A Secure and Scalable Communication Framework for Inter-Cloud Services

Ali Sajjad

School of Mathematics, Computer Science & Engineering
City University London

This dissertation is submitted for the degree of
Doctor of Philosophy

September 2015

THE FOLLOWING PARTS OF THIS THESIS HAVE BEEN REDACTED FOR COPYRIGHT REASONS:

p 7: **Fig 1.2.** International Data Corporation survey.
p 8: **Fig 1.3.** International Data Corporation survey.
p 43: **Fig 2.13.** Architectural view of Google Secure Data Connector.

Supervisors

Prof. Muttukrishnan Rajarajan (City University London)

Prof. Andrea Zisman (The Open University)

Prof. Theo Dimitrakos (British Telecom/University of Kent)

ابّو ، امّی ، رحمٰی ، زعیم اور عیشه کے لیے

"For Abbu, Ammi, Ruhma, Zaim and Eesha"

رازِ حیات پوچھ لے خضرِ نجسته گام سے
زنده هرایک چیز هے کوششِ نا تمام سے
(علامه محمّد اقبالؒ)

Ask Khidr, him of the blessed feet, for the secret of life
'Everything that is alive is due to unsuccessful effort'
(Iqbal)

Acknowledgements

I would like to express my gratitude to my advisers, Muttukrishnan Rajarajan, Andrea Zisman and Theo Dimitrakos, firstly for agreeing to take me on as a PhD student and then for their continuous guidance in form of fruitful discussions and useful advises. Special thanks to Theo for arranging further funding for me after the third year, without which it would have been very difficult, if not impossible, for me to complete this thesis.

I am also grateful to Constantino Carlos Reyes-Aldasoro and Steven Furnell for being on my committee and for their timely corrections and comments that helped me greatly in improving this thesis.

On a personal note, I would like to thank and pay tribute to my parents who have made immense efforts and sacrifices throughout my life so that I could focus all of my energies on my academic pursuits. I admit that I have not been able to do justice to their efforts and their prayers and unconditional support are the only reasons that I have been able to accomplish anything. I am also deeply thankful to my wife, who has supported me in these last four years while pursuing her own doctoral research and raising our two wonderful children in parallel. Thank you Ruhma for your phenomenal multi-tasking skills and for being a rock for our family. Lastly, I do not know how to express in words my eternal love for the sources of my motivation and inspiration, my brilliant children Zaim and Eesha, whose laughter takes away all of my worries and weariness. I love you.

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Ali Sajjad
September 2015

Abstract

A lot of contemporary cloud computing platforms offer Infrastructure-as-a-Service provisioning model, which offers to deliver basic virtualized computing resources like storage, hardware, and networking as on-demand and dynamic services. However, a single cloud service provider does not have limitless resources to offer to its users, and increasingly users are demanding the features of extensibility and interoperability with other cloud service providers. This has increased the complexity of the cloud ecosystem and resulted in the emergence of the concept of an Inter-Cloud environment where a cloud computing platform can use the infrastructure resources of other cloud computing platforms to offer a greater value and flexibility to its users. However, there are no common models or standards in existence that allows the users of the cloud service providers to provision even some basic services across multiple cloud service providers seamlessly, although admittedly it is not due to any inherent incompatibility or proprietary nature of the foundation technologies on which these cloud computing platforms are built. Therefore, there is a justified need of investigating models and frameworks which allow the users of the cloud computing technologies to benefit from the added values of the emerging Inter-Cloud environment. In this dissertation, we present a novel security model and protocols that aims to cover one of the most important gaps in a subsection of this field, that is, the problem domain of provisioning secure communication within the context of a multi-provider Inter-Cloud environment. Our model offers a secure communication framework that enables a user of multiple cloud service providers to provision a dynamic application-level secure virtual private network on top of the participating cloud service providers. We accomplish this by taking leverage of the scalability, robustness, and flexibility of peer-to-peer overlays and distributed hash tables, in addition to novel usage of applied cryptography techniques to design secure and efficient admission control and resource discovery protocols. The peer-to-peer approach helps us in eliminating the problems of manual configurations, key management, and peer churn that are encountered when

setting up the secure communication channels dynamically, whereas the secure admission control and secure resource discovery protocols plug the security gaps that are commonly found in the peer-to-peer overlays. In addition to the design and architecture of our research contributions, we also present the details of a prototype implementation containing all of the elements of our research, as well as showcase our experimental results detailing the performance, scalability, and overheads of our approach, that have been carried out on actual (as opposed to simulated) multiple commercial and non-commercial cloud computing platforms. These results demonstrate that our architecture incurs minimal latency and throughput overheads for the Inter-Cloud VPN connections among the virtual machines of a service deployed on multiple cloud platforms, which are 5% and 10% respectively. Our results also show that our admission control scheme is approximately 82% more efficient and our secure resource discovery scheme is about 72% more efficient than a standard PKI-based (Public Key Infrastructure) scheme.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3DES Triple Data Encryption Standard

AES Advanced Encryption Standard

AH Authentication Header

API Application Programming Interface

B2B Business-to-Business

BT British Telecom Ltd.

CA Certificate Authority

CaaS Connectivity as a Service

CD-ROM Compact Disc Read-Only Memory

CRM Customer Relationship Management

CSP Cloud Service Provider

dDoS Distributed Denial-of-Service

DH DiffieHellman Key Exchange

DHT Distributed Hash Table

DoS Denial-of-Service

EC2 Elastic Compute Cloud

ESP Encapsulated Security Payload

FE Functional Encryption

GENI Global Environment for Network Innovations

HTTP HyperText Transfer Protocol

I/O Input/Output

IaaS Infrastructure as a Service

IBE Identity-Based Encryption

ICMP Internet Control Message Protocol

ICV Integrity Check Value

ICVPN Inter-Cloud Virtual Private Network

ID IDentifier

IDC International Data Corporation

IDEA International Data Encryption Algorithm

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IKEv2 Internet Key Exchange protocol version 2

IP Internet Protocol

IPsec Internet Protocol Security

ISAKMP Internet Security Association and Key Management Protocol

ISO International Organization for Standardization

IT Information Technology

KDC Key Distribution Center

L2TP Layer 2 Tunnelling Protocol

LAN Local Area Network

Mbps Megabits per second

MitM Man-in-the-Middle

MQV Menezes-Qu-Vanstone

NAT Network Address Translation

NIC Network Interface Card

NIST National Institute of Standards and Technology

P2P Peer-to-Peer

PaaS Platform as a Service

PBC Pairing-based Cryptography

PID Peer Identifier

PKI Public Key Infrastructure

QEMU Quick EMUlator

REST REpresentational State Transfer

RTT Round-Trip Time

S3 Simple Storage Service

SA Security Associations

SaaS Software as a Service

SADB Security Association Database

SDC Secure Data Connector

SPI Security Parameters Index

SSL Secure Socket Layer

TAP network TAP

TCP Transport Control Protocol

TLS Transport Layer Security

TTL Time-To-Live

TTP Trusted Third Party

TUN network TUNnel

UDP User Datagram Protocol

UML User-Mode Linux

URI Universally Resource Identifier

URL Universal Resource Locator

UUID Universally Unique IDentifier

VLAN Virtual Local Area Network

VM Virtual Machine

VMM Virtual Machine Monitor

VNE Virtual Network Edge

VNG Virtual Network Generator

VNM Virtual Network Manager

VNR Virtual Network Router

VoIP Voice over IP

VPC Virtual Private Cloud

VPN Virtual Private Network

WAN Wide Area Network

ZKPP Zero-Knowledge Password Proof

Chapter 1

Introduction

1.1 Overview of Cloud Computing

Cloud computing is an extension of the Grid Computing [67] where computing resources are delivered as a service, typically over the Internet. It follows a distributed computational model of a large pool of shared, and usually, virtualised computing resources like storage, processing power, memory, applications, services, and network bandwidth. The users of a cloud computing service provider (CSP) can be provisioned and de-provisioned resources as per their demand in real-time. Its architecture can be split into two parts, front-end and back-end. The front-end is a network accessible interface to the users, organizations, and applications that are using the cloud services. The back-end is typically a large-scale data centre with a huge pool of storage, computational and network resources. The cloud service provider gives the end users access to the cloud-based applications through the front-end interface, which is usually a web interface, either in form of a web portal or REST (REpresentational State Transfer) based Application

Programming Interface (API) calls, to manage and use the back-end resources they have purchased. The end users are generally oblivious to the tools and techniques being used to provide them with these services (hence the phrase in the cloud). The main benefits thus achieved are the reduction or elimination of infrastructure costs, improved manageability and dynamic adjustment of resources to meet the changing computation, I/O or networking demands. According to the National Institute of Standards and Technology (NIST) [115], there are three basic models associated with cloud computing:-

1. Software as a Service (SaaS)

2. Platform as a Service (PaaS)

3. Infrastructure as a Service (IaaS)

In the SaaS model, software applications are offered to end users through a web browser or a thin client. These are almost entirely stored and managed in the cloud and the users do not manage the cloud infrastructure and platform on which the application is running. This is suitable for applications in the domain of social networking, collaboration, media and content management etc. Examples include Customer Relationship Management (CRM) systems like salesforce.com [136], email services like Gmail [33], Outlook.com [119] etc., online games like FarmVille [163] etc. and storage services like Wuala [74] and Dropbox [83] etc.

The PaaS model provides a Computing Platform, which is usually a customised environment meant to facilitate the design, development and deployment of applications. it typically includes operating system, programming language, execution environment, database, web server and load balancer etc. This moves

the responsibility of managing the underlying hardware and software layers off the application developer and to the cloud service provider. Examples include Amazon Elastic Beanstalk [5], Google App Engine [72] and Windows Azure Compute [118].

The IaaS model provides creation of virtual hardware resources including virtual machines, virtual networks and virtualized storage. Cloud providers offer these resources on demand from their large resource pools installed in data centres across the globe. The users have to install and manage operating systems on the machines as well as their application software. Examples include services like Amazon Elastic Compute Cloud(EC2) [7], Amazon Simple Storage Service (S3) [8], Flexiant [66] and Rackspace Cloud [131].

There are primarily four cloud deployment models, as recommended by the National Institute of Standards and Technology (NIST):

1. Private Cloud
2. Public Cloud
3. Hybrid Cloud
4. Community Cloud

Private Cloud is a model in which the infrastructure is operated solely for a single organization. It might be managed internally or by a third-party and also might be hosted internally or externally. This is used usually to address concerns related to data security and trust issues.

In the Public Cloud model, the cloud services are made available to the general public by a Cloud Service Provider (CSP). The services might be free (Gmail

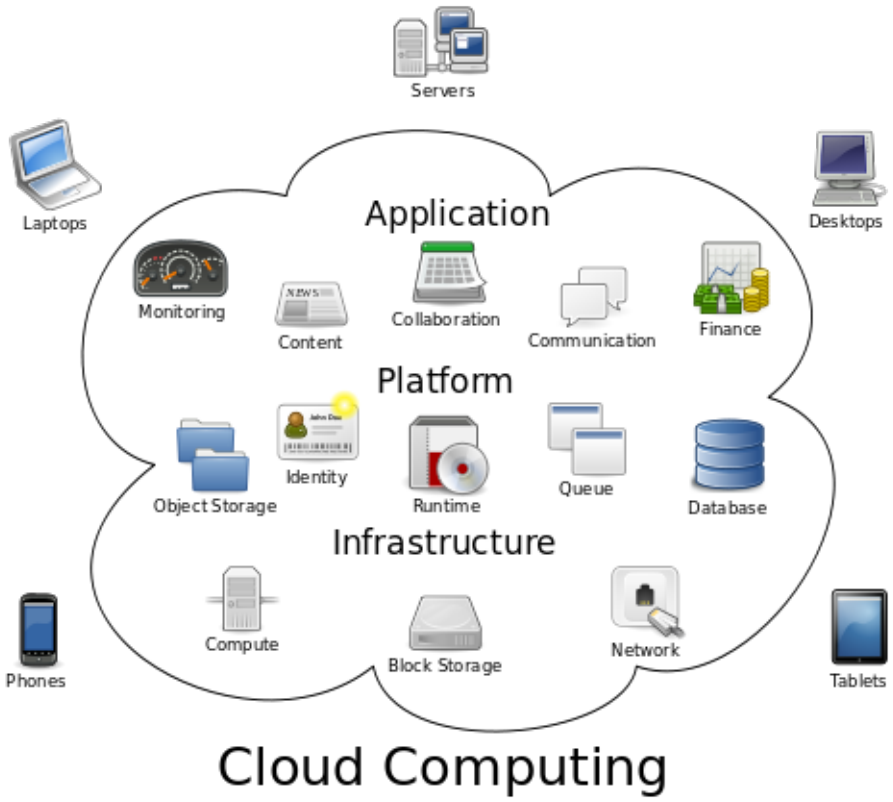


Figure 1.1: A logical view of cloud computing, showing examples of the three basic service models i.e. Infrastructure, Platform and Application/Software that can be accessed from a variety of computational devices [161]

etc.) or offered on a pay-per-use model (Amazon EC2 etc.).

In the Community Cloud model, the cloud infrastructure is shared by several organisations with a common policy, security and/or legal considerations. This helps to reduce costs as compared to a private cloud as it is shared by larger set of organisations.For example government departments requiring access to the same information related to infrastructure, such as hospitals, roads, electrical stations, etc., can utilize a community cloud.

A Hybrid Cloud is the combination of two or more clouds (private, community

or public) to offer the benefits of multiple deployment models. For example, if the existing private cloud infrastructure is not able to handle the user load, the cloud can shift workloads between public and private hosting without any service degradation to the users.

1.2 Characteristics of Cloud Computing

The current buzz around the cloud computing paradigm is due to a number of key benefits that it provides, which also makes it an interesting research domain in both academia and the industry [12], [38], [37] . Some of these benefits are :-

- It provides its users with a very low management overhead.
- It gives fast and easy access to a wide range of applications and services.
- It incurs low maintenance cost on its users as a third party is responsible for the base operations.
- It has the flexibility to scale up and down the resources provided to the users depending on their real-time requirements.
- Its users have the choice to access their services and applications wherever they are from a large variety of devices.
- It offers a cost-effective business model as the users pay only for what they use.

Fig 1.2 shows a survey done by International Data Corporation (IDC) [70] indicating why customers want to incorporate the cloud computing paradigm in

Figure 1.2: A survey showing the answers when 244 IT executives were asked to rate the importance of a variety of cloud services that benefit their organizations. This chart shows the percentage of respondents rating each benefit a 4 or 5, on a 1 (not important) to 5 (very important) scale [70].

their businesses. It is abundantly clear from this figure that there are two key driving factors for the rapid adoption of cloud computing. The first is its economic benefits, and the second being the speed, flexibility and ease of use that it offers to its users for managing and using their IT resources.

1.3 Challenges of Cloud Computing

Most of the currently available IaaS 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. The main reason behind this is that IaaS is the most basic kind of cloud offering as it only delivers raw resources whereas PaaS and

IaaS solutions deliver complex systems like development tools, dynamic libraries, and application life-cycle management suites etc. [69], [147] .

In the cases when higher-level and more abstract concerns need 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 services towards its clients will degrade. The idea of an Inter-Cloud [34] [44] [143] 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.

There are three fundamental steps in the life cycle of a service in the 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 the 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 [68] 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

Figure 1.3: A survey showing the answers when 244 IT executives were asked to rate the top challenges facing adoption of cloud computing in their organizations in 2008. This chart shows the percentage of respondents rating each challenge a 4 or 5, on a 1 (not important) to 5 (very important) scale [70].

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. This can be demonstrated by the results of a survey done by International Data Corporation (IDC) [70] in December 2009. As shown in Fig 1.3, the top challenge identified by *87.5 %* of the sources is *Security*. These results show case the importance of addressing security issues in this respect.

1.4 Research Problem

In the light of the above discussion, we advocate that an inter-cloud security solution is highly desirable that would provide a framework enabling seamless and secure communication between the different 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 their consumers flexible and scalable control over the core security aspects of their services like encryption, secure communication and key management etc. There have been previous attempts to address these security issues concerning the nature of the problem that we are addressing, but most of these have been of limited scope. This has been mainly due to the assumptions made for the creation of the models of the computation and communication architectures. We discuss these in much further detail in Chapter 2.

Most of the existing solutions in this domain are based on some variation of a centralised point-of-control scheme for all of the security concerns of a communication model, which does not scale well as the entities in that model increase. This is especially a major concern in a multi-cloud communication paradigm where the number of virtual machines can increase or decrease dynamically depending on the application and user work loads. To cater for this particular issue, we need a secure communication approach that is flexible to adapt to the churn of virtual machines and also to their deployment locations. This involves having a decentralised control mechanism as well as having an architecture that is tolerant of participating entities leaving the communication framework unannounced or join-

ing it dynamically. We aim to demonstrate the efficiency of our proposed solution in this regard by first modelling the dynamics of the communicating entities in a multi-cloud scenario and then measuring the performance of our solution with respect to how quickly can it admit the newly joining members into the communication framework and how resilient it is to members leaving the framework.

Further limitation of most of the existing work is the amount of manual work needed to set up the solutions they are employing. This is in terms of practical work needed for the installations, dependency resolutions, configurations and operations etc. The scaling problem comes into play here as well as the number of entities that need to be managed like this increases. This is also aggravated by the lack of dynamic network configurability in most cloud providers caused by the inherent limitations of the fixed network architectures offered by these providers. Therefore, one of our security engineering objective is to address the requirement of minimum manual configuration and deployment and focus on a launch-and-forget type of solution. In order to achieve this objective, we have to ensure the security and consistency of the entities participating in the communication framework as well. We demonstrate this by constructing this feature as a component of our architecture that will have the following properties:-

1. Centralised policy-based specification for the operation and security associations of the communication framework.

2. Distributed mechanism to carry out the policy's actions, where each entity will have the responsibility for its own configuration.

3. The policy actions should be idempotent and consistent, i.e. the multiple applications of a policy rule will not change the result beyond the initial ap-

plication, until the next deviation is propagated in the distributed system by the configuration management component.

Another area of limitation in related literature is comprehensive security management in general and key management in particular, as this issue is either handled in a trivial manner with very little details provided or not discussed at all in any length. For us the security of the overall solution is of paramount importance, but it also needs an efficient and scalable design. We achieve this aim by trying to include the concepts of *application partitioning* and *security by isolation* in our solution. More specifically, we have come up with an intra-application sandboxing architecture where we partition our solution into different parts and then securing each part by using a set of different security schemes. As each part is logically separated by the other ones and employing different security schemes, its compromise cannot directly affect the whole solution. The main challenge we have to address here is how to partition the system into meaningful parts and which security scheme to then design and apply for each part. We investigate the partitioning of the solution with respect to the different stages of its life-cycle and describe the design of the security schemes that we have designed to be applied at those stages.

Lastly, to determine the efficiency of our solution we have to measure its performance and compare it with similar works. This introduces a major challenge as direct comparison with most of the related work is almost impossible because of the issues like vastly different test and simulation environments, architecture of the solutions and availability of reproducible results from the related work. We address this problem by identifying the important performance metrics that are

commonly measured and noted when encountering a work of this nature and then designing suitable experiments. We then demonstrate the results produced in light of those metrics by running those experiments on multiple commercial cloud platforms and comparing the performance of our solution with experiments run in the same environment but without any security enhancements and features.

1.5 Research Objectives

We have done an extensive literature review of the secure communication mechanisms and frameworks that can be used, either in their current forms or with modifications, to protect the communication channels between physical or virtual machines deployed on different cloud platforms. The various limitations, gaps, and shortcomings of the reviewed works are explored further in Chapter 2. By analysing these limitations and gaps in the existing work in detail, we have been able to identify the objectives of our research effort. These objectives will help with the addressing the shortcomings of the existing works, as well as contribute towards improvement of the robustness and security of communication frameworks with-in the scope of the inter-cloud environment. We classify our main objectives in four categories :-

Objective 1: Distributed inter-cloud communication framework

- To design and develop a communication framework that enables the components of an application deployed in an inter-cloud environment to communicate with each other.

- To investigate and design a decentralised command and control mechanism for its management and operation so that it does not have a single point of failure.
- To explore and develop an efficient resource discovery mechanism for the communication framework so that the distributed entities that constitute the communication framework are able to share and manage the keys they require for their security operations.

Objective 2: Security

- To design and develop a security mechanism that ensures the integrity and confidentiality of the network traffic between the components of an application utilising our inter-cloud communication framework.
- To tightly and seamlessly integrate the security mechanism with the inter-cloud communication framework in such a way that it causes minimum overhead for the throughput and latency of the communication.
- To design and develop a security protocol for preventing unauthenticated and unauthorised actors from gaining admission to the communication framework and compare its performance with a standard security protocol.
- To design and develop a security protocol for protecting the resource discovery mechanism used for key management in the communication framework and compare its performance with a standard security protocol.

Objective 3: Scalability

- To design and architect the secure communication framework in such a way that it is able to scale with the increased workload.
- To design and develop an architecture for measuring the load scalability of the communication framework in terms of operations it is able to perform when increasing or decreasing the number of the distributed entities.

Objective 4: Ease of Use

- To design and architect the secure communication framework in such a way that it minimizes the complexity of manual deployment and configuration for its operation in a multi-cloud environment.

In summary, the overall aim of this research is to address the secure, flexible and scalable communication concerns that in our view must be overcome in order to provide holistic provisioning of services to consumers from multiple cloud service providers. We aim to present the architecture and design 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, all on top of secure and private communication between the components of the service across multiple cloud platforms. Our architecture provides a virtual network using resources from multiple cloud providers and offers the capability to transparently run applications on top of this network while catering for the dynamic growth and shrinkage of the components of the service.

1.6 Thesis Contributions

The main contributions of our research effort pertain to the design and architecture of a secure, scalable and robust communication framework for cloud services and applications running on virtual machines in a multi-cloud environment, without persistent and centralised administration of all the secure connections. Based on the detailed discussion in the previous and upcoming sections, we have focused on the following contributions in this domain:-

1. Design and architecture of a scalable *inter-cloud* secure communication framework that works seamlessly with multiple cloud platforms.
2. A novel and efficient security protocol utilising the zero-knowledge proof concept for controlling admission into a cloud service's overlay network.
3. A novel and extremely low-overhead secure resource discovery scheme utilising functional encryption for scalable key management.
4. A novel process of using distributed hash tables as a command and control channel for managing and operating the secure communication framework.
5. Deployment, experimentation and analyses of applications using the secure communication framework on real-life commercial cloud platforms for real-world evaluations.

Furthermore, the research carried out as a part of this effort has resulted in 1 book chapter, 1 international journal publication, 3 international conference publications and 1 patent. The detailed references of these publications and patents have been provided in Appendix C.

1.7 Thesis Outline

The rest of the thesis is organized as follows: In Chapter 2 we present the state-of-the-art related works that address issues related to this domain and identify their gaps and limitations. In Chapter 3 we explain in detail the background of the technologies, methods and algorithms that we have utilised in the formulation of our solution. In Chapter 4 we outline the key methodology for our approach and elaborate on the detailed Inter-Cloud Virtual Private Network (ICVPN) architecture and design, as well as the experimental set up and the evaluation and analysis of the performance results of our implementation. In Chapter 5 we elaborate on the design of the admission control component of our solution and show its comparison with other related methods. In Chapter 6 we present the design and implementation of the secure resource discovery component and compare its efficiency with existing approaches. We conclude in Chapter 7 with a summary of our contributions and achievements.

Chapter 2

Review of Related Work

The concept of the Inter-Cloud has come into its own over the recent years as a logical evolution of the cloud IaaS interoperability. Most of the commercial cloud service providers designed and offered their services in such a way that the users could not easily transition to another cloud service provider offering the same service (this phenomenon is called vendor lock-in). However, due to the costumer demands for more flexibility and choice, there has been some effort to come up with solutions that allow the users to use the resources of multiple cloud service providers to deploy their systems.

Although most of the attention in this area has been given to data storage use cases, it is also clear to see that parts of customer softwares running on virtual machines instances on different cloud platforms must be able to dialogue with each other. One deployed instance of the software must be able to find one or more other instances for a particular interoperability scenario and be able conduct transactions or exchange whatever information or data that is required. Thus, an inter-cloud communication protocol or framework is needed for the discovery and

messaging needs that can support the one-to-one, one-to-many, and many-to-many communication scenarios.

As of present, there is no such universal inter-cloud communication protocol and certainly no universal inter-cloud communication security architecture that exists to address the above problem statement. The aim of our research is to come up with such a secure communication framework and evaluate it with respect to different metrics (discussed in detail in later chapters). In order to start this process , we identify some of the key security challenges relevant to the inter-cloud scenario that we will have to address:-

- Preserving the confidentiality and integrity of data in transit to and from a cloud service instance (typically a virtual machine).
- Unauthorised access to the resources of the communication framework.
- Interception of data in transit (man-in-the-middle attacks).
- Making access to data or keys selectively available to authorised users and entities.

Most of these challenges have been addressed quite successfully in the Internet scenario by utilising Virtual Private Networks (VPN). Virtual private networks have been a mainstay for providing secure remote access over wide-area networks to resources in private organizational networks for a long time [120], [106], [54]. They give the illusion of establishing a connected network by setting up logical connections between end-points and securing them by using specialized software that provides confidentiality and integrity of the traffic flowing between these end-points by using well-known encryption techniques [16], [61], [20]. However,

the thrust of our research effort is to provision a secure virtual private network over a multi-cloud infrastructure.

In order to achieve this goal, we have identified a few requirements that help us in evaluating the related work in relation to the gaps and limitations present in them that make them unsuitable for an inter-cloud environment. Some of these requirements have been discussed earlier in section 1.4 in detail. We were able to establish these requirements as a result of our effort to understand the nature of our target research environment, i.e., the inter-cloud and the objectives of intended users of this environment, as discussed in existing literature like [34], [44], and [143].

Additionally, we were also able to gather requirements for our research by keeping track of the on-going attempts by IEEE (Institute of Electrical and Electronics Engineers) to create technical standards (IEEE P2302 [88]) for inter-cloud interoperability. Although the endeavours of IEEE are still in their infancy with no concrete output thus far, it helped us to discern some useful requirements from them that were in line with our findings from the study of the related work. The main insight we were able to ascertain from analysing this material was that the inter-cloud can not be a single entity, instead it should be a replicated and hierarchical system. This was based on the observation that the current cloud computing platforms, that will comprise the inter-cloud environment, are not yet able to federate and interoperate.

From these sources and more, we were able to identify a few unfulfilled needs and gaps that must be addressed by anyone who wants to research for a viable inter-cloud communication framework with regards to security, scalability and ease-of-use. We have have been able to render some of these needs and gaps

into our research requirements and here we summarise the specific requirements that should be catered for when researching the design and architecture of an inter-cloud secure communication framework.

- The design of an inter-cloud communication framework should be decentralised, no single-point-of-failure should exist in it.
- The access to the resource of a the communication framework should be granted only after valid authentication procedures.
  - The authentication procedures should be cryptographically strong, e.g., user-name/password based methods should be avoided.
  - The authentication procedures should be efficient and easy to automate, e.g., Public Key Infrastructure (PKI) based authentication might be problematic in this situation due to complex key distribution scheme.
- The communication channels should operate on the lowest layer possible of the TCP/IP (Transmission Control Protocol / Internet Protocol)networking model in order to reduce performance overheads.
- The design of an inter-cloud communication framework should be scalable; the framework should be not burdened as the number of communicating entities in it increases.
- The management and configuration of the framework should be as simple and automatic as possible; manual setting up of link and channels should be avoided.

- The communication framework should be practical in terms of performance, i.e., it should be secure as that is a fundamental requirement but it should add as low overheads due to that as possible.
- The design of the framework should be cloud platform independent and not specific to any particular cloud service provider in order to allow the users flexibility of choice as well as avoid vendor lock-in.

As mentioned above, the main thrust of our communication framework is the provisioning of a secure virtual private network in a multi-cloud environment. After an elaborate literature survey process, we were able to classify the existing virtual private network techniques and solutions into four main categories. First is the most common type of approach, i.e, those that model their solutions according to the client-server model [25]. Second is the category of techniques and solutions that construct a complete virtual network [81] consisting of virtual computational and network entities as well as customised routing mechanisms. Third category is composed of techniques that leverage peer-to-peer algorithms and topologies [133] to construct resilient and fault tolerant virtual private networks. Fourth and last is the most recent and relevant types of efforts that offer virtual private networking services on cloud computing platforms [140]. We have picked a few related works for further elaboration in the following sections. This is not an exhaustive selection, rather we chose some main candidate recent works to show-case the approaches they take to solve the problems and the common gaps and limitations in these approaches.

2.1 Client-Server based approaches

The discussion of design and architecture of almost all the client-server based virtual private network solutions can be condensed by just discussing the design and architecture of OpenVPN [167]. It is the most common and popular VPN solution (5 million users worldwide as of June 2015 [168]) used to create secure point-to-point or site-to-site connections for authenticated remote access [116], [166].

An OpenVPN server runs the central VPN software and each client machine needs to install a client software so that they can participate in the extended network. This client application is usually installed and configured by an administrator and is done on a per-machine basis. OpenVPN uses industry standard SSL/TLS (Secure Socket Layer and Transport Layer Security) protocols to provide confidentiality and integrity of the traffic exchanged between its end-points [87], [27], [104]. It is based on a modular networking model that uses the TAP (network tap) and TUN (network TUNnel) virtual networking devices as interfaces between the client and server operating systems. The Universal TAP/TUN driver [100] is a virtual network interface and its main purpose is to provide IP tunnelling support to the operating system. Its appears as a network device to all the applications and users of the operating system and every application that can use a network interface is able to use this virtual network interface as well. OpenVPN listens on the TAP/TUN interface for all the network traffic that is being written to it by the user applications, encrypts it and then sends it to the destination machine, where another OpenVPN client will be present to receive the data from its TAP/-TUN interface, decrypt it and give it to the user application waiting for it.

Gaps and Limitations:

However, the main problem in client/server based approaches like OpenVPN 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. Furthermore, for authentication of the OpenVPN client, a simple username / password based system is used. There is also the possibility of using a PKI based system for client authentication but that adds the further key management complexity as well as the overheads associated with PKI-based approaches. This can be a major drawback as the issue of key distribution among all the participating clients in a VPN is non-trivial, especially when the software itself does not provide any key distribution service and all keys have to be manually transferred to individual hosts [105]. In case of PKI model, an additional requirement of a trusted Certificate Authority 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. 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. Lastly, the amount of data transmitted using these VPN tools increases over time due to the wrapping and tunnelling processes. This is known as the VPN overhead and its cost depends on the amount of meta-data and the encryption techniques used in the VPN. In OpenVPN, the use of TAP/TUN interface can introduce a large overhead, especially for bursty and interactive traffic. We address all of these

problems in our work.

2.2 Virtual Network based approaches

There have been some other VPN solutions for large-scale networks aimed at grid and cluster computing environments, such as VIOLIN [92] and VNET [146], that do not follow a strict client/server model based approach.

2.2.1 VNET

VNET is a layer 2 virtual networking tool that implements a virtual local area network (VLAN) [57] over a wide area network (WAN) using layer 2 tunnelling [154]. It relies on VNET servers running on a Virtual Machine Monitor (VMM) that should have the capability to extract raw Ethernet packets sent by the virtual network card and also the capability to inject raw Ethernet packets into the virtual network card.

The operation of a VNET set up is shown in Fig 2.1. Basically, a VNET server VM in a remote network establishes an TCP/SSL tunnel connection to a VNET server running on a machine, called *proxy*, inside the user's home network. All of the remote virtual machine's communication goes through this tunnel and the goal of the *proxy* is to emulate the remote virtual machine as a local host on the user's home network, in effect presenting it as a member of the same LAN (Local Area Network). So the *proxy's* role is to act as a packet filtering gateway that matches the Ethernet packets that it receives and passes them on to the appropriate destinations by either directly injecting them into the virtual network

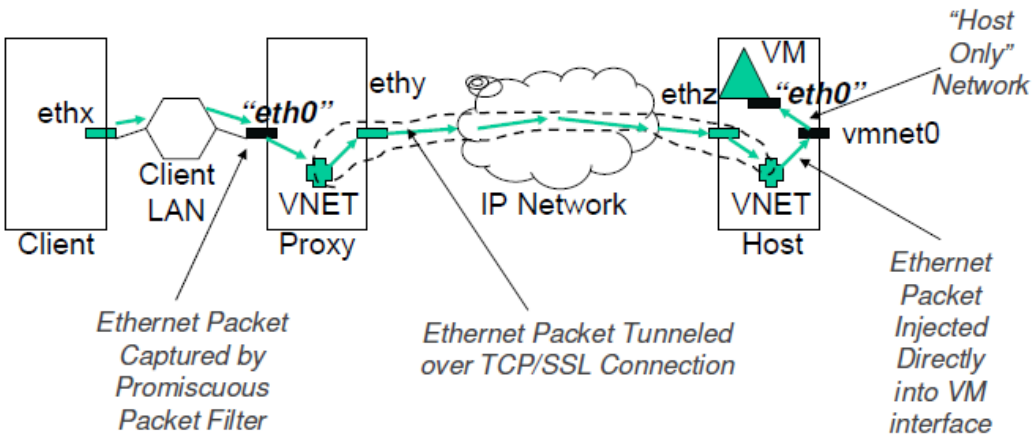


Figure 2.1: Architecture of a VNET based communication framework, showing the establishment of a secure connection between a client and the end host via a *proxy* gateway [146].

interfaces of the VNET servers (in the case of destination being in the remote network) or injecting them into the local LAN (in the case of the destination being in the local network).

Gaps and Limitations:

The motivation of this approach is to tackle the user's lack of administrative control at remote grid sites to manipulate network resources like routing and resource reservations etc. but it suffers from the previously discussed problem of complex and manual configuration, though trying to emulate 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 *proxy's* network interface by the VNET server software.

2.2.2 VIOLIN

The VIOLIN (Virtual Internetworking on OverLay INfrastructure) architectural design offers is a small-scale virtual network with virtual routers, switches and end-hosts. the complete design is composed of three layers, as shown in Fig 2.2. The low-level plane is the actual layer 3 IP network, the mid-level plane denotes an overlay infrastructure such as PlanetLab, and the top-level plane denotes a set of VIOLIN entities that are created on the overlay infrastructure. There are three types of VIOLIN entities which correspond to real network entities i.e., end-hosts, switched LAN, and routers. All entities in the VIOLIN are implemented in software and are hosted by User-Mode Linux (UML) [53] enabled virtual machines as virtual appliances. This kind of design is aimed at allowing for the dynamic establishment of a private layer 3 virtual network among virtual machines.

Gaps and Limitations:

VIOLIN does not offer dynamic or automatic network deployment or route management to set up the virtual network. Virtual links are established between the virtual appliances using encrypted UDP (User Datagram Protocol) [132] tunnels that have to be manually set up and are not self-configuring, making it cumbersome to establish inter-host connections in flexible and dynamic fashion.

2.3 Peer-to-Peer based approaches

There have been many peer-to-peer based VPN solutions proposed or implemented, utilising various technologies such as multicast trees, gossip protocols,

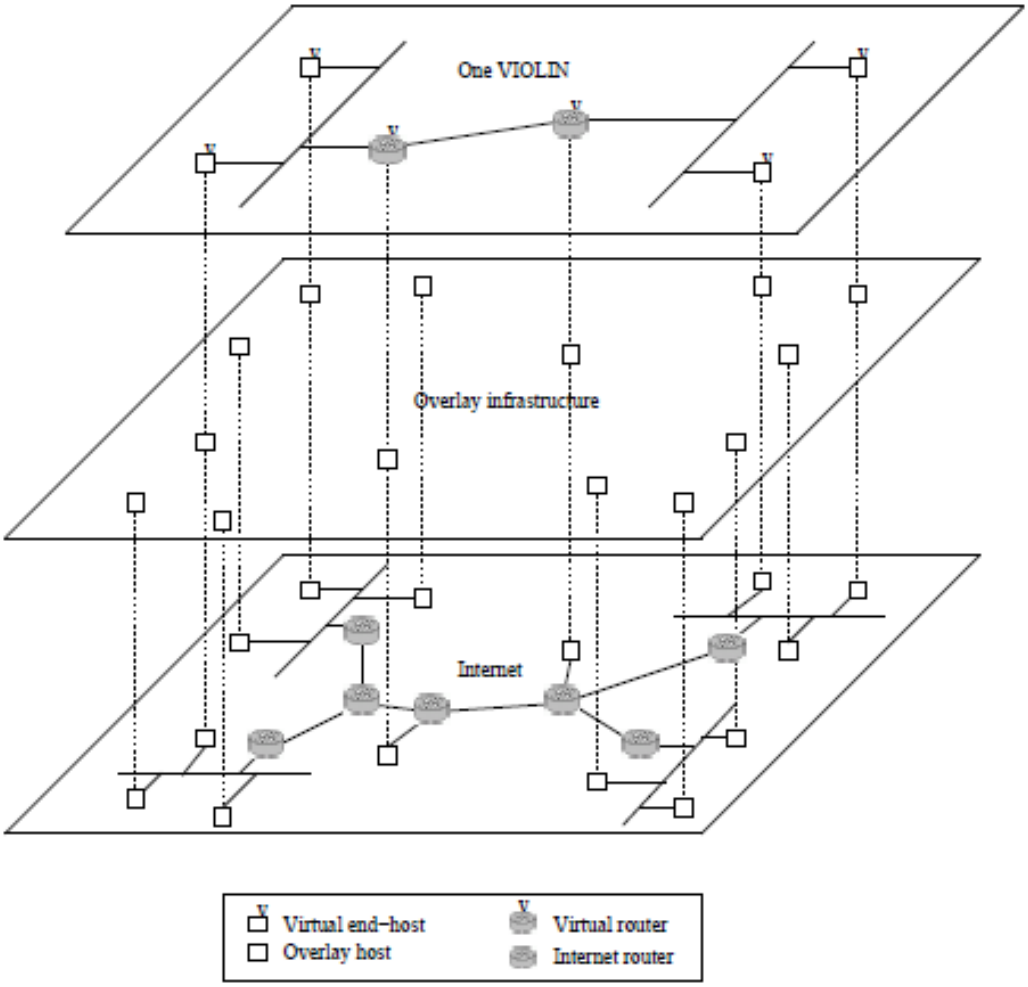


Figure 2.2: The architectural design of VIOLIN, showing the three composing planes i.e., the bottom plane being the actual layer 3 network, the PlanetLab overlay infrastructure acting as the middle plane, and the set of VIOLIN entities that are created in the top plane [92].

and overlay broadcasts [94], [164], [75], [165], and [123]. We discuss two peer-to-peer VPN solutions here, i.e., Hamachi [107] and N2N [50], that have come up as peer-to-peer alternatives to the centralized and client/server model based VPNs.

2.3.1 Hamachi

Hamachi is a shareware application that is capable of establishing emulated direct links between computers that are behind NAT (Network Address Translation) firewalls. Thus it gives the illusion that the network peers on the internet are connected to each other as if they were on the same local network. A back-end cluster of servers is managed by the vendor and clients have to install the client software on the end-user computers. The vendor managed VPN servers are publicly accessible from the client's network and each client can establish and maintain a control connection to the server cluster. When a connection is successfully established, the client goes through a user-name/password based login process which authenticates the client to the server. This is followed by a discovery process which is used to determine the topology of the client's internet connection, specifically to detect the presence of NAT and firewall devices on its route to the public internet. This is followed by a synchronization process that is used to share the status and information of the client's connectivity details with other members of its network. After all this is done, the client can construct peer-to-peer tunnels with other clients using virtual network interfaces and NAT traversal techniques (if the client is behind a NAT gateway or firewall). It is mainly used for internet gaming and remote administration.

Fig. 2.3 shows how a Hamachi server helps two hosts establish direct virtual communication links between each other while they are behind private NAT firewalls. In client A's session message to the Hamachi server, A shares its private socket address with it. The Hamachi server records client A's reported private

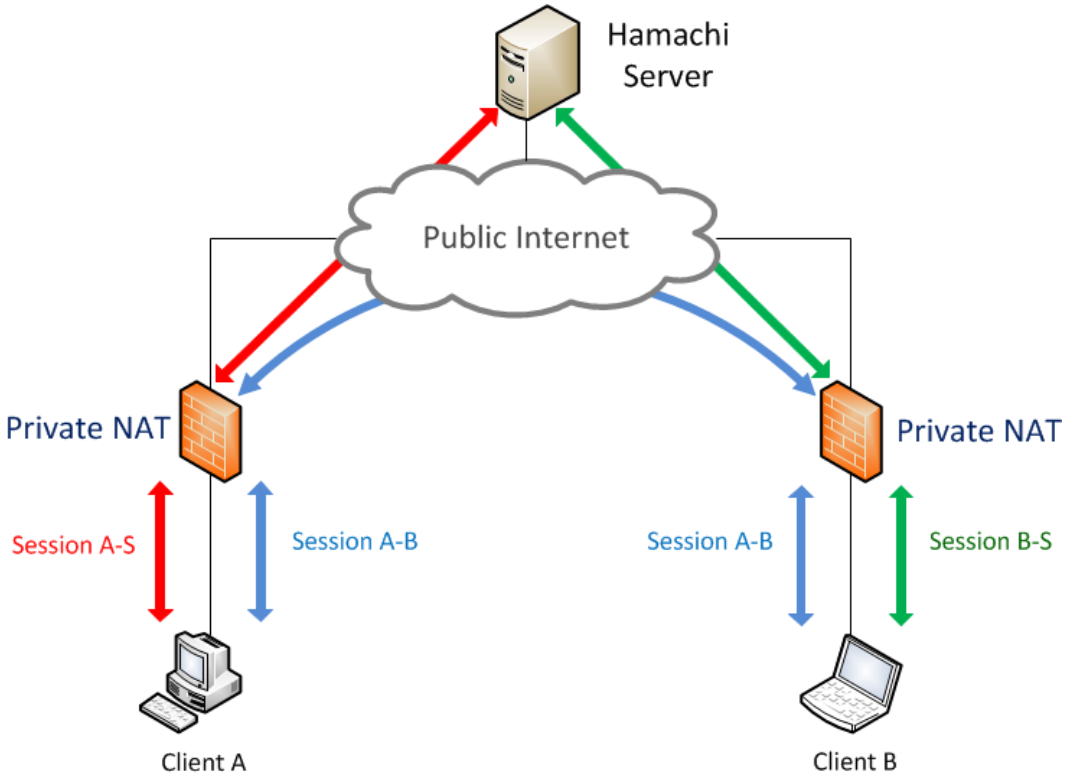


Figure 2.3: The Hamachi architecture for linking fire-walled peers. The fire-wall functionality is usually provided by private NAT devices that are transparent to the end-users of this service.

socket address, along with A's public socket address as observed by the Hamachi server itself. Similarly, when client B establishes its session, the Hamachi server records B's private socket address and it's public socket address as well. After this, client A sends a request message to the Hamachi server asking for help connecting with client B. In response, the Hamachi server sends B's public and private socket addresses to A, and sends A's public and private socket addresses to B. Now, client A and client B can each start sending UDP (User Datagram Protocol) [132] datagrams directly to each other using this session information.

Gaps and Limitations:

Hamachi 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. This means that each client essentially has to deal with the overheads 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), thus placing restrictions on the network size. The keys used for connection encryption and authentication are also controlled by the vendor's servers and individual users do not initially control who has access to their network. While it offers to support different kinds of key distribution mechanisms [108], the actual implementation apparently only offers a Key Distribution Center (KDC) based approach [122], which requires all peers of a VPN to establish trusted relationship with each other through the central Hamachi website. Thus, it is not able to offer the users the feature of independent VPN deployments.

2.3.2 N2N

N2N is a layer 2 VPN solution which does not require a centralized back-end cluster of servers like Hamachi and the encryption keys are not managed or controlled by the vendor. Each N2N node has a encryption key pre-shared among the users that have been invited to join the peer-to-peer overlay. 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

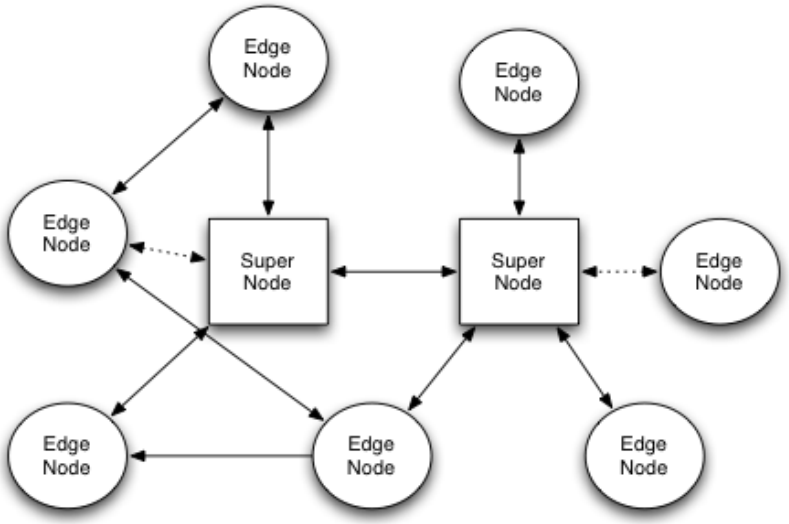


Figure 2.4: The N2N overlay network architecture showing the two kinds of nodes i.e., Super Nodes and Edge Nodes. The figure depicts an example overlay where two Super Nodes are connected to the Edge Nodes in a star topology, and the communication between the Edge Nodes has to pass through the Super Nodes (shown with dashed lines) [50].

[139] keys are used for link encryption. The N2N edge nodes are identified uniquely by a 48-bit MAC address and a 128-bit community name. Edge nodes use virtual Ethernet devices [100] to establish encrypted UDP tunnels between each other.

Gaps and Limitations:

As it operates on layer 2, the users of the overlay have to configure their IP addresses and other network parameters. It also assumes node membership as relatively static with edge nodes rarely leaving or joining the network over their life cycle. This is certainly not true in the cloud computing domain where the virtual machines are very expendable and can be created and destroyed quite frequently. Lastly, the peer discovery method of N2N utilises the overlay broad-

casting method [41]. This will increase the communication overhead of the system, as the number of of peers increases.

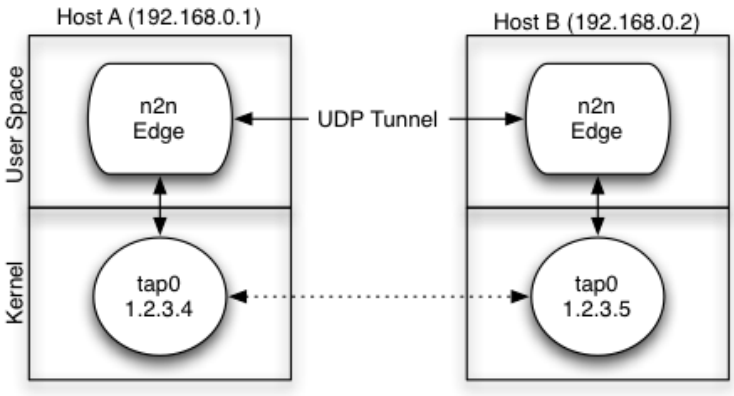


Figure 2.5: Tunnelling between N2N nodes, with the logical communication passing through the UDP tunnel in user-space but the physical signals pass through the tap devices in the kernel-space [50].

2.4 Cloud based approaches

In recent years, there have been a lot of attempts to investigate scalable and secure virtual private network solutions for the cloud computing environment. These include research efforts like Dynamic IP-VPN [79], IPsecVPN [89], and Connectivity-as-a-Service [40]. Also, 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 [6], Google Secure Data Connector [71] and CohsiveFT VPN-Cubed [47]. These are aimed at enterprise customers to allow them to access their resource deployed on the

vendor's cloud over an IPsec (Internet Protocol Security) [55] based virtual private network.

2.4.1 Dynamic IP-VPN

Dynamic IP-VPN is a research effort that aims to provides a virtual private network for a private cloud deployment, using some dynamic features provided by a next generation network. The next generation network used by this system is GENI (Global Environment for Network Innovations) [23], which provides a virtual test-bed for networking and distributed systems research. The main attractive feature GENI for this system is its flexibility and programmability, i.e., it allows the users to program not only the end hosts of their experimental network but also the switches in the core of their network. This, in turn, allows them to experiment with customised network layer protocols. Furthermore, all of the networking equipment is virtualised and its components are made available to users as resources. In that sense, its a more advanced version of VIOLIN [92], discussed earlier in this chapter.

Fig. 2.6 shows the architecture that the next-generation network infrastructure for the private cloud platform must offer their users. This architecture consists of the following four dynamic components.

1. The edge node, called the Virtual Network Edge (VNE), that connects the end-user terminals to the overlay network.

2. The forwarding node, called the Virtual Network Router (VNR), that relays data sent from one VNE to another VNE.

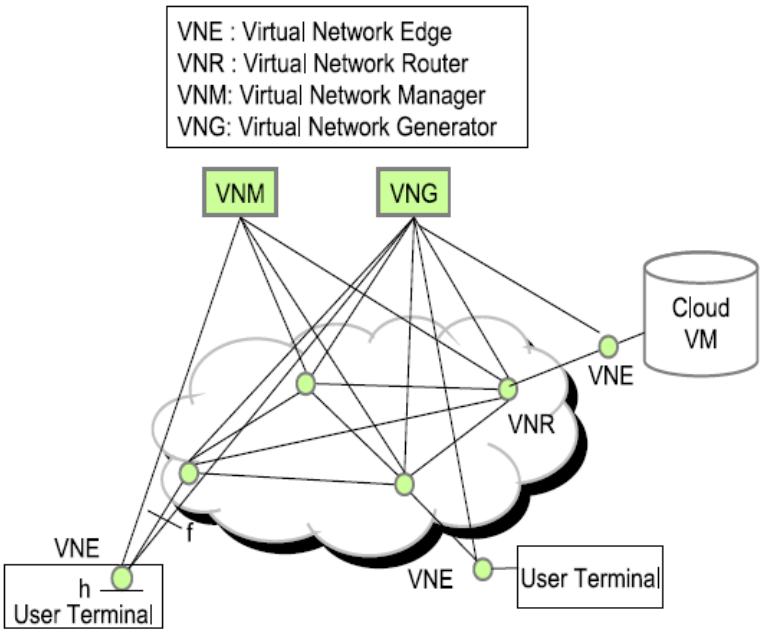


Figure 2.6: Architecture of the Dynamic IP-VPN showing the four dynamic components comprising the system and their deployment locations in an overlay [79].

3. The signalling node, called the Virtual Network Generator (VNG), that helps in the authentication and authorisation of the end-user terminal ahead of connecting the terminal to the overlay network.

4. The routing node, called the Virtual Network Manager (VNM), that computes the optimal route for the path between two VMEs and sends configuration information to VNRs located on that path in order to establish a connection between the two VNEs.

The authors implement this architecture on a GENI test-bed and measure the throughput results on a LAN environment with a 100 Mbps (Megabits per second) line speed specification. They get the throughput of approximately 8 Mbps for an encrypted tunnel session between two VNEs.

Gaps and Limitations:

The Dynamic IP-VPN architecture requires a specialised networking infrastructure that supports software-defined networking, a feature most of the current cloud platforms do not support. Furthermore, the authors assume that the private cloud platform deploying their solution will support multiple protocol programmability in order to offer both Layer 2 and Layer 3 virtual private networks, using IPsec, L2TP (Layer 2 Tunnelling Protocol) and SSL-based VPNs. This is also an unrealistic assumption in our opinion as typically a private cloud deployment usage scenario is useful for a small company or user group, which usually do not have the need of such flexibility. Lastly, the performance of the network throughput really suffers due to the requirement of so much flexibility and even on a 100 Mbps LAN they get an overhead of 92%.

2.4.2 IPsec VPN

IPsec VPN is a research effort that aims to establish a secure VPN between different private networks, that are connected via the Internet, in order to construct a secure closed user group. The authors describe the two kinds of IPsec VPN architectures currently being used, which they term as Full-Mesh IPsecVPN and Hub-and-Spoke IPsecVPN. In case of Full-Mesh IPsecVPN, they envision a model where the private networks are connected to each other directly via an IPsec gateway device, with all the IPsec gateways connected in a mesh topology [62]. Their graphical interpretation of this model is given in Fig. 2.7.

In case of Hub-and-Spoke IPsecVPN, they envision a model where all the private networks connect via IPsec to a central IPsec gateway called the Hub,

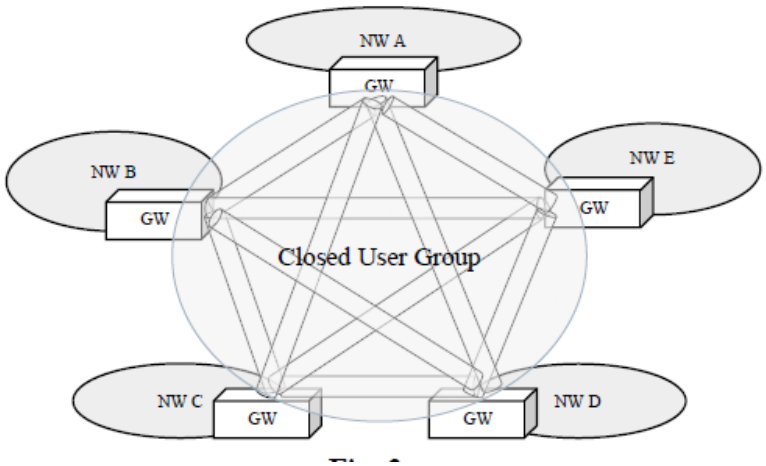


Figure 2.7: Architecture of the Full-Mesh IPsecVPN, where each IPsec gateway (GW) of a private network (NW) is connected to all the other gateways [89].

and all communication between any of the private networks is relayed through the Hub. Their graphical interpretation of this model is given in Fig. 2.8.

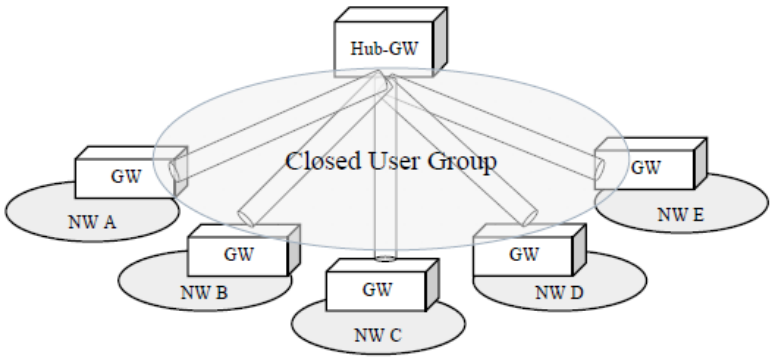


Figure 2.8: Architecture of the Hub-and-Spoke IPsecVPN, where each IPsec gateway (GW) of a private network (NW) is connected only to the Hub gateway (Hub-GW) [89].

The authors then go on to elaborate the limitations of these architectures. In the case of the Full-Mesh IPsecVPN, they discuss the difficulty of dealing with scalability issues related to policy management on the IPsec gateways as the number of private networks increases. In the same vein, it will be difficult to add

new private networks in an existing group seamlessly, as this will require adding new policy in each IPsec gateway. In the case of the Hub-and-Spoke IPsecVPN, they mention the problem of increased load on the Hub gateway as the traffic between the private networks increases. The Hub gateway is also a single point-of-failure in this model.

To address these concerns, they propose a hybrid model which is a modification of the Hub-and-Spoke IPsecVPN, whose architecture is shown in Fig. 2.9. In this model, they address the load management problem of the Hub-and-Spoke by using the IKEv2 (Internet Key Exchange protocol version 2) Mobility and Multihoming Protocol (MOBIKE). The authors use MOBIKE as a load balancing tool for their Hub gateway and in case of additional network traffic, they propose to reconfigure the system so that the *problem* IPsec gateways are able to communicate to each other directly without using the Hub gateway as a relay.

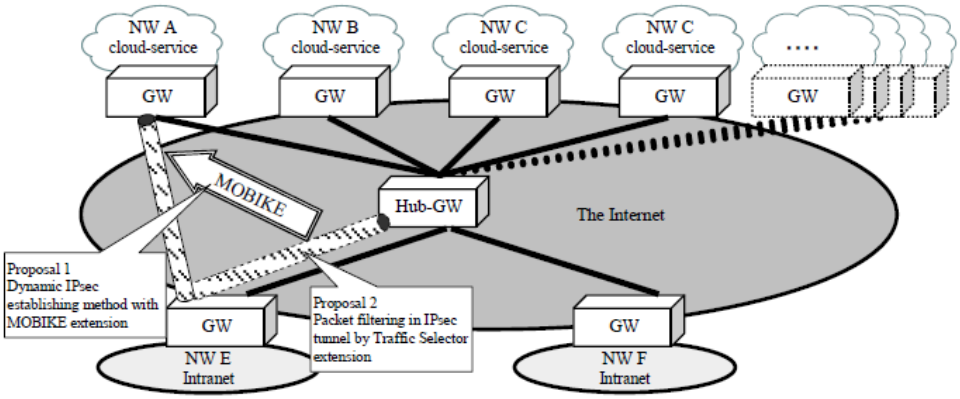


Figure 2.9: Architecture of the hybrid IPsec-VPN model, where the Hub gateway (Hub-GW) is extended using the MOBIKE or Traffic Selector extension in order to perform load-balancing operations [89].

The authors have not implemented or simulated this architecture, so we cannot make use of any network performance results of this solution.

Gaps and Limitations:

The main purpose of the MOBIKE extension is to enable a remote access VPN user to move from one IP address to another, without re-establishing all the security associations with the IPsec gateway. The MOBIKE extension updates only the outer (tunnel header) addresses of IPsec Security Associations (SA), the addresses and other traffic selectors that are in use inside the tunnel stay the same. In this way the mobility of the user is invisible to the applications using the VPN.

Although the authors claim that this extension to the IPsec protocol can be utilised for its unintended use as a load-balancer, they do not mention how they detect or identify the *problem* traffic in the first place. Furthermore, there is still the issue of the Hub gateway still being a single point-of-failure for this architecture, an issue that is not addressed by the MOBIKE extension.

2.4.3 Connectivity as a Service (CaaS)

Connectivity as a Service (CaaS) for intra-cloud and inter-cloud communications [40] is a research effort that aims to offer secure communication in a cloud-based collaborative environment. They propose and implement an Electronic Contract (e-Contract) based solution for intra-cloud and inter-cloud communication between organisation involved in Business-to-Business (B2B) collaboration [77]. The main idea is to offer this solution as a trusted 3rd party web service to the collaborating parties.

The graphical depiction of their architecture is given in Fig. 2.10. The collaborating organisations *A*, *B* and *C* reside within their administrative domains. In

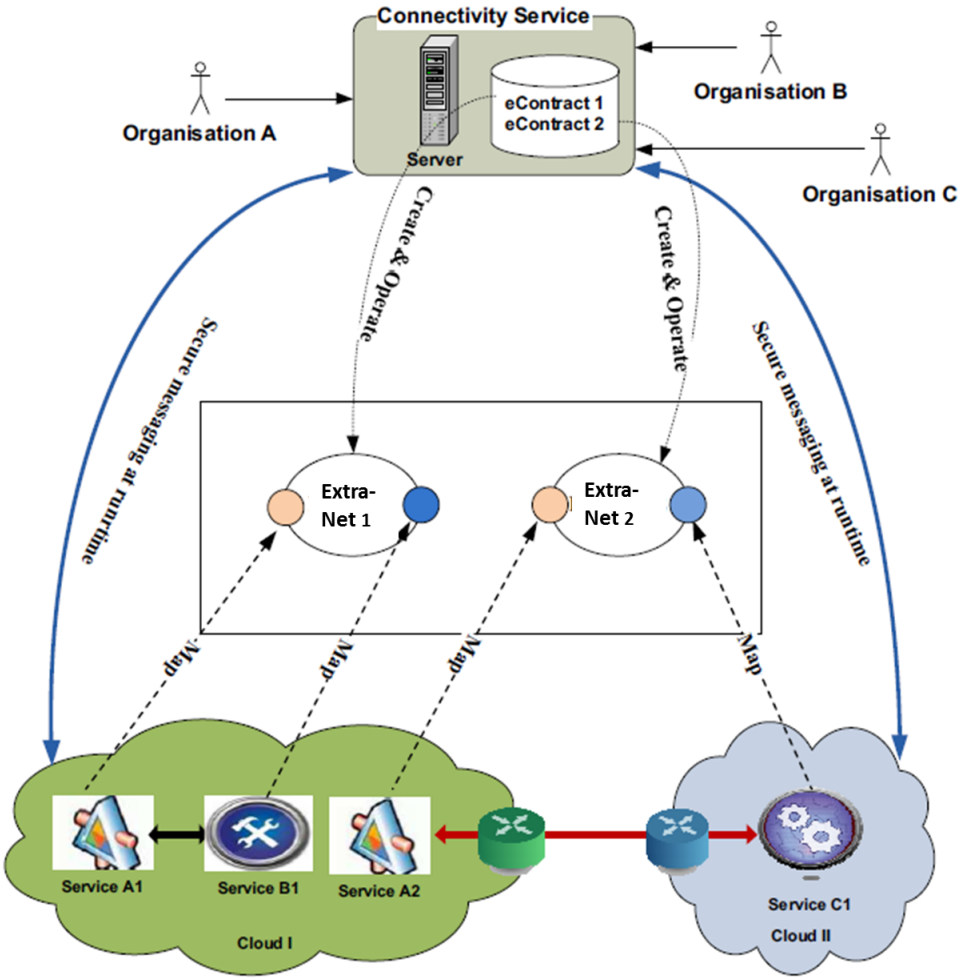
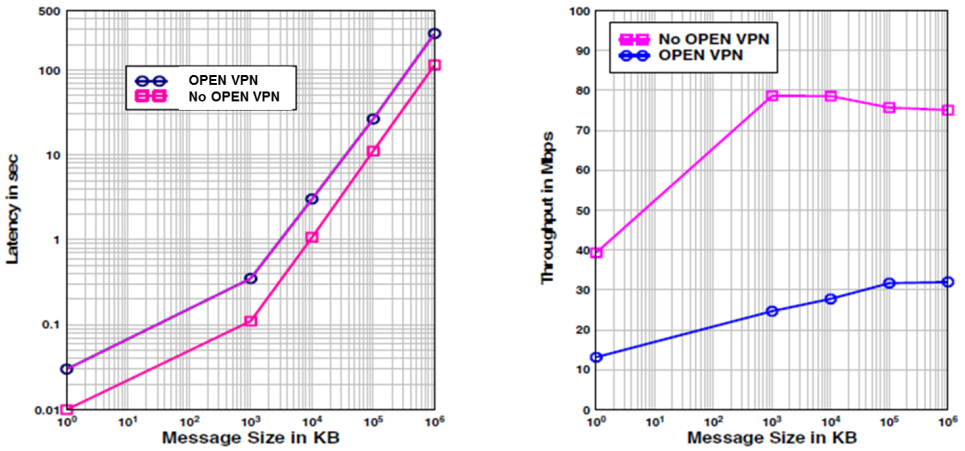


Figure 2.10: Architecture of the eContract-based secure intra-cloud and inter-cloud connectivity service. The figure shows two extranets formed by the Connectivity Service that offer services from both Cloud I and Cloud II [40]

order to collaborate, an e-Contract is negotiated among the organisations and signed by each collaborator. The connectivity web service can then configure and form an *extranet*, according to the e-Contract's specifications. After the successful formation of the *extranet*, the services deployed by the organisations can communication with each other within the *extranet*. The *extranet* itself is a OpenVPN based VPN link between specified VMs of the collaborating organisations.

The authors have conducted experimental evaluation of the performance cost of their architecture in both intra-cloud and inter-cloud environments. For the intra-cloud performance evaluation, they ran tests on three VMs hosted in Amazon EC2 [7] and measured the observed differences in latency and throughput between the VMs in the same *extranet*. They noted an overhead of 100% in both latency and throughput when using their solution as opposed to insecure communication.



(a) Latency: VPN vs. No VPN *(b) Throughput: VPN vs. No VPN*

Figure 2.11: Performance of the eContract-based secure inter-cloud connectivity service in terms of effect on latency and throughput. compared in the presence and absence of OpenVPN tunnels [40].

For the inter-cloud performance evaluation, they ran data transfer tests between two machines running on different LANs. Their comparison of the latencies and throughputs of the two set of tests is shown in Fig. 2.11. They observed that the performance cost of using their solution for inter-cloud communication is much higher, i.e., ranging from 300% for small sized data (1 KB) to 135% for large sized data (1 GB).

Gaps and Limitations:

The authors have based the security and architecture of the secure communication aspect of the their solution on OpenVPN [167]. Hence they suffer from its inherent drawback as discussed earlier in this chapter, i.e., a single point-of-failure. Furthermore, they don't address the issue of distributing keys or other authentication credentials that is required for ensuring the security of the communication channels. In the same vein, they also don't mention how they map the actual machines that form the *extranet*, to their web based 3rd party connectivity service.

Lastly, the latency overheads of more than 100% in case of intra-cloud communication, and more than 300% in case of inter-cloud communication, make their solution extremely undesirable for most cloud-based network applications. In case of most B2B applications, which is their target category of applications, this much overhead will cause an unacceptable increase in time required to send and receive business transactions.

2.4.4 Amazon Virtual Private Cloud (Amazon VPC)

Amazon Virtual Private Cloud (Amazon VPC) is a cloud based VPN solution provided by Amazon that lets its customers provision a logically isolated section of the Amazon cloud as a virtual network to its customers. The customers control and manage their virtual networking environment, including the selection of IP address ranges, creation of subnets, and configuration of routing tables and network gateways. The customers can leverage certain security features provided by the Amazon VPC, like security groups and network access control lists, to help control network access to Amazon EC2 instances in each subnet. In addition to

this, customers can create VPN connections between their local network and the Amazon VPC by using a hardware VPN device hosted by Amazon and bridging it with a hardware VPN device installed in their local network. The logical view of an Amazon VPC deployment is shown in Fig. 2.12.

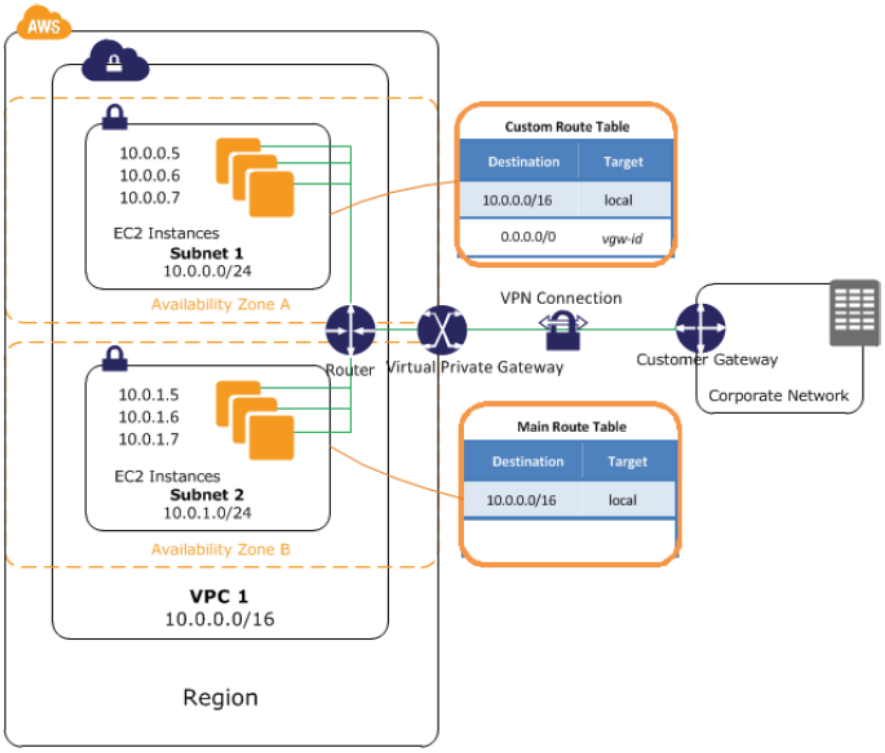


Figure 2.12: Architecture of an Amazon VPC deployment. Access to the EC2 instances in Zones A and B is provided through a Virtual Private Gateway (VPG), which acts as the end-point of the VPN between the customer gateway and the Amazon cloud [6]

2.4.5 Google Secure Data Connector

Google Secure Data Connector (SDC) is a client-side tool that lets its users establish an encrypted connection between their local network and Google tunnel servers. The Google tunnel servers are also used to validate whether a user is

authorized to request access to the specified resources hosted on the Google cloud. After the user validation, the tunnelling protocol allows SDC to connect to a Google tunnel server, authenticate, and encrypt the traffic that flows between the user's network and the Google cloud.

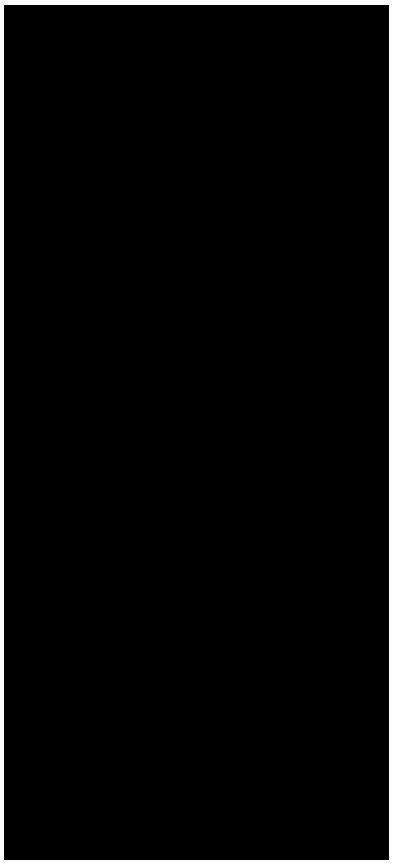


Figure 2.13: Architectural view of Google Secure Data Connector. The VPN is established between the Tunnel Servers hosted on Google premises and the Secure Data Connector hosted on the customer network [71]

2.4.6 CohsiveFT VPN-Cubed

CohsiveFT VPN-Cubed is a virtualised network appliance that is hosted on third-party cloud services and acts as a a router, firewall, and VPN concentrator. Its

main difference from the previous two products is that it lets its customers extend their VPN across multiple public and private clouds to create one logical group of federated resources. It also provides the capability of forming encrypted tunnels using IPsec to secure all traffic coming to and from its virtual appliances. Similar to the previous two examples, users can extend their local network into the federated VPN using dedicated hardware IPsec devices on their premises.

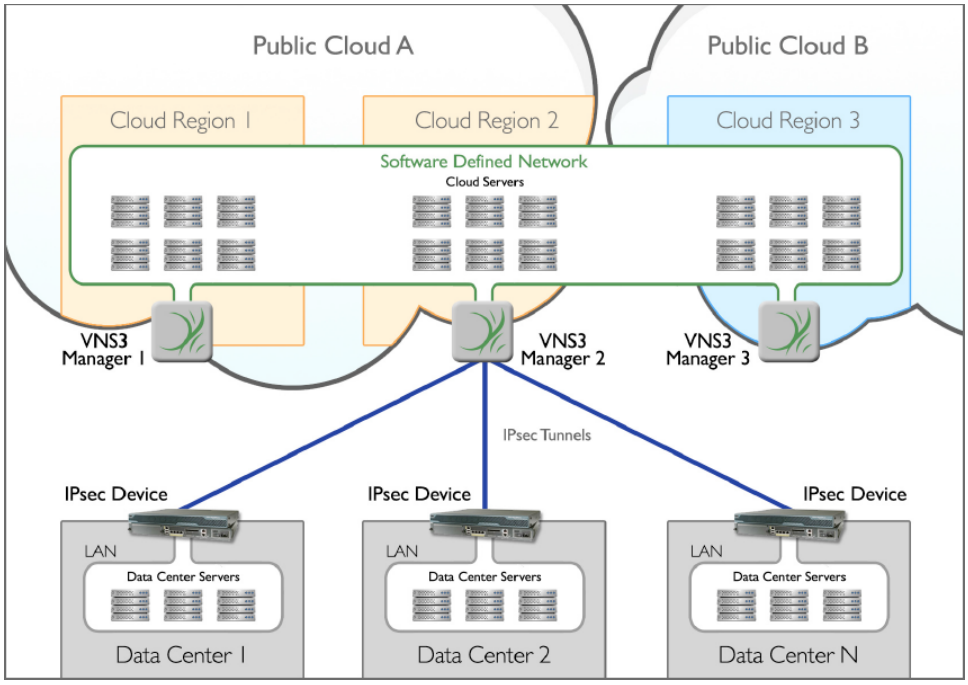


Figure 2.14: A multi-cloud deployment scenario in VPN-Cubed®. In this deployment, three VNS3 Manager servers are hosted on three different cloud regions but only one of them (VNS3 Manager 2) is acting as the IPsec gateway for the IPsec devices of the customer data centers [47]

Gaps and Limitations:

Although these products allow the possibility to leverage the cloud providers' APIs to flexibly grow and shrink their networks, the management and configu-

ration 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 set up an IPsec device on their premises that connects to an IPsec gateway in the VPC running as a virtual appliance which integrates the enterprise's network with the VPC subnet in the cloud. Most importantly, with the exception of [47], these solutions are locked to single cloud vendor and [47] 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.

2.5 Chapter Summary

In this chapter we have reviewed the existing solutions that can be used to set up secure communication links between multiple parties. We have explained their architecture in detail so that its easy to see whether these solutions can be used in an inter-cloud environment. We have also discussed the deployment models utilised by these solutions as the ease and automation of deployment of a secure communication solution in the cloud ecosystem is one of our main objectives. Therefore, we have focused on four classes of deployment strategies, that is, client-server, virtual network, peer-to-peer, and cloud computing.

For the client-server model, we have highlighted that although its administration is simple due to the inherent centralised control, it is also its biggest limitation from the point-of-view of scalability. Therefore, it becomes difficult to manage as the number of parties participating in the VPN increase, and it also exposes a

single point-of-failure.

For the virtual network model, we have shown that although these types of solutions offer the flexibility of constructing highly customised networks, it incorporates an inherent management and administrative overhead into the system. This is due to the fact that heavy customisation does not go hand-in-hand with flexibility, therefore any change in configuration of the virtual networks becomes a arduous manual task. This problem only increases in magnitude as the number of members of such a system increases, thus effecting the scalability of the solution as well.

For the peer-to-peer (P2P) model, we have highlighted that although the basic approach of using P2P technology is sound, the two solutions incur overheads with respect to scalability and ease of administration and management. This is basically due to the way they structure their topologies, forming a mesh network in one case and assuming a fixed and static peer membership in the other. This makes scalability an issue for the former solution, and peer churn comes in as a problem for the later.

For the cloud based model, we have discussed three popular commercial solutions. These solutions have the advantage of being designed specifically for the cloud environment, therefore scalability and ease of deployment is usually not an issue for them. However, some of these give us the problem of vendor-lock-in as they are tailor made for a specific cloud platform. Furthermore, they almost do nothing to make it easy to manage and configure the network elements of their solutions, which are usually left for the customers to administer as they would a traditional network infrastructure.

In the next chapter, we discuss a collection of methods and techniques that we

believe can be utilised to design a secure communication framework according to our objectives. We will elaborate in detail techniques concerning de-centralised control, network tunnelling, key distribution and functional encryption. This will give the reader ability to better understand the motivation for the design of different components of our solution.

Chapter 3

Background

In the previous chapter we have identified the gaps in the existing research domain that we want to fill with our research efforts by building on the state-of-the-art research currently available in security engineering methodologies. As mentioned previously, the main difference between existing work and our research effort is that the existing work predominantly focuses on providing remote access to users over wide-area networks or single/tightly-integrated cloud service providers, whereas our research effort aims towards a scalable communication framework that enables the provisioning of secure virtual private networks in a generic multi-cloud environment.

In order to achieve this aim, we have detailed the core requirements earlier in Section 1.4 and Chapter 2, which need to be fulfilled by our research process to come up with the design and architecture of an inter-cloud secure communication framework. A summarised and consolidated list of important characteristics and features for the communication framework that appear out of these requirements is as follows:-

- Decentralised and scalable architecture.
- Efficient data and configuration management.
- Low communication and performance overheads.
- Secure and strong access control.
- Decentralised and scalable key distribution.
- Efficient and low-overhead encryption techniques.

In order to cater for these characteristics and features, we evaluated a large number of mechanisms and techniques that we identified as being relevant to our stated aims. Of these, we highlight a non-exhaustive selection of the techniques and mechanisms, which according to our reviews can contribute directly towards the achievement of our research goals. Therefore, to address the issues of constructing the base of a decentralised and scalable communication framework in which the number of communicating entities can increase or decrease dynamically we study and analyse peer-to-peer overlays. To cater for the efficient data and configuration management concerns in a widely distributed environment we look into distributed data structures like the Distributed Hash Table.

To focus on the low communication and performance overheads for our secure communication framework we investigate these overheads for all the components of the framework, specifically for tunnelling protocols, as almost all of the research efforts in the related work have focused on using application layer protocols for this purpose whereas in our opinion network layer protocols like IPsec will offer better performance and reduced communication overheads, although at

the cost of added complexity in terms of configuration management. However, these added costs can also be mitigated in our research approach by utilising the distributed data structures for efficiently sharing and enforcing the IPsec related configurations. To address the challenges of secure access control and decentralised and scalable key distribution in a multi-cloud environment, we investigate the suitable techniques of key exchanges and key agreements. And lastly, to keep the overheads of core encryption mechanisms low, we try to avoid the traditional symmetric and asymmetric techniques and analyse the functional encryption approach and its variants.

In the upcoming sections in this chapter, we describe in detail the discerning aspects of these mechanisms and techniques that are beneficial to our goals and form the foundations of our further research contributions. We also highlight when, how and where we make use of these mechanisms and techniques to fulfil our research requirements.

3.1 Peer-to-Peer Overlays

An overlay network can be seen as a network of virtual or logical links and connections that exists on top of another network. As any other computer network, the main purpose of an overlay network is to facilitate the sharing of resources between the nodes comprising of that overlay, but at a higher level of abstraction than the more traditional computer networks. This higher abstraction level has proven to be quite useful in providing features like scalable and robust wide-area routing, efficient network-wide search of data and resources, discovery of nearby nodes, redundant and scalable data storage, anonymity and privacy, and

extensive scalability and fault tolerance [9].

Peer-to-peer (P2P) overlay networks are a type of distributed systems of resource sharing nodes. But instead of conforming to a client-server model in which client nodes act as consumers of resources and centralised server nodes act as the producers of resources, all the nodes in a P2P overlay share the resources among themselves and can act simultaneously both as the consumer and provider of resources.

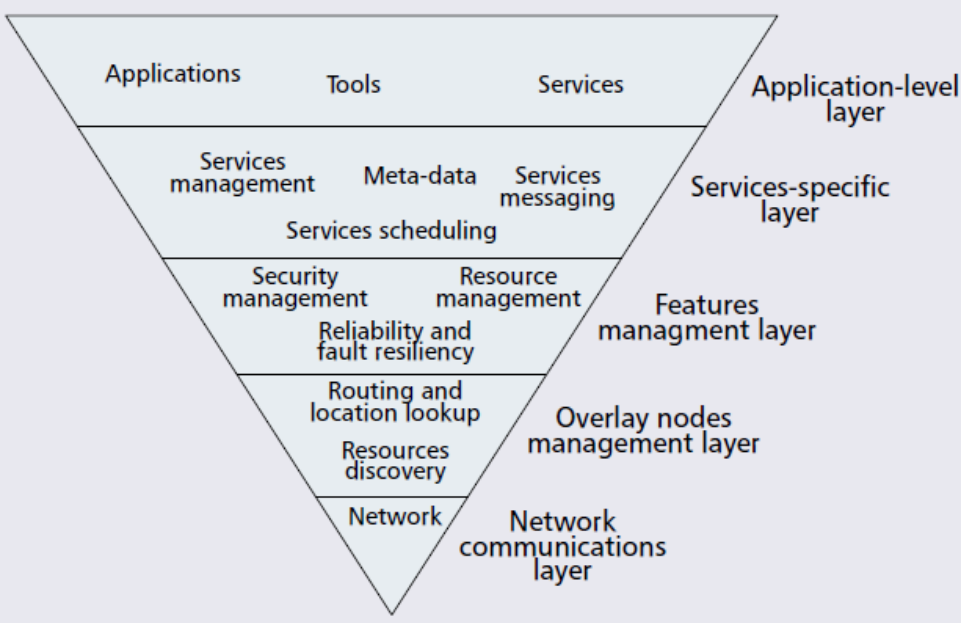


Figure 3.1: An abstract P2P overlay network architecture from [110]

P2P overlays have been used very successfully in a wide range of application domains, over a large spectrum of communication frameworks like telecommunication networks and the world wide web [24]. However, the overall architectural model of a P2P overlay remains quite consistent despite this vast range and is discussed in some detail in [110] and depicted in Fig. 3.1, showing typical components of a P2P overlay. A brief description is as follows:-

- The *Network Communications* layer addresses the network characteristics like connectivity over the Internet or any other communication infrastructure.
- The *Overlay Nodes Management* layer handles the management of peer nodes, which typically includes peer/neighbour-node discovery and P2P routing algorithms.
- The *Features Management* layer addresses the issues related to security, reliability, and fault tolerance of the P2P overlays.
- The *Services Specific* layer deals with the application-specific components present in the lower layers of the model, typically relating to task scheduling, content and file management etc.
- The *Application-level* layer deals with the applications and services which make use of the underlying P2P overlay model layers for their advantage.

Depending on how the nodes in peer-to-peer overlay networks are linked to each other, we can categorize the P2P networks as either structured or unstructured. A structured peer-to-peer network uses a globally consistent protocol to make sure that any peer node can efficiently route a search to some other peer node that has the resource required by that node. In order to achieve this efficiency the peers are organized and managed by following a specific set of rules and algorithms. This in turn leads to overlays with specific topologies and properties.

An unstructured peer-to-peer network, on the other hand, does not follow any rules to ensure any kind of structure in its consistency and are completely decentralized, at least in theory because in practice most implementations do use some

level of centralization. Their main advantage is that the overlay can be easily constructed as a new peer that wants to join the overlay network can copy existing links of another peer and then form its own links over time. However, on the down side, if a peer wants to find a resource in the overlay, its request has to be flooded through the network to find as many peers as possible that have that resource. Moreover, there is a possibility, albeit very small, that the queries may never be resolved.

To meet the requirements of our research effort, we investigate the inner working of a structured peer-to-peer overlay network known as Kademlia [114] in more detail in the next section. This peer-to-peer overlay has the added advantage of supporting distributed hash table for information exchange between the different peer nodes of the overlay. This gives us the capability of addressing the decentralisation and scalability problem space as well as the provisioning of a core mechanism that we can build on further for the purposes of efficient data and configuration management of the secure communication framework.

3.2 Distributed Hash Tables

Most structured P2P networks make use of a data structure known as a Distributed Hash Table (DHT), which provides the same service as a traditional hash table but in it the responsibility for maintaining the mapping from keys to values is distributed among different nodes i.e., instead of mapping each key to a particular array slot, as done in a hash table, the distributed hash tables maps each key to a particular node or peer. More specifically, it employs a keyspace partitioning scheme that splits the ownership of an abstract keyspace (for example a set of

160-bit strings) among the participating nodes. An overlay network can connect these nodes and allows them to find the owner of any given key in the keyspace.

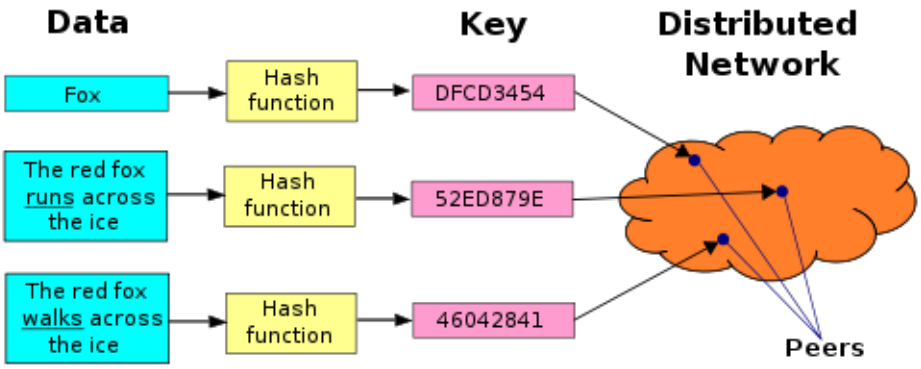


Figure 3.2: An operational description of a distributed hash table [162]

Structured peer-to-peer overlay networks based on DHT support the scalable storage and retrieval of *key*, *value* pairs on the overlay network which is very helpful when we need to store and retrieve meta-data related to the virtual private network management. Given a *(key, value)* pair, a store operation *put(key,value)* can be invoked to store the marshalled data objects corresponding to the value on the P2P overlay. Similarly, given a *key*, a retrieval operation *value=get(key)* can be invoked to obtain the data object corresponding to the key from the P2P overlay. Each peer node maintains a small routing table containing the Peer ID (IDentifier) and the IP addresses of the neighbouring peer nodes. Different structured peer-to-peer overlay systems based on distributed hash tables systems have different naming, management, and organisation schemes for their data objects, key space, and routing algorithms. In theory, DHT-based systems can guarantee that, on average, any data object can be located with-in *O(log N)* overlay hops, where *N* is the number of peer nodes in the overlay. Existing structured peer-

to-peer overlay systems like Chord [145], Pastry [135] and Tapestry [170] have been widely used to provide scalable and fast information storage and retrieval services for a vast variety of applications. We have leveraged the *Kademlia* algorithm [114] to cater for the storage and retrieval requirements of our problem space.

Kademlia works by assigning each peer node a node-identification number called *Peer ID* in a 160-bit key space. Its discerning DHT storage principle is that the *key*, *value* pairs are stored on peers with Peer IDs *closest* to the key, which is also a 160-bit number. A Peer ID based routing algorithm is used to locate peers near a destination key and guarantees that on average, any data object can be located in $O(logN)$ peer hops, N being the number of peers in the overlay. Kademlia uses a novel *Exclusive OR* (XOR) metric for distance between points in the key space, as XOR is symmetric and it allows the peer nodes to receive lookup queries from the same distribution of peer nodes which are present in their routing tables. This is because every time a message is sent by a peer node, it includes its Peer ID, allowing the receiving peer node to record the sender peers Peer ID and IP address etc. in its routing table.

To locate the *key*, *value* pairs, Kademlia utilises the notion of *distance* between two Peer IDs. For two 160-bit Peer IDs, *a* and *b*, the distance between them is defined as their bit-wise Exclusive OR, i.e., $\forall a, b$

$a \oplus b = b \oplus a$,

$a \oplus b = 0$, and

$a \oplus b > 0$ (if $a \neq b$).

XOR also offers the triangle inequality property, i.e., $\forall a, b$

$(a \oplus b) + (b \oplus c) \geq a \oplus c$

$\because\ a \oplus c = (a \oplus b) \oplus (b \oplus c)$ and $a + b \geq a \oplus b\ \forall a \geq 0, b \geq 0$.

Furthermore, XOR is uni-directional, that is, for any given point *x* and distance $\Delta > 0$, there is exactly one point *y* such that $x \oplus y = \Delta$. This uni-directional property makes sure that all lookup operations for the same key converge along the same overlay route, regardless of the starting peer location. This improves the lookup operation performance as caching the *key*, *value* pairs along the lookup overlay route alleviates hot spots.

All peer nodes in the overlay network store a list of IP address, Port No., Peer ID triples for the peers having the distances between 2^i and 2^{i+1} from themselves. These lists are called k-*buckets*. Each *k*-bucket is sorted by last time seen, with the least recently accessed peer at the head of the list and the most-recently accessed at the tail of the list. For small values of *i*, the *k*-buckets will be usually empty as most probably no appropriate peer node will exist, whereas for large values of *i* the list can grow up to the value of *k*. The *k* is a system-wide replication parameter and is chosen empirically such that the probability of any given *k* nodes failing within an hour of each other is minuscule.

The Kademlia routing protocol consists of four operations: PING , STORE, FIND_NODE, and FIND_VALUE.

1. PING pings a peer node to check if it is online.
2. STORE instructs a peer node to store a *key*, *value* pair.
3. FIND_NODE takes a 160-bit ID as input and returns IP address, Port No., Peer ID triples for the *k* peers it knows about that are closest to the input ID.
4. FIND_VALUE is similar to FIND_NODE, in that it returns IP address, port

No., Peer ID triples, except in the case when a peer has received a STORE for the key, in which case it just returns the stored value.

One of the most important operations that a Kademlia peer performs is to locate the *k* closest peers to a given Peer ID. This lookup operation starts by picking α peer nodes from its closest non-empty *k*-bucket, and then sending parallel asynchronous FIND_NODE requests to those α peers, where α is a pre-chosen system-wide concurrency parameter. If the FIND_NODE request fails to return a peer node that is closer than the peers already seen, it resends the FIND_NODE request to all of the *k* closest peers it has not already queried. To find a *key*, *value* pair, a peer starts by performing a FIND_VALUE lookup to find the *k* peers nodes with Peer IDs closest to the key and halts immediately when any node returns the value.

To join the network, a peer A must contact an already participating peer B. Peer A inserts peer B into the appropriate *k*-bucket, and then performs a peer lookup for its own Peer ID. Finally, peer A refreshes all *k*-buckets farther away than its closest neighbour, and during these refreshes it populates its own *k*-buckets and adds itself into other peers *k*-buckets as required.

3.3 IPsec

IPsec, short for Internet Protocol Security [55], is an end-to-end protocol suite based on an Internet Engineering Task Force (IETF) standards [15]. It is used for securing IP layer communication by authenticating and encrypting each IP packet of a communication session. IPsec consists of a set of protocols for nego-

tiation of cryptographic parameters and keys to be used during a communication session and establishing a mutually authenticated session between the communicating hosts. IPsec can be used to secure the IP traffic between a pair of hosts (host-to-host), between a pair of network gateways/router (network-to-network), or between a network gateway/router and a host (network-to-host) [35]. One of the main advantages of IPsec is that as it operates on the Network Layer of the TCP/IP model, applications do not need to be redesigned in order to use it rather it is transparent to most of the application protocols using it. Another advantage of IPsec is that it is currently supported and implemented in almost all operating systems kernels.

IPsec can operate in two modes:

- Transport Mode
- Tunnel Mode

Transport Mode is used to protect end-to-end communication between two hosts. In this mode, only the IP packet payload is encrypted and/or authenticated. The routing behaviour of the packet remains intact as the IP header is neither modified nor encrypted. However, if the Authentication Header is used, the IP addresses cannot be modified as this will invalidate the hash value. Therefore, Authentication Header is incompatible with NAT (Network Address Translation).

Tunnel Mode is used to encapsulate IP packets inside another IP packet and sent to the destination. In this mode, the entire IP packet is encrypted and/or authenticated and then encapsulated into a new IP packet with a new IP header. This mode is used to create virtual private networks by allowing the formation of tunnels between two hosts. As this mode encapsulates the complete IP header

as well as the payload, it allows the source and destination IP addresses to be different from those of the encompassing packet and hence support NAT traversal.

IPsec mainly uses two sub-protocols to perform its security operations:-

- Encapsulated Security Payload (ESP)
- Authentication Header (AH)

Encapsulated Security Payload (ESP) protocol protects the IP packet data from malicious third party attacks by encrypting its contents using symmetric cryptography algorithms such as 3DES (Triple Data Encryption Standard) [18] and AES (Advanced Encryption Standard) [1]. It also provides the security services of authentication and integrity for the IP packets.

Authentication Header (AH) protocols provides authentication and integrity protection for the IP packets. It can also protect against replay attacks, in which the attackers can capture packets during transmission and attempt to re-inject them back onto the transmission at a later time. It operates by computing a cryptographic hash-based Integrity Check Value (ICV) over all the fields of a IP packet, except the ones which are modified during transit, for example the TTL (Time-To-Live) and the IP checksum etc. It stores this ICV in a newly-added AH header along with some other parameters and sent to the receiving host.

IPsec uses Security Associations (SA) to establish and share security attributes between hosts that want to secure their communication using either AH or ESP. An SA includes attributes like cryptographic algorithms, IPsec mode, encryption keys, and other network parameters. The framework for establishing security associations is provided by the Internet Security Association and Key Man-

agement Protocol (ISAKMP). Security Associations are stored inside a Security Association Database (SADB) in each host. Upon receiving an IP packet secured using IPsec, three fields are used to locate the correct SA from the SADB, which are the sender's IP address, IPsec Protocol (ESP or AH), and the Security Parameters Index (SPI). A similar operation is performed when sending an IP packet using IPsec.

To meet the requirements of our research effort, we make use of the IPsec protocol suite instead of the more commonly used OpenVPN. This is done in light of the above background as well as the comparison studies between the two like [99], which evaluate IPsec as the better choice in terms of throughput and latency performance results. This better performance is due to IPsec being implemented in the kernel space, as opposed to the user space implementation of OpenVPN on most operating systems, as well as because of location of the IPsec in layer 3 of the TCP/IP model as opposed to the layer 5 location of OpenVPN.

3.4 Internet Key Exchange

As described in the previous section, IPsec can be used to provide confidentiality, integrity, and source authentication for IP packets transmitted from a source to a destination. These security services are provided by sharing some meta-data and security parameters between the IP source and destination. This include the type of security services provided (AH or ESP), the possible cryptographic algorithms that will be used to provide the security services, and the encryption/decryption keys. The IPsec protocol suite itself does not contain features of sharing this required meta-data and security parameters automatically and efficiently between

the IP source and destination, and it especially becomes cumbersome as the number of hosts that want to use IPsec for secure communication increases. This is where Internet Key Exchange (IKE [76] or IKEv2 [95]) comes in, which is a protocol that can be used to solve this problem, at least partially, by helping to set up a shared session secret between the two hosts. The relationship between IPsec and IKE is shown in Fig. 3.3 in context of the standard TCP/IP networking model.

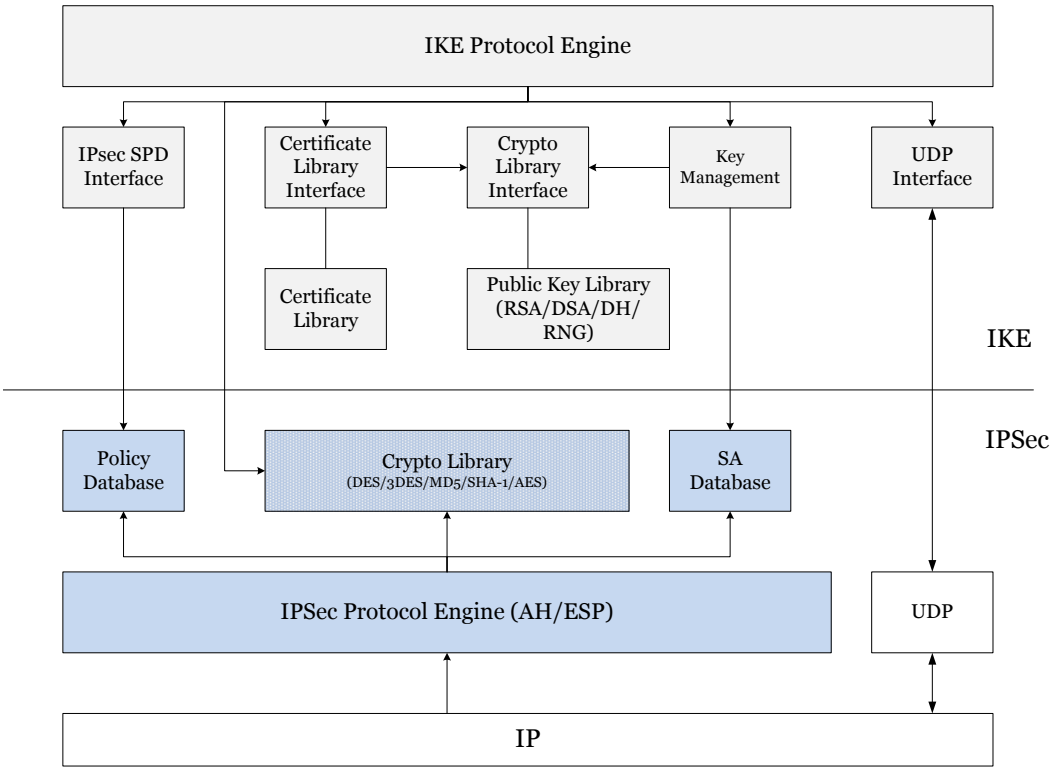


Figure 3.3: Architecture of the IPsec and IKE protocols

The IKE protocol starts by performing mutual authentication between the two hosts who want to secure their communication using IPsec. It then establishes an Security Association (SA) that includes security meta-data and parameters that

can be used to establish SAs for Authentication Header (AH) or Encapsulating Security Payload (ESP) and a set of cryptographic algorithms that are to be used by the SAs to encrypt and secure the traffic. The same functions are performed on the other end of the connection as well and a common set of cryptographic algorithms to be used between the hosts is negotiated and agreed upon. After the IKE protocol has completed its operations, the IPsec stack on the hosts machines has the required information needed to start the actual secure transmission of data. This information required by IPsec is parameters like an AES key, IP addresses of the source and the destination, TCP or UDP ports of application layer protocols that are to be protected, and the type of IPsec tunnel that is to be created.

The IKE protocol is generally supported in most of the operating systems that support IPsec. The latest versions of the Microsoft Windows operating systems fully support the IKEv2 protocol, where as almost all major distributions of Linux-based operating systems support different open source implementations of IKEv2 as well.

3.5 Key Agreement Protocols

In 1976, Whitfield Diffie and Martin Hellman came up with an algorithm that allows two parties to agree on a shared secret key over an insecure communications channel, without having any prior knowledge of each other [52]. This algorithm is now known as Diffie-Hellman (DH) Key Exchange and is widely used for the purpose of key agreement. The basic steps for reaching the key agreement between two communicating parties, Alice and Bob, are as follows:-

- Alice and Bob agree on a finite cyclic group G and a generating element $g \in G$
- Alice picks a random natural number a and sends $g^a \pmod p$ to Bob
- Bob also picks a random natural number b and sends $g^b \pmod p$ to Alice
- Alice calculates the key $K = (g^b)^a \pmod p = g^{ab} \pmod p$
- Bob calculates the key $K = (g^a)^b \pmod p = g^{ab} \pmod p$

Only values of *a* and *b* need to be private in order to secure the key agreement protocol against a passive eavesdropper. In order to compromise the system, a passive attacker will have to find the value of $g^{ab} \pmod p$, given the generator *g*, modulus *p*, and the values of $g^a \pmod p$ and $g^b \pmod p$. This is known as the Diffie-Hellman Problem (DHP) and is considered to be a hard problem for the recommended generators and modulus values.

However, Diffie-Hellman Key Exchange is vulnerable to an active attacker as it does not offer any authentication of the communicating parties. Thus, a Man-in-the-Middle (MitM) attack is possible where the attacker can establish two different DH key exchanges with Alice and Bob, effectively appearing as Alice to Bob, and vice versa. This will allow the attacker to to decrypt, then re-encrypt, the messages passed between Alice and Bob without them noticing anything. To safeguard Diffie-Hellman against an active attacker, we can use some of its authenticated variants like MQV (Menezes-Qu-Vanstone) [101]. MQV is an authenticated key exchange protocol, that is, it tries to combine key exchange with a mutual authentication of both communicating parties.

All of the following operations use a finite cyclic group G of prime order q and a generating element $g \in G$. We use $|q|$ as the bit length of q, that is, $|q| = \lceil \log_2 q \rceil$. So $\ell = |q|/2$.

The basic steps for reaching the key agreement between two communicating parties, Alice and Bob, are as follows:-

- Alice has a long term private key $a \in \mathbb{Z}_q$ with corresponding public key $A = g^a$; similarly Bob has a key pair $(B,\ b)$
- Alice picks a random number x and calculates $X = g^x$; similarly Bob picks y and $Y = g^y$
- Alice calculates $d = \overline{X} = 2^\ell + (X \mod 2^\ell)$; similarly Bob calculates $e = \overline{Y} = 2^\ell + (Y \mod 2^\ell)$
- Alice sends X and A^d to Bob; similarly Bob sends Y and B^e to Alice
- Alice computes $\sigma_A = (Y \cdot B^e)^{x+ad}$
- Bob computes $\sigma_B = (X \cdot A^d)^{y+be}$

So both Alice and Bob get to the same authenticated session key $K = \sigma_A = \sigma_B$.

$$
\begin{aligned}
\sigma_A &= \sigma_B \\
(Y \cdot B^e)^{x+ad} &= (X \cdot A^d)^{y+be} \\
(Y \cdot B^{\overline{Y}})^{x+a\overline{X}} &= (X \cdot A^{\overline{X}})^{y+b\overline{Y}} \\
(g^y \cdot g^{b\overline{Y}})^{x+a\overline{X}} &= (g^x \cdot g^{a\overline{X}})^{y+b\overline{Y}} \\
(g^y + b\overline{Y})^{x+a\overline{X}} &= (g^x + a\overline{X})^{y+b\overline{Y}}
\end{aligned}
\tag{3.1}
$$

MQV is an authenticated key exchange in that it tries to combine the key exchange with a mutual authentication of both involved parties. However, this does not map well to existing communication frameworks, in particular SSL. In SSL, the client and server have distinct roles; the client gains assurance that it knows the correct server public key through the server's certificate, but the authentication of the client, when applied at all, is separated from the key exchange (the client computes a signature with its private key, and the algorithm for that signature needs not be related to the one used for the key exchange).

Another viewpoint on the subject is that if a putative SSL client has a certificate with a MQV public key (assuming that use of MQV with SSL was formalized and implemented), then the client can use that certificate only to authenticate with SSL servers who uses MQV and happen to have a MQV key pair which uses the same elliptic curve. This is rather restrictive, and contrasts with the usual situation where the client has a generic signature certificate which can be used in many other contexts. As the purpose of a secure and authenticated key exchange protocol is directly related to our research requirements, we discuss the modification of this protocol as part of our research effort in detail later in Chapter 5.

3.6 Functional Cryptography

Encryption is a well known and established technique for securely sharing data between users and processes over insecure or untrusted networks and storage mediums. Traditionally it had been done with the help of a secret key that was shared priori between two or more parties that wanted to share data with confidentiality. There exists a large class of algorithms in literature that uses this

traditional primitive, where usually the same key is used for both the encryption and decryption process, known as Symmetric Encryption [63]. Some examples of popular symmetric encryption techniques are Twofish [139], Serpent [10], AES [1], Blowfish [138], RC4 [63], 3DES [63], and IDEA [137].

However, while these symmetric encryption techniques might be acceptable for a small and cohesive set of users, they were clearly infeasible for larger networks such as Internet, which may consist of millions of users. So about thirty years ago, a radically new encryption method was invented in the form of public key cryptography [52], where two or more parties can securely communicate with each other without having to agree to a priori mutually shared secret. Now-a-days, the use of this public key encryption or asymmetric encryption is ubiquitous in all sort of communication networks, from secure web communication to disk encryption.

However, for some emerging domains, like our focus area the inter-cloud communication, this notion of public key encryption is insufficient. This is mainly due to the inherent lack of access granularity of the public key encryption techniques, i.e., a user can either decrypt and access the entire message or he learns nothing at all about the message, other than perhaps its length. In most modern cloud services, it is often desirable to associate a decryption policy with the encrypted message and only the users who can satisfy the policy being able to decrypt the message. In more generic terms, we want to be able to only give access to a function of the plaintext message to an authorised user. Public key cryptography is not really helpful in this types of scenarios and this is where functional encryption comes in. We discuss the secure formulation of a secure resource discovery protocol as part of our research effort in detail later in Chapter 6 which uses differ-

ent variants of the functional encryption scheme described briefly in this section below.

Table 3.1: Primitives of Functional Encryption

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

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 [155] 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.

To describe it more formally but briefly, a functional encryption scheme (FE) for a functionality F defined over $(K,\ X)$ is a sequence of four algorithms (setup, keygen, encryption, decryption), as given in Table 3.1. These must satisfy the correctness condition $y = F(k,\ x)$ with the probability *1*, for all $k \in K$ and $x \in X$. The set K is called the key space and the set X is called the plaintext space. The functional encryption technique also requires that the key space K contain a special key called the empty key ϵ.

The empty key ϵ in K gathers all the information about the plaintext that intentionally leaks from the ciphertext, such as the length of the encrypted message. Therefore, anyone is able to apply $dec(\epsilon, c)$ on a ciphertext $enc(pp, x) \xrightarrow{R} c$ and get all the information about x that intentionally leaks from c.

Another advantage of functional encryption is that many encryption concepts

and constructions can be viewed as special cases of functional encryption. For example, we can show that public key encryption is a simple example of functional encryption. Let $K := 1, \epsilon$ and consider the following functionality F defined over $(K; X)$ for some plaintext space X:

$$F(k, x) = \begin{cases} x & \text{if } k = 1 \\ len(x) & \text{if } k = \epsilon \end{cases} \tag{3.2}$$

A secret key for $k = 1$ decrypts the valid ciphertexts, while the empty key $k = \epsilon$ simply returns the length of the plaintext message. Hence, this functionality syntactically defines the standard public key encryption method.

Identity-Based Encryption [141], [30], [45], Predicate Encryption [31] and Attribute-Based Encryption [80] are some examples of sub-classes of functional encryption. We show by giving a few examples that how functional encryption captures these encryption concepts.

3.6.1 Predicate Encryption

In many peer-to-peer applications, a plaintext message $x \in X$ is usually a *key-value* pair $(ind,\ m) \in I \times M$ where ind is the index of the pair, m is the payload message, I is the index space, and M is the payload message space. For example, in an email message, the index will be usually set as the sender's name or email address while the payload might be the contents of the email message.

In this context, the Functional Encryption functionality in terms of a polynomial-

time predicate $P : K \times I \rightarrow \{0,1\}$ over $(K \cup \{\epsilon\}, (I \times M))$ is defined as,

$$F(k \in K,\ (ind,\ m) \in X) := \begin{cases} m & \text{if } P(k,\ ind) = 1, and \\ \perp & \text{if } P(k,\ ind) = 0 \end{cases} \tag{3.3}$$

Now, let c be an encryption of $(ind,\ m)$ and let sk_k be a secret key for $k \in K$. Then, the function $dec(sk_k,\ c)$ will decrypt the message payload in c when $P(k,\ ind) = 1$ but will reveal nothing new about m otherwise.

3.6.2 Identity-based Encryption

An Identity-Based Encryption (IBE) scheme is an encryption scheme where any arbitrary string can be a valid public key. For example, email addresses and dates etc. can also be public keys. Its main advantage is that communicating parties may encrypt messages and verify signatures with no prior distribution of keys required between individual participants. This is obviously very useful in a lot of cases where distribution of authenticated keys is inconvenient or impractical due to technical or administrative constraints.

Identity-Based Encryption can be formally described as a Predicate Encryption scheme where :

- The key space is $K := \{0,\ 1\}^* \cup \{\epsilon\}$
- The plaintext is a pair $(ind,\ m)$ where the index space $I := \{0,\ 1\}^*$

- The predicate P on $K \times I$ is defined as,

$$P(k \in K,\ ind \in I) := \begin{cases} 1 & \text{if } k = ind, and \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

For these Identity-Based Encryption systems to properly support the empty key ϵ functionality, the ciphertext must explicitly include the ind and the length of the message $len(m)$ in the clear.

3.6.3 Attribute-Based Encryption

An Attribute-Based Encryption (ABE) scheme is an encryption scheme where the secret key of a user and the ciphertext are dependent upon attributes that the user possesses. Due to this, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext. Subsequently, this makes it possible to express complex access policies using these attributes.

The Attribute-Based Encryption concept has been refined into two types: Key-Policy ABE and Ciphertext Policy ABE.

In Key-Policy ABE, the attributes are assigned to a ciphertext when that ciphertext is created. The policies are assigned to users and keys by an authority, which is usually the same entity that creates the keys. A key can decrypt only those ciphertexts whose attributes satisfy the assigned policy.

In Ciphertext Policy ABE, the users of the system are assigned certain attributes. The users then receive a key from an authority for their set of attributes. A policy is associated with the ciphertext at the time of encryption. If a user's

attribute set satisfies the policy, he can use his key to decrypt the ciphertext.

3.7 Chapter Summary

In this chapter, we give a detailed explanation of methods and techniques that we have researched and selected for utilisation in our secure communication framework. We have included the techniques that will help us in achieving our research objectives, as stated in Chapter 1. Some of these methods and techniques will form the core of some of the components of our solution, so we have described them in detail and focused on the aspects that make them suitable for our use.

We started by describing the peer-to-peer overlays and their types, as these form the basis of the architecture of our framework. A structured peer-to-peer overlay will be the foundation of our approach to address the issues regarding the inter-cloud nature of our target environment. Therefore, we describe the specific peer-to-peer protocol that we will use in our implementation in some detail, i.e., Kademlia. We focused on its efficient storage and retrieval aspects using the distributed hash table data structure, as well as the node joining and peer discovery mechanisms.

We also described the IPsec protocol suite in some detail, along with its different modes of operations. This is important as we use IPsec and its associated technologies to provide the underlying confidentiality and integrity in the communications links established as a result of using our framework. We also described the Internet Key Exchange protocol, which we don't actually use in our solution, but it acts as our motivation to provide the key distribution functionality between the nodes using our solution, using the distributed hash table as a distribution

mechanism.

We also described the Diffie-Hellman key exchange protocol that can be used to generate the actual session keys that are required to perform the encryption operations in IPsec. We described in detail one of its authenticated variant known as MQV, which addresses the Man-in-the-Middle attacks that are possible in the standard Diffie-Hellman scheme. This variant becomes our guiding motivation with which we design a modified version of authenticated key generation and exchange protocol for use in our solution.

Lastly, we described the concept of functional cryptography and some of its types in some detail. This is attractive to us as it gives us the opportunity to control the granularity of the decryption process by associating it with proving the ownership of a predicate, attribute, or policy. This is especially useful in cloud computing scenarios if we are able to share the security credentials efficiently and securely, by ensuring that only an authorised user is able to access and decrypt these security credentials and we can control the granularity of the authorisation.

In the next chapter, we highlight when, how and where we make use of these mechanisms and techniques to construct our solution. We will see in detail how these methods provide us with further advantages, in addition to the accomplishment our main objectives. Furthermore, we will explain the methodology, techniques and tools that we have used to implement the prototype of our model framework and the various performance results that we have measured by deploying this prototype on three cloud service providers.

Chapter 4

Inter-Cloud VPN Overlay

In the previous chapters we have identified the gaps in the existing research domain that we want to fill with the techniques and protocols developed as part of our research efforts and by applying the applied security research methodologies. In this respect, the main objectives of our research effort to design and architect a secure communication framework that works efficiently in a inter-cloud environment are fourfold.

Firstly, to construct a scalable virtual private overlay network between virtual machines deployed on multiple cloud platforms. The design of this overlay network has to conform to the constraints and requirements detailed in the previous chapters. We also have to take into consideration that the number of virtual machines can increase or decrease dynamically during the life-cycle of a cloud application and that the overlay network also has to cater for this churn. However, the triggering of this churn is currently external to our scope of research.

Secondly, to include a scalable key distribution mechanism as an integral part of the framework, which will cater to the encryption and decryption-keys related

requirements of the security mechanisms and protocols that will be needed to be designed, implemented and evaluated as the core contributions of our research effort.

Thirdly, to have a design that involves as minimal manual configuration management as possible in order to make the final solution very easy to deploy. This is important also due to the fact that most of the existing cloud platforms require a lot of manual effort to correctly set up and manage even a small or medium cloud service. So when addressing service deployment and operation at an inter-cloud level, maximum automation is a highly desirable and attractive prospect for any cloud vendor.

Fourthly and lastly, we want to ensure the confidentiality and integrity of the communication as well as the sensitive data and meta-data that is exchanged between different components of the communication framework.

In order to achieve these objectives, we have come up with a collection of mechanisms and techniques that can be integrated to formulate an efficient and scalable secure communication framework that is able to satisfy our research requirements in an inter-cloud environment. In the coming sections we describe in detail the peer-to-peer model architecture we have come up with in order to address the inter-cloud nature of our target environment. Although the basic purpose of its design is to handle the distributed and dynamic placement of the virtual machines on multiple cloud providers, we will also elaborate how we have added to a standard peer-to-peer design. These additions and modifications have provided us with further advantages to accomplish our research objectives. After that we explain the methodology, techniques and tools that we have used to implement the prototype of our model and the various performance results that we

have measured by deploying and evaluating this prototype on one academic and two commercial cloud platforms.

4.1 Design and Architecture

The design and architecture of our inter-cloud secure communication framework is inspired by a collection of techniques like Virtual Private Networks [148] (VPN) and Peer-to-Peer (P2P) Overlays [9]. Network virtualization techniques like VPNs and P2P Overlays have been shown to provide their users with legacy communication functionalities of their native network environments, despite the topology, configuration and management architecture of the 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 the complications and complexities of managing a physical network are abstracted by the overlay network, enabling the virtual machines deployed on multiple clouds to benefit from a customised communication network typically only available in physical local-area environments. However, there are some problems in using the traditional designs and architectures of Virtual Private Networks and Peer-to-Peer Overlays for our secure communication framework that we have to address before utilising them in our solution.

Traditionally, most of the private network solutions for similar problem spaces require the direct and continuous control of a centralised administration entity over every aspect of the overlay network, consisting of all the participants that constitute and facilitate the operation of the service being deployed and run on the multiple cloud providers. Such a central controller provides services to au-

thenticate, secure and police the interactions amongst peers. These centralised solutions make it almost necessary to provide complex support and management functionalities to meet the user demands of smooth and continuous operation.

Furthermore, to robustly handle the loads generated by a large number of users, significant infrastructure resources and services like mirroring or redundant instances and load-balancers must be set aside, incurring additional costs for the service owner. Peer-to-Peer overlays, on the other hand, are designed to offer improved scalability, flexibility and availability in a distributed fashion without extensive reliance on centralised servers or resources. For these reasons, such overlay networks have been used very successfully to provide specialized application layer services like voice over IP (VoIP) e.g., Skype [19] and file sharing e.g., Bittorrent [46].

Having discussed the shortfalls of the centralised approach used in virtual private networks above, we also need to highlight some common problems present in current Peer-to-Peer techniques that adversely effect our research efforts. These problems are concerned with the seamless bootstrapping of services like peer discovery and resource advertisement and discovery. In all structured Peer-to-Peer overlays, a joining peer is required to have relevant information about at least one other peer that is already in the overlay, and the current structured Peer-to-Peer overlays do not offer a scalable solution for the seamless bootstrapping of these services. This can be seen in the specifications of the popular structured Peer-to-Peer overlays [14] and [91].

Therefore, a key research contribution is the following architecture of a scalable communication framework that can bootstrap multiple Peer-to-Peer overlays, each able to provide the VPN functionality over multiple cloud providers' infras-

tructure. We strive to explain the detailed design of our core secure communication framework in two segments. In the first segment (Section 4.1.1) we describe the architecture of the core communication framework in terms of its salient components and their relationship with each other. In the second segment (Section 4.1.2) we describe the core work-flow and protocol design that is followed in order to establish the secure communication links between these components.

4.1.1 Inter-Cloud VPN Overlays

The core research innovation employed in our Inter-Cloud VPN framework is the loose and dynamic integration of two tiers of peer-to-peer overlays, i.e., a 'universal peer-to-peer overlay' and service or application-specific 'VPN overlays'. The universal overlay is the top tier peer-to-peer overlay that every peer node (virtual machine) participating in the formation of a secure virtual private network is expected to join. Therefore, the main role of the universal overlay is to act as a facilitator that provides services to bootstrap and launch the individual VPN overlays. Each VPN overlay is formed dynamically and is logically separate from the universal overlay, however, some of the peer nodes of a VPN overlay are a subset of the super peer nodes of the universal overlay.

A distinct benefit of our framework design is that a single universal peer-to-peer overlay can be used to provide a scalable and secure infrastructure service for initiating and binding multiple VPN overlays on top of different cloud platforms, as long as the peer nodes of the universal overlay are accessible to the peer nodes of the VPN overlays deployed on these cloud platforms. The universal overlay itself can be initiated either by the service owner, a cloud broker or the

cloud service providers and kept in operation for an indefinite amount of time, thus providing a long-term and reliable universal launch service to individual VPN overlays.

The Fig. 4.1 shows a reference deployment scenario where the universal overlay has been initiated on two different cloud service providers and the peer nodes of the universal overlay are accessible from the VPN peer nodes running on virtual machines that have been deployed on three cloud service providers. To avoid creating a chaotic and confused depiction, Fig. 4.1 only illustrates the composition of a single VPN overlay over the three cloud service providers. However, the benefits of the flexibility and decentralised nature of our architecture is not hard to conceptualise here as a large number of VPN overlays can be constructed with the help of the universal overlay, depending on the requirements of the users and the cloud service providers.

As discussed earlier, the universal overlay is the overlay that the peer nodes of the underlying VPN overlays have to join and therefore it helps with the bootstrapping activity of the VPN overlay peers. Thus the universal overlay essentially adds a layer of abstraction over the underlying VPN overlays. This abstraction can also come in handy when providing other functions 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 eventually peered overlay resources, which in this case are virtual machines, and spans 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 the VPN service.

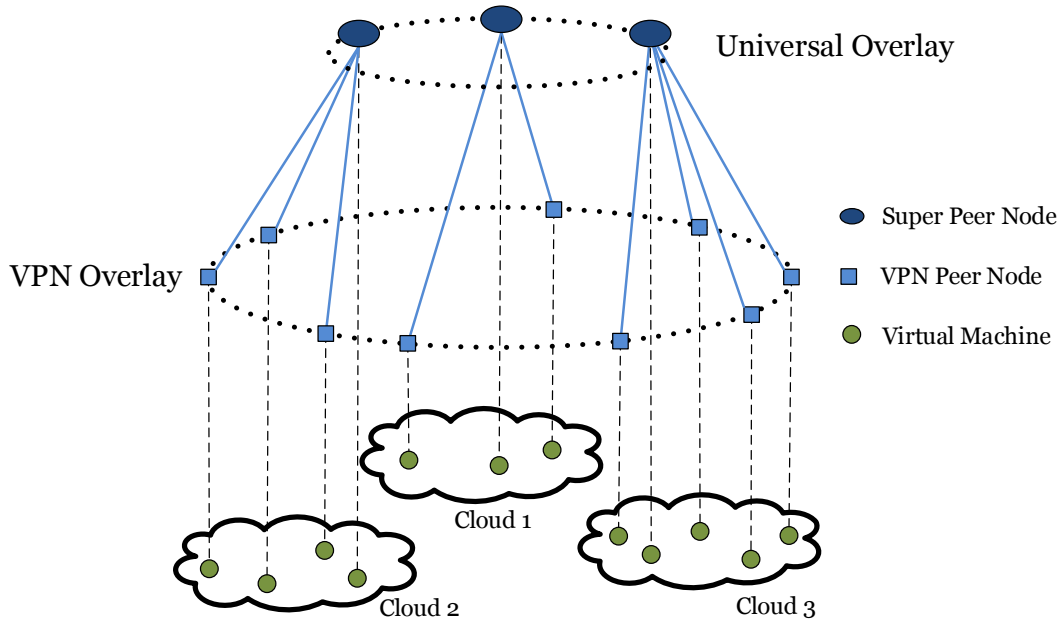


Figure 4.1: The two-tiered architecture for the Inter-Cloud VPN, with the nodes of the Universal Overlay acting as the super peers whereas the nodes of the VPN overlay acting as normal peers

As depicted in Fig. 4.1, a service owner/user, a cloud broker, or a cloud service provider could itself be a peer node in the universal overlay and a subset of the universal overlay peer nodes can act as super-peers for the peer nodes of the VPN overlay for a particular cloud service or application. 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. As both the universal and the VPN overlay nodes are basically running on VMs provisioned from different cloud providers, they can be demoted or promoted from these overlays respectively based on parameters like performance and availability.

To join the universal overlay, each peer needs to acquire a unique identifica-

tion 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 cloud service or application that this particular VM belongs to and that cloud service's or application's secret key. This data is embedded in a secure cache on the virtual machine by a VM contextualization service [13], when it is provisioned for the cloud service deployment and the same contextualization service is used to install the peer-to-peer client in the VM[1].

After the bootstrapping phase, the peer follows the admission control protocol described in Chapter 5, Section 5.3 for a validated admission into the overlay, using the cloud service's or application's secret key as the required password.

After the completion of the bootstrapping process, the VPN peer requests for enrolment with one of the super peer node in the universal overlay. The super peer checks its enrolment policy to see if the requesting peer node passes the requirements. After the successful enrolment process, the peer authenticates itself with the super peer using the secret key provided to it by the VM contextualization service. For this, the peer follows the admission control protocol described in Chapter 5, Section 5.3 for an authenticated admission into the overlay, using the cloud service's or application's secret key as the required password.

If the authentication is successful, the peer asks for the list of the neighbouring peers in its overlay that are part of the same cloud service. The super peer can query for all the peer nodes that belong to a particular cloud service by using the *serviceID* and return the results to the requesting peer. Similarly, the peer

[1] See Appendix A for a more detailed description

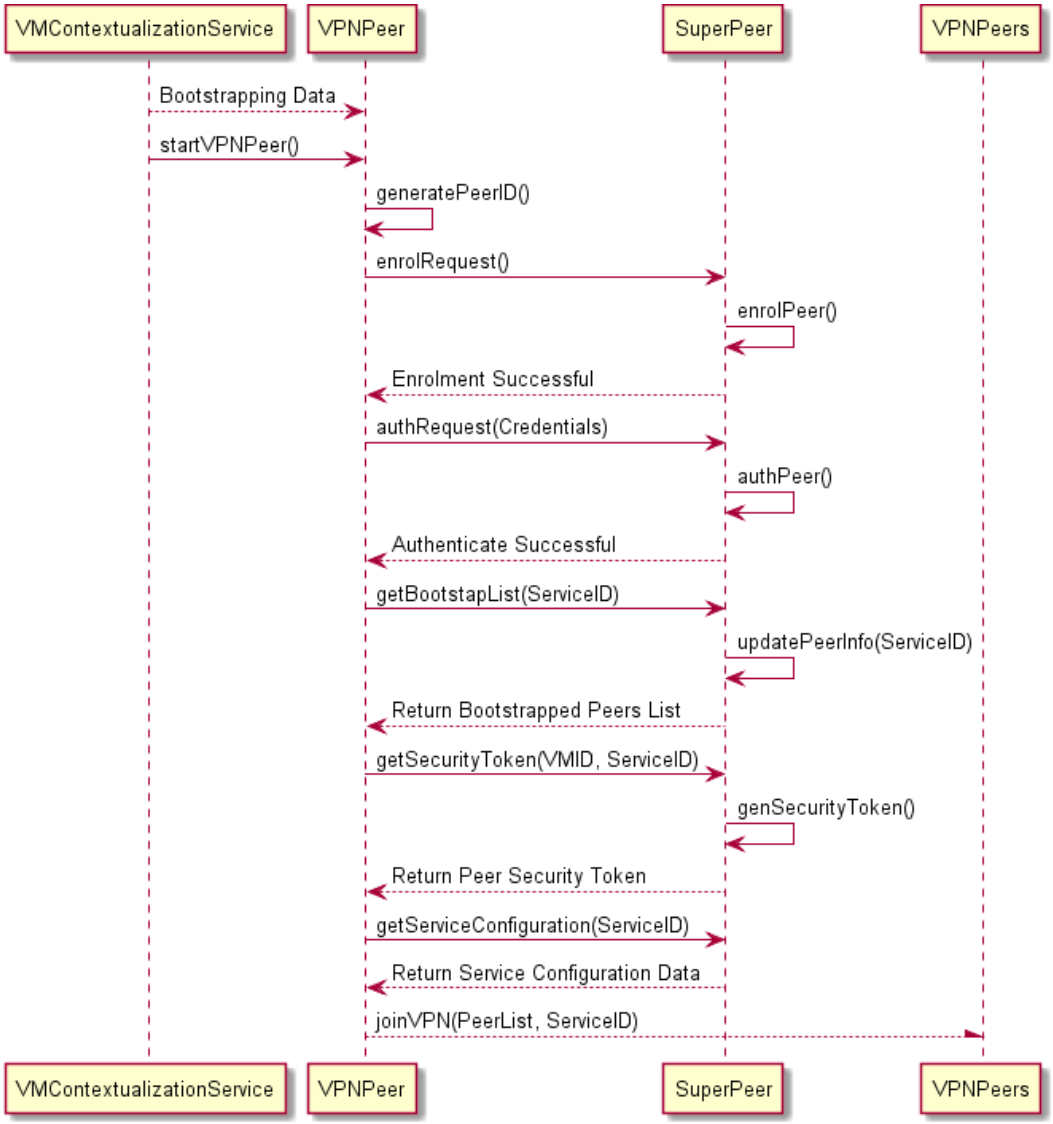


Figure 4.2: Sequence diagram depicting the steps undertaken for the formation of a VPN Overlay, with the VM Contextualization service bootstrapping the process and the SuperPeer facilitating with secure enrolment and automatic configuration etc.

asks for the security tokens/keys that it needs for use in the protocol to form secure communication channels with other peers of its VPN overlay and the initial configurations that it requires to follow for the protocol.

In the current implementation we use and compare two models; the Public Key Infrastructure model (PKI) where the super peer act as Certificate Authority (CA) and can issue signed certificates to the authenticated peers which are valid for a fixed time duration. The peers can use these certificates to validate each other as well as use them in the security protocols which support the PKI model. In the upcoming Chapter 6, Section 6.4 we also develop and evaluate an alternative to the PKI model which uses Functional Cryptography to solve the same problem more efficiently.

After all this information is made available to the peer, it joins the VPN overlay and starts with the process of constructing secure communication tunnels with other peers of the same VPN overlay according to the policies it has received in the configuration data. The configuration data can be updated dynamically and all the peers check with the super peers periodically so that they can apply and use the latest policies according to the service demands. A sequence diagram describing this flow of operations is depicted in Fig.4.2.

In a typical usage scenario, the service/application owner is responsible for provisioning virtual machines from cloud service providers to deploy and run their services. These virtual machines are considered as the peers of the VPN overlays and the complete life-cycle of the peers is handled by a peer-to-peer client embedded in the appliance image used to instantiate a virtual machine on a cloud platform.

However, a further advantage of the universal overlay approach is that the peers of a VPN overlay can get, update and modify the peer-to-peer client program dynamically from the super-peers in the universal overlay. The program to be run is signed by the super-peers for validity and it can check for updated ver-

sions of itself by querying for the associated *serviceID* in the persistent store of the DHT of the universal overlay.

4.1.2 Secure Virtual Private Connections

The main components of the peer-to-peer client used to construct a virtual private network in our model are shown in Fig. 4.3. 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 (C&C) channel for the ICVPN solution, key-based routing, peer discovery, bootstrapping service and overlay maintenance service. All of these services are constructed by implementing the Kademlia protocol mentioned in detail in chapter 3 section 3.2.

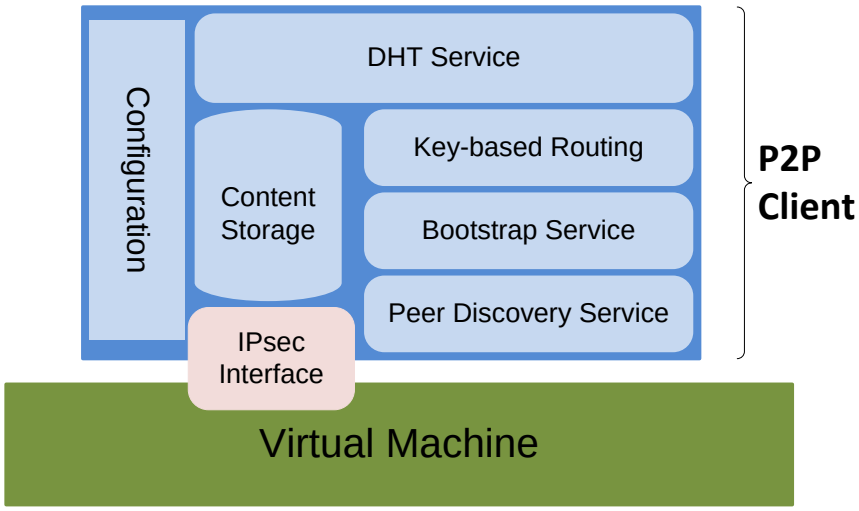


Figure 4.3: Architecture of a Inter-Cloud VPN *P2P Client* node, the architecture being identical for both super peer nodes in the Universal Overlay and VPN peer nodes in a VPN Overlay

In addition to these peer-to-peer specific components, we specify and set-aside a secure content storage for the client where sensitive data like keys, pass-

words, 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 pulling new configurations from the super peers. The configuration component manages both the peer-to-peer related configurations as well as the policies used to configure the IPsec tunnels between the peers for the use of the higher-level services using the client to provide the secure communication framework.

The peer-to-peer client software sets up and configures the IPsec Security Associations according the cloud service/application network security policy, which is advertised by the cloud service/application 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 Peer-to-Peer client's IPsec interface.

The key feature of our Inter-Cloud VPN is establishing a secure communication tunnel 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 policy of that particular service. To achieve this, we make use of the Internet Protocol Security (IPsec) protocol suite [55] to authenticate and encrypt each IP packet of a particular 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 IPsec session and negotiation of cryptographic keys to be used during the ses-

sion, we employ the Internet Key Exchange protocol [95], which can make use of standard cryptographic primitives like public key cryptography [52] and AES [1]. In our current implementation, we have used and compared two key exchange methods. First is the PKI certificate-based method where the super peers act as a Certificate Authority (CA) and 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. Second is the authenticated key exchange scheme, described below, which is used to derive a secure session key which can be used in the Cipher-Block-Chaining (CBC) mode with AES to ensure the confidentiality of the traffic exchanges between the peers using the tunnel [82].

This second approach removes the Diffie-Hellmans well-known susceptibility to an active Man-in-the-Middle attack. This is done by providing a way to mutually authenticate the key exchange between communicating peers. In most traditional systems, this is done by depending on digital signatures backed by a centrally managed PKI. However, it has been shown from a practical point of view that deploying and managing a central PKI can be a complex and problematic experience as evident from the DigiNotar and Comodo incidents [103]. PKIs require too many managerial as well as computational and communicational resources, which are not easy to commit by a small scale cloud service customer. Especially in our target use case, where such customers wants to use the resources of multiple cloud providers, they typically does not want to deal with issues like cross-carrier authentication, certificate revocation lists, and other complexities.

It is therefore a much simpler approach to avoid PKIs altogether, especially

when developing secure commercial products. Hence, we augment the Diffie-Hellman key exchange with secure hash usage at the start of the key exchange and so PKI is not needed for this approach to mutually authenticate the key exchange. The session keys generated as a result of this method, 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. The peers of the VPN overlay use the following protocol to agree on a secret key ***S*** and parameters for establishing the IPsec tunnels between the VMs for secure communication. This protocol comes into action immediately after the communicating peers have completed the resource discovery phase and want to proceed to the secure communication phase.

All of the following operations use a finite cyclic group ***G*** of prime order ***p*** and a generating element ***g*** ∈ ***G***. The initiating peer ***A*** generates its ephemeral key pair before entering the secure communication phase. The peer begins the exchange by sending a Hello message to the other peer. The Hello message contains the peer ID of the peer. The peer ID is a unique 160-bit random string (***PID***) that has been generated by the peer-to-peer algorithm (in this case Kademlia) and can be used to index and look up credentials and configuration data from the overlay DHT for a particular peer. The responding peer ***B*** replies with a Hello message of its own, containing its ***PID***. On its receipt of the response, peer ***A*** sends the cyclic group generator ***g***, the prime ***p*** and $\widehat{A} = g^a \pmod p$ to the peer ***B***. A hash of the public parameters ***g, p***, $\widehat{A}$, and the Hello message of responder ***B*** is performed and sent in the same message to prevent active Man-in-the-Middle attacks.

$$hash(g \| p \| \widehat{A} \| B(Hello))$$

All subsequent messages also contain a hash image that is used to link the messages together. This allows rejection of false messages injected during an exchange by an active Man-in-the-Middle attacker. On receipt of the above message, peer ***B*** checks the hash using the received public parameters for ***A*** and its own Hello message. If it matches, it generates its own random secret value ***b*** and computes its public parameter $\widehat{B}$, i.e., $\widehat{B} = g^b \pmod p$, and sends it to ***A*** with the hash. It then calculates the result as,

$$\widehat{R} = (\widehat{A})^b \pmod p$$

Now ***A*** can deduce the same result as,

$$\widehat{R} = (\widehat{B})^a \pmod p$$

For the calculation of the shared secret ***S***, first a total hash H_τ of all the received and sent messages in the current exchange is calculated by both peers. The final shared secret is the hash of a concatenation of the $\widehat{R}$, the PID's of ***A*** and ***B***, and the H_τ.

$$S = hash(\widehat{R} \| PID_A \| PID_B \| H_\tau)$$

The PIDs act as the context fields and H_τ as a nonce value, as recommended in [39].

4.2 Prototype Implementation

We have implemented a working prototype of Inter-Cloud VPN architecture and its constituent Peer-to-Peer clients using the Java programming language [59] that can be deployed on Linux-based operating systems [153]. We were motivated by the following reasons to use Java for the prototype development, after suffering from some initial problems with other technologies:-

- A large number of cryptographic and peer-to-peer protocols and libraries have been developed in Java and are easily available.
- It was easier to integrate components of our framework with the VM Contextualizer service as this service is also developed in Java and is exposed as a Java Remote Procedure Call (RPC).
- We were able to utilise the Java NIO2 libraries to develop multi-threaded, non-blocking and scalable core networking components that perform much better than a traditional socket-based implementation.
- There is excellent documentation and useful community help easily available online for Java related technologies.

The choice of using Linux as the base operating system to run and test our prototype instances due to the following reasons:-

- As it is an open source operating system, we did not have to worry about licenses.

- After cursory evaluation, it became clear to us that instances of Linux based virtual machines launch much faster on most cloud computing platforms than their proprietary competitor.
- The IPsec protocol suite is fully implemented and available in the most recent Linux based operating systems.
- A large number of useful tools are available in Linux for the management of IPsec tunnels and connections like *ipsec-tools*, *racoon*, *OpenSwan* and *StrongSwan* etc.

Other than the core components of our communication framework, the implementation of our core research contributions (mechanisms and protocols) was also done using open source libraries and APIs. Specifically, we chose the *BouncyCastle* library [124] to implement and batch together most of the required cryptographic primitives, as well as the PKI alternatives that were required for the comparisons. This was due to the fact that *BouncyCastle* is one of the most lightweight and extensive cryptographic libraries that is designed with very strong emphasis on standards compliance and adaptability. In the same vein, we used the *cpabe* library [26] for building the Functional Encryption mechanisms, and the *TomP2P* library [28] for its implementation of the *Kademlia* [114] peer-to-peer protocol and the overlay DHT. In addition, we use the commercially available *BT Compute Cloud* platform [32], *Flexiant FlexiScale* cloud platform [66], and a Xen hyper-visor based cloud platform [17] from *ATOS Origin* as our experimental testbed.

4.3 Experimental 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 cloud platforms provided by ATOS Origin, British Telecom Ltd., and Flexiant Ltd.

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 platform geographically located in Ipswich, England, Flexiant cloud platform located in Livingston, Scotland, and ATOS cloud platform located in Barcelona, Spain. We define service latency as the inter-cloud round-trip time taken by a HTTP (Hyper Text Transfer Protocol) [65] request, issued by a service component on one cloud, to get a response from the target service component on a different cloud. Similarly, service throughput is the inter-cloud network throughput between service components deployed on different clouds.

In this section we introduce the methodology we used to design, conduct and evaluate our experiments. We firstly explain the metrics that we measure in our experiments, and then the methodology we used to measure them.

4.3.1 Latency Evaluation Methodology

In the domain of network performance measurement, latency is traditionally defined in term of round-trip time (RTT), that is, the length of time it takes for a

source to send a data packet to a destination and receive a reply from the destination. Latency is used most commonly in form of RTT as it can be measured from a single point, so as long as the source and destination are using a well-defined and consistent network protocol, there is no need to set up or manage a latency experiment on the destination.

An example of such protocol is Internet Control Message Protocol (ICMP) [129], which is usually implemented on all network devices and operating systems. The ICMP contains a large number of message types that can be used for network diagnostic or traffic control purposes. The ones typically used for measuring latency are the *Echo Request* and *Echo Reply* messages. Various network tools are available that utilise these ICMP messages to measure network latency, packet loss and traffic jitter etc.

4.3.1.1 Measurement Tools

Some of the tools commonly used for accurate measurement of latency in computer networks are ping [121], traceroute [42], MTR [97], PathPing [117], and nmap [111]. However, all of these use the ICMP messages described earlier in some form or another to calculate the RTT latency between network hosts. The most prevalent of these is the ping tool that gives the results of the ICMP echo request and reply messages in the form of a statistical summary of the reply packets received. This includes the minimum RTT, maximum RTT, mean RTT, and the standard deviation of the RTT.

However, the ping utility has been abused in the past as a form of Denial-of-Service attack on some networks. This was historically done as a Ping Flood

[160] or Ping of Death [112]. Although most of the ping related security vulnerability have been fixed over the years, most of the commercial network environments still disable or block ICMP traffic coming into their networks. As a result it is impossible to use ICMP based tools to conduct any latency related experiments on these networks.

Therefore, we make use of the Httping tool [156], which works on the same principal as the ping tool, but instead of using the ICMP messages, it uses the HTTP protocol requests and response messages to measure the latency. As this latency is measured at the application layer, instead of the network layer as in ping, the RTT time is usually greater for HTTP requests. So to keep the size of the requests minimum, we only send a HTTP request and retrieve only the Header field of the HTTP reply message, which contains the required timestamps used in calculating the RTT.

4.3.2 Throughput Evaluation Methodology

Throughput is traditionally defined as the amount of data that is successfully sent from the source host to the destination host via a network link. It is one of the key metrics that has to be measured when investigating the performance of a network protocol or application. In case of a LAN environment where the network link is usually Point-to-Point, the *maximum theoretical* throughput is usually very close to the channel capacity of the physical link, this is,

$$Throughput \approx \frac{Bitrate \times Transmission\ Time}{Round\ Trip\ Time}$$

However, this is not the case in WAN links, like those encounters in the inter-cloud environment. For example, the TCP [36] throughput may be quite limited as compared to the theoretical channel capacity of the links between the communicating hosts, as it is affected by every component along the route from the source to the destination, including all the hardware and software components like switches, routers and forwarding, and routing protocols. Therefore, as its not possible to measure the maximum theoretical throughput, we have to measure the *achievable* TCP throughput, that depends on the network link capacity, the TCP/IP stack implementation on the networking components, their processing power, NIC (Network Interface Card) speeds, and the buffer sizes on end hosts. Hence, we need a tool that is able to handle this complexity for us and can give us accurate achievable throughput results.

4.3.2.1 Measurement Tools

For conducting and measuring the throughput experiments on our solution, we make use of the Iperf [152] tool, which measures the throughput between two hosts by measuring the amount of data sent over a fixed interval of time. Iperf is quite commonly used in both academic and commercial to gather and evaluate network performance statistics [4], [130], [43], and [126]. Iperf is also attractive for us as it allows variations in many TCP parameters like Window Size and amount of data to be transmitted. However, Iperf needs to be run for longer interval of times or in multiple bursts, to counter the effects of TCP Slow Start. Slow Start [144] is the congestion control mechanism used by the TCP to avoid sending more data than the network is capable of transmitting. For our experiments, we

overcome the effects of TCP Slow Start by first measuring the minimum amount of data that needs to be transmitted between VMs on different cloud platforms that will give us stable and reliable throughput results.

4.3.2.2 Data Size for Throughput Experiments

We measure the minimum amount of data that is required to be transmitted between three different cloud platforms, which are BT (British Telecom), Flexiant and ATOS cloud platforms. Finding out a fixed data size to be transmitted in the ICVPN throughput experiments is important for two main reasons. Firstly, it will greatly simplify the process of measuring throughput between different cloud platforms as it is the otherwise the only variable parameter required by the Iperf tool. Secondly, a minimum size is also important to bring down the bandwidth costs incurred when transferring data in or out of a commercial cloud. In order to measure achievable throughput reliably, we set up the Iperf tool on these cloud platforms and run data transmission tests between them, gradually increasing the amount of data transmitted in each successive test.

We start by transmitting *1 MB* random data between the cloud platforms and go up to *50 MB* of data, and measure the throughput achieved for each run. This upper limit of 50 MB was chosen on the basis of empirical inference from experiments done in the similar domains, as in [93] and [157]. We consider each run of these 50 tests as one experiment and repeat each experiment two more times at different times of the day to get a more rounded vision of the achievable throughput between the cloud platforms. The results of the 50 tests for each experiment between ATOS and BT cloud platforms are given in Fig. 4.4, giving us

150 data points in total.

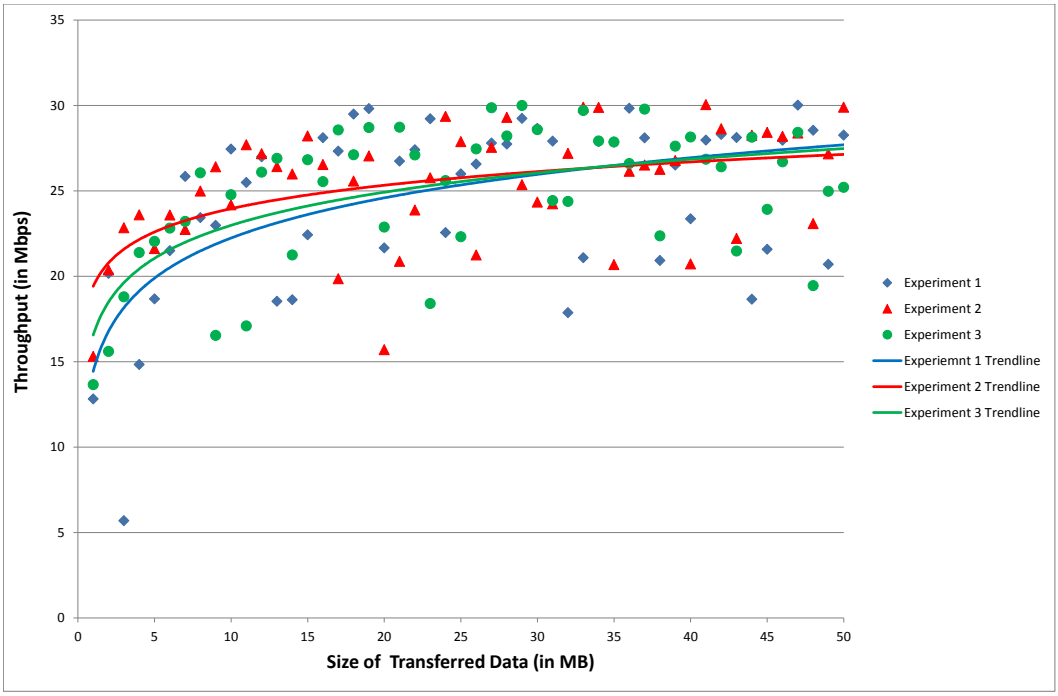


Figure 4.4: Plot of 150 throughput measurements of 1-50 MB data transfers between ATOS and BT cloud platforms in order to find the most stable 3-tuple measurements

We plot the trend-lines for each experiment in Fig. 4.4, that depicts the throughput as a function of data size, for transfers from ATOS to BT cloud platforms. We got almost identical results for the opposite direction experiments as well (BT to ATOS cloud platforms), so we only include single-direction results here. These trend-lines conform to the typical TCP throughput behaviour [109], that is, it increases exponentially with increase in data size and then stabilises as the network capacity is reached, thus following a power law with a long tail. This is due to the well-known [169] strong correlation between throughput and data size. However, we are mainly interested in finding out that against what amount of data are we able to achieve the most *stable* throughput results among the three experi-

ments. We define the most stable throughput results as the results having the *least* amount of standard deviation among them for a fixed amount of data. From Fig. 4.4, we can see that the most stable throughput was achieved for *39 MB* data size.

Similarly, the results of the 50 tests for each experiment between BT and Flexiant cloud platforms are given in Fig. 4.5 and Fig. 4.6. We conducted the throughput experiment in both directions between the BT and Flexiant cloud platforms because of the major difference in the throughput results depending on the direction. When transmitting data from BT to the Flexiant cloud platform, the throughput trend shows a very stable rate from almost the start and maintains it for most of the experiment data points. The average throughput achieved in this direction was *47.31 Mbps*.

This changes to an average throughput of *117.32 Mbps* when transmitting data from Flexiant to the BT cloud platform, which is more than double the throughput seen in the reverse direction. In addition to that, throughput trend-lines also follows the typical and expected throughput behaviour that was also observed in Fig. 4.4. Nevertheless, these difference do not fundamentally effect our evaluation as our focus is on the stability of the throughput to find out the optimal data size to be transmitted, not the non-conformance of the trend-lines to the known throughput behaviour.

Furthermore, as it is very difficult to ascertain the exact and detailed knowledge of the underlying physical wide-area network connectivity between the two cloud service providers, we cannot speculate on the reason of this difference based on direction of transmission. However, such differences are not unheard of in this domain and are usually due to differences in upstream and downstream

traffic throttling policies, differences in routes chosen by the IP packets, and firewall policy issues. We did not see similar issues between BT and ATOS cloud platforms, therefore we only include the experiment results for tests conducted from ATOS to BT cloud platforms.

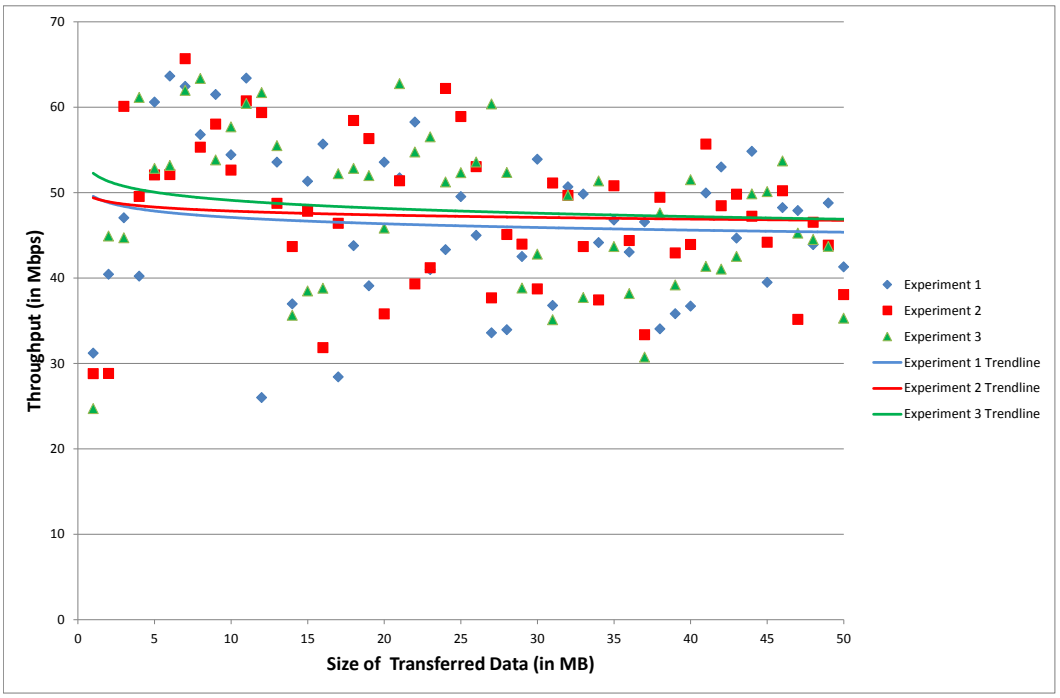


Figure 4.5: Plot of 150 throughput measurements of 1-50 MB data transfers *from* BT *to* Flexiant clouds in order to find the most stable 3-tuple measurements

As in the case of ATOS - BT throughput experiments, we again plot the trendlines for each experiment in this case as well, as shown in Fig. 4.5 and Fig. 4.6. We observe that the most stable throughput was achieved for *32 MB* of data transmitted from BT to Flexiant cloud platform and *17 MB* for the reverse direction.

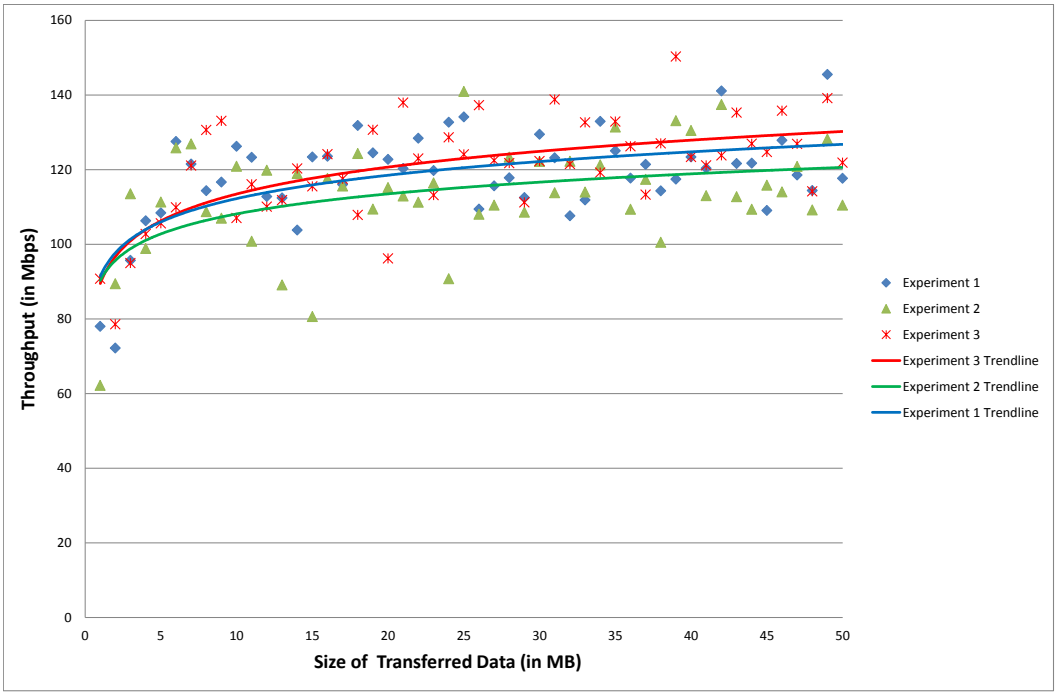


Figure 4.6: TPlot of 150 throughput measurements of 1-50 MB data transfers *from* Flexiant *to* BT clouds in order to find the most stable 3-tuple measurements

The summarised results for the most stable achieved throughput for all one hundred and fifty experiments conducted on the three cloud platforms are given in Table 4.1. Based on the mean data size of these results, we chose the fixed data size of *30 MB* when conducting throughput experiments on the ICVPN solution.

Table 4.1: Throughput results with least standard deviation against corresponding transmitted data size

Cloud Platforms	**Throughput Experiments**	
	Least Std Dev	*Data Size*
ATOS - BT	0.43	39 MB
BT - Flexiant	0.44	32 MB
Flexiant - BT	0.77	17 MB

4.3.3 Scalability Evaluation Methodology

Although there is no generally accepted definition of scalability, it can be identified as the ability of a system to handle and process increasing amount of work load efficiently. In the domain of computer networks, a system is usually said to be scalable if the addition of resources proportional to the increase of work load increases or maintains its current level of performance. An ideal and desirable characteristic of scalable solutions is that the increase in resources lead to a linear increase in service capacity. In other words, if there are a n entities that are affecting the work load of a system, then the amount of resources required to process the increased load must increase less than n^2 [58].

Analysing and measuring the scalability of a system is considered quite important, especially when designing new architectures [78], as the design and architecture of a system has the greatest influence on its scalability. In fact, a better design can give better scalability to a system than better hardware or more code optimisation and fine-tuning. Therefore, it is important to note here that scalability and performance are two separate entities. In fact, sometimes optimising a system for maximum performance can sometimes adversely affect the scalability of that system [29].

4.3.3.1 Measurement Tools

There are different types of scalability discussed in the literature that can be measured and analysed for different kind of systems, protocols and architectures. These include load scalability, space scalability, space-time scalability, and structural scalability [29] etc. However, for analysing the architecture of Inter-Cloud

VPN solution we analyse and evaluate its load scalability. This is generally defined as the the ability of a system to function normally under increased work loads, without requiring the use of exponentially increased resources. This type of scalability is appropriate for evaluation in the case of Inter-Cloud VPN, as although we have a distributed peer-to-peer architecture, the main resource that is managing the bulk of the operations at the start up phase of the VPN overlays is the number of super peer nodes in the Universal Overlay.

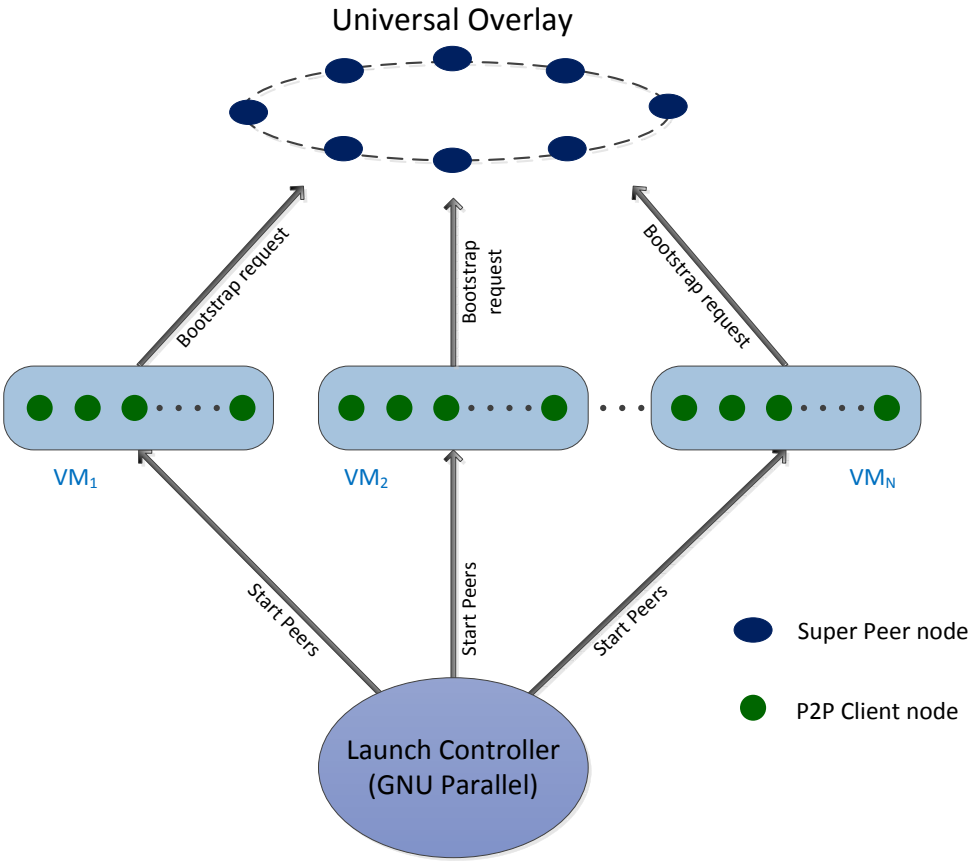


Figure 4.7: Design of the load scalability experiment to measure the effects of increasing the numbers of parallel bootstrapping requests from the VPN peer nodes (P2P Clients) to the Universal Overlay

Therefore, in order to measure the load scalability of our solution, we set up an

environment where we can run the experiment of increasing the number of super peers and measuring the number of requests per second that they can handle from the peer-to-peer clients. In order to do so efficiently, we make use of the GNU Parallel tool [149]. GNU parallel is a Linux shell tool that can be used for executing programs in parallel, using one or more computers. This is useful for us as it is logistically and financially hard for us to create thousands of VMs on our cloud platform test-beds, therefore we use a limited number of VMs in each of our cloud platforms and instantiate and large number of peer-to-peer clients on each VM. The design of this experiment is shown in Fig.4.7.

So in other words, we use GNU Parallel as a load management tool. Its job is to launch a small bash script that starts a peer-to-peer VPN client and give it the address of the super peer as input. In case of multiple super peers, we chose the super peer, to be used by the peer-to-peer client for bootstrapping, in a round-robin fashion [142]. This method is chosen in order to distribute the load uniformly among the participating super peers. Our GNU Parallel script is able to split the input list of super peers and pipe it into the launch commands in parallel.

Furthermore, the super peers are multi-threaded and implement asynchronous I/O so that they do not block on the client's bootstrapping requests and can act in a more responsive manner. We accomplish this by using the Netty API, which is an asynchronous event-driven network application API that is commonly used for development of high performance network protocols [113].

4.4 Experimental Results and Analysis

In this section we illustrate the results of our experiments. We firstly present the latency, throughput and scalability results we measured on our experimental test-bed comprising of three cloud platforms, and then analyse them in comparison with alternate approaches.

4.4.1 Service Latency

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 either the same hypervisor or the same data center.

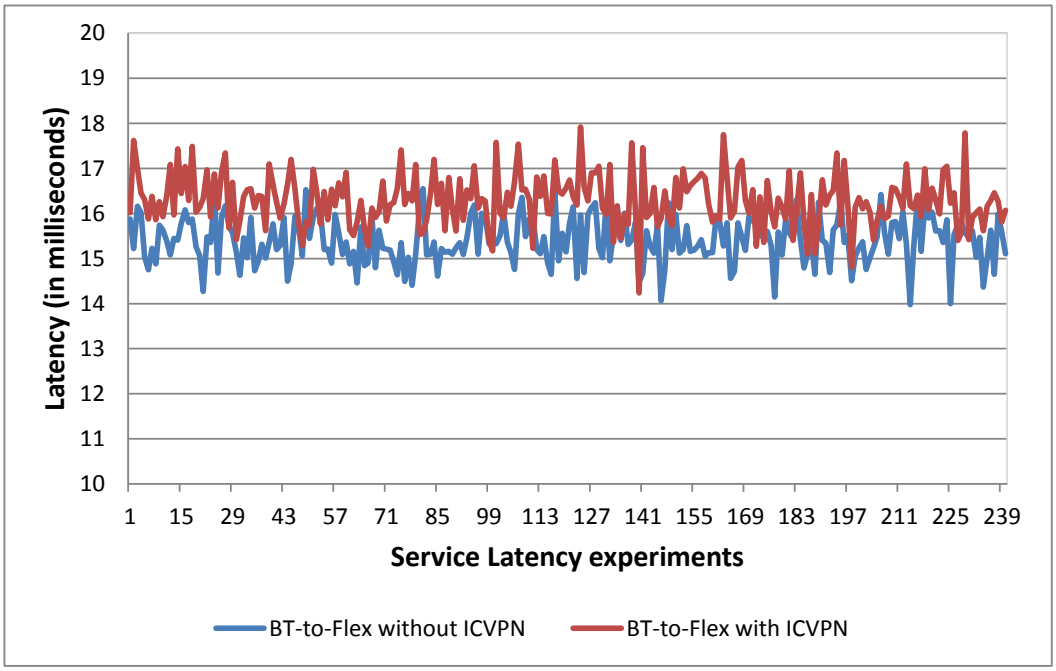


Figure 4.8: Service latency of 240 HTTP HEAD round-trip time request-response messages *from* BT *to* Flexiant clouds

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. The main benefit of using the HTTP HEAD request is that the HTTP HEAD reply message does not contain a message-body, thus reducing its size to a minimum.

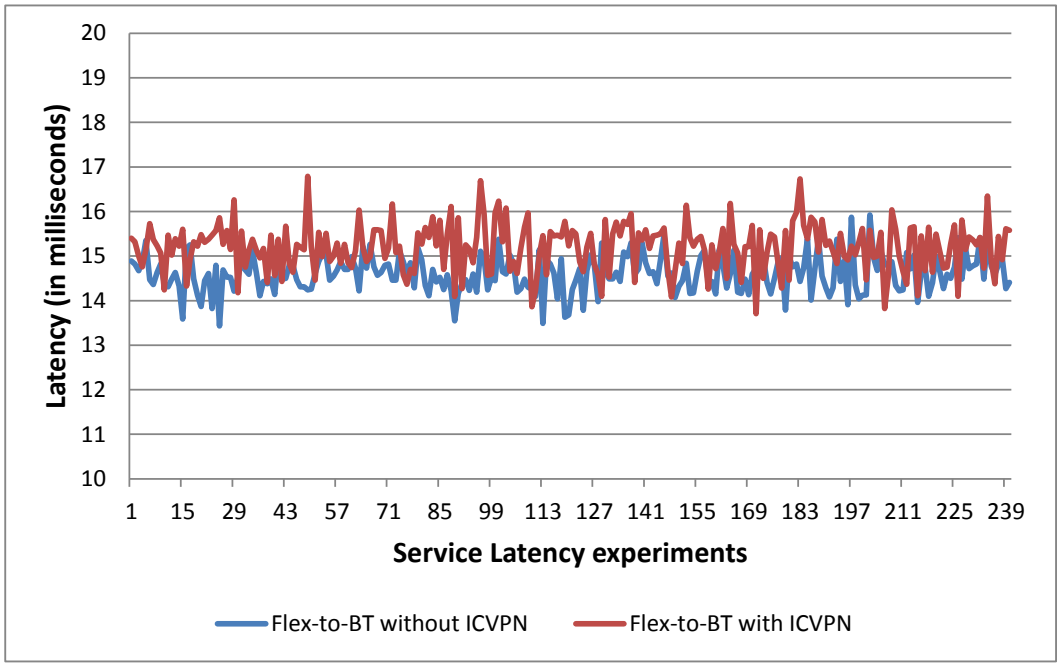


Figure 4.9: Service latency of 240 HTTP HEAD round-trip time request-response messages *from* Flexiant *to* BT clouds

We computed the average latency by running 10 experiments very hour for a period of 24 hours, firstly without using the Inter-Cloud VPN solution and then with it. The results for the 240 experiments run between BT and Flexiant cloud platforms are shown in Fig. 4.8 and Fig. 4.9. As is clear from the graphs, we get consistent latency results independent of the direction of the experiments and the variation of latency within the experiments itself is also quite minimal. Catering

for the bi-directional latency, we get a mean overhead of *5.06%* when using the Inter-Cloud VPN solution.

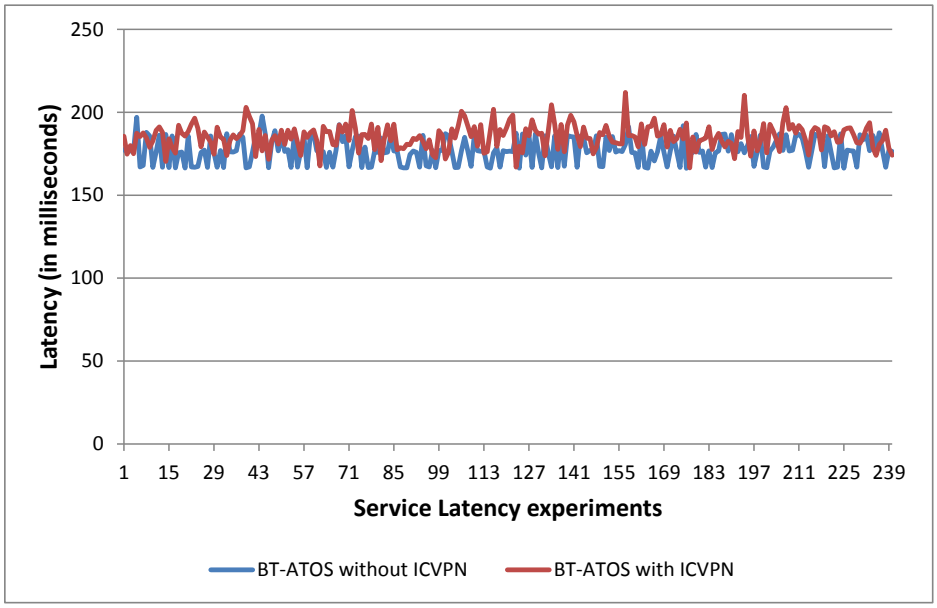


Figure 4.10: Service latency of 240 HTTP HEAD round-trip time request-response messages *from* BT *to* ATOS clouds

We repeated the same set of service latency experiments for BT and ATOS cloud platforms. The results for these 240 experiments run between BT and ATOS cloud platforms are shown in Fig. 4.10 and Fig. 4.11. In this case as well we get consistent latency results independent of the direction of the experiments and the variation of latency within the experiments itself is minimal too. Catering for the bi-directional latency in this case, we get a mean overhead of *5.35%* when using the Inter-Cloud VPN solution.

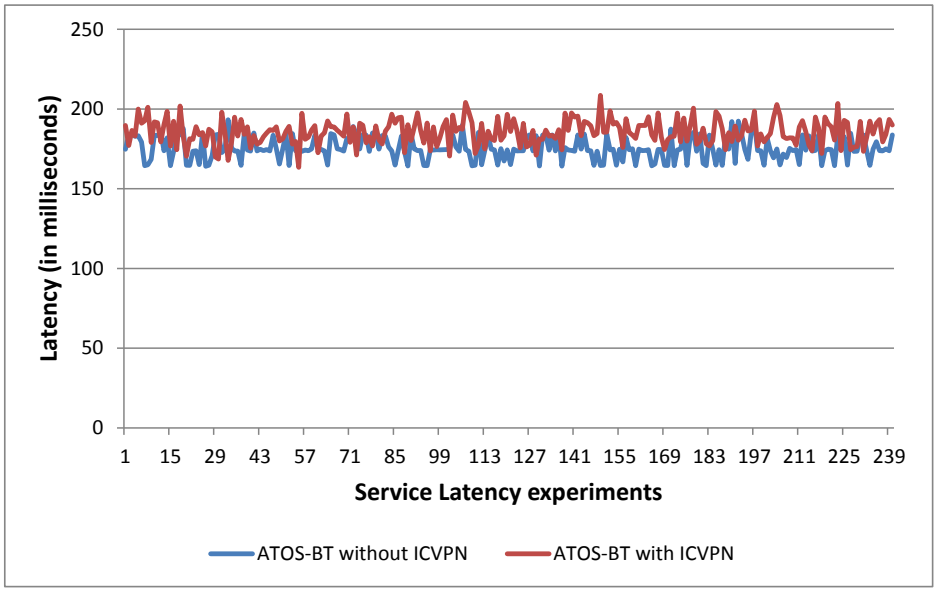


Figure 4.11: Service latency of 240 HTTP HEAD round-trip time request-response messages *from* ATOS *to* BT clouds

Looking at the results, we can see that using our solution only has a small impact on the HTTP latency, increasing it just by about *5%*. For further analysis we collect the network traffic dump when running our experiments, using the tcp-dump 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 protocol handshake phase is over, the latency performance is almost same in the comparative experiments.

4.4.2 Service Throughput

We measure the throughput between components of the service deployed on our three test-bed cloud platforms. We measured the throughput in both directions by transferring *30 MB* of data, a size chosen earlier in this section according to the results gathered from experiments conducted in Section 4.3.2.2. We computed the average throughput by running 10 experiments every hour for a period of 24 hours, firstly without using the Inter-Cloud VPN solution and then with it. The results are shown in Fig. 4.12 and Fig. 4.13.

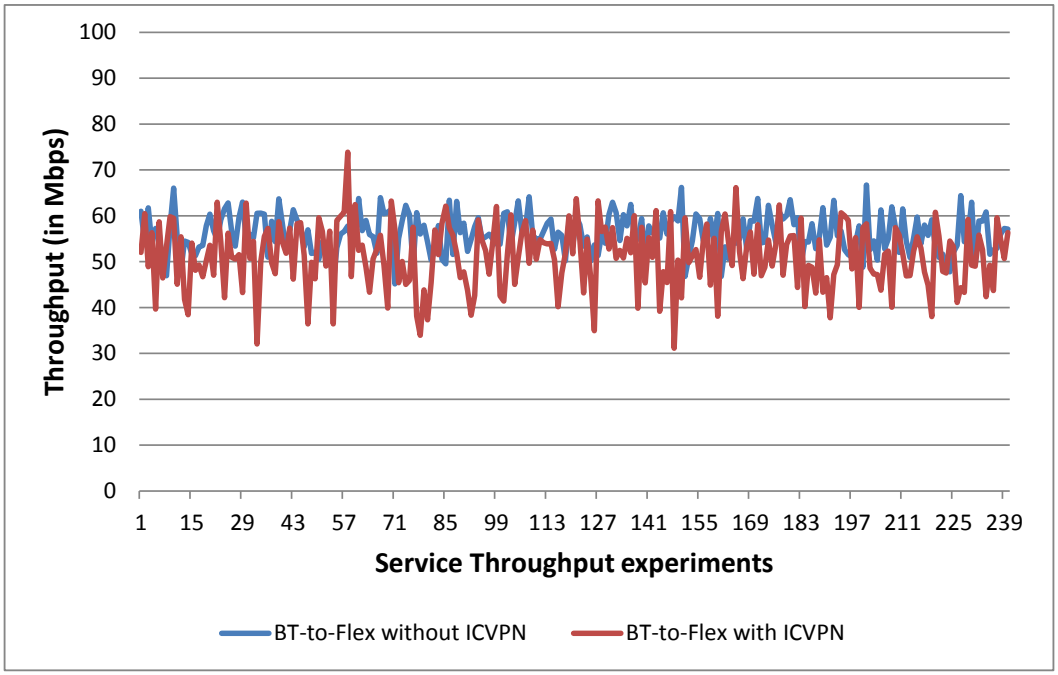


Figure 4.12: Throughput of 240 data transmission experiments *from* BT *to* Flexiant clouds

From the throughput results, the first thing that stands out is the difference in the throughput values depending on the direction of transferring the data. We have discussed this issue in some detail in Section 4.3.2.2 previously.

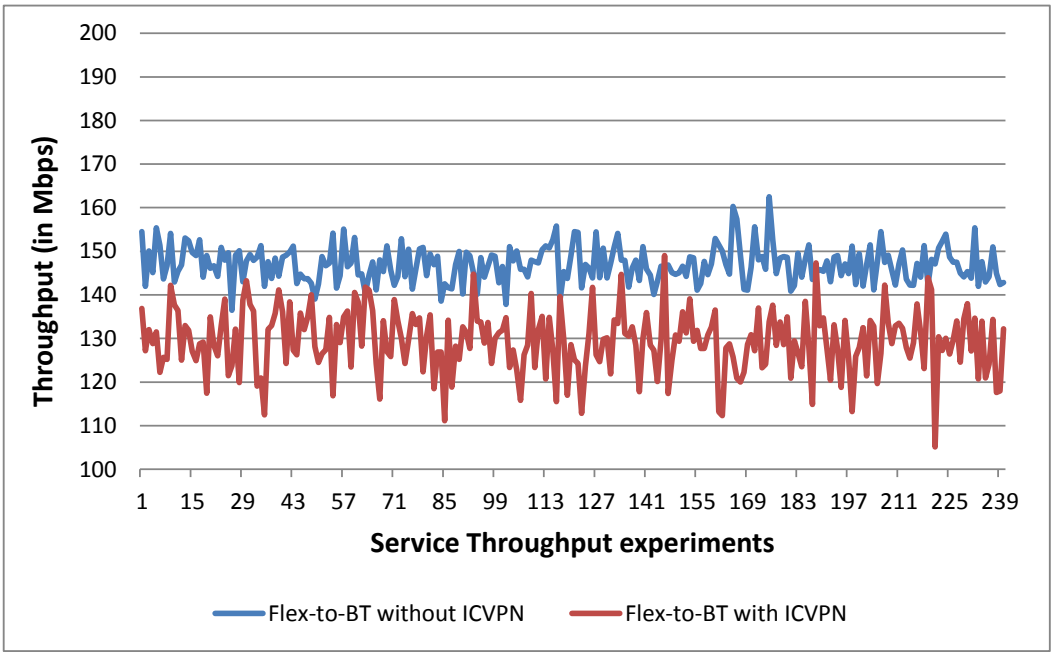


Figure 4.13: Throughput of 240 data transmission experiments *from* Flexiant *to* BT clouds

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.

4.4.3 Service Scalability

As we have mentioned above, one of the main motivation of using peer-to-peer overlays in our solution is their ability to scale as the number of virtual machines in the inter-cloud VPN service increases with the possible increase in the workload. Some cloud services and applications can easily expand to hundreds, even

thousands, of virtual machines across multiple clouds and it is important that our solution is able to cope with this sort of scalability.

Therefore, in order to measure the scalability of our solution, we observe the scale-up behaviour of the super peers of our universal overlay as more and more P2P clients request to enrol and join their respective VPN overlays. The metric that we use to measure the scalability is the number of bootstrapping requests that a super peer can service per second as more and more VPN peers try to join an overlay.

For this measurement, due to the limitation of resources and privileges in our test-bed cloud providers, instead of launching thousands of VMs to emulate a large number of peers trying to join an overlay, we launch only a few VMs containing the P2P client in each cloud provider but create more and more instances of the peer in the same VM to simulate a heavy workload. On the other hand, we increase the number of super peers handling the bootstrapping and observe how many requests they were able to process per second by looking into their log files. Again, due to the limitation of resources and privileges, we limit the number of virtual machines acting as the dedicated super peers to 4 on each of our two test-bed cloud providers used in this experiment, BT and ATOS.

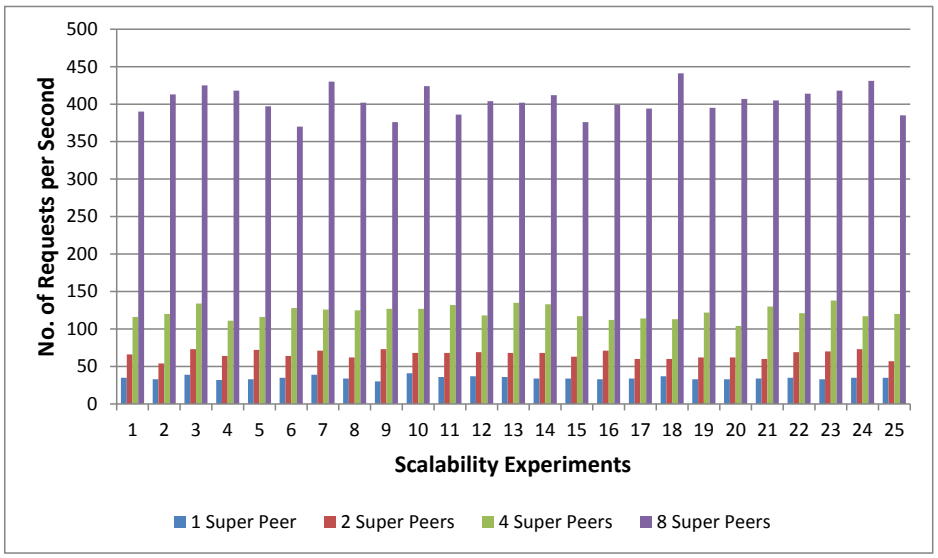


Figure 4.14: Experiments measuring bootstrapping requests processed per second against increasing number of Super Peers

Fig. 4.14 shows the results of experimental measurements of the number of bootstrapping requests processed by the Inter-Cloud VPN solution, by doubling the number of super peers in the universal overlay starting from 1 up to 8. To generate the bootstrapping load, we create 10 VMs on each cloud platform which contain the P2P client, and each VM then instantiates 25 instances of the P2P client simultaneously in order to generate the required work load for the super peers, using the GNU Parallel tool as the launch controller. The same tool is also used to send the list of available super peers to the launch scripts. We repeated this experiment periodically 25 times in order to cater for the adverse effects of jitter, and with 1, 2, 4, and 8 super peers. As it is apparent from Fig. 4.14, we did

not actually observe a lot of jitter and observed very low variation in the number of bootstrap requests per second for all quantities of the super peers.

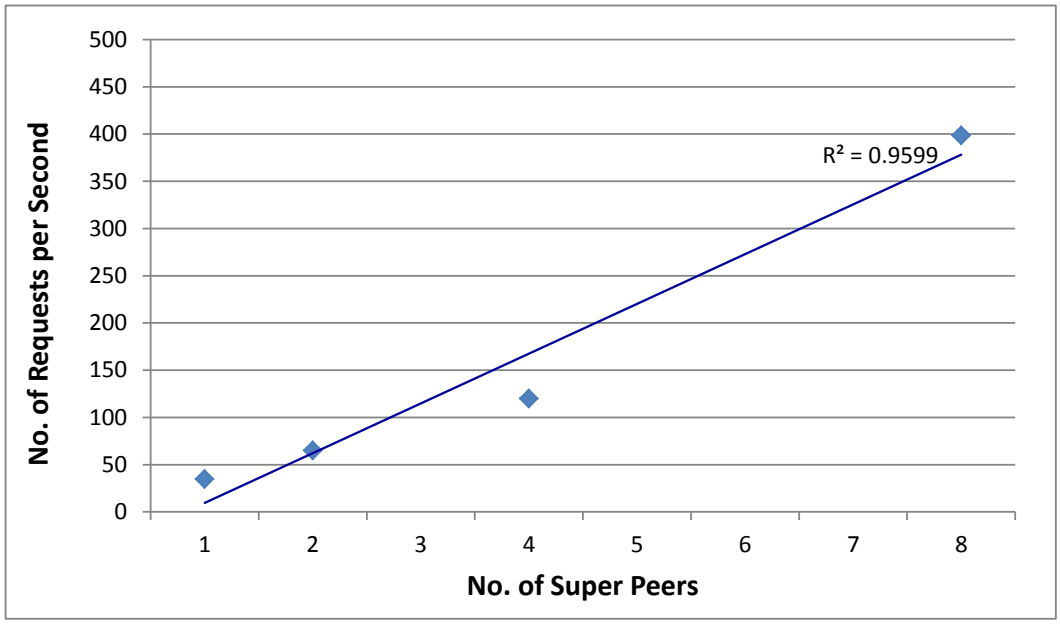


Figure 4.15: Trend of increasing the number of Super Peers on the average number of bootstrapping requests processed per second

As we can see in Fig 4.15, our solution was able to handle more bootstrapping requests per second on average, as we increased the number of super peers. As observed from the trend-line in Fig. 4.15, our solution is able to scale linearly to hundreds of requests per second against the number of super peers in the Universal overlay. We attribute this linear progression to the distributed design of our solution and this progression can be scaled to handle even thousands of requests per second by linearly increasing the number of super peers in the system.

4.5 Chapter Summary

In this chapter, we have given a detailed description of the design and implementation details about our Inter-Cloud VPN overlay solution. We have expanded on the techniques and mechanisms that we mentioned briefly in Chapter 3 and show how we modify and integrate them to come up with an efficient framework. This has been done by trying to model the architecture based on a realistic reference deployment scenario, where we have consider a cloud service that is deployed on multiple real-world cloud platforms.

Therefore, we started by detailing our novel scheme of employing a two-tiered overlay network, with the a single universal overlay focusing on management level functionalities and multiple, per-service VPN overlays focusing on the secure communication aspects. The universal overlay provides a scalable service to initiate and bind multiple VPN overlays to different cloud services, so the primary service that it provides is the integrated bootstrapping of the VPN peers of a cloud service. It accomplishes this by using the DHT, both as a distributed data store for data sharing, as well as a command & control channel. The peers of the universal overlay act as super-peers for the nodes of the underlying VPN overlays, so they can enrol, authenticate, bootstrap and join nodes to a particular VPN overlay based on the cloud service requesting the VPN service.

The VPN overlays provide the core security functionalities required to establish and maintain secure communication links between the nodes constituting a VPN overlay. It does that by utilising some of the protocols available in the IPsec protocol suite and the security meta-data, like the session keys, that are made available to it by the universal overlay. We detail the authenticated key sharing

protocol that we use to generate and share the session keys between the peers of the VPN overlay, as well as the design and architecture of our P2P VPN client that is embedded in the image of each participating VM.

Lastly, we gave the implementation details of the solution and how we evaluated its performance as deployed on two real-world commercial cloud platforms. We carried out a large number of experiments, over a long time interval, to evaluate the performance of our framework in terms of latency, scalability and throughput. From the results that we gathered, we were able to show that our devised solution incurred a minimum overhead of approximately 5 % in terms of latency, and an overhead of about 10 % in terms of throughput. With respect to scalability, we showed that the replication of super peers leads to a linear increase in the number of bootstrapping requests processed per second.

In the next chapter, we describe in detail how we solve the problem of secure admission of a peer of a VPN overlay with the help of a super peer in the universal overlay. We list and describe some of the traditional solutions of this problem and show how they are not really suitable for an inter-cloud model. A secure admission control process is really important as it is a requirement of the scalable key sharing and secure communication scheme detailed in the current chapter, as these schemes are effective only after the peers join the overlay as the result of a secure admission process.

Chapter 5

Inter-Cloud VPN Admission Control

5.1 Admission Control in Peer-to-Peer Systems

It is important to control the admission of peers in the Inter-Cloud VPN framework to safeguard and protect the resources, credentials, data and meta-data present in the DHT of the universal overlay from unauthorised and malicious users. Although the inter-cloud network has a complex and heterogeneous environment, we have to employ efficient and scalable security mechanisms that can protect our communication with minimal performance overhead. In this chapter we present the design and implementation of the Admission Control scheme that we have incorporated in the Inter-Cloud VPN solution.

5.1.1 Definition

In the scope of our work, we define admission control as the process following which a peer of a VPN overlay can securely join and enrol with a super peer in

the universal overlay. In Peer-to-Peer networks, bootstrapping a new peer is a well-known issue,that is, there is a need for the new peer to discover the required configurations and peers of the overlay to successfully join the network and access resources.

5.1.2 Bootstrapping using Server Lists

There are some traditional and commonly used solutions for this issue in peer-to-peer networks. One solution is to use a public server-based peer lists, where the address of the public server is either embedded in the P2P clients or is very well-known. There are some obvious security problems with this approach, for example, an attacker can compromise all the peer-to-peer overlays by compromising the server. Even without compromising the server, an attacker can just impersonate the server and feed the clients bad or malicious information that can be used to compromise the client itself. Lastly, it is difficult to inform and update the clients of any changes in the server's address.

5.1.3 Bootstrapping using Peer Caches

Another solution is to use client-based peer caches, that contain information of the last-known peers. This approach assumes a successful and secure initial peer-to-peer bootstrap process, which is an assumption that we don't make in our approach. Furthermore, the participants of a cloud server in a inter-cloud environment can join and leave quite rapidly, hence the cached information can become stale quite quickly. Another problem to consider is that whether you trust the cached information from a peer in absence of any formal authentication

process.

5.1.4 Bootstrapping using Random Probing

A yet another solution is to use random address probing to actively find peers. This is usually done with in the scope of a local area network as the common underlying technique is to use broadcast transmissions. Hence, this is obviously not suitable for the wide network scope of the inter-cloud environment. Furthermore, securing broadcast communication protocols usually carry a large performance penalty. The presence of a malicious impersonate will need to be considered in this situation as well, as in the peer cache based solution discussed above.

5.1.5 Bootstrapping using Multicast

In this bootstrapping technique, peers discover other peers in their domain by listening to a well-known global multicast address. This assumes that all participating peers will be reachable by at least one peer that has multicast connectivity. In the case where this special peer does not have multicast connectivity, it will try using directed broadcasts. If directed broadcasts are blocked (due to their common use in denial-of-service attacks), the peers will only be able to use previously cache peers until another special peer in the same multicast scope is introduced to the overlay. As an additional optimization, peers can solicit asynchronous announcement by using expanding ring searches, in which TTL-limited query requests are sent to the global multicast address and the TTL is increased gradually until there is a response.

However, regardless of the bootstrapping method being utilised, the require-

ment and importance of secure admission control is obvious as scalable key management and secure communication schemes are effective only after the peers join the overlay in a secure admission process. This is also useful to thwart the well-known vulnerability of P2P networks to Sybil attacks [56], where a peer or a collection of peers can claim or impersonate multiple identities in a peer-to-peer network.

5.2 Threat vectors affecting Inter-Cloud VPN Admission Control

While the concept and design of structured peer-to-peer overlay networks is highly robust and scalable, that very scale and flexibility of peer nodes can greatly increase the exposure of the overlay network to malicious peer nodes. For instance, if a malicious peer node is able to join a VPN overlay, it can place a malware on the distributed hash tables that can be later shared and run within the scope of virtual machines comprising of the affected VPN overlay. Thus, the attacker's ability to harvest sensitive information and even use the compromised virtual machines as desired (e.g., bot-nets, email account harvesting, denial of service etc.) will be significantly more than the compromise of a single host or an isolated virtual machine. Therefore, in addition of using encrypted tunnels and secure key and resource sharing schemes, we also have to address the security threats the admission control process as this process precedes all the other processes and protocols in our, and in fact all, communication frameworks.

To come up with an efficient, scalable and secure admission control mech-

anism for our Inter-Cloud VPN architecture, we consider the following common security vulnerabilities and attacks that should be addressed to ensure that no malicious nodes are able to join an overlay network. However, we do not aim to identify all possible threats and the corresponding solutions; instead, we start from an analysis of the common types of attacks discussed in the related literature and discuss them in the context of the admission control in our inter-cloud virtual private network.

5.2.1 Confidentiality Attacks

All most all of the current commercial and non-commercial cloud IaaS platforms support multi-tenant operations. This means that multiple users share the same computation, storage, and network resources but the cloud platforms can logically distinguish between different users, thus the users do not share or see each other's data, processing and network traffic. However, recent research has shown that it is possible to utilise cross-VM side-channel attacks to extract information from a target VM running on the same hardware as an attackers VM [134].

This introduces the possibility of VMs of different users sharing the same network resources and the possibility, in our case, for an unauthorised users to sniff the network traffic of VMs that want to join a VPN overlay. Although the IPsec protocol utilised by our solution protects against it, this kind of network sniffing can be timed before the peer has enrolled with the universal overlay and joined a VPN overlay, effectively before it has the time to use IPsec. Therefore, it will be possible for the attacker to sniff sensitive data, passwords and meta-data transmitted or received by a peer, before it gets admitted into a VPN overlay, compromising

the confidentiality of our solution.

Another possible attack vector is that an attacker can gain unauthorised access to a running VM itself by exploiting a known or even zero-day vulnerability in a software present on the VM or the operating system itself. So even if the VM is initially trusted at the provisioning stage, it may be compromised later by exploits that are discovered in it by an attacker. Therefore, it will be possible for the attacker to retrieve and use sensitive data, passwords and meta-data etc. from the VM storage if it is stored there unencrypted.

5.2.2 Integrity Attacks

In the same vein as the previous discussion, if an attacker is able to modify the data transmitted between the peers, it should not be able to compromise the admission control process. For example, if an attacker can modify the content of a peer's enrolment request during transit, the super peer should be able to detect the tempering and discard that request.

5.2.3 Authentication Attacks

It is possible for an attacker to intercept all traffic between a peer and a super peer and inject its own data instead. This can lead to a man-in-the-middle attack where the attacker can masquerade as a super peer to the peer and force it to join a compromised VPN overlay, or the attacker can masquerade as a peer and dupe the super peer into admitting it into a secure VPN overlay. Lastly, the attacker may also be able to read all the traffic sent between the communication entities. This kind of attack is especially easy in wireless environments, where the traffic can

be easily intercepted by anyone who is equipped with the right tools and is with in range of communication devices.

5.2.4 Availability Attacks

One of the salient benefits of using structured peer-to-peer model as the core of our communication framework is that the services it provides have higher robustness against failure. However, denial-of-service attacks are still possible against individual peer nodes within an overlay if the attacker possesses sufficient resources to carry our a sustained and persistent attack. For example, a network of malicious peer nodes that is controlled by the same attacker could simultaneously launch lookup queries for a particular key in the VPN overlay's distributed hash table. This can overload the peer node responsible for the key and even crash it. However, we can again mitigate these types threats with mass replication (of key-value pairs) strategies discussed earlier. Therefore, in our research effort we focus our energies in designing an admission control protocol that stops the malicious peer nodes from joining the overlay in the first place.

5.3 Security protocol for Inter-Cloud VPN Admission Control

As mentioned before in Chapter 4, we utilise a VM contextualisation service [13] to customise and provision VM images on different cloud platforms. To address the security threats described earlier, we modify the service to embed our peer-to-peer client and some bootstrapping information in each VM launched by the

service on the available cloud platforms. Because of this start-up routine, we can begin from a reasonable assumption that the cloud service or application owner that wants to establish an Inter-Cloud VPN between its VMs, and the super peers of the universal overlay, both have already agreed to a secret value or a password and that secret information is part of the bootstrapping information embedded by the VM contextualisation service in each VM. So when a VM created by this process is first started on a cloud platform, it is free of any malware and it has a trusted and secure P2P client provisioned on it. Therefore, a fresh virtual machine containing our P2P client has all the pre-requisites present that are required for it to undergo an overlay admission control process.

5.3.1 The Admission Control Protocol

Our admission control scheme is used when the peer bootstraps for the first time. It is motivated by the concept of Zero-Knowledge Password Proof (ZKPP), which is a generic and interactive method for one party (the prover) to prove to another party (the verifier) that it knows the value of a secret password, without revealing anything else to the verifier [21]. The use of this concept is very suitable for our security protocol as it allows a super peer to authenticate itself to a bootstrap peer without exchanging the password.

5.3.1.1 Using the Embedded Secret

The concept of a basic Zero-Knowledge Password Proof protocol can be explained in form of a simple interaction between two peers, Alice (A) and Bob (B). We assume that Alice and Bob share a secret *S*. Now, to share a secure

session key among themselves, Alice creates a secure random key *K*, and sends it to Bob encrypting it with S, i.e., $M = ENC_S(K)$. As Bob already knows the secret *S*, it can decrypt this message from Alice and retrieve the session key *K*, i.e., $K = DEC_S(M)$. Now Alice and Bob can use the secure key *K* for establishing a secure communication channel between them.

As we can see from the above interaction, the secret *S* itself is never transmitted on the wire between Alice and Bob, therefore, there is no direct way for an attacker to sniff it out from the network traffic. However, it is possible for a malicious eavesdropper to save the message *M* sent by Alice and run a dictionary or brute force attack against the secret *S* on it at his leisure. This type of exchange is susceptible to other kinds of offline and replay attacks as well. Therefore, we use an enhanced Zero-Knowledge Password Proof protocol based on [151], in which a Man-in-the-Middle attack cannot be used to obtain enough information to be able to guess a secret password by brute force, without further interactions with the peers for each guess.

5.3.1.2 Securing the Embedded Secret

If we consider the possibility that an attacker has somehow stolen sensitive data about a VPN peer from sources external to the overlay, like the external password database of the user or client of the service, then he will also be able to gain admission into the VPN overlay. To cater for this possibility, we modify our admission control protocol such that instead of using the stored secret password, we use a password-based key derivation function like *scrypt* [127] to generate a new password P from the stored secret password p and a cryptographic salt s,

and securely erase the stored password p from the VM. The same key derivation function is used on the super peers to generate the verifier v, which is then stored in the DHT of the universal overlay.

The inclusion of the key derivation function in our admission control protocol makes it resistant to a wide range of integrity attacks, from dumb/brute force attacks to rainbow table attacks. Furthermore, as it this function is designed to be computationally intensive, thus we can substantially limit the amount of parallelism that an attacker can use in a sustained attack.

5.3.1.3 The Complete Protocol

In our admission control scheme, the super peer stores the password in the form of a three element tuple $(ServiceID,\ Password,\ salt)$. The $ServiceID$ is the identifying common attribute of the VMs that together constitute the single cloud service that is being deployed on multiple cloud platforms. It is represented by a version 4 Universally Unique IDentifier (UUID) [102], and is 128-bits in length.

The salt s is generated as a random number. The private key is generated by using the SHA-2 hash function in the following manner:-

$$x = H(s \parallel H(\ ServiceID \parallel P))$$

The password verifier is generated as,

$$v = g^x \bmod n$$

where g is a generator of the multiplicative group and n is a safe prime. The

Table 5.1: Notations for the Inter-Cloud VPN Admission Control protocol

Symbol	Explanation
n	A large prime number. All computations are performed modulo *n*
g	A primitive root modulo *n* (often called a generator)
s	A random octet string used as the salt
p	The peer's password
P	The peer's password strengthened using a key derivation function
x	A private key derived from the password and salt
v	The peer's password verifier
u	Random scrambling parameter, publicly revealed
$H()$	One-way hash function e.g. SHA-1, SHA-2 etc.
$m \parallel n$	The two quantities *m* and *n* concatenated
K	Session key

description of other mathematical notations is given in Table 5.1.

The authentication process for admission into the overlay is initiated by the peer when it is started in its VM. Upon contacting the super peer, the peer receives the salt stored on the super peer, indexed under its Service ID. After its reception, the peer can calculate x as its Service ID and password P is already embedded in the VM. Now the peer generates a random number a, uses it to calculate A and sends the result to the super peer.

$$A = g^a \pmod{n}$$

The super peer does a similar operation to calculate B and also adds the public

verifier to it, before sending B and a random scrambling parameter u to the peer.

$$B = (v + g^b) \pmod{n}$$

Both sides can now construct the shared session key. The peer constructs it as:

$$\begin{aligned} S_A &= (B - g^x)^{a+ux} \pmod{n} \\ &= (v + g^b - g^x)^{a+ux} \pmod{n} \\ &= (g^x + g^b - g^x)^{a+ux} \pmod{n} \\ &= (g^b)^{a+ux} \\ K &= H(S_A) \end{aligned} \tag{5.1}$$

The super peer constructs it as:

$$\begin{aligned} S_B &= (A.v^u)^b \pmod{n} \\ &= (g^a.g^{ux})^b \pmod{n} \\ &= (g^{a+ux})^b \pmod{n} \\ K &= H(S_B) \end{aligned} \tag{5.2}$$

Both sides now possess the same and secure shared session key K based on the respective formulae. To complete the authentication, now they need to prove to each other that their keys are identical. In order to do so, the peer constructs the message M_A and sends it to the super peer,

$$M_A = H(H(g) \oplus H(n) \parallel H(ServiceID) \parallel s \parallel A \parallel B \parallel K)$$

The super peer will calculate M_A using its own K and compare it against the message received from the peer. If it does not match, the authentication fails. If it does match, the super peer issues its own proof to the requesting peer by sending it M_B.

$$M_B = H(A \| M_A \| K)$$

The peer will compute the expected response using its own *K* to verify the authenticity of the server. If it is a match, both parties are now authenticated. The summarized protocol is given in Table 5.2.

Table 5.2: The Admission Control protocol

Peer		***Super Peer***
	$\rightarrow$	$(lookup\, s, v)$
$x = H(s, P)$	$\leftarrow s$	
$A = g^a$	$A \rightarrow$	
	$\leftarrow B, u$	$B = v + g^b$
$S = (B - g^x)^{(a+ux)}$		$S = (A.v^u)^b$
$K = H(S)$		$K = H(S)$
$M_A = H(A, B, K)$	$M_A \rightarrow$	$(verify\, M_A)$
$(verify\, M_B)$	$\leftarrow M_B$	$M_B = H(A, M_A, K)$

5.3.2 Protocol Security Analysis

In this section we try to analyse our admission control protocol in terms of its strength and resistance against the confidentiality, integrity and authentication attacks described earlier in this chapter.

5.3.2.1 Mitigating Confidentiality Attacks

In the context of confidentiality attacks, the parameters that a passive attacker sniffing the network traffic between the peer and super peer node is able to ascertain are s, A, B and u. However, the knowledge of these parameters is not enough to formulate the value of S, as that requires the attacker to know the values of a, b and the verifier component v.

Furthermore, in case of an active attacker that is able to modify the parameters being exchanged between the peer and super peer nodes, the modification will result in both parties computing different values of S, and subsequently different values of the secret key K. This will cause the $(verify\, M_A)$ and $(verify\, M_B)$ checks to fail on the super peer and the peer node respectively and the modification attempt will be detected.

5.3.2.2 Mitigating Integrity Attacks

In the context of integrity attacks, we are able to get strong integrity in our scheme even when using weak or short passwords p, as the attacker can only attempt *one* guess per run of the protocol, making this scheme very resistant to dictionary and brute-force attacks. Furthermore, the super peer does not need to store the password, thus even an attacker who has stolen the super peer data cannot masquerade as the VPN peer unless he first does a brute force search for the password. Additionally, on the VPN peer side, as mentioned above, the secret is never stored in plain format, rather it is stored in encrypted form using a strong key derivation function that is resistant to all known integrity attacks.

5.3.2.3 Mitigating Authentication Attacks

In the context of authentication attacks, our scheme enables to ensure the proof of identity of the super peer node due to the use of the verifier v, as only the valid super peer knows the correct value of v and it is never transmitted on the wire during the run of the admission control protocol. Thus, a malicious attacker trying to impersonate the super peer will have to use an incorrect value of v and will force the peer node to formulate a different secret key K.

Furthermore, after a peer is authenticated and joins the overlay, its session key is kept in a secure cache and is valid for a brief time period. Our scheme can utilize previous session keys to generate new session keys to take advantage of key-continuity and avoiding overloading the authentication system. As the service or applications owner's password is deleted from the virtual machine as soon as it has been used as an input in the password-based key derivation function, an attacker won't be able to compromise other VMs of the same user even if he has been able to gain access to one of the VMs that has been provisioned with the Peer-to-Peer client. Another key feature of our scheme is that although it uses some elements of asymmetric encryption, it does not need a trusted third party. Thus we are able to avoid the overhead of a purely PKI-based scheme.

5.3.2.4 Mitigating Availability Attacks

In the context of availability attacks, an attacker may be able to target a super peer with a denial-of-service attack (DoS), rendering it unable to receive or process the admission control mechanism. These types of attacks are very hard to defend against, especially for the distributed DoS attacks (dDoS). Our scheme does

not specifically defend against these type of attacks, however, most commonly used approaches in order to mitigate their affects are robust connection/session management mechanisms, so that each connection/session consumes minimum resources, and massive replication of services, so that the attack surface is increased substantially for the attacker.

Therefore, our use of structured peer-to-peer overlays as the foundation of our communication framework is beneficial in this regard as they have in-build mechanisms for replication of resources like content and key-value pairs of the distributed hash table.

5.4 Prototype Implementation

We have implemented a working prototype of our Inter-Cloud VPN Admission Control scheme and integrated it with the overall Inter-Cloud VPN architecture. In this way, the admission control scheme is available as part of the Peer-to-Peer ICVPN client, discussed in detail in the previous chapter. Therefore, both the super peer and peer nodes contain the same implementation of our admission control scheme, the only difference being that the super peer also contains the mapping of the cryptographic salt s to the verifier v in its distributed hash table (DHT of the universal overlay).

This scheme has been implemented using the Java programming language [59] that can be deployed on Linux-based operating systems [153]. We were motivated by the following reasons to use Java and Linux as the core technologies for the prototype development, after suffering from some initial problems with other technologies and platforms:-

- Java has a large number of relatively easy-to-use cryptographic and peer-to-peer protocols and libraries.
- Java has excellent documentation and easily available online community-based help.
- As it is an open source operating system, we did not have to worry about licenses.
- Linux offers built-in support for creation and management of private keys, public keys and other cryptographic parameters in form of OpenSSL crypto library.

Other than the core components of our communication framework, the implementation of our core research contributions (mechanisms and protocols) was also done using open source libraries and APIs. Specifically, we chose the *BouncyCastle* library [124] to implement the majority of the cryptographic operations needed for our zero-knowledge password proof based admission control and Inter-Cloud VPN admission control schemes, as well as the PKI-based alternative that was required for the comparisons. This was due to the fact that *BouncyCastle* is one of the most light-weight and extensive cryptographic libraries that is designed with very strong emphasis on standards compliance and adaptability. In the same vein, we used the *TomP2P* library [28] for its implementation of the *Kademlia* [114] structured peer-to-peer protocol and the overlay DHT. Additionally, we used the commercially available *BT Compute Cloud* platform [32], *Flexiant FlexiScale* cloud platform [66], and a Xen hyper-visor based cloud platform [17] from *ATOS Origin* as our experimental test-bed.

5.5 Experimental Evaluation

In this section we present the results of a series of experiments we conducted to evaluate the efficiency of our Inter-Cloud VPN admission control protocol, when used in a service deployed on three different cloud IaaS providers. We evaluate this efficiency by comparing two variants of our protocol against the standard PKI-based protocol [22] that is second most common method used to authenticate and grant admissions in secure peer-to-peer environments. We have intentionally not compared our scheme against the most common method used for authentication and admission control i.e., password or shared secret based admission control. This is due to the fact that even though it is a low-cost and efficient method, the weaknesses and limitations of systems based on this method are numerous and well known [128].

5.5.1 Methodology

To evaluate the performance of the Inter-Cloud VPN admission control protocol, we measure the time taken by a peer to gain admission to the universal overlay. We have chosen this particular metric as the basis of our evaluation as it encompasses the whole life-cycle of the admission control process in a peer-to-peer environment. We have conducted this evaluation by launching the universal overlay on BT, Flexiant and ATOS cloud platforms, therefore the super peer nodes supporting the evaluated admission control protocols were running on all the three test-beds. Thereafter, we started the VPN peer clients on virtual machines on each cloud platform as well and measured the time taken by them to successfully join target super peers on different cloud platforms. These VPN peers also

supported the evaluated admission control protocols. Repeating the experiments measuring the joining times from different sources to different destinations gives us a fair idea of the performance overheads of our admission control scheme, as all other attributes remain same during the evaluations.

In order to conduct comparative evaluation of our scheme, we conduct the experiments using three different methods of admission control:-

1. We evaluate the PKI-based method, where the peer uses its RSA private key to sign its admission request and sends it to a super peer in the universal overlay. In this scenario, each peer also possesses a principal name and its own certificate, and also the certificate of the Certificate Authority. The super peer plays the role of the Certificate Authority, as discussed earlier, and is thus able to authenticate and admit a valid peer, as it has the record of the peer's public key when it had issued that particular peer its certificate.

2. We implement and evaluate the variant of our admission control protocol in which we do not secure the embedded secret. In this scenario we make use of a simple password that has been embedded in the peer VM by the VM contextualisation service at the time of the virtual machine provisioning. The reason for not securing the embedded secret, as discussed above, is that it relies on passing it through a password derivation function which is a time-consuming process and therefore incurs performance penalties. Although it is an interactive method, we have implemented the protocol in such a way that all interaction between a peer and the super peer is automated.

3. We implement and evaluate an improvement upon our previous admission control scheme (ZKPP Admission Control) by strengthening the simple

password used in it by using a strong key derivation function (scrypt) and securely erasing the original password from the virtual machine. This safeguards the system in the scenario where the password database of the peer virtual machine might be stolen some time after it has been provisioned on a cloud platform. Therefore, this method increases the security of the protocol but we are interested in finding out the performance cost associated with it.

5.5.2 Experimental Results

We implemented the three admission control schemes, described above, in our cloud platform test-bed. We evaluate the performance of these designs from a peer's perspective, that wants to gain admission in to its peer-to-peer overlay. We do this by measuring the runtime cost of the admission control phase between the peer and a super peer, as observed by the peer. We define the runtime cost of the admission control process as the time duration between the sending of the admission control request by the peer and the reception of a successful enrolment response from the super peer. This time is logged by the peer in a log file created by the P2P client on the host VM.

Fig. 5.1 shows the time duration results of admission control process using the three methods, for 100 separate trials conducted over a 24-hour period. In this case, only the BT cloud platform was used as the experimental test-bed. Only a single cloud platform was used so that we can observe the effects of the admission control protocol with being affected by network latency issues of an inter-cloud environment. Network latency does exist in a single cloud platform,

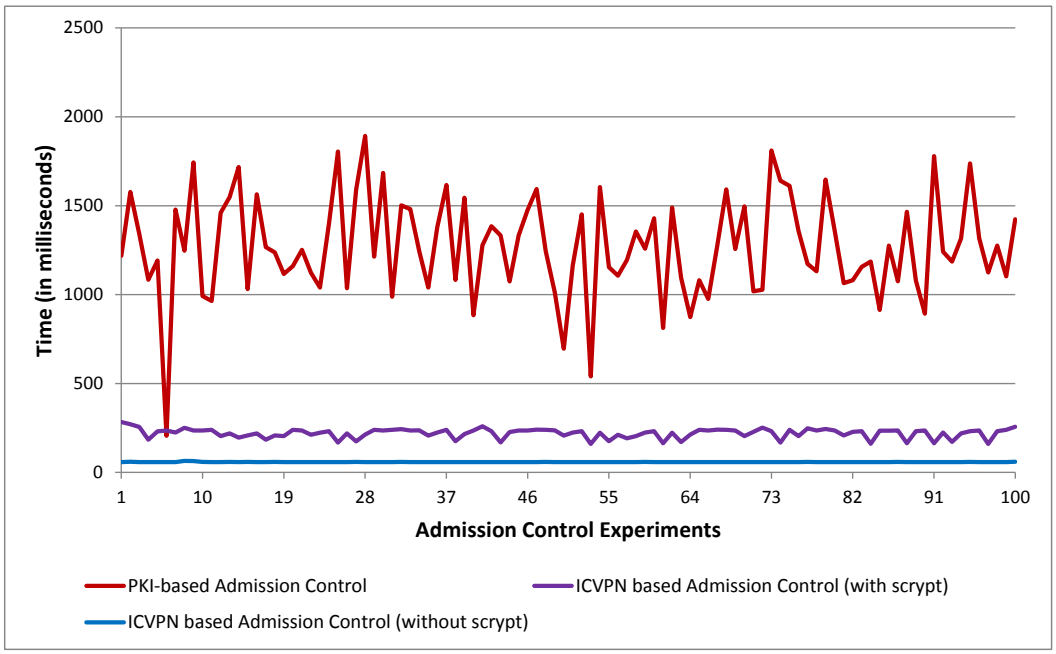


Figure 5.1: Duration of the admission control process for 100 trial instances using the PKI-based and ICVPN methods, on a single cloud platform (BT)

but it is typically very less (around *2.5* ms) and with negligible jitter.

We modified the P2P VPN client so that the same peer is able to make admission control requests using the three different authentication mechanisms. Similarly, the super peer is able to handle all three types of requests by launching a new thread for the processing of each request. We can see from the graph in Fig. 5.1 that, on average, the duration of the admission control process using the PKI-based approach is much greater (*1270.41 milli-seconds*) than that for our Inter-Cloud VPN schemes, with or without using the *scrypt* key strengthening function (*220.65 milli-seconds* and *58.59 milli-seconds* respectively). Similarly, the variation in the time duration results in case of the former is much greater than that for the later scheme.

Fig. 5.2 shows the time duration results of admission control process using the

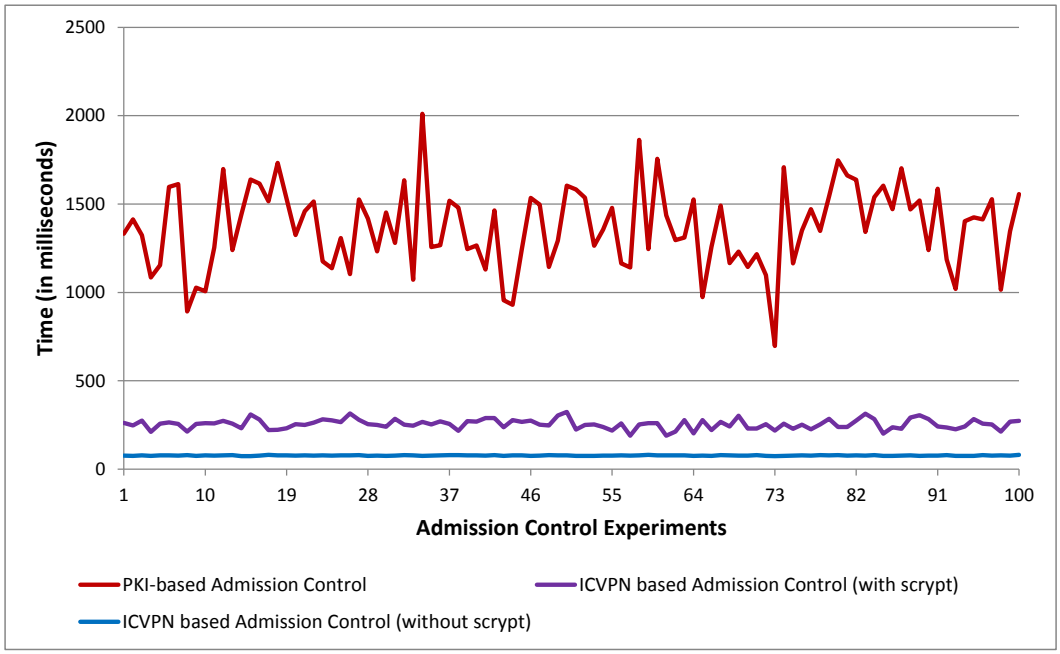


Figure 5.2: Duration of the admission control process for 100 trial instances using the PKI-based and ICVPN methods, between BT and Flexiant cloud platform

three methods, for 100 separate trials conducted over a 24-hour period. In this case, two different cloud platform were used as the experimental test-bed, i.e., the BT and Flexiant cloud platforms. This was done so that for this set of trial we can observe the effects network latency on the admission control process. The super peer VMs were running on the BT cloud platform and the peer nodes on the Flexiant cloud platform.

We can see from the graph in Fig. 5.2 that, similar to the single cloud platform trials, on average the duration of the admission control process using the PKI-based approach is much greater (*1368.18 milli-seconds*) than that for our Inter-Cloud VPN schemes, with or without using the *scrypt* key strengthening function (*254.83 milli-seconds* and *77.17 milli-seconds* respectively). Similarly, the variation in the time duration results in case of the former is much greater

than that for the later scheme.

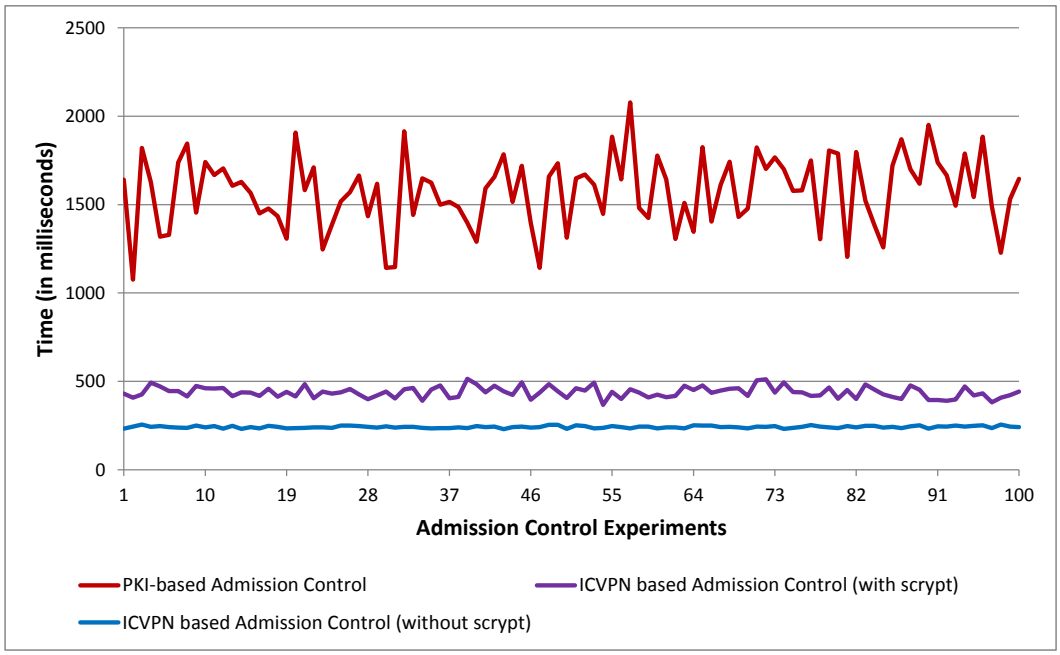


Figure 5.3: Duration of the admission control process for 100 trial instances using the PKI-based and ICVPN methods, between BT and ATOS cloud platform

Fig. 5.3 shows the time duration results of admission control process using the three methods, for 100 separate trials conducted over a 24-hour period. In this case too, two different cloud platform were used as the experimental test-bed, i.e., the BT and ATOS cloud platforms. This was done for the same reason as described in the case of the previous results. The super peer VMs were running on the BT cloud platform and the peer nodes on the ATOS test-bed.

We can see from the graph in Fig. 5.3 that on average the duration of the admission control process using the PKI-based approach is much greater (*1575.03 milli-seconds*) than that for our Inter-Cloud VPN schemes, with or without using the scrypt key strengthening function (*438.97 milli-seconds* and *241.98 milli-seconds* respectively). Similarly, the variation in the time duration results in case

of the former is much greater than that for the later scheme.

5.5.3 Results Analysis

Given the known stable network latency between the different cloud platforms available to us, we profiled the implementation of the PKI based design and its cryptographic libraries at the code level, in order shed some light on the reasons behind its large overhead in terms of both time and jitter.

Table 5.3: Average time taken by the admission control trials

Cloud	**Admission Control Mechanisms**		
Platforms	*PKI*	*ICVPN (with scrypt)*	ICVPN (without scrypt)
BT	1270.41 ms	220.65 ms	58.59 ms
BT - Flexiant	1368.18 ms	254.83 ms	77.17 ms
BT - ATOS	1575.03 ms	438.97 ms	241.98 ms

The main reason for the relative slowness of the PKI-based design is the required use of cryptographically secure random number generation in this method. This introduces the largest time and jitter penalty for our PKI-based measurements, as the peer has to wait for system events to gather enough entropy from entropy sources like disk reads, network activity, mouse movement, key presses, etc. to generate this kind of random number. In a typical cloud hosted virtual machine, these events are usually quite stable or uniform (as in case of disk and network activity), or totally absent (as in case of mouse and keyboard activity). Therefore, it takes a much longer time to generate the required random number of reasonable strength, and increasing the overhead of a PKI based solution as a result.

We also note that the second method (*ZKPP+scrypt*), although being more secure than the third method, adds quite a performance overhead on the admission control duration. Thus, although the Inter-Cloud VPN admission control method takes about *162* milli-seconds more than the less-secure ZKPP based method, it is still approximately *82%* more efficient than a PKI-based solution. The cumulative average of all the experiment trials are given in Table 5.3.

Furthermore, the time cost of the third admission control solution can be further reduced by adjusting the *CPU cost*, *memory cost* and *parallelisation cost* parameters of the scrypt algorithm, or even by using a simpler password strengthening algorithm other than scrypt e.g., bcrypt, PBKDF2 or even SHA-1.

5.6 Chapter Summary

In this chapter, we have given a description of the design and implementation details of our attempt to solve the problem of secure admission of a peer of a VPN overlay. We discussed some of the traditional solutions of this problem and then explained why they are not really suitable for an inter-cloud model, mainly due to the lack of authentication of the peer nodes trying to join an overlay network. So we try to come up with an efficient solution, that should be able to protect our communication with minimal performance overhead, as well as be integrate-able with the complex and heterogeneous environment of the inter-cloud.

We started by describing the threat vectors for Inter-Cloud VPN admission control process. We considered the attacks on confidentiality of the communication process, especially as unauthorised users might be able to sniff the network traffic of VMs that want to join a VPN overlay. So we have to make sure that its

not possible for the attacker to sniff sensitive data and meta-data transmitted or received by a peer, before it gets admitted into a VPN overlay. We also considered attacks on the integrity of the data communicated between the peers. So an attacker should not be able to modify the content of a peer's enrolment request during transit. We also considered attacks on the identity and availability of the peers and super peers.

We then give the details of our security protocol for the admission control process, which is used when the peer goes through the bootstraps process for the first time. We have based the core of the protocol on zero knowledge password proof, which is very suitable for our security model as it allows a peer to authenticate itself to a bootstrapping peer without exchanging the password. Furthermore, it is resistant to dictionary attacks and it does not need a trusted third party. We further modified this scheme such that instead of using the stored password in the zero-knowledge password proof, we use scrypt as the key derivation function.

We describe the implementation of this scheme in our system and measured the time taken by a peer to gain admission to the universal overlay for its evaluation. We used three different methods for authentication to do this comparison, which are, PKI-based method using RSA algorithm, the zero-knowledge password proof method, and the zero-knowledge password proof method strengthened with scrypt. We showed the time taken by the admission control process for 100 experimental instances using these three methods, with the zero-knowledge password proof method performing the best and the PKI-based method the worst.

In the next chapter, we describe in detail how we solve the problem of secure service based resource discovery in our framework. This is a well known issue in the peer-to-peer overlays and the data about a peer's resources has to be

obtained in such a manner that this information remains private and confidential between the valid peers of a VPN overlay. We describe our threat model for this problem in detail and then describe the security model that gives us the desired solution by using the functional cryptography based methods. At the end we show the performance evaluation of our solution again the traditional PKI model.

Chapter 6

Inter-Cloud VPN Secure Resource Discovery

6.1 Resource Discovery

Resource discovery is the process by which the users or client of a distributed system are able to search for resources required for their operations. Typically these resources are in form of services and devices etc. and are advertised or hosted in a networked environment by describing some of their attributes e.g., keywords, URL (Universal Resource Locator), URI (Universal Resource Identifier), and other forms of identifiers [86]. Usually specialised services, known as directory services, are used to store all the information related to the available resources, as well as implementing the functionality of resolving user queries [85]. However, most of the current resource discovery efforts tend to focus on expressive resource descriptions and extensive query predicates [3], [11], [14], and [91]. These efforts may differ in the way in which they name resources and how these

names are resolved to the target network location. However, in one way or another, all resource discovery mechanisms utilise attribute-based naming schemes and semi-structured resource descriptions [2].

As we have already discussed in the earlier chapters, most structured peer-to-peer networks use a distributed hash table (DHT) that handles resource placement and discovery, as well as ensuring a bounded number of hops for every search query. However, an important issue still facing resource discovery mechanisms, especially in distributed and peer-to-peer environments, is how to provide resource discovery techniques that allow their users to locate the resources of their interest securely, but still efficiently, especially in large-scale environments like the inter-cloud.

6.2 Service based Resource Discovery

In the Inter-Cloud VPN solution, after a peer in a VPN overlay has been successfully and secure bootstrapped using the scheme described above in Chapter 5, it needs to discover resources like VM ID, IP address and port numbers of the neighbouring peers in its overlay in order to construct IPsec tunnels between itself and the peers it wants to communicate with. These required resources have to be obtained in such a manner that this information remains private and confidential between the valid peers of a VPN overlay. Therefore, we are interested in devising a resource discovery scheme that is secure, but it also has to be scalable as we are operating in a decentralised environment.

As may be recalled from Chapter 3, Section 3.2, we make use of the *Kademlia* protocol [114] to provide us with the basic functionalities of a peer-to-peer overlay.

Two of its four basic operations can be used for the purposes of resource discovery in our solution, namely FIND_NODE and FIND_VALUE. Therefore, a peer can send FIND_NODE requests those super peers whose contact information has been embedded in the virtual machine by the VM contextualisation service. In the case of the absence of this contact information or to use more up-to-date information, it can also send the FIND_NODE request to the super peer it has already communicated with in the admission control process.

The advantage of the later method is that as every peer of a VPN overlay has to go through the admission control process, the super peers responsible for handling admission control can keep a running list of the peers currently present in a VPN overlay in the DHT of the universal overlay. However, both of these resource discovery methods have some security issues discussed below.

6.3 Threat vectors affecting Inter-Cloud Resource Discovery

Most of the directory services currently used for the purposes of resource discovery do not have security primitive built-in in their architecture. This is mostly due to the fact that these are usually intended to be used with in the local and internal scope of an organisation, which is assumed to be a trusted and secure environment. Whenever there are security requirements, these services fall back to the use of SSL/TLS protocols, or some variant of them, in order to fulfil those requirements [84], [51].

However, due to the inter-cloud scope of our working environment, we are

aware of the threats of malicious attackers trying to compromise the system using attacks like eavesdropping on network traffic, spoofing of communication end-points, tampering with in-flight packets (Man-in-the-Middle attack), and denial of service etc. Although we do use some encryption techniques for authentication and admission control purposes in our Inter-Cloud VPN framework, as discussed in detail in Chapter 5, encryption alone is insufficient to deal with all of these attacks to compromise the security of resource discovery.

6.3.1 Information Confidentiality

Eavesdropping attacks can be targeted at the network traffic between the virtual machines, either in a single cloud platform or between multiple cloud platforms. This can be done quite easily using tools called network sniffers or network analysers, e.g., tcpdump [90], Wireshark [125], network taps etc., which can be either software or hardware. These types of attacks can be thwarted by encrypting the traffic sent between the communicating components of a cloud service, hence the use of IPsec tunnels to encrypt the connections between the peers of a VPN overlay.

6.3.2 Traffic Tampering

This an active version of the eavesdropping attacks, where the attacker is placed between the communicating entities and can either modify the packets they are sending to each other or even impersonate them both. Furthermore, the attacker may even choose to not deliver the packets at all, or he may deliver the packets out-of-order. Lastly, the attacker may also be able to read all the traffic sent

between the communication entities. This kind of attack is especially easy in wireless environments, where the traffic can be easily intercepted by anyone who is equipped with the right tools and is with in range of communication devices.

To handle this class of attacks, we can again use protocols based on symmetric and asymmetric encryption techniques, that provide the features authenticated key exchange and mutual authentication. These protocols typically assume that the identities have already been established for all the participants, in other words the problem of spoofing has been handled, which might be difficult to achieve in a large network.

6.3.3 Denial of Service

In this type of attack, the attacker attempts to deny the users access to the resource discovery service. This can be done by either inundating the service with a large number of simultaneously requests, which will overload the system and cause it to crash. It can also be accomplished by exploiting a vulnerability in the service itself to compromise its operation. Or it can be done in the form of a Man-in-the-Middle attack as well where the attacker just does not forward the traffic between some or all of the communicating entities.

These types of attacks are very hard to defend against, especially for the distributed DoS attacks. Most commonly used approaches in order to mitigate their affects are robust connection/session management mechanisms, so that each connection/session consumes minimum resources, and massive replication of services, so that the attack surface is increase for the attacker.

6.3.4 Peer Spoofing

To join a peer-to-peer overlay, each peer needs to acquire a unique identification number called Peer Identification number ($PeerID$). In most structured P2P systems, this is done by the peer itself by choosing a random number from a large identity space. For instance, Kademlia protocol assigns a random 160-bit string to each peer. The content in the DHT also has a 160-bit string as its ID, which is known as *ContentID* or *infohash*. However, this approach is vulnerable to Sybil attacks [56], the basic idea of which is that an attacker can create and inject a large number of false peers in the overlay.

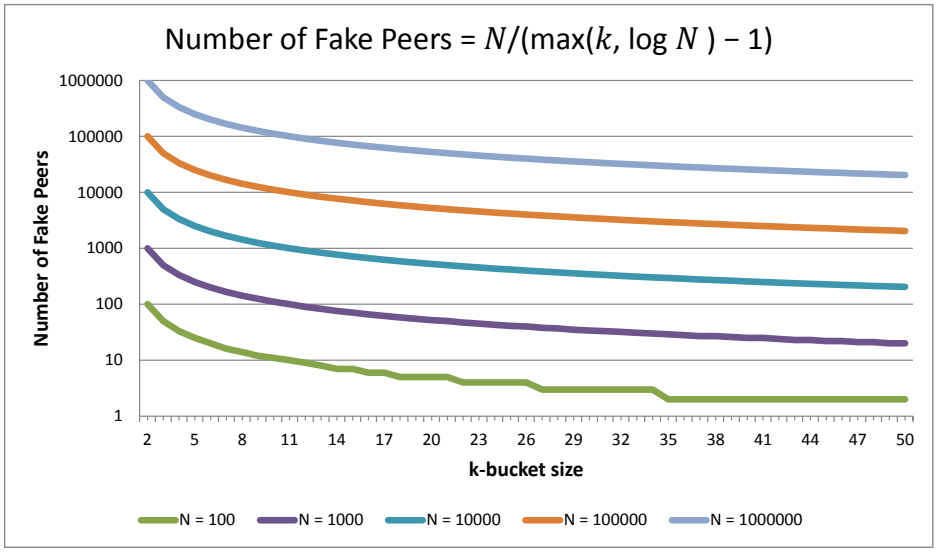


Figure 6.1: No. of fake peers required to intercept all inter-peer communication in a Kademlia overlay of size N

The attacker can assign or manipulate the false peers' $PeerID$ at will, and

thus subvert the functioning of the peer-to-peer overlay. Due to the *k*-bucket mechanism used in Kademlia, an attacker can effectively intercept peer messages if he has at least one false peer among every peer's *k* closest neighbours. This means that an attacker only has to inject $N/(k-1)$ false peers in an overlay. However, we know that the average number of hops in routing a message in a Kademlia overlay is $O(\log(N))$. As $k \leq 8$ in most default deployments of Kademlia, $O(\log(N))$ will become larger than k as the number of peers in an overlay increase. Therefore, a better calculation of the number of false peers required in an overlay is $N/(\max(k, \log(N))) - 1$.

6.4 Security protocol design for Inter-Cloud VPN Resource Discovery

A common way of dealing with this issue it 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 solution model, it can work by designating a stable super peer as the Certificate Authorities (CA) for a VPN overlay's peer nodes. The CA can assign peer IDs to the peers and signs a certificate that binds the *serviceID* of the cloud service or application 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.

6.4.1 Proposed Solution

We propose a decentralized solution that overcomes the above mentioned scalability problems by utilizing a functional encryption based scheme [49]. 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 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. 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 6.1.

Table 6.1: Four-tuple Functional Encryption

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

For Inter-Cloud VPN, 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 [158] as Identity-Based Signatures, and in [80] 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. We have discussed the basic background of these systems in Chapter 3. 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. There are two versions of the ABE, Key Policy ABE and Ciphertext-Policy ABE.

6.4.1.1 Key Policy Attribute based Encryption (KP-ABE)

In KP-ABE, attributes are assigned to a ciphertext when creating the ciphertext and policies are assigned to users/keys by an authority which created the keys. A key provides an access formula that operates over the set of attributes that must evaluate to true for decryption to yield the plaintext message. A key can decrypt only those ciphertexts whose attributes satisfy the policy.

6.4.1.2 Ciphertext-Policy Attribute based Encryption (CP-ABE)

In CP-ABE, the users of the system are assigned different attributes and each user is issued a key from an authority for its set of attributes. The ciphertext contains a policy (which is a Boolean predicate over the attribute space) and if the users attribute set satisfies the policy, they can use their key to decrypt the ciphertext. Another attractive feature of this scheme is that it is collusion resistant i.e. multiple users cannot pool their attributes together to decrypt a ciphertext.

6.4.1.3 Bilinear Pairing

Most of the Functional Encryption schemes are based on Pairing-based Cryptography (PBC) [98], which uses a pairing between elements of two cryptographic groups to a third group in order to formulate cryptographic systems. If the combining of elements of the two groups yields an element of the third group, that is linear in each of its arguments, then this pairing is called a Bilinear Pairing.

In groups constituting a bilinear mapping, for example the Weil pairing [159] or Tate pairing [150], generalizations of the computational DiffieHellman problem are considered to be impractical while the simpler decisional DiffieHellman problem can be solved using the pairing function. Bilinear pairings have been used to design many cryptographic systems for which no other practical implementation was known to exist, and these include the Functional Encryption schemes like Identity-based Encryption and the Attribute-based Encryption. The Bilinear Pairing is often formally defined as follows in most cryptography literature:-

Let r be a prime number.

Let G_1 and G_T be cyclic groups of prime order r.

Let G_2 be a group, which is not necessarily cyclic, where each element has order dividing r.

Let $P \in G_1$ and $Q \in G_2$ be the generators of G_1 and G_2 respectively.

A bilinear pairing e on (G_1, G_T) is a computable map,

$$e : G_1 \times G_2 \rightarrow G_T$$

for which the following is true:

1. Bi-linearity: $e(P^a, Q^b) = e(P, Q)^{ab} \quad \forall\ a, b \in \mathbb{Z}$

2. Non-Degeneracy: $e(P, Q) \neq 1$

3. Computability: e has to be efficiently computable

6.4.2 Secure Resource Discovery

Once a peer has joined its overlay network, it needs to discover its neighbours and the resources offered by them, to establish secure IPsec tunnels. After the establishment of the tunnels, the deployed cloud service or application will be able communicate securely with its different components. In order to achieve this secure resource discovery process, we have designed the following protocol scheme based on the Functional Encryption predicates discussed earlier in the section. A simplified step-wise description of the scheme is as follows:-

1. A super peer sets up its own Master Secret $\boldsymbol{ms}$ and Public Parameters $\boldsymbol{pp}$.

2. The super peer generates a private key for itself using the $\boldsymbol{ServiceID}$ and its own $\boldsymbol{PeerID}$ as the public key i.e. $\boldsymbol{Pub_{SP}}$ = $\boldsymbol{ServiceID} \wedge \boldsymbol{SuperPeerID}$, for each service the super peer is managing.

3. The VPN peer requests for $\boldsymbol{pp}$ on boot up from the super peer.

4. The VPN peer sends a Provisioning Request to super peer, encrypted using the super peer's public key ($\boldsymbol{Pub_{SP}}$)

5. The super peer issues a private key to the VPN peer encrypted by its own private key, against the public key $\boldsymbol{Pub_{VPN}}$ = $\boldsymbol{VMID} \wedge \boldsymbol{PeerID} \wedge \boldsymbol{ServiceID}$

6. The super peer inserts the VPN peer's public key in the universal overlay DHT to keep a record of issued private keys;

$$\boldsymbol{key(ServiceID) = value(List\ of\ VMID)}$$

7. For each peer i, the super peer adds its public key in its VPN overlay ;

$$\boldsymbol{key(VMID_i) = value(Pub_{VPN_i})}$$

8. The VPN peer requests list of other peers from super peer, which returns the result of $\boldsymbol{key(ServiceID)}$, encrypted using $\boldsymbol{Pub_{VPN} = PeerID \wedge ServiceID}$

This protocol and its security features are discussed in detail in the following sections:

6.4.2.1 System Setup

Most of the Functional Encryption based techniques like Identity-based Cryptography and Attribute-based Cryptography rely on a trusted third party that is tasked with the generation of private keys for the whole system. This trusted third party is typically called Private Key Generator (PKG). Before any encryption or decryption can take place, the PKG must also generate a *master private key* and a *master public key*. This is denoted by the $setup(1^\lambda)$ function in Table 6.1. In our protocol we denote these credentials as $\boldsymbol{ms}$ and $\boldsymbol{pp}$ respectively. The PKG is only involved in the initial phase of the process and it is not required to be continually involved in the normal operation of the Functional Encryption mechanisms after the keys have been generated.

In the Inter-Cloud VPN architecture, we can use the super peer nodes as a distributed PKG, as these nodes are also mainly used for the initial management and peer bootstrapping processes. In case of multiple super peers acting as the PKG, we can utilise standard techniques like threshold cryptography to distribute the PKG functionality among the participating super peers. This will ensure that $\boldsymbol{ms}$ is not available on a single super peer, thus enhancing security of the master private key as well as avoiding a single point-of-failure.

The master public key $\boldsymbol{pp}$ is put on the DHT storage of the Universal Overlay by the super peer as a publicly accessible parameter, indexed against the $ServiceID$ of the VPN overlay for which it is required. Thus each $\boldsymbol{ms}$ and $\boldsymbol{pp}$ pair is unique and bound to a single VPN overlay, and hence to the individual cloud service that is to be deployed on multiple cloud platforms. This is to ensure the security compartmentalisation of VPN overlays, and as a result the deployed cloud service as well, so that they do not share the same security parameters.

6.4.2.2 Key Generation

When a peer bootstraps and registers with the super peer, the super peer with the PKG functionality has to generate the peer's private key based on its three attributes. These three attributes that the super peer requires are its $VMID$, $PeerID$, and $ServiceID$. The $VMID$ of a peer is globally unique and is assigned to its host VM by the hyper-visor or the cloud platform that launched the VM. It is not generally possible the change its value from inside the VM as it is not stored in the VM, rather it resides only on the hyper-visor running the VM or in some cases the cloud platform can store it as well. It is also difficult to guess as it is

usually a long random number, e.g., in case of BT cloud platform it is a 128 bit number.

However, the peer can query the its hyper-visor or the cloud platform for its value and then send it to the super peer securely by encrypting it with the super peer's public key. A method for securely querying a hyper-visor or the cloud platform for a $VMID$ is usually provided by the vendor in form of an API call. The hyper-visors and the cloud platforms can also ascertain the source of this query and only supply the valid results if the query originates from the VM itself. The other attributes of $PeerID$ and $ServiceID$ have been already discussed in the last chapter. In order to construct the super peer's public key, the peer requires to know the attributes of pp, $SuperPeerID$, and $ServiceID$. The latter two of these are conveniently already embedded in the VM image itself, whereas the pp attributed can be obtained from the universal overlay DHT at run time, as it is indexed against a well-known key, i.e., $ServiceID$.

6.4.2.3 Key Distribution

In the traditional Functional Encryption mechanisms, the PKG will send the user this private key via a secure side channel. In the Inter-Cloud VPN architecture, however, we can take advantage of the universal overlay DHT and the private key of the super peer itself to streamline this key distribution process to the peers. In our design, the super peer node can generate a private key for itself using the same process as described above. In fact this step needs to take place before any key distribution occurs for the VPN peers in the secure resource discovery process. The attributes that it uses for this purpose are its own $PeerID$, which in

this case is called $SuperPeerID$, and the $ServiceID$ of the cloud service being deployed.

Now possessing its own private key, the super peer can sign the private key it generates for the requesting peers. As the $ServiceID$ is a required attribute in the super peer's public key, and its knowledge is private to the valid peers running in the VMs where this information was embedded, thus only the valid peers are able to construct the super peer's public key and able to decrypt and acquire their private keys from the PKG super peer.

6.4.2.4 Public Key Repository

Along with issuing the peers with their private keys, the super peer also builds a repository for the public keys, in the form of the conditional attributes against which a particular private key was generated. Each record in this repository is kept in the form of a key-value pair in the VPN overlay DHT. The key used for this purpose is the $VMID_i$ of the VM on which the peer is running, and the value against this key is the public key of the same peer Pub_{VPN_i}. Also, this value is encrypted by the super peer using its private key before placing it in the DHT, so that only a peer that is in possession of the super peer's valid public key is able to read it.

This feature offers the functionality, to any other peer in the VPN overlay, of uploading or sharing data through the DHT in such a way that only the intended recipient peer is able to read it. All it needs it order to achieve this goal is to know the $VMID_i$ of the intended recipient and it will be able to get its public key using the resource discovery lookup operation on the DHT. Encrypting the data to be

shared with this public key will ensure that only the intended recipient is able to read it, as only that peer's private key can be used to decrypt it.

6.4.2.5 Peer Address Resolution

However, in order to communicate directly with the recipient peer, the sending peer must have the knowledge of its IP address or the DNS name, as the $VMID$ is the virtual machine UUID that cannot be used itself as a logical network address. This problem is overcome again by using a simple API call provided by all hypervisors and cloud platforms vendors, which returns a virtual machines IP address or its DNS name if queried with its $VMID$. Furthermore, in cloud platforms where each virtual machines is assigned at least one public IP address by default (e.g., Flexiant), we can use this public IP address as the value of the $VMID$ parameter, instead of the virtual machine UUID. This will further streamline the process of the acquiring a neighbouring peer's public key as it will eliminate the need of making an API call to the cloud platform.

6.4.2.6 Neighbour Peer Discovery

To keep a record of peers that have successfully bootstrapped and have been issued their private keys, the super peer maintains a running list of these peers in the *universal* overlay's DHT. This list is composed of the concatenated $VMID$ parameters of all the peers currently active in the VPN overlay, and is indexed in the DHT against the $ServiceID$ parameter as its key. As this list is stored in the universal overlay, the VPN peers cannot access it directly, as the resource discovery mechanism is local to each overlay network. The super peer is also

make use of this list, as it helps it to monitor the population of the VPN peers. Therefore, if it sees a trend of increase in this population, it can start up more super peer's in the universal overlay in order to handle the increased work load, or vice versa.

Therefore, if a VPN peer wants to get the list of all peers in its VPN overlay, it has to query it from the super peer. On reception of this query, the super peer will send the list to the the requesting VPN peer, encrypted with the public key of that peer that it has on record. In this way, on the valid peer will be able to get the list of all its neighbour peers by decrypting it with its private key. As mentioned in the previous paragraph, if the cloud platform is assigning public IP addresses to its virtual machines, then this will be a list of public IP addresses of the all the peers currently active in the overlay. This will greatly simplify the process of discovering all the neighbouring peers in a VPN overlay, that is, a peer wishing to communicate with all the peers in its overlay will be able to acquire their IP addresses using just one DHT lookup operation.

6.5 Prototype Implementation

We have implemented a working prototype of our Inter-Cloud VPN Secure Resource Discovery (ICVPN SRD) protocol and integrated it with the overall Inter-Cloud VPN architecture. In this way, our Secure Resource Discovery scheme is available as part of the Peer-to-Peer ICVPN client, whose design and architecture has been discussed in detail chapter 4. Therefore, all the super peer and peer nodes contain the same implementation of our Secure Resource Discovery scheme.

This scheme has been implemented using the Java programming language [59] that can be deployed on Linux-based operating systems [153]. We were motivated by the following reasons to use Java and Linux as the core technologies for the prototype development, after suffering from some initial problems with other technologies and platforms:-

- Java has a large number of relatively easy-to-use cryptographic and peer-to-peer protocols and libraries.
- Java has excellent documentation and easily available online community-based help.
- As it is an open source operating system, we did not have to worry about licenses.
- Linux offers built-in support for creation and management of private keys, public keys and other cryptographic parameters in form of OpenSSL crypto library.

Other than the core components of our communication framework, the implementation of our core research contributions (mechanisms and protocols) was also done using open source libraries and APIs. Specifically, we chose the *BouncyCastle* library [124] to implement the majority of the cryptographic operations needed for the PKI-based alternative P2P resource discovery scheme that was required for the comparisons. This was due to the fact that *BouncyCastle* is one of the most light-weight and extensive cryptographic libraries that is designed with very strong emphasis on standards compliance and adaptability.

To implement the functional encryption schemes, especially the CP-ABE and IBE related functions, we used the *cpabe* [26] and *libfenc* crypto libraries. These are the only extensible libraries that offer implementations for Attribute-Based and Identity-based encryption schemes. An isue that we encountered at this step of the implementation was that these libraries are implemented in C programming language [96] whereas all of other development code was in Java. Therefore, we had to implement wrappers functions to call the native C code from the Java using Java Native Interface (JNI) [73]. Although this is not the most efficient and seamless approach, this was the only option available to use other than implementing the complete libraries in Java.

In the same vein, we used the *TomP2P* library [28] for its implementation of the *Kademlia* [114] structured peer-to-peer protocol and the overlay DHT. Additionally, we used the commercially available *BT Compute Cloud* platform [32], *Flexiant FlexiScale* cloud platform [66], and a Xen hyper-visor based cloud platform [17] from *ATOS Origin* as our experimental test-bed.

6.6 Evaluation Methodology

One of the standard metrics to be measured in order to analyse the performance of a resource discover mechanism is the latency of a resource lookup operation. Therefore, we also focus to design and conduct experiments that enable us to accurately measure the time taken by our secure resource discovery protocol to find and return a set of resources. In case of solutions based on peer-to-peer networks, such as ours, this includes the time required for each peer to process the lookup query, as well as the time required to establish connection to the next

peer to whom the lookup query is to be forwarded, in case the required data is not found in the current peer. In such environments, the time required for a lookup operation increases with the increase in the number of peers in the overlay network.

6.6.1 Cost of DHT Lookups

However, we can remove the cost of the DHT lookups from our measurements as their theoretical overhead is known to be $O\log(N)$ for Kademlia. This means that we will get a fixed value of latency for a particular value of N. But due to the nature of actual runtime experiments, including the measurement of this metric will add unhelpful noise to the results, as we know that the value of N can actually vary during the life-time of a peer-to-peer overlay. Furthermore, the replication and intelligent routing mechanisms implemented by the peer-to-peer protocol also affect the measurement of this metric.

6.6.2 PKI-based Design for Comparison

As mentioned earlier, one of the main overheads in peer-to-peer overlays is related to the performance of the resource discovery lookup operations 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 secure resource discovery system, like the one described in SDS [48].This alternative design uses the traditional hybrid of asymmetric and symmetric cryptography. However, we cannot directly compare our results with the ones provided by SDS due a num-

ber of architectural and operational differences. For example, SDS is based on the client-server architecture, it uses a global multicast address for the search purposes of service discovery and its experiments are run on physical hardware whose specifications are very different to the VMs used in our test-bed cloud platforms. Therefore, we take the basic design of a PKI based architecture and implement and evaluate it in our own test-bed environment.

In the PKI based design, a principal name and a X.509v3 certificate is associated with every peer, which can be used to prove the peer's identity. The certificates are supposed to be signed by a well-known Certificate Authority (CA), whose own public key is supposed to be known by every peer. In our implementation, we enhance a super peer with the functionality of a Certificate Authority. Each peer is issued a signed certificate upon authenticated completion of the bootstrapping process. The public certificates of the all the peers in the overlay are stored in the overlay DHT, indexed against their $ServiceID$, so that the peers can easily query for and receive any other peer's public certificate. A peer can now publish or share any data on its overlay's DHT by encrypting it with its private key. Similarly, a peer can query the overlay DHT with resource discovery requests and get the resulting data back which can then be decrypted by the data-owning peer's public key.

6.6.3 Experimental Results

We implemented both the Functional Encryption based secure resource discovery design and the PKI based secure resource discovery design on our cloud platform test-bed. We evaluate the performance of these two designs by measur-

ing the runtime cost of their resource discovery operations. We define the runtime cost for both designs as the time duration between the start and end of a secure resource discovery lookup operation.

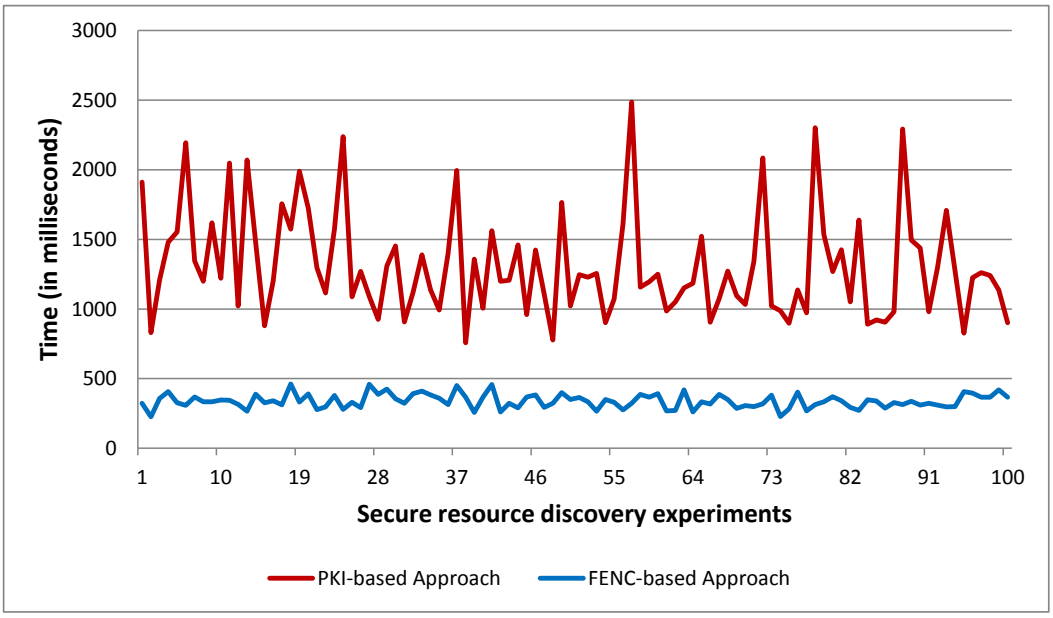


Figure 6.2: Secure resource discovery for 100 runtime trials between PKI and Functional Encryption based approaches in ICVPN, on a single cloud platform

Fig.6.2 shows the results of doing a secure resource discovery lookup operation for 100 separate trials, conducted over a 24-hour period. In this case, a single cloud platform was used as the experimental test-bed. We deployed four VMs on this test-bed, one pair installed with the super peer and peer using the PKI based resource discovery approach and the other pair installed with the super peer and peer using the Functional Encryption based resource discovery approach. We can see from the graph that, on average, the runtime of the secure resource discovery process using the PKI-based design is much greater (*1313.52 milli-seconds*) than that for our Functional Encryption based scheme

(*338.81* milli-seconds). Similarly, the variation in the time duration results in case of the former is much greater than that for the later scheme.

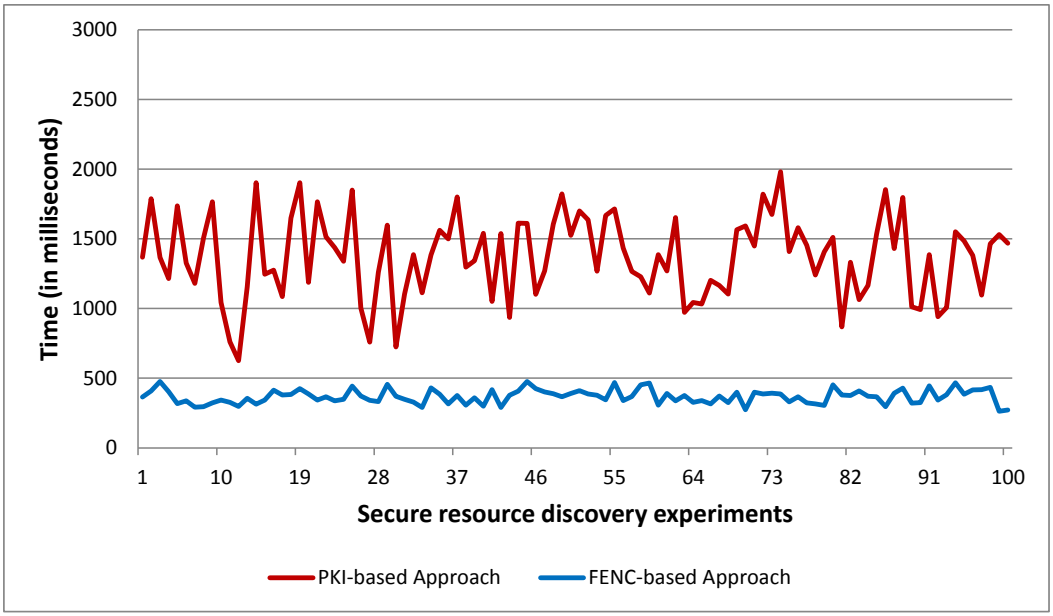


Figure 6.3: Secure resource discovery for 100 runtime trials between PKI and Functional Encryption based approaches in ICVPN, between BT and Flexiant cloud platform

Similarly, Fig.6.3 shows the results of doing a secure resource discovery lookup operation for 100 separate trials, conducted over a 24-hour period. In this case, two different cloud platform were used as the experimental test-bed, i.e., the BT and Flexiant cloud platforms. Again, we deployed four VMs on this test-bed, one pair installed with the super peer and peer using the PKI based resource discovery approach and the other pair installed with the super peer and peer using the Functional Encryption based resource discovery approach. The super peer VMs were hosted on the BT cloud platform and the peer nodes on the Flexiant test-bed.

We can see from the graph in Fig.6.3 that, similar to the single cloud platform trials, on average the runtime of the secure resource discovery process using the PKI-based design is much greater (*1373.47 milli-seconds*) than that for our Functional Encryption based scheme (*368.09* milli-seconds). Similarly, the variation in the time duration results in case of the former is still much greater than that for the later scheme.

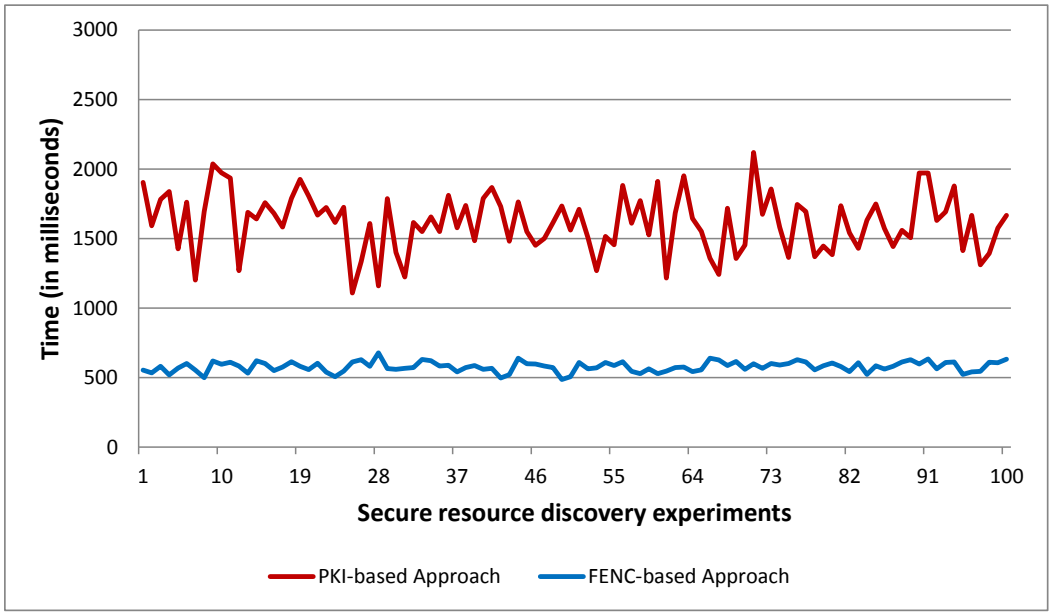


Figure 6.4: Secure resource discovery for 100 runtime trials between PKI and Functional Encryption based approaches in ICVPN, between BT and ATOS cloud platform

Lastly, Fig.6.4 shows the results of doing a secure resource discovery lookup operation for 100 separate trials, conducted over a 24-hour period. In this case as well, two different cloud platform were used as the experimental test-bed, i.e., the BT and ATOS cloud platforms. Here too we deployed four VMs on these test-beds, one pair installed with the super peer and peer using the PKI based resource discovery approach and the other pair installed with the super peer and

peer using the Functional Encryption based resource discovery approach. We again hosted the super peer nodes on the BT cloud platform, and the peer nodes on the ATOS test-bed.

We can see from the graph in Fig.6.4 that, similar to the single cloud platform trials, on average the runtime of the secure resource discovery process using the PKI-based design is much greater (*1615.66 milli-seconds*) than that for our Functional Encryption based scheme (*578.22* milli-seconds). Similarly, the variation in the time duration results in case of the former is still much greater than that for the later scheme.

6.6.4 Results Analysis

Given the known stable network latency between the different cloud platforms available to us, we profiled the implementation of the PKI based design and its cryptographic libraries at the code level, in order shed some light on the reasons behind its large overhead in terms of both time and jitter. The main reason for the relative slowness of the PKI-based design is the required use of cryptographically secure random number generation in this method. This introduces the largest time and jitter penalty for our PKI-based measurements, as the peer has to wait for system events to gather enough entropy from entropy sources like disk reads, network activity, mouse movement, key presses, etc. to generate this kind of random number. In a typical cloud hosted virtual machine, these events are usually quite stable or uniform (as in case of disk and network activity), or totally absent (as in case of mouse and keyboard activity). Therefore, it takes a much longer time to generate the required random number of reasonable strength. The

Functional Encryption based approach, on the other hand, does not have such a requirement. For it to work, it just needs attributes that are usually in an alpha-numeric form.

Furthermore, a peer in this approach does not need to get a key signed by the super peer CA to perform encryption, thus eliminating the need to get a signed certificate from super peers upon bootstrapping. Due to these main design and implementation specific reasons, our scheme incurs about *74.2%* less overhead than a PKI based scheme.

6.7 Chapter Summary

In this chapter, we described in detail how we solve the problem of secure service based resource discovery in our framework. The resources we focus on discovering were the VM IDs, IP addresses and port numbers of the neighbouring peers in a VPN overlay. The knowledge of these resources is important as they are required to construct IPsec tunnels between a peer and the other neighbouring peers that it wants to communicate with. We described how we make use of the Kademlia peer-to-peer protocol's resource discovery features to achieve this goal. However, we still needed to address the issue that the data about a peer's resources has to be gathered in such a manner that it remains private and confidential between the valid peers of a VPN overlay.

So we started by describing the threat vectors for this problem in detail. This mostly focused on Sybil attacks, where a malicious attacker can create or impersonate a large number of fake peers in an overlay and use them to disrupt or compromise the data and communication. We analysed how this type of attack

would work in a Kademlia based deployment and worked out the number of fake peers that will be required to intercept all the peer communication in a Kademlia based overlay. We discovered that it will require only 14 % fake peers to compromise a Kademlia overlay using a default k-bucket size.

We then gave the details of our security protocol for the resource discovery process, which is used after the admission control process has been successfully completed. We described how a certificate authority based scheme can help us in securing the resource discovery process, but also highlighted its problems with centralisation and scalability with regard to the overlay size. Therefore, we presented in detail our novel secure resource discovery model that utilises functional encryption techniques to accomplish our goals securely and efficiently.

Lastly we described the implementation details of this scheme in our system and related the experiments we have conducted that measure the time taken by a peer to discover the identity of other peers currently in its overlay, that is, the VPN overlay with the same ServiceID. We compared the results of our functional encryption based scheme with a PKI-based method using RSA algorithm that we have implemented in the same environment. We showed the time taken by the secure resource discovery process for 100 experimental instances using our scheme incurred approximately 75 % less overhead than the PKI based scheme.

Chapter 7

Conclusions

7.1 Achievements

In this thesis, we present a secure and scalable communication framework for cloud services/applications deployed in an inter-cloud environment. We employ the decentralisation and resilience afforded by structured peer-to-peer overlays to design a novel command and control architecture for managing and operating the secure communication framework. The construction of a decentralised and distributed command and control mechanism is one of the main objectives for our research effort, and is achieved by constructing two tiers of peer-to-peer overlay networks, with the upper tier overlay acting as a *universal overlay*, spanning across multiple cloud platforms and undertaking the general management related responsibilities, whereas the lower tier *VPN overlay* operates within the scope of a single cloud service that is being deployed on virtual machines on these multiple cloud platform, with the main responsibility of encrypting the communication between these virtual machines according to the security polices set by the cloud service owner. Therefore, we are able to offer secure communication functionality to multiple cloud services by using a separate VPN overlay for each service,

while a single universal overlay acts as the overseer of all the operational VPN overlays.

We also utilise the inherent ability offered by almost all structure peer-to-peer overlay networks to handle growing amount of work load. To design and architect a *scalable* communication framework is one of our main research objectives. The main challenge in this regard is that although a single peer-to-peer network might be able to address this issue relatively easily by using its overlay churn management protocol, our architecture design uses two tiers of peer-to-peer overlays. Although the universal overlay might be a single overlay network in its tier, it has to instantiate and manage multiple VPN overlays in the lower tier. In order to address this challenge, we designate a subset of the peers in the universal overlay as super peer nodes for the underlying VPN overlay, with at least one super peer node in each participating cloud platform. This approach provides us with a stable bootstrapping point, as well as addressing the issue of peer churn in the VPN overlays, as we can increase the number of super peers in the universal overlay as the work load in the underlying VPN overlays increases, or vice versa.

Another main objective that we set out in the beginning of this research is that the communication between the components of the deployed cloud service should be secure. We achieve this goal by making use of the IPsec protocol to form the VPN links between the virtual machines constituting a VPN overlay, thus providing confidentiality and integrity for all the data exchanged between the constituent components of the deployed cloud service. However, this proves to be just the last lag of the complete security life cycle, and we design a comprehensive and novel combination of application partitioning and security-by-isolation schemes to formulate the secure establishment and operation of the IPsec links

between the virtual machines of the deployed cloud service. This is due to the fact that the sharing and management of IPsec session keys for establishing secure communication tunnels is a complex challenge in a peer-to-peer environment. As structured peer-to-peer overlay networks have been primarily designed for fast and scalable content distribution, security considerations have not been the focus of the design in most of the existing implementations. Therefore, as a result of this there are a few well-known security vulnerabilities associated with a peer-to-peer overlay system, especially concerned with the identity of the peers and the ease with which they can be spoofed.

So in addition to design and implement a protocol for the generation, sharing and management of IPsec session keys for establishing secure communication tunnels, we address the security limitations of the command and control mechanism of the communication framework as well. We identify two crucial stages in the life cycle model of a peer-to-peer communication framework where the application of a security model would maximise the secure operation of the whole framework, before the third and final stage of establishing IPsec links. The first of these stages is the admission control stage, where the P2P clients running on the virtual machines of a particular cloud service seek to enrol with the super peers of the universal overlay in order to join their specific VPN overlay. The second stage is the resource discovery stage, where the peer nodes of a particular VPN overlay wish to discover their neighbouring peers in the same overlay so that they can begin the process of securely sharing the IPsec session keys.

The security of these three stages is of paramount importance in order to secure the complete life cycle of the inter-cloud virtual private network. According to the principals of security engineering, we decide to secure each of these stages

using different security mechanisms, so that the compromise of the security of one stage, however improbable, would not make it easy for an attacker to compromise the whole system.

Therefore, for the first stage we devise a secure admission control process that relies on a protocol that we construct according to the concept of zero-knowledge password proof. As its enabling feature, we allow the owners of the cloud service to contextualise their virtual machine images with some security meta-data, which can be used by our admission control protocol implemented in the P2P client. These security meta-data can be in the form of simple and easy to remember passwords, as one of the benefit of using a zero-knowledge password proof schemes is that the actual password is never transmitted between the peers and the super peers. However, we also offer and evaluate the option of strengthening this scheme further by increasing the entropy of the password by using a key strengthening mechanism.

For the second stage, we devise a secure resource discovery process based on Functional Encryption schemes. The resources can be anything, like the identifiers of virtual machines in a VPN overlay, keys, and other security credentials etc. For this process, we use the peer identities and their cloud service and virtual machine attributes to come up with a protocol, that can encrypt and store the resources such that only the P2P clients having the right attributes and peer identity are able to decrypt them. Therefore, it allows us to meet our objective of secure and scalable key distribution as well, because this scheme too relies on the super peers of the universal overlay.

A salient feature of our research effort is the practical evaluation of the performance of our solution in a real-world deployment, as we are fortunate enough

to be given access to two commercial and one academic cloud IaaS platforms to use as our test-bed. As a result, we are able to show that our architecture presents a minimal latency and throughput overhead of creating and maintaining the Inter-Cloud VPN connections among the virtual machines of a cloud service deployed on multiple cloud platforms, being 5% and 10% respectively. Similarly, we demonstrate the performance of our admission control scheme as having approximately 82% more efficiency as compared to a PKI based scheme, and the secure resource discovery scheme being about 72% more efficient for the same comparison.

7.2 Challenges and Limitations

The main challenge we face is the diversity of cloud platforms being used by the cloud infrastructure as a service providers, as each cloud platform has its own management API for the purposes of account management, virtual machine management, identity management, etc. We try to overcome this challenge by customising a virtual machine contextualisation tool that allows us to install and embed programs and data inside virtual machine images before launching them in their respective host cloud platforms. This affords a very valuable level of control to conduct our experiments in realistic conditions over multiple cloud infrastructure environments. However, this is not a perfect solution as it requires the contextualisation service to be extended every time a new cloud platform is needed to be supported.

Another major challenge is using various cryptographic APIs to manage the different kinds of security credentials and parameters used in the different pro-

tocols and mechanisms that we have to implement. So although the security engineering principals of system partitioning and security-by-isolation makes for a robust solution, it also makes for a quite complex implementation. The most difficult part is to deal with the integration issues between modules that use different encoding structures and standards for their security credentials. For example, most of the Public Key Infrastructure APIs use the Abstract Syntax Notation One (ASN.1) standard for the certificates and public, private keys, whereas most IPsec implementations in Linux require the credentials to be in PEM (Privacy-Enhanced electronic Mail) format, and the library implementing Functional Encryption uses Basic Encoding Rules (BER) and Distinguished Encoding Rules (DER).

Another challenge that we address in our work is the complexity of manual configuration and dependency resolution for the deployment of P2P clients in the virtual machines. The later issue has been resolved by the contextualisation tool mentioned before, and the remaining peer related configuration aspects were automated by using the DHT of the peer-to-peer overlays as the shared storage for the configuration data and implementing a periodic update feature in the P2P clients to check for their service related updates regularly against known index/-content keys in the DHT. However, the former issue is more challenging, especially in case of virtual machines' networking and IPsec related configurations. This is due to the reason that only one of our cloud platforms (Flexiant) assigns public IP addresses to all of its virtual machines, while the other two cloud platforms only offer NAT based networking, which assigns private IP addresses to virtual machines which all map to a single public IP address. This breaks the end-to-end design of the network layer and makes these virtual machines inaccessible from external networks.

This is a tough problem to overcome as it also makes it impossible to use IPsec in AH mode, as it is incompatible with NAT. This is because the Source IP Address field of the IP packet header is included in the integrity checking process of the IPsec AH mode, and as this field is changed in a NAT environment, the integrity check fails at the recipient. In order to overcome these issues, we have to implement a *NAT Traversal* module in our P2P clients that is able to detect if the peer is behind a NAT firewall and modify the IPsec policy accordingly to adapt for this situation.

7.3 Future Work

We also highlight some the limitations of our research effort that we intend to address as future work. From the security perspective, one of the main issues currently not addressed in this work is the handling of Denial-of-Service attacks. In the case of peer-to-peer overlays, the most common form of a Denial-of-Service attack is to flood the overlay with bogus traffic, thus preventing the peers with processing legitimate traffic. This type of attacks are very difficult to handle, even by increasing the computational and network resources, as the attack can also be scaled by distributing it. Although there are some methods that can be utilised to mitigate the effects of a dDoS (Distributed Denial-of-Service) attack, they are effective only against a limited number of simultaneous attacks. However, this problem has to be addressed to make the communication framework as robust as possible.

Another feature that needs to be focused on in the future is the dynamically update the components of the communication framework. In our architecture, an

additional advantage of using our two-tiered overlay approach is that the peers of a VPN overlay can get, update and modify the P2P client program dynamically from the super-peers in the universal overlay. This possibility can be further enhanced to install or run any program on the virtual machine hosting the peer node. This will add the valuable feature of a dynamic code delivery service to this architecture. The program to be run can be signed by the super-peers for validity and it can check for updated versions of itself by querying for the associated $ServiceID$ in the persistent store of the universal overlay's DHT.

Overall, we feel that the research contributions made in this thesis with regard to the ICVPN architecture can be very useful to the cloud service providers and developers to highlight the high level requirements of users and applications that want to construct a secure communication framework to fulfil their needs. The ICVPN research effort has also validated two novel specific state-of-the-art security protocols, providing insights into their use in an inter-cloud environment.

Lastly, a major achievement of our research efforts is its real-world deployment and evaluation. Simulations and emulations of such large scale systems as ours are not a substitute for deployment, as in most cases it is impossible to understand ahead of time the impact of the environment on technology and this is usually critical to system design. Real-world deployment and evaluation is the only way to fully investigate the complex interactions between the cloud computing applications, the underlying network overlays, and the inter-cloud environment.

Appendix A

Virtual Machine Contextualization

Most of the current cloud Infrastructure-as-a-Service providers enable rapid provisioning of virtual machines for their users. This in turn allows the users of the cloud platforms to scale up and down their cloud services and applications on-demand. However, this flexibility in virtual machine provisioning introduces a new set of challenges for dynamic service configuration, one of the main challenges being the contextualisation of the virtual machines.

1 Contextualisation

We define contextualisation is a set of processes and mechanisms that enable us to modify the virtual machine images in such a way that external data and programs can be placed inside well known locations in these images. VM contextualisation is a valuable tool especially in the use case scenarios of inter-cloud and multiple cloud platforms for reasons of interoperability. Most cloud service providers offer IaaS services that are not interoperable with each other. In this respect, VM contextualisation can be used for enabling interoperability between IaaS providers by by-passing their native platform services.

2 Architecture

The VM Contextualization service of the OPTIMIS toolkit [64] provides us with two capabilities. The first capability is a bootstrap mechanism to prepare a VM image for embedding the appropriate context in it. The second capability provides a mechanism for creating ISO CD-ROM (International Organization for Standardization Compact Disc Read-Only Memory) images that contain the context data and operating system specific scripts for processing the context data.

The main purpose of the VM Contextualisation tool is to prepare a VM image in such a way that it is able to receive the context data in a reusable fashion. It operates by mounting a VM image using the QEMU quick machine emulator [60] and modifying it to include a collection of programs and scripts that are usually run at the operating system boot up. When a virtual machine instantiated from a contextualised VM image is launched in a cloud platform, it can run these programs and scripts and access contextualisation data held within an ISO image attached with the VM.

3 Advantages

The use of an ISO image as a mechanism to store the contextualisation data provides a facility to separate the contextualisation data from the VM image and removes the time consuming process of creating multiple VM images with different contextualisation needs. Furthermore, all cloud computing and virtualisation platforms provide the feature of attaching ISO images with virtual machines, making this tool universally usable. This also enhances the security of the contextualisation process, as sensitive security related data and meta-data is not stored on

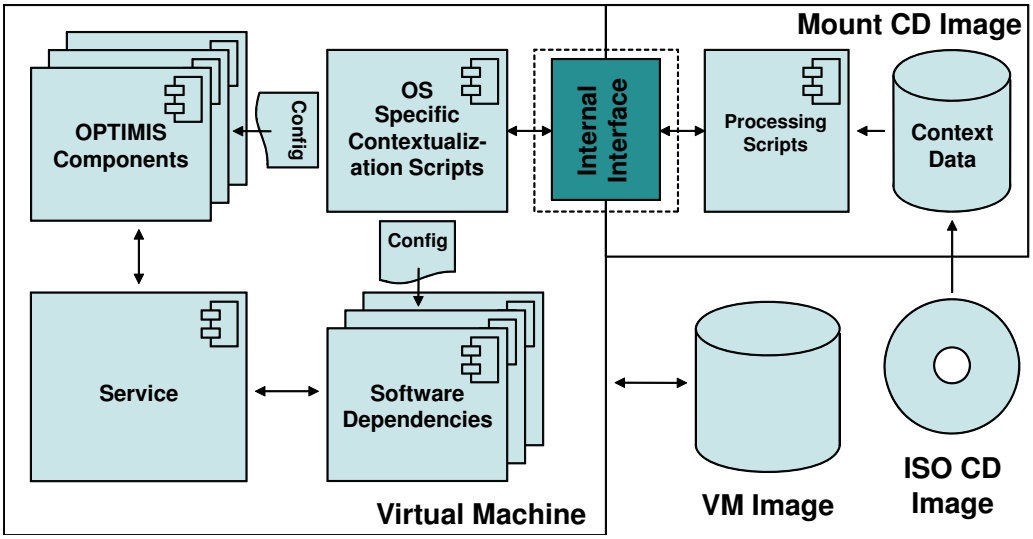


Figure A.1: Interaction between VM image and ISO Image at run time [13]

the VM itself but on the ISO image, where it can be periodically refreshed or even deleted by the VM contextualisation tool.

Figure A.1 shows the instance-level contextualisation process of a VM when it starts its execution at system boot up. During the boot process, the contextualisation tool mounts the ISO image containing the contextualisation data and the programs and scripts. These scripts and programs can perform all sorts of operations on the guest operating system of the virtual machines, like modifying configuration of installed programs, install additional software and resolve their dependencies, and establish network connections with other virtual machines etc. etc. Furthermore, these programs can be made to execute in daemon mode, in order to offer a permanently running contextualisation functionality.

Appendix B

IPsec Policy

A sample listing of the IPsec policy applied to a running instance of a VPN overlay is given in the listing below. The IP addresses of the VPN peers are detected by the Peer-to-Peer client software dynamically as they are subject to change in a cloud environment each time a VM reboots. This particular policy is encrypting all the ICMP traffic between two peer nodes and all TCP traffic that is incoming and outgoing on ports *80* and *8080* (standard ports for a web server and Apache Tomcat server) as this is the traffic belonging to the applications that the service wants to encrypt.

Sample policy

```
spdadd 82.223.250.28 217.33.61.85 icmp -P in ipsec
               esp/transport//require
               ah/transport//require;
spdadd 217.33.61.85 82.223.250.28 icmp -P out ipsec
               esp/transport//require
               ah/transport//require;
spdadd 82.223.250.28 217.33.61.85[80] tcp -P in ipsec
               esp/transport//require
               ah/transport//require;
spdadd 217.33.61.85[80] 82.223.250.28 tcp -P out ipsec
               esp/transport//require
               ah/transport//require;
```

```
spdadd 82.223.250.28[80] 217.33.61.85 tcp -P in ipsec
               esp/transport//require
               ah/transport//require;
spdadd 217.33.61.85 82.223.250.28[80] tcp -P out ipsec
               esp/transport//require
               ah/transport//require;
spdadd 82.223.250.28 217.33.61.85[8080] tcp -P in ipsec
               esp/transport//require
               ah/transport//require;
spdadd 217.33.61.85[8080] 82.223.250.28 tcp -P out ipsec
               esp/transport//require
               ah/transport//require;
spdadd 82.223.250.28[8080] 217.33.61.85 tcp -P in ipsec
               esp/transport//require
               ah/transport//require;
spdadd 217.33.61.85 82.223.250.28[8080] tcp -P out ipsec
               esp/transport//require
               ah/transport//require;
```

Appendix C

Publications and Patents

1 Book Chapter

Johan Tordsson, Karim Djemame, Daniel Espling, Gregory Katsaros, Wolfgang Zielgler, Oliver Waldrich, Kleopatra Konstanteli, *Ali Sajjad*, Muttukrishnan Rajarajan, Georgina Gallizo and Srijith K. Nair, ”Towards Holistic Cloud Management”, Book chapter in ”European Research Activities in Cloud Computing”, Jan 2012, Cambridge Scholars Publishing.

2 Journals

Sajjad, A., Rajarajan, M., Zisman, A., Dimitrakos, T.,”A Scalable and Dynamic Application-level Secure Communication Framework for Inter-Cloud Services”, Elsevier Journal of Future Generation Computer Systems (FGCS), March 2015.

3 International Conferences

1. *Sajjad, A.*, Rajarajan, M., Zisman, A., Nair, S. K. & Dimitrakos, T., ”Dynamic virtual private network provisioning from multiple cloud infrastructure service

providers", 4th European Conference, ServiceWave 2011, 26-28 Oct 2011, Poznan, Poland.

2. *Sajjad, A.*; Zisman, A.; Rajarajan, M.; Nair, S.K.; Dimitrakos, T., "Secure communication using dynamic VPN provisioning in an Inter-Cloud environment", 18th IEEE International Conference on Networks (ICON), 428-433, 12-14 Dec. 2012, Singapore.

3. *A. Sajjad*, M. Rajarajan, and T. Dimitrakos, "A low-overhead secure communication framework for an inter-cloud environment", International Conference on Intelligent Cloud Computing, 24-26 Feb 2014, Muscat, Oman.

4 Patent

1. *Sajjad, A.*, El-Moussa, F., "Application Level VPN", IRF No. A32672/E01921, BT IPD, April 2014.

References

[1] Federal Information Processing Standard PUB 197. Announcing the advanced encryption standard (aes), 2001. 59, 66, 85

[2] Serge Abiteboul. *Querying semi-structured data*. Springer, 1997. 141

[3] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *ACM SIGOPS Operating Systems Review*, pages 186–201. ACM, 1999. 140

[4] Les Cottrell Ajay Tirumala and Tom Dunigan. Measuring end-to-end bandwidth with iperf using web100. In *Web100, Proc. of Passive and Active Measurement Workshop*, 2003. 93

[5] Amazon. AWS Elastic Beanstalk, 2015. URL http://aws.amazon.com/elasticbeanstalk. 3

[6] Amazon. Virtual private cloud, 2015. URL http://aws.amazon.com/vpc. xv, 32, 42

[7] Amazon. Amazon Elastic Compute Cloud, 2015. URL http://aws.amazon.com/ec2. 3, 40

[8] Amazon. Amazon Simple Storage Service, 2015. URL http://aws.amazon.com/s3. 3

[9] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. *SIGCOMM Comput. Commun. Rev.*, January 2002. 51, 75

[10] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A proposal for the advanced encryption standard. *NIST AES Proposal*, 174, 1998. 66

[11] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, 2004. 140

[12] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4): 50–58, April 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672. URL http://doi.acm.org/10.1145/1721654.1721672. 5

[13] Django Armstrong, Karim Djemame, Srijith Krishnan Nair, Johan Tordsson, and Wolfgang Ziegler. Towards a contextualization solution for cloud platform services. In *CloudCom*, pages 328–331, 2011. xviii, 80, 119, 177

[14] Ken Arnold, Robert Scheifler, Jim Waldo, Bryan O'Sullivan, and Ann Wollrath. *Jini Specification*. Addison-Wesley Longman Publishing Co., Inc., 1999. 76, 140

[15] Randall Atkinson. Security architecture for the internet protocol. In *RFC 1825*, 1995. 57

[16] A Balasubramanian, A Hemanth Kumar, and R Prasanna Venkatesan. An optimized and secured vpn with web service. *Networking and Communication Engineering*, 6(2), 2014. 18

[17] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003. 89, 129, 158

[18] William C Barker and Elaine B Barker. Recommendation for the triple data encryption algorithm (tdea) block cipher. In *National Institute of Standards & Technology*, 2012. 59

[19] Salman Baset and Henning Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *CoRR*, 2004. 76

[20] Brian Beach. Virtual private cloud. In *Pro Powershell for Amazon Web Services*, pages 67–88. Springer, 2014. 18

[21] S.M. Bellovin and Michael Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *Research in Security and Privacy, 1992 IEEE Computer Society Symposium on*, pages 72–84, May 1992. 120

[22] K. Berket, A. Essiari, and A. Muratas. Pki-based security for peer-to-peer information sharing. In *Peer-to-Peer Computing, 2004. Proceedings. Pro-*

ceedings. Fourth International Conference on, pages 45–52, Aug 2004. 130

[23] Mark Berman, Jeffrey S. Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. Geni: A federated testbed for innovative network experiments. *Computer Networks*, 61(0):5 – 23, 2014. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.bjp.2013.12.037. URL http://www.sciencedirect.com/science/article/pii/S1389128613004507. Special issue on Future Internet Testbeds Part I. 33

[24] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, and Bernd Pollermann. World-wide web: the information universe. *Internet Research*, 2(1): 52–58, 1992. 51

[25] Alex Berson. *Client-server architecture*. McGraw-Hill, 1992. 21

[26] J Bethencourt, A Sahai, and B Waters. Advanced crypto software collection: The cpabe toolkit, 2015. URL http://acsc.cs.utexas.edu/cpabe/. 89, 158

[27] Luca Boccassi, Marwan M Fayed, and Mahesh K Marina. Binder: a system to aggregate multiple internet gateways in community networks. In *Proceedings of the 2013 ACM MobiCom workshop on Lowest cost denominator networking for universal access*, pages 3–8. ACM, 2013. 22

[28] Thomas Bocek. TomP2P: A P2P-based high performance key-value pair storage library, 2015. URL http://tomp2p.net/. 89, 129, 158

[29] André B. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2Nd International Workshop on Software and Performance*, WOSP '00, pages 195–203, New York, NY, USA, 2000. ACM. ISBN 1-58113-195-X. doi: 10.1145/350391.350432. URL http://doi.acm.org/10.1145/350391.350432. 99

[30] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001. 68

[31] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004. 68

[32] BritishTelecom. BT Compute Cloud, 2015. URL https://cloud.btcompute.bt.com. 89, 129, 158

[33] Paul Buchheit. GMail, 2004. URL http://mail.google.com. 2

[34] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. 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. 7, 19

[35] James Henry Carmouche. *IPsec virtual private network fundamentals*. Pearson Education, 2007. 58

[36] C Stephen Carr, Stephen D Crocker, and Vinton G Cerf. Host-host com-

munication protocol in the arpa network. In *Proceedings of the Spring Joint Computer Conference*, pages 589–597. ACM, 1970. 93

[37] M. Carroll, A. van der Merwe, and P. Kotze. Secure cloud computing: Benefits, risks and controls. In *Information Security South Africa (ISSA), 2011*, pages 1–9, Aug 2011. 5

[38] Daniele Catteddu. Cloud computing: Benefits, risks and recommendations for information security. In Carlos Serro, Vicente Aguilera Daz, and Fabio Cerullo, editors, *Web Application Security*, volume 72 of *Communications in Computer and Information Science*, pages 17–17. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-16119-3. doi: 10.1007/978-3-642-16120-9_9. URL http://dx.doi.org/10.1007/978-3-642-16120-9_9. 5

[39] L. Chen. *Recommendation for Key Derivation Using Pseudorandom Functions*. NIST Special Publication 800-108, October 2009. 87

[40] Shiping Chen, S. Nepal, and Ren Liu. Secure connectivity for intra-cloud and inter-cloud communication. In *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*, pages 154–159, Sept 2011. doi: 10.1109/ICPPW.2011.54. xv, 32, 38, 39, 40

[41] Yang-hua Chu, Aditya Ganjam, TS Eugene Ng, Sanjay G Rao, Kunwadee Sripanidkulchai, Jibin Zhan, and Hui Zhang. *Early experience with an internet broadcast system based on overlay multicast*. School of Computer Science, Carnegie Mellon University, 2003. 32

[42] Cisco. Understanding the ping and traceroute commands. In *Cisco IOS Software Releases*, Aug 2014. 91

[43] Kimberly Claffy, Greg Miller, and Kevin Thompson. The nature of the beast: Recent traffic measurements from an internet backbone. In *Proceedings of INET*, volume 98, pages 21–24, 1998. 93

[44] LLC Cloud Strategy Partners. Ieee intercloud interoperability and federation framework. *Computer Society*, 2015. 7, 19

[45] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001. 68

[46] Bram Cohen. The BitTorrent protocol specification, 2001. URL http://www.bittorrent.org/beps/bep_0003.html. 76

[47] CohesiveFT. VPN-Cubed, 2014. URL http://www.cohesiveft.com/vpncubed. xv, 32, 44, 45

[48] Steven E Czerwinski, Ben Y Zhao, Todd D Hodes, Anthony D Joseph, and Randy H Katz. An architecture for a secure service discovery service. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 24–35. ACM, 1999. 159

[49] Amit Sahai Dan Boneh and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012. 147

[50] Luca Deri and Richard Andrews. N2N: a layer two Peer-to-Peer VPN. In *Resilient Networks and Services*, Lecture Notes in Computer Science, pages 53–64. Springer Berlin Heidelberg, 2008. xiv, 27, 31, 32

[51] Brian Desmond, Joe Richards, Robbie Allen, and Alistair G Lowe-Norris. *Active Directory: Designing, Deploying, and Running Active Directory*. " O'Reilly Media, Inc.", 2008. 142

[52] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, November 1976. 62, 66, 85

[53] Jeff Dike. *User Mode Linux*, volume 2. Prentice Hall Englewood Cliffs, 2006. 26

[54] Wei Dong and Zhen Ya Zhang. Research on virtual private lan service signaling protocol and its application. *Applied Mechanics and Materials*, 543:2585–2588, 2014. 18

[55] Naganand Doraswamy. *IPSec : the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall PTR, 2nd ed. edition, 2003. 33, 57, 84

[56] John Douceur. The sybil attack. In *Peer-to-Peer Systems*. Springer Berlin / Heidelberg, 2002. 116, 145

[57] P Draft Standard. 802.1 q/d10, ieee standards for local and metropolitan area networks: Virtual bridged local area networks, 1997. 24

[58] Leticia Duboc, David S. Rosenblum, and Tony Wicks. A framework for modelling and analysis of software systems scalability. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 949–952, New York, NY, USA, 2006. ACM. ISBN 1-59593-375-

1. doi: 10.1145/1134285.1134460. URL http://doi.acm.org/10.1145/1134285.1134460. 99

[59] James Gosling et. al. *The Java Language Specification*. Addison Wesley, 2nd ed. edition, 2000. 88, 128, 157

[60] Fabrice Bellard. QEMU: open source process emulator, August 2015. URL http://wiki.qemu.org/Main_Page. 176

[61] Ilhem Fajjari, Nadjib Aitsaadi, Michał Pióro, and Guy Pujolle. A new virtual network static embedding strategy within the clouds private backbone network. *Computer Networks*, 62:69–88, 2014. 18

[62] Tse-Yun Feng. A survey of interconnection networks. *Computer*, 14(12): 12–27, Dec 1981. ISSN 0018-9162. doi: 10.1109/C-M.1981.220290. 35

[63] Niels Ferguson and Bruce Schneier. *Practical cryptography*, volume 141. Wiley New York, 2003. 66

[64] A. J Ferrer, F. Hernndez, J. Tordsson, E. Elmroth, C. Zsigri, R. Sirvent, J. Guitart, R. M Badia, K. Djemame, and W. Ziegler. OPTIMIS: a holistic approach to cloud service provisioning. In *First International Conference on Utility and Cloud Computing*, December 2010. 176

[65] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol - http/1.1, June 1999. 90

[66] Flexiant. Flexiant, your cloud simplified, 2015. URL http://www.flexiant.com/. 3, 89, 129, 158

[67] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid - enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15:2001, 2001. 1

[68] Gartner. Cloud consumers need brokerages to unlock the potential of cloud services, July 2009. URL http://www.gartner.com/it/page.jsp?id=1064712. 7

[69] Gartner. Survey analysis: Buyers reveal cloud application adoption plans through 2017, November 2014. URL http://www.gartner.com/document/2883318. 7

[70] Frank Gens. New idc it cloud services survey: Top benefits and challenges, December 2009. URL http://blogs.idc.com/ie/?p=730. xiii, 5, 6, 8

[71] Google. Secure data connector, 2012. URL http://code.google.com/securedataconnecto. xv, 32, 43

[72] Google. Google App Engine, April 2015. URL https://developers.google.com/appengine. 3

[73] Rob Gordon. Java native interface. *Prentince Hall PTR*, 1998. 158

[74] Dominik Grolimund. Wuala - a distributed file system. *Google Tech Talk*, October 2007. URL www.youtube.com/watch?v=3xKZ4KGkQY8. 2

[75] Jungsoo Han. Distributed hybrid p2p networking systems. *Peer-to-Peer Networking and Applications*, pages 1–2, 2014. 27

[76] D. Harkins. Internet key exchange (ike). In *RFC 2409*, 1998. 61

[77] R Haywood. Business to business (b2b). *Key Concepts in Public Relations*, page 35, 2009. 38

[78] Mark D. Hill. What is scalability? *SIGARCH Comput. Archit. News*, 18 (4):18–21, December 1990. ISSN 0163-5964. doi: 10.1145/121973.121975. URL http://doi.acm.org/10.1145/121973.121975. 99

[79] H. Hiroaki, Y. Kamizuru, A Honda, T. Hashimoto, K. Shimizu, and H. Yao. Dynamic ip-vpn architecture for cloud computing. In *Information and Telecommunication Technologies (APSITT), 2010 8th Asia-Pacific Symposium on*, pages 1–5, June 2010. xiv, 32, 34

[80] Susan Hohenberger and Brent Waters. Attribute-based encryption with fast decryption. In *Public Key Cryptography*, pages 162–179, 2014. 68, 147

[81] Ines Houidi, Wajdi Louati, and Djamal Zeghlache. A distributed virtual network mapping algorithm. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 5634–5640. IEEE, 2008. 21

[82] R. Housley. Using advanced encryption standard (aes) ccm mode with ipsec encapsulating security payload (esp), 2005. 85

[83] Drew Houston and Arash Ferdowsi. Dropbox, 2015. URL http://dropbox.com. 2

[84] Timothy A Howes, Mark C Smith, and Gordon S Good. *Understanding and deploying LDAP directory services*. Addison-Wesley Longman Publishing Co., Inc., 2003. 142

[85] Adriana Iamnitchi and Ian Foster. On fully decentralized resource discovery in grid environments. In *Grid Computing - GRID 2001*, pages 51–62. Springer, 2001. 140

[86] Adriana Iamnitchi, Ian Foster, and D Nurmi. A peer-to-peer approach to resource discovery in grid environments. In *IEEE High Performance Distributed Computing*, 2002. 140

[87] Kohei Ichikawa, Hirotake Abe, Hiroaki Yamanaka, Eiji Kawai, Shinji Shimojo, et al. A network performance-aware routing for multisite virtual clusters. In *Networks (ICON), 2013 19th IEEE International Conference on*, pages 1–5. IEEE, 2013. 22

[88] Intercloud Working Group. P2302 - Standard for Intercloud Interoperability and Federation (SIIF). *IEEE Computer Society*, January 2012. URL https://standards.ieee.org/develop/project/2302.html. 19

[89] K. Ishimura, T. Tamura, S. Mizuno, H. Sato, and T. Motono. Dynamic ip-vpn architecture with secure ipsec tunnels. In *Information and Telecommunication Technologies (APSITT), 2010 8th Asia-Pacific Symposium on*, pages 1–5, June 2010. xiv, xv, 32, 36, 37

[90] Van Jacobson, Craig Leres, and S McCanne. The tcpdump manual page. *Lawrence Berkeley Laboratory, Berkeley, CA*, 1989. 143

[91] Michael Jeronimo and Jack Weast. *UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play*, volume 158. Intel Press, 2003. 76, 140

[92] Xuxian Jiang and Dongyan Xu. VIOLIN: virtual internetworking on overlay INfrastructure. In *In Proc. Of The 2nd Intl. Symposium On Parallel And Distributed Processing And Applications*, 2003. xiv, 24, 27, 33

[93] Guojun Jin and B. Tierney. Netest: a tool to measure the maximum burst size, available bandwidth and achievable throughput. In *Information Technology: Research and Education, 2003. Proceedings. ITRE2003. International Conference on*, pages 578–582, Aug 2003. 94

[94] Pierre St Juste, Heungsik Eom, Benjamin Woodruff, Corey Baker, and Renato Figueiredo. Enabling decentralised microblogging through p2pvpns. *International Journal of Security and Networks*, 8(3):169–178, 2013. 27

[95] Charlie Kaufman. Internet key exchange protocol version 2 (ikev2). In *RFC 5996*, 2010. 61, 85

[96] Brian W Kernighan, Dennis M Ritchie, and Per Ejeklint. *The C programming language*, volume 2. Prentice-Hall Englewood Cliffs, 1988. 158

[97] Matt Kimball. Network diagnostics. In *BitWizard*, Aug 2014. 91

[98] Neal Koblitz and Alfred Menezes. *Pairing-based cryptography at high security levels*. Springer, 2005. 149

[99] I. Kotuliak, P. Rybar, and P. Truchly. Performance comparison of ipsec and tls based vpn technologies. In *9th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 217–221, Oct 2011. 60

[100] Maxim Krasnyansky. Virtual Tunnel, 2015. URL http://vtun.sourceforge.net. 22, 31

[101] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003. 63

[102] Paul J Leach, Michael Mealling, and Rich Salz. A universally unique identifier (uuid) urn namespace. *IETF RFC 4122*, 2005. 122

[103] Neal Leavitt. Internet security under attack: The undermining of digital certificates. *Computer*, 44(12):17–20, December 2011. ISSN 0018-9162. doi: 10.1109/MC.2011.367. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6096548. 85

[104] Tzong-Jye Liu, Chi-Bin Chou, and Chuan-Mu Tseng. P2p traffic classification in encrypted tunnels. In *Communications (APCC), 2013 19th Asia-Pacific Conference on*, pages 597–602. IEEE, 2013. 22

[105] Madhusanka Liyanage and Andrei Gurtov. Securing virtual private lan service by efficient key management. *Security and Communication Networks*, 7(1):1–13, 2014. 23

[106] Madhusanka Liyanage, Mika Ylianttila, and Andrei Gurtov. Ip-based virtual private network implementations in future cellular networks. *Handbook of Research on Progressive Trends in Wireless Communications and Networking*, 1:44, 2014. 18

[107] LogMeIn. Hamachi - a zero-configuration virtual private network, 2015. URL https://secure.logmein.com/products/hamachi2. 27

[108] LogMeIn. Hamachi security, 2015. URL https://secure.logmein.com/products/pro/security.aspx. 30

[109] Dong Lu, Yi Qiao, Peter Dinda, Fabian E Bustamante, et al. Characterizing and predicting tcp throughput on the wide area network. In *25th IEEE International Conference on Distributed Computing Systems*, pages 414–424. IEEE, 2005. 95

[110] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005. xv, 51

[111] Gordon Lyon. nmap: Network mapper. In *Phrack Magazine*, Aug 2014. 91

[112] Margaret Rouse. Ping of Death, August 2014. URL http://searchsecurity.techtarget.com/definition/ping-of-death. 92

[113] Norman Maurer. *Netty in Action*. Manning Publications Co., 1st ed. edition, November 2014. 101

[114] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002. 53, 55, 89, 129, 141, 158

[115] P Mell and T Grance. Draft NIST working definition of cloud computing. *http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf*, 2009. URL http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf. 2

[116] Du Meng. Implementation of a host-to-host vpn based on udp tunnel and openvpn tap interface in java and its performance analysis. In *Computer Science & Education (ICCSE), 2013 8th International Conference on*, pages 940–943. IEEE, 2013. 22

[117] Microsoft. Pathping. In *Technet Microsoft*, Aug 2014. 91

[118] MicroSoft. Windows Azure, February 2015. URL http://www.windowsazure.com. 3

[119] MicroSoft. Outlook.com, 2015. URL http://www.outlook.com. 2

[120] Ahmad Moradi, Andrea Lodi, and S Mehdi Hashemi. On the difficulty of virtual private network instances. *Networks*, 2014. 18

[121] Mike Muuss. The story of the ping program. In *US Army Research Laboratory*, Aug 2014. 91

[122] B Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, 1994. 30

[123] Daiyuu Nobori and Yasushi Shinjo. Vpn gate: a volunteer-organized public vpn relay system with blocking resistance for bypassing government censorship firewalls. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 229–241. USENIX, 2014. 27

[124] Legion of the Bouncy Castle. Bouncy castle java cryptography apis, 2015. URL http://www.bouncycastle.org/java.html. 89, 129, 157

[125] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006. 143

[126] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespignani. Dynamical and correlation properties of the internet. *Physical Review Letters*, 87(25):258701, 2001. 93

[127] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. *IETF Network Working Group*, 2012. 121

[128] Adrian Perrig. Shortcomings of password-based authentication. In *9th USENIX Security Symposium*. ACM, August 2000. 130

[129] J. Postel. Internet control message protocol. *IETF RFC 792*, 1981. 91

[130] Ravi Prasad, Constantinos Dovrolis, Margaret Murray, and KC Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *Network, IEEE*, 17(6):27–35, 2003. 93

[131] Rackspace. Rackspace: The open cloud company, 2014. URL http://www.rackspace.com. 3

[132] David Patrick Reed. User datagram protocol (udp). *IETF*, August 1980. 26, 29

[133] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 99–100. IEEE, 2001. 21

[134] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party

compute clouds. In *Proceedings of the 16th ACM conference on Computer and Communications Security*, pages 199–212. ACM, 2009. 117

[135] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for Large-Scale Peer-to-Peer systems. In *Middleware*. ACM, 2001. 55

[136] salesforce. Salesforce, 2015. URL http://www.salesforce.com. 2

[137] Bruce Schneier. The international data encryption algorithm (idea). *Dr Dobb's Journal-Software Tools for the Professional Programmer*, 18(13): 50–57, 1993. 66

[138] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption*, pages 191–204. Springer, 1994. 66

[139] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. *The Twofish encryption algorithm: a 128-bit block cipher*. John Wiley & Sons, Inc., New York, NY, USA, 1999. 31, 66

[140] Jeff Sedayao. Implementing and operating an internet scale distributed application using service oriented architecture principles and cloud computing infrastructure. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 417–421. ACM, 2008. 21

[141] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Pro-*

ceedings of CRYPTO 84 on Advances in cryptology, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc. 68

[142] Madhavapeddi Shreedhar and George Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, 1996. 101

[143] Stelios Sotiriadis and Nik Bessis. An inter-cloud bridge system for heterogeneous cloud platforms. *Future Generation Computer Systems*, 2015. 7, 19

[144] W Richard Stevens. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *IETF RFC 2001*, 1997. 93

[145] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001. 55

[146] Ananth I Sundararaj and Peter A Dinda. Towards virtual networks for virtual machine grid computing. In *In Proceedings of the 3rd USENIX Virtual Machine Research And Technology Symposium*, 2004. xiii, 24, 25

[147] T.H. Szymanski. Impact of future trends on exascale grid and cloud computing. In JulianMartin Kunkel, Thomas Ludwig, and HansWerner Meuer, editors, *Supercomputing*, volume 8488 of *Lecture Notes in Computer Science*, pages 215–231. Springer International Publishing, 2014. ISBN 978-3-319-07517-4. doi: 10.1007/978-3-319-07518-1_14. URL http://dx.doi.org/10.1007/978-3-319-07518-1_14. 7

[148] Andrew S. Tanenbaum and David J. Wetherall. Virtual private networks. In *Computer Networks*, page 821. Prentice Hall, 5th edition, October 2010. 75

[149] O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011. URL http://www.gnu.org/s/parallel. 101

[150] John Tate. Duality theorems in galois cohomology over number fields. In *Proceedings of an International Congress on Mathematics Stockholm*, pages 288–295, 1962. 149

[151] D. Taylor, T. Wu, N. Mavrogiannopoulos, and T. Perrin. *Using the Secure Remote Password (SRP) Protocol for TLS Authentication*. RFC 5054, November 2007. 121

[152] Tirumala, Ajay and Qin, Feng and Dugan, Jon and Ferguson, Jim and Gibbs, Kevin. Iperf: The TCP/UDP bandwidth measurement tool, August 2015. URL https://iperf.fr/. 93

[153] Linus Torvalds et al. The linux kernel. *http://www.kernel.org*, 2015. 88, 128, 157

[154] W Townsley, A Valencia, Allan Rubens, G Pall, Glen Zorn, and Bill Palter. Layer two tunneling protocol (l2tp). *IETF*, August 1999. 24

[155] Alan M Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 42:230–265, 1936. 67

[156] Van Heusden, FOLKERT. Testing http throughput and latency with httping, August 2014. URL http://www.vanheusden.com/httping/. 92

[157] Guohui Wang and T.S.E. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *Proceedings of IEEE INFOCOM*, pages 1–9, March 2010. 94

[158] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25910-7. doi: 10.1007/11426639_7. URL http://dx.doi.org/10.1007/11426639_7. 147

[159] André Weil. Sur les fonctions algébriquesa corps de constantes fini. *CR Acad. Sci. Paris*, 210:592–594, 1940. 149

[160] Wikipedia. Ping Flood, August 2014. URL http://en.wikipedia.org/wiki/Ping_flood. 92

[161] Wikipedia. Cloud Computing, December 2015. URL http://en.wikipedia.org/wiki/Cloud_computing. xiii, 4

[162] Wikipedia. Distributed Hash Tables, December 2015. URL http://en.wikipedia.org/wiki/Distributed_hash_table. xvi, 54

[163] Michele Willson and Tama Leaver. Zynga's farmville, social games, and the ethics of big data mining. *Communication Research and Practice*, pages 1–12, 2015. 2

[164] David Isaac Wolinsky, Panoat Chuchaisri, Kyungyong Lee, and Renato Figueiredo. Experiences with self-organizing, decentralized grids using the grid appliance. *Cluster computing*, 16(2):265–283, 2013. 27

[165] D.I Wolinsky, Kyungyong Lee, P.O. Boykin, and R. Figueiredo. On the design of autonomic, decentralized vpns. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, pages 1–10, Oct 2010. 27

[166] Wang Xing, Liu Guaiguai, and Wang Lixia. Build a provincial business system remote maintenance channel using open vpn. *Information & Communications*, 1:013, 2013. 22

[167] J. Yonan. OpenVPN - an open source SSL VPN solution, 2008. URL http://openvpn.net/. 22, 41

[168] J. Yonan. Facts about OpenVPN, 2015. URL https://openvpn.net/index.php/about-menu/openvpn-facts.html. 22

[169] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. On the characteristics and origins of internet flow rates. In *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '02, pages 309–322. ACM, 2002. ISBN 1-58113-570-X. doi: 10.1145/633025.633055. URL http://doi.acm.org/10.1145/633025.633055. 95

[170] B. Y Zhao, Ling Huang, J. Stribling, S. C Rhea, A. D Joseph, and J. D Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *Selected Areas in Communications, IEEE Journal on*, January 2004. 55