique-stub link to each destination m has length e_m. ($e_m = \infty$ if there is no such clique-stub link to m.)

First, as an input to the protocol, the clique shares its local topology information $\boldsymbol{d} = (d_1, \ldots, d_K)$ and $\boldsymbol{e} = (e_1, \ldots, e_M)$ by invoking $[\boldsymbol{d}]$ = SHARE($\boldsymbol{d}$) and $[\boldsymbol{e}]$ = SHARE($\boldsymbol{e}$). The shares $[\boldsymbol{D}^1]$ of its own distance vector $\boldsymbol{D}^1 = (D^1_1, \ldots, D^1_M)$ is initialized to $[\boldsymbol{e}]$, where D^1_m is the current shortest distance to each destination m at the beginning of the step 1.

Then, at each step t ($1 \leq t$), the clique transfers $\boldsymbol{D}^t$ to its neighbors via each clique-level link $k = 1, \ldots, K$ by invoking TRANSFER($k, [\boldsymbol{D}^t]$). In turn, via each link k, this clique receives the shares $[\boldsymbol{D}^t_k]$ of a neighbor's distance vector $\boldsymbol{D}^t_k = (D^t_{k,1}, \ldots, D^t_{k,M})$, where $D^t_{k,m}$ is the current shortest distance from this neighbor connected via link k to each destination m at the beginning of the step t. Using these shares $[\boldsymbol{D}^t_1], \ldots, [\boldsymbol{D}^t_K]$ of distance vectors, this clique obtains the shares of its new distance vector defined as a function

$$\begin{aligned} \boldsymbol{D}^{t+1} &= \mathit{UpdateDistance}(\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{D}^t_1, \ldots, \boldsymbol{D}^t_K) \\ &= \left(\min_{0 \leq k \leq K} C^t_{km} \middle| \, m = 1, \ldots, M\right), \end{aligned} \quad (1)$$

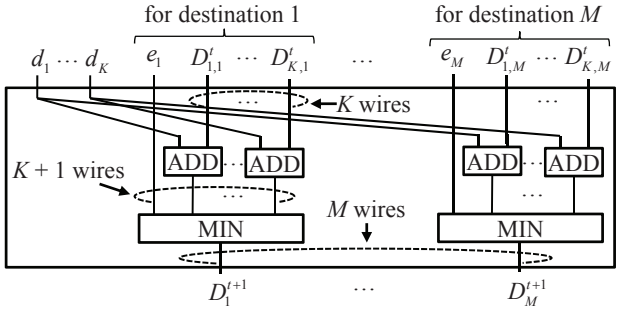

Fig. 3. Block-level (not gate-level) circuit representation of *UpdateDistance* defined in Eq. (1)

where

$$C^t_{km} = \begin{cases} e_m, & k{=}0, \\ d_k + D^t_{km}, & k{\neq}0, \end{cases}$$

by invoking $[\boldsymbol{D}^{t+1}]$ = COMPUTE(*UpdateDistance*, $[\boldsymbol{d}]$, $[\boldsymbol{e}]$, $[\boldsymbol{D}^t_1], \ldots, [\boldsymbol{D}^t_K]$).

Finally, this protocol converges at a step t_{max} (i.e., the diameter of the network, which is estimated as at most 10 in a Tier-1 network [19]). The border routers in the clique obtain their routing table as an output by computing a function

$$\begin{aligned} \boldsymbol{r} &= \mathit{NextHop}(\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{D}^{t_{max}}_1, \ldots, \boldsymbol{D}^{t_{max}}_K) \\ &= \left(\operatorname{argmin}_{0 \leq k \leq K} C^{t_{max}}_{km} \middle| \, m = 1, \ldots, M\right), \end{aligned} \quad (2)$$

by invoking $[\boldsymbol{r}]$ = COMPUTE(*NextHop*, $[\boldsymbol{d}]$, $[\boldsymbol{e}]$, $[\boldsymbol{D}^{t_{max}}_1]$, $\ldots$, $[\boldsymbol{D}^{t_{max}}_K]$) and recovering $\boldsymbol{r}$ = RECOVER($[\boldsymbol{r}]$).

IV. Feasibility Study

To verify feasibility of the proposed solution described in Section III-E, we examine extra latency incurred by SMC protocol is comparable with respect to the convergence time in typical routing algorithms. At every step of the solution, each clique of routers invokes COMPUTE on *UpdateDistance* implemented by the generic SMC protocol [17]. The SMC protocol requires each router to exchange messages with every other router in its clique and to perform computation on the received messages, thus, incurs extra latency for both computation and communication compared to non-secure versions of routing protocols.

The latency of each *UpdateDistance* function, T_{update}, is

$$T_{update} = T_{comm} + T_{comp} \quad (3)$$

where T_{comm} and T_{comp} are latencies for communication and computation, respectively.

Figure 3 shows logic circuit for the function *UpdateDistance*. Although a function in SMC is represented in a logic circuit of AND, OR, and NOT gates, each gate involves communications among routers unlike the usual logic circuits. Thus, T_{comm} depends on the $size$ (the total number of gates) and the $depth$ (the number of gates computed in sequence due to their dependency) of the circuit, and is formulated as

$$T_{comm} = size \cdot s/B + depth \cdot P \quad (4)$$

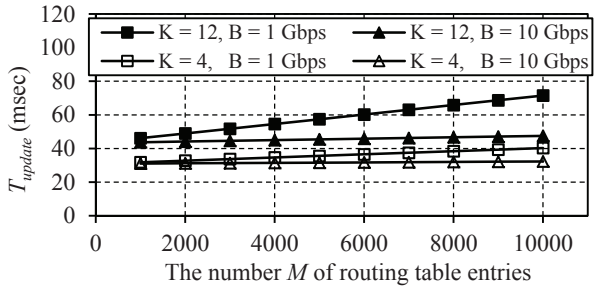

Fig. 4. Latency of an invocation of COMPUTE on *UpdateDistance*

where s is the message size, B and P are the bandwidth and one-way propagation delay, respectively, of peering links connecting border routers in a clique. Appendix A shows $size = 74MK$ and $depth = 10 + 9\log(K+1)$ hold for each *UpdateDistance*.

For each of these gates in the circuit, each router performs a set of computations on small integer operands. Since, as shown in Figure 3, the circuit in question can be decomposed per each destination, we can leverage parallel computation as follows. T_{comp} can be multiplicatively reduced as follows,

$$T_{comp} = size \cdot T_{gate}/n_{parallel} \tag{5}$$

where T_{gate} is the time required for computation per gate and $n_{parallel}$ is the degree of parallelism.

We evaluate the latency T_{update} derived from the equation (3) with typical SP's network configurations. Figure 4 shows T_{update} as functions of the number of routing table entries, M. As the SMC protocol is performed by an inter-subslice clique (i.e., peering routers collocated together within a single city area [9], [15]), and thus P is set to 1 millisecond. Since such short peering links have large bandwidth, B is set to 1 Gbps and 10 Gbps. The degree K depends on the topology of SP's slice. According to actual measurement results [20], we set $K = 12$ (average degree of backbone routers) and $K = 4$ (average degree of POPs). The message size s needs to be 3 bits, since the secret sharing scheme [18] encodes one bit of secret into shares of size roughly $\log L$ bits and we suppose $L = 6$ as the number of tier-1 InPs that the SP operates on. According to [21], a recent commodity 1.35 GHz GPGPU can perform computation for each gate (5 additions, 4 multiplications and one modulo [2]) in 57 clocks, which is $T_{gate} = 42$ nanoseconds on each core, and since it has 240 cores, $n_{parallel} = 240$. Memory bandwidth is not considered since it is much larger than network bottleneck.

From Figure 4, we observe that the latency overhead T_{update} is sub-hundred milliseconds for a large routing table with up to 10k entries. As a result, the proposed solution will converge within a second because it requires as many number of invocations of *UpdateDistance* as the diameter of the network, which is estimated as at most 10 in a Tier-1 network with 471 routers [19]. Since the convergence time of well-engineered OSPF is in the order of sub-seconds [22], we conclude that the convergence time in the proposed solution is comparable to that in typical routing algorithms, thus, the proposed solution is secure, yet, no worse than the existing routing algorithms in terms of convergence time.

[2]Computation for each gate is rather simple because the SMC protocol under our consideration is an information-theoretic scheme, not a cryptographic scheme.

V. Discussion

A. Correctness of MDR

For the sake of brevity, instead of providing the formal proof of the correctness of MDR calculation, this section discusses an intuitive sketch of the proof. Our approach can be viewed as running a secure version of the Bellman-Ford algorithm in a network of the cliques of routers. Information on such a clique can never be reverse-engineered by any single InP, since inputs/outputs of each clique is securely separated via SHARE/RECOVER, the local computation is conducted securely via COMPUTE, and the communication between neighboring cliques is secured via TRANSFER. Correctness and security of COMPUTE, SHARE and RECOVER are provided in SMC [14] and the correctness of routing is attributed to that of the standard Bellman-Ford algorithm [23].

B. Security of MDR

Any solution to a security problem needs thorough analysis of the security level it provides. We suppose that semi-honest InPs try to reverse engineer the SP's routing by mounting two types of attacks on the proposed solution. First, adversaries may attempt to decode confidential information from its shares stored on top of semi-honest InPs. Obviously, this attack is impossible because the secret sharing scheme [18] used in the solution is information-theoretically secure under our assumption that there is no collusion between two or more InPs. Second, side channel attack may be launched that exploits information such as timings of computation/message transmission and message size. In the proposed solution, all the primitive invocation sequence and timing are fixed and computational time and message size are constant. In other words, timing and message size of our protocol only depend on the size of the network (M and t_{max}), which are constant thus not confidential. As a result, the proposed solution has no side channel that leaks values of confidential inputs ($\boldsymbol{d}$ and $\boldsymbol{e}$) to a semi-honest InP.

C. OSPF

Although our method can be naturally applied to RIP, its application to OSPF is not so straightforward. In fact, our primitives are generic enough to be applied to any distributed algorithm, but should it be naively applied, it is suboptimal, since the SMC for Dijkstra's algorithm is costly. Alternatively, since OSPF area border routers exchange distance vectors but not link state information, our solution can be applied to the inter-area distance vector routing in OSPF. However, this approach suffers from the same performance problem as discussed in [24]. Applicability of our method to OSPF is left for our future work.

VI. Related Work

Security issues in network virtualization have just begun attracting attentions. Keller et al. [25] have identified the problem of *accountability* in hosted virtual networks. In contrast, our work addresses confidentiality and is complementary to the accountability and discusses fundamental security requirements in virtualized environments.

A few proposals exist for confidentiality of topology information in conventional inter-domain networking [26], [27], where several operators need to cooperatively provide end-to-end paths. These studies have not adopted any sophisticated computation techniques like SMC and simply hide the information not necessary for the computation or disclose some of confidential information required for the computation.

A large body of work in SMC studies distributed computation without disclosing each party's confidential information [14], [16], [17]. Unfortunately, the generic SMC protocols including [16], [17] and specific SMC protocols (e.g., privacy-preserving shortest path [28] and inter-domain routing between ASes [29]) are not applicable to distributed routing since they assume every pair of the parties has a communication channel between them, while distributed routing is to establish such logical channels. We break down the problem into smaller local SMC problems that require only the local communications. To the best of our knowledge, our work is the first to extend and apply the SMC protocol to distributed routing.

VII. Conclusion

We posit that operational confidentiality is crucial for enabling virtual collocation of SPs on top of InPs via network virtualization (NV) for the real business scenarios. We focus on Minimum Disclosure Routing (MDR) to enable an SP to route packets without disclosing routing information to InPs and propose that the extension to the generic Secure Multiparty Computation (SMC) achieves MDR securely. Our study reveals that the proposal is feasible since the extra latency overhead incurred in the convergence time in our secure routing protocol is within sub-seconds in large Tier-1 ISP networks and is comparable to the convergence time in well-engineered intra-domain routing algorithms. Our solution presented in this paper sheds light on the path for network virtualization to be used to resolve all the challenges for ISPs of today, (1) *footprint*, (2) *cost*, (3) *availability*, and especially (4) *operational confidentiality*, concurrently.

References

[1] "GENI: Global Environment for Network Innovations," http://www.geni.net/.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[3] A. Nakao, R. Ozaki, and Y. Nishida, "CoreLab: An Emerging Network Testbed Employing Hosted Virtual Machine Monitor," *Proc. ROADS '08*, December 2008.

[4] N. Feamster, L. Gao, and J. Rexford, "How to Lease the Internet in Your Spare Time," *ACM SIGCOMM Computer Communications Review*, pp. 61–64, January 2007.

[5] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications magazine*, vol. 47, no. 7, pp. 20–26, 2009.

[6] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path splicing," *ACM SIGCOMM CCR*, vol. 38, no. 4, pp. 27–38, 2008.

[7] D. McLoughlin and D. Aaker, *Strategic Market Management: Global Perspectives*. John Wiley & Sons, 2010.

[8] G. Saloner, A. Shepard, and J. Podolny, *Strategic management*. John Wiley, 2001.

[9] W. Norton, "The evolution of the US Internet peering ecosystem," *The 31st NANOG meeting*, 2004.

[10] "Steel in the Air, AT&T/Cingular Cell Tower Lease Renegotiation," http://www.steelintheair.com/Cingular-and-ATT-Wireless-Cell-Tower-Lease-Negotiations.html.

[11] P. Nattermann, "Best practice does not equal best strategy," *The McKinsey Quarterly*, vol. 2, no. 2000, pp. 22–31, 2000.

[12] W. Parkhurst, *Cisco OSPF command and configuration handbook*. Cisco Press, 2002.

[13] F. Chung, M. Garrett, R. Graham, and D. Shallcross, "Distance realization problems with applications to Internet tomography," *Journal of Computer and System Sciences*, vol. 63, no. 3, pp. 432–448, 2001.

[14] O. Goldreich, *Foundations of Cryptography, volume 2, Basic Applications*. Cambridge University Press, 2004.

[15] Qwest Business, "Qwest Network Maps," http://www.qwest-business.com/demos/network-maps.html.

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

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

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

[19] R. Fukumoto, S. Arakawa, T. Takine, and M. Murata, "Analyzing and modeling router–level Internet topology," *LNCS*, vol. 5200, pp. 171–182, 2008.

[20] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Transactions on networking*, vol. 12, no. 1, pp. 2–16, 2004.

[21] M. Papadopoulou, M. Sadooghi-Alvandi, and H. Wong, "Micro-benchmarking the GT200 GPU," Computer Group, ECE, University of Toronto, Tech. Rep., 2009.

[22] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *ACM SIGCOMM CCR*, vol. 35, no. 3, pp. 35–44, 2005.

[23] N. Lynch, *Distributed algorithms*. Morgan Kaufmann, 1996.

[24] M. Thorup, "OSPF areas considered harmful," *Private paper*, 2003.

[25] E. Keller, R. Lee, and J. Rexford, "Accountability in hosted virtual networks," in *Proc. of VISA*. ACM, 2009, pp. 29–36.

[26] J. Vasseur and A. Farrel, "Preserving topology confidentiality in inter-domain path computation using a path-key-based mechanism," RFC 5520, April 2009.

[27] "3GPP TS 23.228 V5.15.0," June 2006.

[28] J. Brickell and V. Shmatikov, "Privacy-preserving graph algorithms in the semi-honest model," *LNCS*, vol. 3788, p. 236, 2005.

[29] S. Machiraju and R. Katz, "Verifying global invariants in multi-provider distributed systems," in *Proc. SIGCOMM HotNets*, 2004, pp. 149–154.

Appendix

The $size$ and the $depth$ of the *UpdateDistance* circuit is calculated as follows. Each wire in the circuit carries a distance value at most 16 (as in RIP) and thus encoded into five bits. Each ADD block (a five-bit adder) is a sequence of five one-bit full adders (each with size 5 and depth 2), and thus has size 25 and depth 10. Since we have $M \cdot K$ ADD blocks in parallel, the upper half of the circuit has size $25MK$ and depth 10. Each MIN block (a minimum selector with $K+1$ inputs) is a $\log(K+1)$-deep binary tree of K minimum selectors with two inputs (each with size 49 and depth 9), and has size $49K$ and depth $9\log(K+1)$. Because we have M MIN blocks in parallel, the lower half of the circuit has size $49MK$ and depth $9\log(K+1)$. In total, the entire circuit of *UpdateDistance* has $size = 25MK + 49MK$ and $depth = 10 + 9\log(K+1)$.