SDN-Based Private Interconnection

Shlomi Dolev[1] Shimrit Tzur-David[1,2]

[1] Department of Computer Science, Ben-Gurion University of the Negev, Israel.
{dolev, tzurdavi}@cs.bgu.ac.il.
[2] Software Engineering Department, Azrieli - College of Engineering, Israel

1 Introduction

Cloud computing is one of the fastest growing opportunities for enterprises and service providers. Enterprises use the Infrastructure-as-a-Service (IaaS) model to build private and public clouds that reduce operating and capital expenses and increase the agility and reliability of their critical information systems. In order to fulfil these needs, service providers build public clouds to offer on-demand, secure, multi-tenant IT infrastructure to potential costumers that now can use cloud services using a public cloud infrastructure.

The Open Networking Foundation (ONF) report on the infrastructure between datacenters [**?**] states that fast-changing and demanding enterprise and carrier business requirements force changes in network architecture. However, these changes were focused on datacenter server and storage virtualization, while the underpinning network architectures have stagnated with respect to both scalability and manageability.

Due to the growth of businesses and the advent of Big Data, the private clouds are augmented with external resources known as public clouds. The use of public cloud requires efficient and well performed connectivity between private and public clouds. This resulting "hybrid" cloud should provide transfer and sharing of data. Furthermore, this transfer has to be private.

SDN enables more deterministic, more scalable, more manageable and as we present in this paper, also private virtual networks between the local datacenters that reside in the private cloud, to the public resources in the public cloud. These virtual networks are called the *hybrid cloud*.

In this work we present a private hybrid cloud in which all the information that pass across the cloud is information theoretic secured. I.e., unless there is a coalition of several routers in the cloud, the information cannot be revealed. This is done by using secret sharing scheme together with SDN to ensure privacy. Encryption with (n,k) *secret sharing* scheme $(n \geq k)$ is done by creating n shares from the data such that only by having at least k shares, the data can be decrypted. In the cloud notations, assume that the data has to be sent from the private datacenter. The source in the private datacenter creates n shares from the data and sends them to the destination at the public cloud through the hybrid cloud. The SDN controller manages the routes of these shares such that no router sees k or more shares. This way, we ensure that only the destination at the public cloud that gets all the shares, can decrypt the data, resulting in a private channel in the hybrid cloud. When $n > k$, we allow $n-k$ shares to get lost, due to congestion or even by malicious routers.

Our Contribution. The contributions of our paper are as follows; To the best of our knowledge, we are the first to use secret sharing for a unicast communication over SDN architecture. We show that secret sharing can be very useful to achieve private channel when two parties communicate over a multipath network. Whereas many papers have some contribution on the area of public cloud security, i.e., securing data in the cloud, to the best of our knowledge, we are the first to target the problem of theoretical secured channel *to* the public cloud. We show that even if there is a probability to the existence of coalition of several routers in the hybrid cloud, we can still bound the probability for privacy violation.

2 Gaining Privacy with Maximum Flow Technique

Problem Definition. As mentioned, our goal is to provide private interconnection between datacenters. In that case, we have a service provider that uses the cloud to extend its available services by using virtualized servers on the cloud. This service provider has to be able to guarantee to its customers that the interconnection between those datacenters is private, and therefore, to the eyes of any potential customer, these interconnection is transparent. We would like to have a solution that is information theoretic secured, i.e., there is no key that by revealing it, privacy is compromised.

A secret-sharing scheme is a method by which a dealer distributes n shares to parties such that only authorized subsets of at least k parties can reconstruct the secret. In our case, when one datacenter intends to communicate with another datacenter, it creates n shares of its message and distributes these shares to the network. The SDN controller routes these shares to the target datacenter in a way that no router on the way sees k or more shares. This way, we achieve a private channel between the datacenters. In our case, k can be equal to n if we assume that the channels are reliable and nodes do not omit or corrupt shares. When $n > k$, approach similar to forward erasure correcting or error correcting, such as the Berlekamp Welch technique [**?**] can be used to overcome erasures and even corruptions.

We examine the problem as graph theory problem. We are given a graph $G = (V,E)$, a source node s, and a sink node t. Each node $v \in V$ has a determined non-negative capacity c_v. Our goal is to push as much flow as possible from s to t in the graph. Each path has a flow f_{p_i}, the rule is that the sum of the flows of all paths that each node sees cannot exceed its capacity, formally, for each node v, $\Sigma_{p_i|v\in p_i} f_{p_i} \leq c_v$.

Reduction to the Maximum Flow Problem. We now show that there is a reduction between our problem and the known maximum network flow problem where each edge e in E has an associated non-negative capacity c_e, where for all non-edges it is implicitly assumed that the capacity is 0. In this problem, the goal is to push as much flow as possible from s to t in G. The rules are that no edge can have flow exceeding its capacity, and for any vertex except s and t, the flow into the vertex must equal the flow out from the vertex.

The original graph, G, does not have to be directed. In order to reduce our problem to the maximum flow problem, we expand each vertex u in G to an directed edge (u_1,u_2). The input of the algorithm is the graph G, the source and sink, s and t, and the threshold of the secret sharing scheme, k. The general idea is first to add a directed edge (u_1,u_2) for each vertex u in G (except s and t). The capacity of this new edge is the capacity of the vertex u, c_u. Then each edge (s,u) from the source is replaces by the directed edge (s,v_1). Respectively, each edge directed to the sink (v,t) is replaces by (v_2,t). Then for each other edge (u,v) in G we add, in case G is undirected, two directed edges, (u_2,v_1) and (v_2,u_1). In order to eliminate the possibility that edge sniffers will reveal the secret, the capacity of these edges is $k-1$. In our paradigm, since the capacity of the nodes in the original graph also represents $k-1$, all the capacities of all edges are equal. Note, that the expanded graph is directed to force the paths to pass through the edges of the expanded vertexes but the input graph can be either directed or undirected. In case of a directed graph, we still add a directed edge (u_1,u_2) for each vertex u, but for each other directed edge (u,v), we add only the edge (u_2,v_1) (and not also (v_2,u_1) as we did for an undirected G).

The SDN-Based Solution. The SDN Controller defines the route of each flow that occurs in the data plane. The controller computes a route for each flow, and adds an entry for that flow in each of the routers along the path. With all complex functions subsumed by the controller, routers simply manage flow tables whose entries can be populated only by the controller. Communication between the controller and the routers uses a standardized protocol and API. Most commonly this interface is the OpenFlow specification [**?**]. In an SDN architecture, each router forwards the first packet of a flow to the SDN controller, enabling the controller to decide whether the flow should be added to the router flow table. When a packet of a known flow is encountered, the router forwards it out the appropriate port based on the flow table. The flow table may include some additional information dictated by the controller. With the decoupling of the control and data planes, SDN enables applications and services to deal with a single abstracted network device without concern for the details of how the device operates. Network services see a single API to the controller. Thus, it is possible to quickly create and deploy new applications to orchestrate network traffic flow to meet specific enterprise requirements for performance or security.

In our case, after creating the n shares, the sender adds an index i, $1 \leq i \leq n$ to each share, this index becomes a part of the flow id and thus for each original flow, the controller handles n "first" packets of these n initiated flows. Furthermore, each router may have more than one entry but up to $k-1$ entries for each flow, where k is the threshold of the secret sharing scheme. By creating the paths based on the maximum flow algorithm, we can program the controller to route the shares such that no router sees k or more shares.

Coalitions and Trust. Our paradigm works as long as the nodes in the paths do not share their data. Our building block assumption is that the network is reliably managed by the controller, i.e., the controller can be trusted. If this is not the case, either by a malicious controller or by an adversary that controls the controller, the controller can simply forward all shares to a single router to reveal the secret. In order to minimize the risk, the network can be partitioned in multiple controller domains; each controller is responsible for one domain. With a wise partitioning, it will be more difficult for a single controller to break the privacy. In addition to the controller, we also assume that the routers (the nodes in the graph notations) behave according to the controller routes. If there is a coalition of two or more nodes, the current solution might fail. Assume that, with probability p, two nodes share their data, in that case, they should be considered as one node, and the algorithm should rerun with the new topology. We analyse the probability of saving the privacy of the channel under this coalition even if it is unknown to the controller. This probability depends on the graph topology and the values of n and k.

References

1. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.
2. ONF. OpenFlow-Enabled Hybrid Cloud Services Connect Enterprise and Service Provider Data Centers. ONF Solution Brief, 2012.
3. Loyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes. US Patent 4 633 470, December 1986.