A Low-Overhead Secure Communication Framework for an Inter-Cloud Environment

Ali Sajjad[1], Muttukrishnan Rajarajan[1], and Theo Dimitrakos[2]

[1] City University London, London, UK
[2] British Telecom Ltd, Adastral Park, Ipswich, UK

Abstract. Most of the current cloud computing platforms offer Infrastructure as a Service (IaaS) model, which aims to provision basic virtualized computing resources as on-demand and dynamic services. Nevertheless, a single cloud provider may not have limitless resources to offer to its users, hence the notion of an Inter-Cloud environment where a cloud can use the infrastructure resources of other clouds. However, there is no common framework in existence that allows the service owners to seamlessly provision even some basic services across multiple cloud service providers, albeit not due to any inherent incompatibility or proprietary nature of the foundation technologies on which these cloud platforms are built. In this paper we present a novel solution which aims to cover a gap in a subsection of this problem domain. Our solution offers a security architecture that enables service owners to provision a dynamic and service-oriented secure virtual private network on top of multiple cloud IaaS providers. It does this by leveraging the scalability, robustness and flexibility of peer-to-peer overlay techniques to eliminate the manual configuration, key management and peer churn problems encountered in setting up the secure communication channels dynamically, between different components of a typical service that is deployed on multiple clouds. We present the implementation details of our solution as well as experimental results detailing the overheads of our solution carried out on two commercial clouds.

1 Introduction

Most of the currently available Cloud Computing solutions are mainly focused on providing functionalities and services at the infrastructure level, e.g., improved performance for virtualization of compute, storage and network resources, as well as necessary fundamental functionality such as Virtual Machine (VM) migrations and server consolidation etc. In the cases where higher-level and more abstract concerns like dynamic configuration and application level security are needed to be addressed, existing Infrastructure as a Service (IaaS) solutions tend to focus on functional aspects only. Furthermore, if a cloud's computational and storage infrastructure resources are overloaded due to increased workloads, its service towards it clients will degrade. The idea of an Inter-Cloud [1] has been gaining much traction to address such a situation, where a cloud can borrow the required infrastructure resources of other clouds. However, in order to progress from a basic cloud service infrastructure to a more adaptable cloud service ecosystem, there is a great need for tools and services that support and provide higher-level concerns and non-functional aspects in a comprehensive manner, e.g., automatic provisioning of value-added services like application and communication security.

The OPTIMIS project [2] has been a recently completed effort in this regard, which strived to provide a holistic approach to cloud service provisioning by offering a single abstraction for multiple coexisting cloud architectures. OPTIMIS addressed various high-level concerns in this domain like trust, risk, eco-efficiency and cost, however a major concern of high importance is the provisioning of a secure communication framework to the services utilizing the resources of different cloud IaaS providers. The usage pattern of these services is usually quite flexible. on one hand they might be directly accessed and managed by end-users, and on the other hand their access and management might be brokered and orchestrated by Cloud Service Providers (CSP) or third-party Cloud Brokers [3] for their customers.

There are three fundamental steps in the life cycle of a service in a cloud computing ecosystem; the construction of the service, the deployment of the service to one or more IaaS clouds and lastly the operational management of the service. In the resulting scenarios, the presence of multiple IaaS providers in the cloud ecosystem is the key issue that needs to be addressed by any inter-cloud security solution. A major goal of service owners is to select IaaS providers in an efficient way in order to host the different components of their services on appropriate clouds. In this respect, third-party cloud brokers [3] can play a major role in simplifying the use, performance and delivery of the cloud services. These brokers can also offer an inter-mediation layer spanning across multiple cloud providers to deliver a host of optimization and value-added services which take advantage of the myriad individual cloud services e.g., aggregation of different services or arbitration for a best-match service from multiple similar services. For the numerous interaction possibilities among these parties, whatever the usage scenarios maybe, the security of data and the communication between the consumers of the service and its multiple providers is of paramount importance.

In the light of the above discussion, it is advocated that an inter-cloud security solution is highly desirable that would provide a framework enabling seamless and secure communication between the actors of a cloud ecosystem over multiple cloud platforms. Such a solution, however, has to overcome a number of challenges because of architectural limitations. This is because most of the current cloud service platforms, and the multi-tenants environments they offer, make it difficult to give the consumers of their services flexible and scalable control over the core security aspects of their services like encryption, communication isolation and key management. Secure communication is also challenged by lack of dynamic network configurability in most cloud providers, caused by the inherent limitations of the fixed network architectures offered by these providers.

In this work we address the security concerns related to flexibility, scalability and overheads that in our view must be overcome in order to provide holistic provisioning of services to consumers from multiple cloud service providers. We present the design and architecture of an inter-cloud secure communication framework that offers the features of dynamic and scalable virtual network formation, efficient and scalable key management and minimal manual configuration. This framework enables secure and private communication between the components of a service utilising resources of multiple cloud platforms. Our peer-to-peer architecture provides a single virtual network to that service as an overlay of resources from multiple cloud providers and offers the capability to efficiently and transparently run services on top of this network while catering for the dynamic growth and shrinkage of the components of the service.

The rest of the paper is organized as follows: In Section *2* we present the background and related works that address the issues related to this domain. In Section *3* we elaborate on the detailed Inter-Cloud Virtual Private Network (ICVPN) architecture. In Section *4* we present our experimental setup and the analysis of the performance results and overheads of our solution. We conclude in Section *5* with the future directions of our work.

2 Related Work

Virtual Private Networks (VPN) have been a mainstay for providing secure remote access over wide-area networks to resources in private organizational networks for a long time. Well-known tools and softwares like OpenVPN [4] are used to create secure point-to-point or site-to-site connections for authenticated remote access. However, the main problem in client/server based approaches is that they require centralized servers to manage the life cycle of all the secure connections for the participating clients, hence suffering from a single point-of-failure. Another issue is the quite complex and error prone configuration problems especially if you want to construct and manage a large-scale network not having a relatively simple topology, as it would require customized configuration on every client and even more elaborate management and routing configuration on the server-side. Another major drawback is the complexity of key distribution among all the participating clients in a VPN, as the software itself does not provide any key distribution service and all keys have to be manually transferred to individual hosts. In case of the Public Key Infrastructure (PKI) model, an additional requirement of a trusted Certificate Authority (CA) exists that has to issue individual certificates to all the servers and clients constituting a VPN, which incurs an additional communication overhead when forming a virtual private network.

There have been some other VPN solutions for large-scale networks aimed at grid and cluster computing environments, such as VIOLIN [5] and VNET [6], that do not follow a strict client/server model based approach. VNET is a layer 2 virtual networking tool that relies on a VNET server running on a Virtual Machine Monitor (VMM) hosting a virtual machine in a remote network which establishes an encrypted tunnel connection to a VNET server running on a machine (called Proxy) inside the users home network. All of the remote virtual machines communication goes through this tunnel and the goal of the Proxy is to emulate the remote virtual machine as a local host on the users home network, in effect presenting it as a member of the same LAN. The motivation of this approach is to tackle the users lack of administrative control at remote grid sites to manipulate network resources like routing and resource reservations etc. but it suffers from the previously discussing problem of complex and manual configuration though going for the simplicity of a private LAN. Also the scalability will be a big issue for the Proxy as the number of remote virtual machines grows as each will require a secure tunnel connection and corresponding virtual network interface mapped to the Proxys network interface by the VNET server software.

VIOLIN is a small-scale virtual network with virtual routers, switches and end hosts implemented in software and hosted by User-Mode Linux (UML) enabled machines as virtual appliances. It allows for the dynamic establishment of a private layer 3 virtual network among virtual machines, however, it doesnt offer dynamic or automatic network deployment or route management to setup the virtual network. Virtual links are established between the virtual appliances using encrypted UDP tunnels that have to be manually setup and are not self-configuring, making it cumbersome to establish inter-host connections in flexible and dynamic fashion.

P2P VPN solutions like Hamachi [7] and N2N [8] have come up as peer-to-peer alternatives to centralized and client/server model based VPNs. Hamachi is a shareware application that is capable of establishing direct links between computers that are behind NAT firewalls. A backend cluster of servers are used to enable NAT traversal and establish direct peer-to-peer connections among its clients. Each client establishes and maintains a control connection to the server cluster. It is mainly used for internet gaming and remote administration but suffers from scalability issues as each peer has to maintain the connection with the server as well as any other peers it wants to communicate with, ending up with the overhead of a mesh-topology. It therefore offers limited number of peers (16 per virtual network) and limited number of concurrent clients (50 per virtual network). The keys used for connection encryption and authentication are also controlled by the vendors servers and individual users do not initially control who has access to their network.

N2N is a layer 2 VPN solution which doesn't require a centralized back-end cluster of servers like Hamachi but it uses a peer-to-peer overlay network similar to Skype, where a number of dedicated super-nodes are used as relay agents for edge nodes that cannot communicate directly with each other due to firewall or NAT restrictions. The edge nodes connect to a super-node at start-up and pre-shared TwoFish [9] keys are used for link encryption. As it operates on layer 2, the users of the overlay have to configure their IP addresses etc. It also assumes node membership as relatively static with edge nodes rarely leaving or joining the network over their life cycle.

More recently, some commercial cloud computing services have been made available by different vendors that provide a virtual private network inside their public cloud offering and offering the customers some limited degree of control over this network, which is called a Virtual Private Cloud (VPC). Prime examples in this domain are Amazon Virtual Private Cloud [10], Google Secure Data Connector [11] and CohsiveFT VPN-Cubed [12]. These are aimed at enterprise customers to allow them to access their resource deployed on the vendor's cloud over an IPSec [13] based virtual private network. Although these products allow the possibility of leveraging the cloud providers' APIs to flexibly grow and shrink their networks, the management and configuration is as complex as a traditional network as components of the VPC such as internet gateways, VPN servers, NAT instances and subnets have to be managed by the customers themselves. Furthermore, the customers are required to setup an IPSec device on their premises that connects to an IPSec gateway in the VPC running as a virtual appliance which integrates the enterprises network with the VPC subnet in the cloud. Most importantly, with the exception of [12], these solutions are locked to single cloud vendor and [12] provides use of a selective set of cloud providers by placing its virtual appliances as VPN gateways in these cloud infrastructures and allowing the customers to join these gateways in a mesh topology manually.

3 Design and Architecture

In this section we present the design and architecture of our inter-cloud secure communication framework, the Inter-Cloud VPN (ICVPN). The architecture is inspired by two main techniques, namely Peer-to-Peer (P2P) Overlays [14] and VPNs [15]. Network virtualization techniques like VPNs and P2P Overlays have been shown to provide their users legacy communication functionalities of their native network environments, despite the topology, configuration and management architecture of the actual underlying physical network. This fits perfectly with our goal of providing a secure virtual private network as a service to the consumers operating on top of multiple cloud providers. All complications and complexities of managing a physical network can be handled by the overlay network, enabling the services deployed on multiple clouds to benefit from a customized communication network typically only available in physical local-area environments.

3.1 Peer-to-Peer Overlay

The core technique employed by the ICVPN is the use of two tiers of P2P overlays. A Universal Overlay (UO) forms the higher tier overlay and is used to provide a scalable and secure service infrastructure to initiate and bind multiple lower tier VPN overlays (VO) to different cloud services. The universal overlay can be initiated either by the service owner, a cloud broker or the cloud service providers. Its main purpose is to help with the bootstrapping activity of VPN peers of the VPN overlay. It also provides other functionalities such as service advertisement, service discovery mechanisms, and service code provisioning, with minimal requirement for manual configuration and administration.

This approach acts as an aggregation service for the peered overlay resources (which in this case are virtual machines) span across multiple cloud domains to help form a virtual private network. The peers of the universal overlay act as super peers for the nodes of the underlying VPN overlays and let new nodes enrol, authenticate, bootstrap and join a particular VPN overlay based on the cloud service requiring a VPN service.

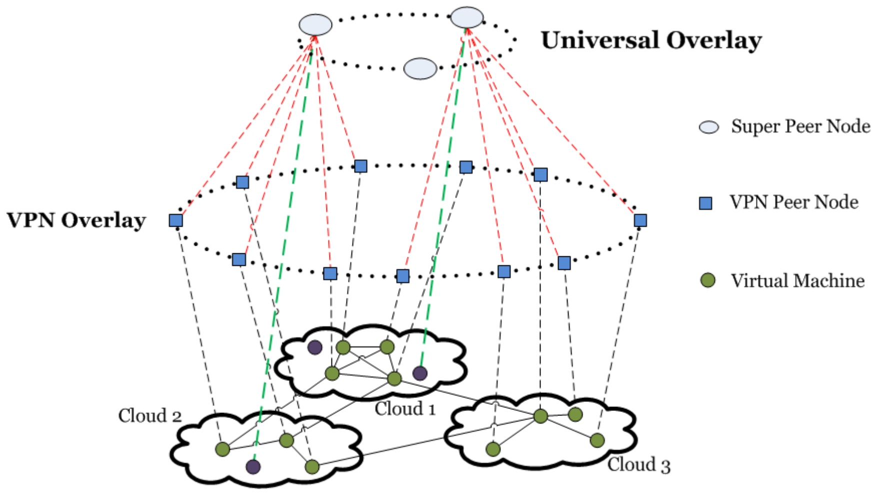

Fig. 1. Two-tiered overlay architecture for the Inter-Cloud VPN solution

As depicted in Fig.1, the service owner/provider or the cloud broker could itself be a peer in the universal overlay and a subset of the universal overlay peers can act as super-peers for the peer nodes of the VPN overlay for a particular cloud service. This enables the service owner/provider or the cloud broker to publish and propagate configuration and other data throughout the universal overlay. This is done by using its super peer as the initial dissemination point and then taking advantage of the scalable content-sharing capabilities of the Distributed Hash Table (DHT) feature of the overlay. The universal overlay peers can join and leave the system dynamically and additional VMs from the cloud providers can be provisioned to act as the universal overlay peers as well.

To join the universal overlay, each peer needs to acquire a unique identification number (PID). This is generated by the peer itself on its first initialization on a VM as a unique 160-bit random number. It also needs some bootstrapping data to validate itself with a super peer for admission into the overlay. The bootstrapping data consists of the IP addresses of the super peers, the ID of the service that this particular VM belongs to and some security-related parameters described later. This data is embedded in a secure cache on the VM by a VM contextualization service [16] when it is provisioned for the service deployment and the same contextualization service is used to install the peer-to-peer client in the VM.

Once the peer has joined its overlay, it needs to discover its neighbours and gather additional configuration data to establish secure tunnels with them so that the deployed service can communicate securely with its different components deployed on different cloud platforms. In order to achieve this, we use the following scheme based on the Functional Encryption predicates.

3.2 Secure Service based Resource Discovery

After joining a peer-to-peer overlay, each peer needs to discover its neighbours for the resources they provide to achieve the secure communication goals of the application using the overlay. Most commonly these resources are configuration and credentials data and the secure communication goals of a typical application pertain to encryption of traffic associated with specific ports and protocols. In most structured P2P systems, the peers must maintain lists of neighbours to achieve this goal efficiently. To populate these lists, peers in a structured overlay usually use distributed trackers or IP Multicast [17]. Although IP multicast offers the feature of scalable group communication needed for efficient resource delivery, it is not suitable for use in our system architecture, this is mainly due to its very limited deployment by ISPs and network carriers as well as the complexity of its architectural design [18]. A P2P tracker is a specialised service that introduces other peers of an overlay to the requesting peer. In order to perform this function, a tracker keeps track of peers as soon as they make a request. A tracker may be deployed as a dedicated server or distributed among the peers of an overlay it self. At first glance the distributed trackers based approach looks very suitable for use in our system architecture as it maps nicely with the functionality of the peers of the universal overlay i.e. well known bootstrapping points. However, this approach is vulnerable to malicious attacks like Denial-of-Service (DoS) and Sybil attacks [19], in which the attacker can subvert the functioning of a peer-to-peer overlay by creating and using a large number of false identities. For ICVPN we focus on the Sybil attack where a malicious attacked can impersonate a number of the Universal Overlay peers to subvert the process of resource discovery.

A common way of dealing with this issue is to use some trusted authority to allocate peer IDs to the participating peers and the peers validate each other by querying the central authority with a validation request. In our model, it can work by designating a set of super peers as the Certificate Authorities (CA) for the overlays other peers. The CA can assign peer IDs to the peers and signs a certificate that binds the serviceID of the cloud service making use of our solution and peer ID within the public certificate of the peer for a limited time duration. The peer then can use this signed certificate to authenticate itself with other peers in the overlay. However, using this Trusted Third Party (TTP) model to validate peers and allocate them their identities can introduce substantial communicational and computational overhead, especially as the number of peers in the overlay increases. We propose a decentralized solution that overcomes the above mentioned scalability problems by utilizing a functional encryption based scheme [20].

In a generic functional encryption scheme, a decryption key describes a function of the encrypted data to the user. This function $F(\cdot,\cdot)$ is modelled as a Turing Machine [21] and an authority possessing a master secret key (msk) can generate a key skk that can be used to compute the function $F(k,\cdot)$ on some encrypted data. Identity-Based Encryption [22], [23], [24], Searchable Encryption [25] and Attribute-Based Encryption [26] are some examples of a Functional Encryption scheme. To describe it more formally but briefly, A functional encryption scheme (FE) for a functionality F dened over (K, X) is a sequence of four algorithms (setup, keygen, encryption, decryption) satisfying the following correctness condition for all $k \in K$ and $x \in X$ is given in Table 1.

Table 1. Four-tuple Functional Encryption

Sequence	Explanation
$setup(1) \rightarrow (pp, msk)$	Generate a public and master secret key pair
$keygen(mk, k) \rightarrow sk$	Generate secrekt key for k
$enc(pp, x) \rightarrow c$	Encrypt message x
$dec(sk, c) \rightarrow y$	Use sk to decrypt c

For ICVPN, we employ a special case of Functional Encryption which falls under the category of systems known as the predicate encryption schemes with public index. For our scheme we make use of the system defined in [26] as Attribute-Based Encryption (ABE), where the decision that which users can decrypt a ciphertext is based on the attributes and policies associated with the plaintext message and the user. In this scheme an authority creates secret keys for the users of the system based on attributes or policies for each user and anyone can encrypt a plaintext message by incorporating the appropriate attributes or policies in the scheme. We describe the simplified step-wise description of our version of this scheme as follows:-

i. A super peer sets up its own Master Secret ms and Public Parameters pp
ii. The super peer generates a private key for itself using the attributes *ServiceID* and *SuperPeerID* as the public key i.e. $Pub_{SP} = ServiceID \wedge SuperPeerID$, for each service the super peer is managing
iii. After bootstrapping, the VPN peer sends a provisioning request to the super peer encrypted by the super peers public key (Pub_{SP})
iv. The super peer issues a private key to the VPN peer encrypted by its own private key, against the public key with attributes $Pub_{VPN} = VMID \wedge PeerID \wedge ServiceID$
v. The super peer inserts the VPN peers public key in the universal overlay DHT to keep a record of issued private keys, against the $key(ServiceID) = value(List\ of\ VMID)$ and for each peer; $key(VM_{ID_i}) = value(Pub_{VPN_i})$

vi. The VPN peer query the universal overlay DHT for lists of other peers and gets the result of $key(ServiceID)$ encrypted using $Pub_{VPN_x} = PeerID \wedge ServiceID$

In Section *5* we show the evaluation results of the performance overhead of using our secure resource discovery scheme as compared to that using a PKI-based approach described earlier in this section.

3.3 Secure Virtual Private Connections

The key feature of our ICVPN is establishing a secure communication network between the peers of the overlay formed over a collection of cloud providers infrastructure. Therefore, after successfully joining the overlay network to become part of a service, a VPN peer starts the process of creating secure tunnels to the other peers of the service it wants to communicate with, according to the functional operations of that particular service. To achieve this, we make use of IPSec [13] to authenticate and encrypt each IP packet of a communication session between the peers, thus creating end-to-end tunnels which provide protection against eavesdropping, message tempering and message forgeries. For establishing mutual authentication between peers at the beginning of the session and negotiation of cryptographic keys to be used during the session, we employ the Internet Key Exchange protocol [27], which can make use of standard cryptographic primitives like public key cryptography [28] and AES [29]. In our approach, we make use of an authenticated Diffie-Hellman based scheme to derive a secure session key which is used in the AES-CBC mode to ensure the confidentiality of the traffic exchanges between the peers using the tunnel [30]. The session keys generated for the IPSec communication are valid for a short period of time and when the keys expire the protocol is run again to come up with new session keys to maintain the IPSec tunnels.

Another practical advantage of this approach is the reuse of existing frameworks and tools which have been thoroughly tried and tested in a myriad of different domains, are widely used and have been adopted in both academic and commercial domain. The main components of the peer-to-peer client used to construct a virtual private network in our model are shown in Fig. 2. These include the standard components required to form a structured peer-to-peer overlay like the Distributed Hash Table (DHT) service, which basically acts as the command-and-control channel for the ICVPN solution, key-based routing, peer discovery, bootstrapping service and overlay maintenance service. All of these services are provided by a modified Kademlia implementation. In addition to these peer-to-peer specific components, we have a secure content storage for the client where sensitive data like keys, passwords, and security tokens etc. are stored. The configuration component is integrated with the overlays DHT so that the clients behaviour can be modified dynamically by push new configurations to it from the super peers. The configuration component manages both the peer-to-peer related configurations as well as the policies used to configure the

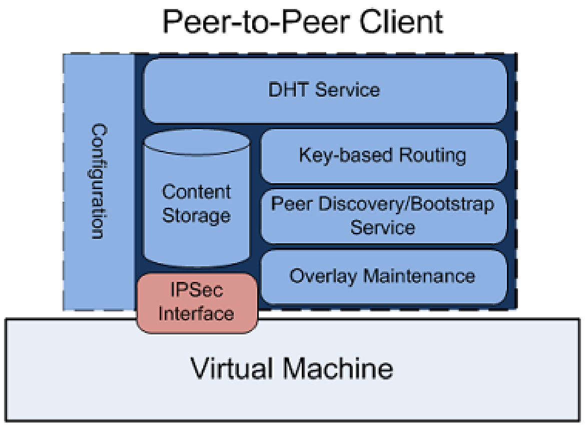

Fig. 2. Architecture of a P2P client in the VPN overlay

IPSec tunnels between the peers for the use of the higher-level services using the client to provide the secure communication framework.

The P2P client software sets up and configures the IPSec security associations according the service network security policy, which is advertised by the service owner through the DHT of the Universal Overlay. The peers of the underlying VPN overlay periodically check for any update in the security policy and apply and enforce any changes on the kernel of the VM through the P2P client's IPSec interface.

4 Implementation

We implement a working prototype of ICVPN using the Java programming language on virtual machines running the Linux operating system. Our implementation is built using open source libraries and APIs. Specifically, we use the BouncyCastle library [31] for most of the cryptographic operations, the cpabe library [32] for the Functional Encrytion based secure resource discovery, and the TomP2P library [33] for its implementation of the Kademlia [34] peer-to-peer protocol and the overlay DHT. In addition, we use BT Compute Cloud [35] and Flexiant Cloud [36] as our cloud service providers.

5 Evaluation

In this section we present the results of a series of experiments we conducted to evaluate the effect of our prototype ICVPN solution upon the network performance of a service deployed on two different cloud IaaS providers. We use a 3-tier web service comprising of database, business logic and presentation components deployed on nine virtual machines hosted on the clouds of British Telecom Ltd.

and Flexiant Ltd., our partners in the EU OPTIMIS project. The purpose of these experiments is to evaluate the architecture being proposed, in terms of service latency and service throughput, in a practical scenario with a service deployed over a real wide-area network, with the BT cloud geographically located in Ipswich, England and Flexiant cloud located in Livingston, Scotland.

5.1 Service Latency

We define service latency as the inter-cloud round-trip time taken by a HTTP request, issued by a service component on one cloud, to get a response from the target service component on a different cloud. We compare the latency between the components of the service deployed on different cloud providers, as the latency between the components in the same cloud is almost negligible as they are usually hosted on the same hyper-visor. We measured the latency by using the round-trip delay of an HTTP HEAD request/response pair, as the components of the web service communicate with each other using HTTP protocol and ICMP, the de facto latency measurement protocol, is blocked in the networks of our cloud providers. We measured the latency readings by running *10* experiments very hour for a period of *24* hours, firstly without using the ICVPN solution and then with it.

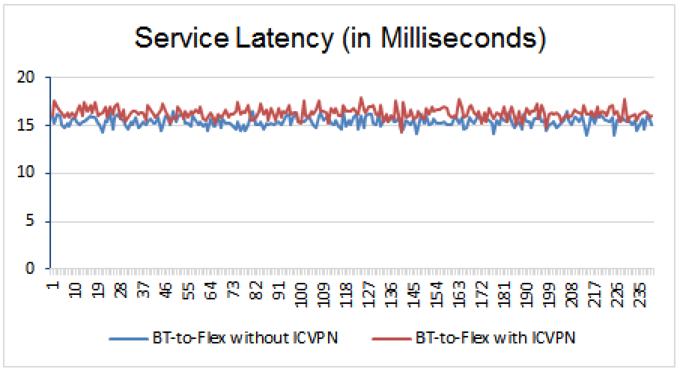

Fig. 3. Service latency of 240 round-trip time experiments from BT to Flexiant clouds

Looking at the results shown in Fig. 3 and Fig. 4, we can see that using our solution only has a small impact on the HTTP latency, increasing it just by about *5%*. For ease of analysis, we collect the network traffic dump when running our experiments, using the tcpdump packet sniffer. We found out from the traffic dumps that the increased delay we encountered is mostly due to the additional packets transmitted and received by the peers for the purposes of key exchange and cryptographic primitives negotiation when establishing an IPSec tunnel. After this initial handshake phase is over, the latency performance is almost same in the comparative experiments.

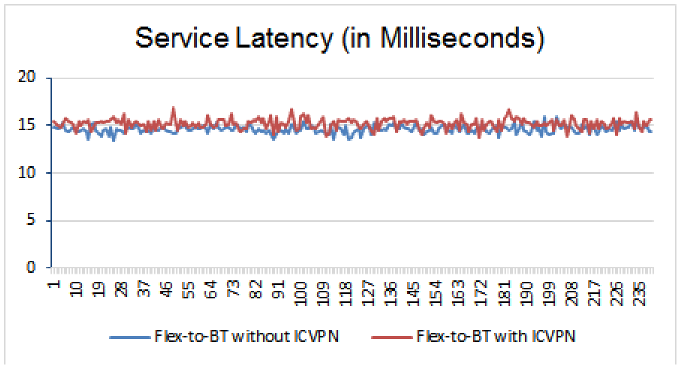

Fig. 4. Service latency of 240 round-trip time experiments from Flexiant to BT clouds

5.2 Service Throughput

We define service throughput as the inter-cloud network throughput between service components deployed on different clouds. We measure the throughput between components of the service deployed on different cloud providers by using Iperf [37], a commonly used network testing tool. We measured the throughput in both directions by transferring 30 MB data, a size chosen empirically to saturate the WAN links between the components and get the throughput results representing realistic conditions. We obtained the throughput measurements by running *10* experiments every hour for a period of *10* hours, firstly without using the ICVPN solution and then applying the security policy to tunnel the traffic through IPSec.

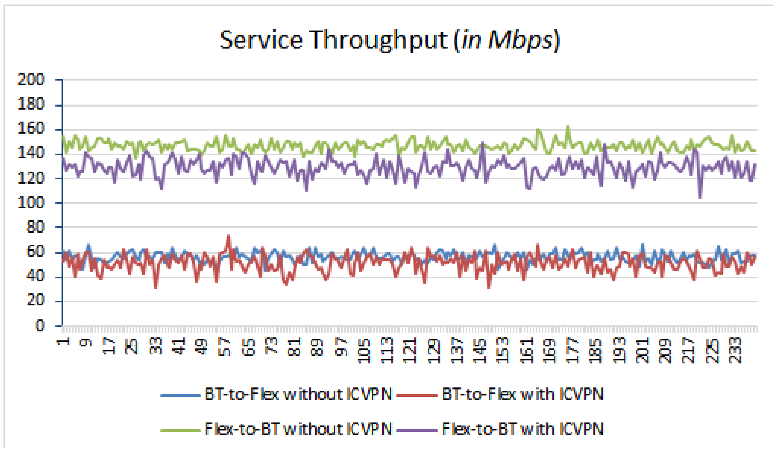

Fig. 5. Service throughput of 240 data transmission experiments in both directions between BT and Flexiant clouds

From the throughput results shown in Fig. 5, the first thing that stands out is the difference in the throughput values depending on the direction of transferring the data. Although we don't have the detailed knowledge of the underlying physical wide-area network connectivity between the two cloud service providers, such readings are not unheard of in this domain and are usually due to differences in upstream and downstream traffic conditions, different routes chosen by the IP packets or network configuration issues. Irrespective of that, by looking at the comparative results it is clear that we just incur a small overhead in the throughput, of about *10%*. By analysing the traffic dumps generated from the throughout test, we can attribute this overhead to the IKE and IPSec handshakes in addition to the extra time taken by the VM kernel in encrypting and encapsulating 30 MB of data for each throughput test.

5.3 Secure Resource Discovery Overheads

One of the main overheads in peer-to-peer overlays related to the cost of the resource discovery after the peers have bootstrapped. Securing this process further adds to this overhead but in an effort to characterise the effect of our secure resource discovery mechanism, we compare it with an alternate design of a PKI-based system where the super peers have the functionality of a Certificate Authority (CA), each peer is issued a signed certificate upon authenticated completion of the bootstrapping process and queries the Universal Overlay DHT for resource discovery and gets the resulting data back which is encrypted by the owning peer using its private key.

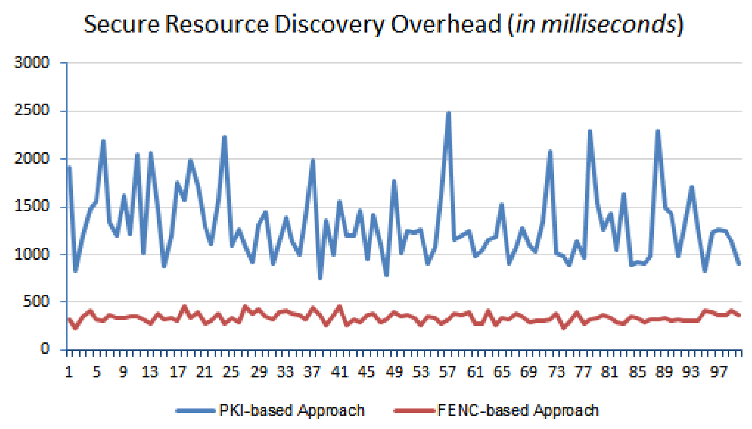

Fig. 6. Secure resource discovery for 100 runtime analysis between PKI and Functional Encryption approaches

We remove the cost of the DHT lookups from our measurements as their theoretical complexity is known to be $O\ log(n)$ for Kademlia DHT but due to the nature of actual runtime measurements they can add unhelpful noise to the data. We define the runtime cost for both designs as the time duration between the start and end of the secure resource discovery process.

From the results shown in Fig. 6, the mean runtime of the PKI-based design is *1313.52* milli-seconds whereas that for our Functional Encryption based scheme is *338.81* milli-seconds. This shows that our scheme incurs about *74.2%* less overhead than a PKI based scheme.

6 Conclusion

In this paper, we present a secure communication framework for services deployed in an inter-cloud environment. We employ the robustness and scalability afforded by structure peer-to-peer overlays to join virtual machines running on different cloud IaaS providers with each other using IPSec tunnels, hence providing confidentiality, authentication and integrity for all the data exchanged between different components of a cloud service. Our solution needs minimal manual configuration as peers are automated to discover the information needed to perform their operations from the Universal Overlay. We also provide a distributed and scalable key management solution for the consumption of the virtual machines to set-up the secure communication channels. Our solution supports the dynamic addition and removal of nodes from the VPN overlay as we use the peer-to-peer DHT not just as a command and control channel for managing the VPN peers but also for the churn management of peers in the VPN overlay. We have evaluated a prototype implementation based on experiments conducted in realistic conditions, over multiple cloud infrastructure environments and found minimal latency, throughput and security overheads of creating and maintaining the ICVPN connections among the participating VMs of a service.

References

1. Buyya, R., Ranjan, R., Calheiros, R.N.: Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In: Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010. (2010)
2. Ferrer, A.J., Hernandez, F., Tordsson, J., Elmroth, E., Zsigri, C., Sirvent, R., Guitart, J., Badia, R.M., Djemame, K., Ziegler, W.: OPTIMIS: a holistic approach to cloud service provisioning. In: First International Conference on Utility and Cloud Computing. (December 2010)
3. Gartner: Cloud consumers need brokerages to unlock the potential of cloud services. http://www.gartner.com/it/page.jsp?id=1064712 (July 2009)
4. Yonan, J.: OpenVPN - an open source SSL VPN solution. http://openvpn.net
5. Jiang, X., Xu, D.: VIOLIN: virtual internetworking on overlay INfrastructure. In: In Proc. Of The 2nd Intl. Symposium On Parallel And Distributed Processing And Applications. (2003)
6. Sundararaj, A.I., Dinda, P.A.: Towards virtual networks for virtual machine grid computing. In: In Proceedings of the 3rd USENIX Virtual Machine Research And Technology Symposium. (2004)

7. LogMeIn: Hamachi - a zero-configuration virtual private network. https://secure.logmein.com/products/hamachi2
8. Deri, L., Andrews, R.: N2N: a layer two Peer-to-Peer VPN. In: Resilient Networks and Services. Volume 5127 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 53–64
9. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: The Twofish encryption algorithm: a 128-bit block cipher. John Wiley & Sons, Inc., New York, NY, USA (1999)
10. Amazon: Virtual private cloud. http://aws.amazon.com/vpc
11. Google: Secure data connector. http://code.google.com/securedataconnecto
12. CohesiveFT: VPN-Cubed. http://www.cohesiveft.com/vpncubed
13. Doraswamy, N.: IPSec : the new security standard for the Internet, intranets, and virtual private networks. 2nd ed. edn. Prentice Hall PTR (2003)
14. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay networks. SIGCOMM Comput. Commun. Rev. (January 2002)
15. Tanenbaum, A.S., Wetherall, D.J.: Virtual private networks. In: Computer Networks. 5th edn. Prentice Hall (October 2010) 821
16. Armstrong, D., Djemame, K., Nair, S.K., Tordsson, J., Ziegler, W.: Towards a contextualization solution for cloud platform services. In: CloudCom. (2011) 328–331
17. Deering, S.E., Cheriton, D.R.: Multicast routing in datagram internetworks and extended lans. ACM Trans. Comput. Syst. (May 1990)
18. Diot, C., Levine, B., Lyles, B., Kassem, H., Balensiefen, D.: Deployment issues for the ip multicast service and architecture. Network, IEEE (2000)
19. Douceur, J.: The sybil attack. In: Peer-to-Peer Systems. Springer Berlin / Heidelberg (2002)
20. Dan Boneh, A.S., Waters, B.: Functional encryption: a new vision for public-key cryptography. Commun. ACM **55**(11) (2012) 56–64
21. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London mathematical society **42** (1936) 230–265
22. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Proceedings of CRYPTO 84 on Advances in cryptology, New York, NY, USA, Springer-Verlag New York, Inc. (1985) 47–53
23. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: CRYPTO. (2001) 213–229
24. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: IMA Int. Conf. (2001) 360–363
25. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: EUROCRYPT. (2004) 506–522
26. Hohenberger, S., Waters, B.: Attribute-based encryption with fast decryption. In: Public Key Cryptography. (2013) 162–179
27. Kaufman, C.: Internet key exchange protocol version 2 (ikev2). In: RFC 5996. (2010)
28. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory (November 1976)
29. 197, F.I.P.S.P.: Announcing the advanced encryption standard (aes) (2001)
30. Housley, R.: Using advanced encryption standard (aes) ccm mode with ipsec encapsulating security payload (esp) (2005)
31. of the Bouncy Castle, L.: Bouncy castle java cryptography apis. http://www.bouncycastle.org/java.html (2013)

32. Bethencourt, J., Sahai, A., Waters, B.: Advanced crypto software collection: The cpabe toolkit. http://acsc.cs.utexas.edu/cpabe (2011)
33. Bocek, T.: TomP2P: A P2P-based high performance key-value pair storage library. http://tomp2p.net (2012)
34. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems, Springer-Verlag (2002)
35. BritishTelecom: BT Compute Cloud. https://cloud.btcompute.bt.com (2013)
36. Flexiant: Flexiant, your cloud simplified. http://www.flexiant.com (2013)
37. Ajay Tirumala, L.C., Dunigan, T.: Measuring end-to-end bandwidth with iperf using web100. In: Web100, Proc. of Passive and Active Measurement Workshop. (2003)