

Security for Cloud Based Services

SABRINA ALI TANDRA
and
SARWARUL ISLAM RIZVI



**KTH Information and
Communication Technology**

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Security for Cloud Based Services

Sabrina Ali Tandra
and
Sarwarul Islam Rizvi

2014-01-27

Master's thesis

Examiner and academic adviser
Professor Gerald Q. Maguire Jr.

School of Information and Communication Technology (ICT)
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

Cloud computing is a new buzzword in the modern information technology world. Today cloud computing can be considered as a service, similar to the way that electricity is considered a service in urban areas. A cloud user can utilize different computing resources (e.g. network, storage, software application), whenever required, without being concerned with the complex underlying technology and infrastructure architecture. The most important feature is that the computing resources are available whenever they are needed. Additionally, users pay only for the resource they actually use. As a result, cloud users can easily scale their information technology infrastructure, based on their business policy and requirements. This scalability makes the business process more agile.

The motivation for this thesis was the need for a suitable set of security guidelines for *ifoodbag* (and similar companies) when implementing web applications in the cloud. The goal of this thesis is to provide security in a system, being developed in another Master's thesis project, to implement the *ifoodbag* web application in a cloud. To achieve this goal, we began by identifying the risks, threats, and vulnerabilities in the system model proposed by these other students for their implementation. A study was made of several different security mechanisms that might reduce or eliminate risks and secure the most vulnerable points in the proposed system's design. Tests of these alternatives were conducted to select a set of mechanisms that could be applied to the proposed system's design. Justification for why these specific mechanisms were selected is given. The tests allowed the evaluation of how each of these different security mechanisms affected the performance of the system. This thesis presents the test results and their analysis. From this analysis a set of mechanisms were identified that should be included in the prototype of the system. In conclusion, we found that DNSSEC, HTTPS, VPN, AES, Memcached with SASL authentication, and elliptic curve cryptography gave the most security, while minimizing the negative impact on the system. Additionally, client & server mutual authentication and a multi-level distributed database security policy were essential to provide the expected security and privacy that users would expect under the Swedish Data Protection law and other laws and regulations.

Keywords: cloud computing, security, risk, vulnerability, performance.

Sammanfattning

Molntjänster är något nytt inom informationsteknikens värld, som kan idag kan liknas vid hur folk i stadsområden köper sin el. Människor som använder sig av molntjänster kan dela och använda olika data (t.ex. olika nätverk, lagringsutrymme och programvara) utan att ha djupare kunskaper om den bakomliggande, komplexa tekniken eller om infrastrukturens uppbyggnad. Den viktigaste egenskapen hos molntjänster är hur man kan dela och komma åt den dator man behöver när man vill och betalar bara för den dator man använder. Molntjänster har resulterat i att företag enkelt kan anpassa sin informationsteknikens-infrastruktur baserat på de policys och krav företaget har.

Motiveringen för denna avhandling är att det behövs lämpliga riktlinjer för företag som t.ex. iFoodBag (och liknande företag) vid integrering av webbapplikationer i molntjänster. Målet för denna avhandling är att ge stabilitet och säkerhet i systemet iFoodBag, ett program som är gjort i ett annat examensarbete. För att uppnå målet började vi med att identifiera risker, hot och sårbarheter i programmets modell och uppbyggnad i det andra examensarbetet. Med en efterföljande undersökning tittade vi på hur flera olika säkerhetsmekanismer antingen minskade eller eliminerade riskerna och såg även på de mest sårbara punkterna i det föreslagna programmets utformning. Med resultaten från denna undersökning genomförde vi tester för att välja vilka mekanismer som skulle fungera bäst med programmet. Motiveringen till varför vi valde dessa mekanismer är baserad på testerna och våran utvärdering av dessa. Vi valde säkerhetsmekanismer baserat på hur de påverkade prestandan i programmet. Denna avhandling presenterar testresultaten och analyserna av dessa. Genom att studera alla resultat valde vi ut de säkerhetsmekanismer som skulle fungera bäst i prototyp-programmet. Sammanfattningsvis kom vi fram till att DNSSEC, HTTPS, VPN och AES, Memcached med SASL autentisering, och elliptisk kurva kryptografi gav högst säkerhet med minst negativ påverkat på programmet. I tillägg såg vi att klient-server ömsesidig autentisering, och flera nivåer databas säkerhetspolicy var nödvändiga för att tillgodose de förväntningarna användare har på programmet när det gäller säkerhet och integritet i enighet med Svenska dataskyddslagstiftningar och andra lagar och förordningar.

Nyckelord: Molntjänster, säkerhet, risk, sårbarhet, utförande.

Acknowledgements

Sarwarul Islam Rizvi

I am thankful to almighty for giving me patience while working with the thesis project. I do remember my beloved parents who have always prayed, wished and, inspired me from some thousand miles away. Without their unconditional love and consistent support I could not manage to reach this stage of life. I am grateful to my lovely wife Elham Khorami, who has always literally inspired and practically insisted me day and night to finish the thesis work. Without her support I cannot even think of finishing my degree. I am thankful to my father and mother-in-laws for their love and prayers for me. Besides, I am thankful to all my friends who have inspired me time to time to finish the Master program. Special thanks to my thesis group mate Sabrina Ali Tandra for her cooperation throughout different phases of the thesis work.

Sabrina Ali Tandra

First of all, I am deeply grateful to almighty for bringing me this far, giving me courage and patience for pursuing my dream. I am thankful to my parents. Without them nothing was possible. It was their support; prayer and love which makes me fight all the odds and reach my goal. I am very thankful to all of my friends to believing in me, supporting me and encouraging me consistently. I am also thankful to my thesis partner Sarwarul Islam Rizvi for his support to make this thesis successful.

Table of contents

Abstract	i
Sammanfattning	iii
Acknowledgements	iv
Table of contents.....	v
List of Figures	ix
List of Tables	xi
List of acronyms and abbreviations	xiii
1 Introduction	1
1.1 Problem definition	1
1.2 Motivation.....	2
1.3 Scope.....	2
1.4 Method and methodology	2
1.5 Structure of this document	3
2 Background	5
2.1 What is Cloud computing?	5
2.2 Characteristics of Cloud Computing.....	6
2.2.1 On-demand self-service.....	6
2.2.2 Broad network access	6
2.2.3 Resource pooling	7
2.2.4 Rapid elasticity.....	7
2.2.5 Measured service.....	7
2.3 Three ways to provide cloud based services	7
2.3.1 Software as a Service (SaaS)	8
2.3.2 Platform as a Service (PaaS)	8
2.3.3 Infrastructure as a Service (IaaS)	9
2.4 Cloud Deployment Models	9
2.4.1 Private cloud.....	9
2.4.2 Public cloud	9
2.4.3 Hybrid cloud	9
2.4.4 Community cloud.....	10
2.5 Scalable model of the <i>ifoogbag</i> web application in cloud	10
2.6 How can using a cloud help a business?	11
2.7 Security issues in cloud.....	12
2.7.1 Confidentiality related attacks	12
2.7.2 Integrity related attacks	14
2.7.3 Accountability Check	14
2.7.4 Availability related attacks	14
3 System breakdown and analysis.....	17
3.1 Name resolving process of ifoodbag web application.....	17
3.1.1 What is the job of a DNS server?	17
3.1.2 How DNS works.....	17
3.1.3 Iterative and Recursive Queries	18
3.1.4 DNS name resolving process	19
3.1.5 Importance of DNS security	20
3.1.6 DNS security issues and vulnerabilities	20
3.1.7 Available solutions	22

3.1.8	DNS Security Extension (DNSSEC)	23
3.2	Distributing load among application servers	25
3.2.1	Why load balancing?	25
3.2.2	Using squid for load balancing	26
3.2.3	Policies to improve squid security.....	27
3.2.4	Alternative approaches for load balancing with Squid.....	27
3.3	HTTP connection between a user's browser and an <i>ifoodbag</i> application server.....	28
3.3.1	Hypertext Transfer Protocol (HTTP)	28
3.3.2	Security issues with HTTP	28
3.3.3	Solutions to improve security in HTTP.....	28
3.4	Communication between the management node and other nodes	29
3.4.1	What is VPN?	29
3.4.2	VPN Tunnels	30
3.4.3	VPN Implementation	31
3.4.4	OpenVPN	31
3.5	Caching web data.....	32
3.5.1	What is a cache?	32
3.5.2	How does Memcached work?.....	33
3.5.3	Advantages of Memcached.....	35
3.5.4	Disadvantages of Memcached.....	35
3.5.5	Security of Memcached.....	35
3.6	Cloud storage.....	36
3.6.1	What is cloud storage?	36
3.6.2	Types of Cloud storage.....	36
3.6.3	Characteristics of Cloud storage.....	37
3.6.4	Advantages and disadvantages of Cloud storage	37
3.6.5	Traditional versus Cloud storage.....	38
3.6.6	Reliability and Security factors about cloud storage	38
3.7	Distributed database	39
3.7.1	What is distributed Database?	39
3.7.2	Types of distributed databases	39
3.7.3	Heterogeneous versus Homogenous DDB	40
3.7.4	Approaches to DDBs	40
3.7.5	Advantages and disadvantages of DDBs.....	41
3.7.6	Security Weakness of distributed database.....	42
3.7.7	Security Components in a DDB	42
3.8	Addressing DDoS attack.....	43
3.8.1	What is a DDOS attack	43
3.8.2	Different types of DDOS attacks	44
3.8.3	Review of work related to prevention, detection, and mitigation of DDoS attack	45
4	Secure system design	49
4.1	Secure name resolving through DNSSEC	49
4.2	Secure browsing through HTTPS.....	50
4.3	Secure Cloud Storage	51
4.4	Method to Secure a DDB in the Cloud.....	52
4.5	Secure Memcached	52
5	Implementation	55
5.1	Secure browsing of ifoodbag web application.....	55

5.2	Secure Communication between Nodes	57
5.3	Secure information in database	59
5.4	Secure information in Cloud Storage	60
6	Results and evaluation	63
6.1	Revisit design issues	63
6.2	Recommendations	64
6.2.1	For Secure Name Resolving	64
6.2.2	For Securing Load Balancers	65
6.2.3	For Secure Browsing	66
6.2.4	For Secure Communication between Nodes	68
6.2.5	Memcache	75
6.2.6	Distributed database	75
6.2.7	Cloud Storage	78
6.3	Security guidelines	81
6.4	General guidelines	81
7	Conclusions and Future Work	83
7.1	Conclusions	83
7.2	Future work	83
7.3	Reflections	84
7.3.1	Social	84
7.3.2	Economic	85
7.3.3	Legal and Ethical issues	85
	References	87
	Appendix A: How to set policy with IPsec	97
	Appendix B: Set HTTP in IIS Web Server	115
	Appendix C: Code to measure time to get response for a SQL SELECT query through VPN Tunnel	117
	Appendix D: Code to measure time required to download certain content with HTTP and HTTPS	119
	Appendix E: Download time comparison for 390MB file	121
	Appendix F: Download time comparison for 3.13 MB file	123
	Appendix G: SQL SELECT query time comparison for different VPN setup	125
	Appendix H: SQL SELECT query time (in milliseconds) comparison for different VPN setup and for different result sizes	129
	Appendix I: Code to measure AES encryption time	131
	Appendix J: Time for inserting data	139
	Appendix K: Time for selecting data	141
	Appendix L: Analysis of data in appendices E & F	143
	Appendix M: Analysis of data in appendix H	145
	Appendix N: Storage security with AES and TripleDES	147
	Appendix O: AES encryption-decryption time in the same machine	157
	Appendix P: AES encryption-decryption time between networked hosts	161
	Appendix Q: TripleDES encryption-decryption time between networked hosts	165
	Appendix R: Analysis of data in Appendix P and Q	169
	Appendix S: Analysis of data in Appendix J and K	171
	Appendix T: Data Push and Pull time without security	173

List of Figures

Figure 1-1: Matt Bishop's three key aspects of security Confidentiality, Integrity, and Availability (Adapted from Figure 1-3 of [2])	1
Figure 2-1: Cloud computing logical diagram [9]	6
Figure 2-2: Cloud service model (Collected and edited from [12],[13])	8
Figure 2-3: A dynamically scalable model for <i>ifoodbag</i> in cloud [3] (Appears here with the permission of the authors.)	11
Figure 3-1: Host A using DNS to identify Host B in Internet (The idea for this figure is based upon [46].).....	17
Figure 3-2: DNS hierarchical tree structure	18
Figure 3-3: Recursive and iterative DNS queries.....	20
Figure 3-4: DNSSEC work process (concept taken from [62])	24
Figure 3-5: Load balancing with Squid.....	26
Figure 3-6: Remote-access VPN.....	30
Figure 3-7: Memcached client is responsible for sending requests to the correct servers	34
Figure 3-8: Homogenous Distributed database (142)	40
Figure 4-1: System design to secure ifoodbag cloud architecture.....	49
Figure 5-1: HTTP & HTTPS experiment setup	56
Figure 5-2: Management Node(s) execute policy to scale up or down number of nodes in Application tier.....	58
Figure 5-3: VPN Experiment scenario	58
Figure 5-4: Database experiment scenario	60
Figure 5-5: Cloud storage experiment scenario	61
Figure 6-1: Management Node(s) execute policy to scale up or down number of nodes in application tier.....	64
Figure 6-2: HTTP & HTTPS experiment setup	66
Figure 6-3: Download time comparison for HTTP and HTTPS (40bit & 128bit): (a) for a 390MB file and (b) for a 3.13MB file.....	67
Figure 6-4: VPN Experiment scenario	68
Figure 6-5: SELECT query time comparison among IPSec, OpenVPN, and normal traffic (i.e. without any security mechanism applied).....	69
Figure 6-6: SQL SELECT query time for different data sizes.....	69
Figure 6-7: 24KB file Encryption/Decryption for AES and 3DES.....	79
Figure 6-8: 4MB file Encryption/Decryption for AES and 3DES	79

List of Tables

Table 3-1: Heterogeneous versus Homogeneous DDB [145].....	40
Table 5-1: HTTP and HTTPS experimental configuration.....	57
Table 5-2: VPN experimental configuration.....	59
Table 5-3: Database experimental configuration.....	60
Table 5-4: Cloud storage experiment configuration	61
Table 6-1: HTTP and HTTPS data transfer times	67
Table 6-2: Table structure of small_database	76
Table 6-3: Table record information for small_database.....	76
Table 6-4: Data insert time with and without AES encryption for small_database.....	77
Table 6-5: SELECT data from small_database with and without AES decryption.....	77
Table 6-6: Table structure of big_database.....	77
Table 6-7: Table data record information for big_database.....	77
Table 6-8: Data insert time with and without AES encryption for big_database	77
Table 6-9: Data select time with and without AES decryption for big_database	78
Table 6-10: AES and 3DES encryption/decryption time for 4MB file.....	81
Table 6-11: Summary of security issues and recommendations.....	81

List of acronyms and abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
CIA	Confidentiality, integrity and availability
DDB	Distributed database
DDBMS	Distributed Database Management system
DoS	Denial of Service
DNS	Domain Naming System
DNSSEC	Domain Naming System Security
DES	Data Encryption Standard
FAT	File Allocation table
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IP	Internet Protocol
IPsec	Internet Protocol Security
IT	Information Technology
LRU	Least Recently Used
L2TP	Layer 2 Tunneling Protocol
NIST	(US) National Institute of Standards and Technology
PaaS	Platform as a Service
PPTP	Point-to-Point Tunneling Protocol
RR	Resource Record
RSA	Rivest-Shamir-Adleman
SASL	Simple Authentication and Security Layer
SaaS	Software as a Service
SOAP	Simple Object Access Protocol
SSTP	Secure Socket Tunneling Protocol
SSL	Secure Socket Layer
TLS	Transport Layer Security
TripleDES	Triple Data Encryption Standard
VPN	Virtual Private Network
WAMP	Windows, Apache, MySQL, and PHP
WWW	World Wide Web

1 Introduction

Ifoodbag is a company who aims to provide services to its customers through its own web based solution. Two KTH master thesis students have proposed a design to implement the ifoodbag web application in cloud environment [3]. We aim to investigate the proposed design and identify potential security issues in it.

This chapter describes the problem addressed in this thesis project. We also discuss the goal of the thesis. The motivation for this problem is stated along with a discussion of why this is an important problem to solve. Next, the scope of the thesis project is stated. The chapter ends with a description of the structure of the entire thesis.

1.1 Problem definition

According to Matt Bishop, the three *key aspects* of security are confidentiality, integrity, availability (CIA) [1]. Confidentiality is the concealment of information or resources. Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change. Availability refers to the ability to use the desired information or resource [1]. Figure 1-1 shows that without ensuring every one of these three aspects, the security of a computer system cannot be ensured.

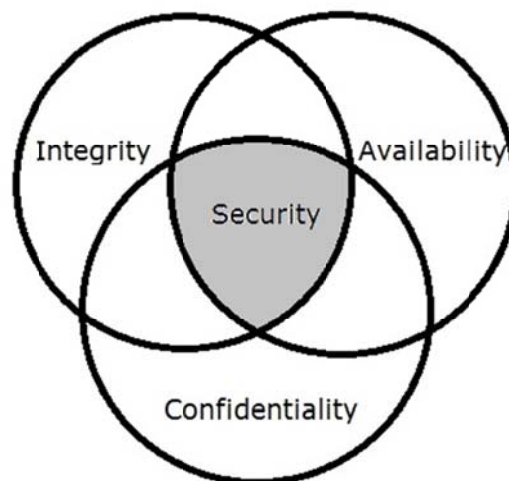


Figure 1-1: Matt Bishop's three key aspects of security Confidentiality, Integrity, and Availability (Adapted from Figure 1-3 of [2])

In this thesis project we seek to identify the countermeasures necessary to fortify the CIA of the model for a computing system, designed by two other Master's project students: Iqbal Hossain and Iqbal Hossain [3]. These two students have designed an infrastructure to provide dynamic scaling of *ifoodbag*'s Web Application running in a cloud environment^{*}.

In our thesis project, we will: (i) identifying potential threats, risk factors and vulnerable points in their proposed system and (ii) propose countermeasures within the context of network security in order to provide enhanced security for their system.

^{*} *Ifoodbag* is a Stockholm based startup offering weekly home delivery of food with personalized recipes. Further details can be found at <http://www.ifoodbag.se/>.

We expect to provide comprehensive guidelines to strengthen the security for *any* company who attempts to implement a dynamically scalable system designed to support applications such as *ifoodbag*'s web application in a cloud.

1.2 Motivation

By using cloud-based solutions, companies do not need to have their own hardware infrastructure to host their application. Thus, they eliminate the need for a large capital investment to purchase this hardware. In their service contract, they of course are paying the costs of the cloud provider purchasing and maintaining this hardware, along with some profit margin for the cloud provider. For these reasons, more and more small companies, such as *ifoodbag*, are becoming interested in using cloud-based solutions. Big companies are also getting interested about cloud based solutions in order to make their business more scalable and robust. However, there is often a lack of security when realizing such cloud-based solutions. In the longer term, the problems caused by this lack of security might inhibit companies from taking advantage of cloud-based solutions.

This thesis project aims to provide guidelines to strengthen the security of the cloud-based infrastructure that has been designed [3] for implementing the *ifoodbag* web based application in a cloud. In addition to *ifoodbag*, other companies can use these guidelines when they design and implement their own solution in a cloud in order to realize a more secure solution.

1.3 Scope

This thesis focuses on information and network security. Physical security, legal compliance, disaster recovery strategy, and risk management are **not** in the scope of this thesis. We do **not** consider what activities the application servers (virtual machines) are supposed to perform, thus the security of the *Ifoodbag* web application itself (e.g. security holes in the application program itself) are not in the scope of this thesis. This means that we will focus on the *interaction* between these servers, and client web browsers via the network. In addition, we do **not** consider the policies defined in the management node to make the system scalable, thus there could be attacks on this scaling mechanism to cause increased expense for the company by unnecessarily scaling their system up. We consider that in the proposed cloud architecture – load balancers are in a demilitarized zone (DMZ). Otherwise, all other nodes in the design reside inside *ifoodbag*'s private network; hence we do **not** focus on the security of networking devices (e.g. routers, switches, firewalls).

1.4 Method and methodology

This section presents the research approach we have taken for this thesis. We present the necessary steps and methods we have used in order to achieve our thesis goal. Qualitative and quantitative research approaches and Engineering design process methodology were adopted to achieve the thesis goal. Seyyed Khandani presents five steps to solve design problems according to the *Engineering Design Process* [4]:

- I. Define the problem
- II. Gather pertinent information
- III. Generate multiple solutions
- IV. Analyze and select a solution
- V. Test and implement the solution

So in order to follow these steps first we must define the problem. In addition to this, we have listed what we want to do in our thesis, what motivates us to do so and, the scope of our work. Next, we have studied related literature and have specified a set of requirements. Based

on these requirements we discuss some alternative mechanisms to provide a solution. We analyzed the alternative mechanisms and selected a set of mechanisms suitable for our thesis project. The choice of mechanisms is based on our own empirical observation and the observations of prior research. Finally, we present a guideline for the system implementers to follow in order to solve the defined problem.

We can divide the whole process into three different phases:

- Literature study** This phase includes study on related work and background of the topic. This study helped to understand the core of cloud computing and its various security issues and their solutions. Also in this step, we have broken down the architecture of ifoodbag's cloud based web application into seven different modules. Based on this breakdown we have analyzed the proposed design for implementing ifoodbag's web application in cloud. A deeper study was made of each module of the design to understand the basic functionality and to identify potential security issues for each module.
- Experiment** The knowledge acquired during the previous phase assisted in designing a set of experiments. This set of experiments was conducted to acquire data that could be used to select the appropriate security mechanism(s) for the proposed ifoodbag cloud architecture.
- Evaluation** In this phase, we analyze our experimental observation as well as exploited observation by other research (i.e. as an *Ex Post Facto* Study [5]). In this phase, our focus was to design a suitable set of security guidelines for ifoodbag to implement its web application in the cloud according to the proposed architecture. We also present some other recommendations to improve the overall security of the application in ifoodbag's cloud.

1.5 Structure of this document

This structure of this thesis is as follows:

- **Chapter 1** introduces the reader to the basic idea behind this thesis project. We wrote this chapter together.
- **Chapter 2** presents a detailed explanation of the scalable architecture designed for implementing *ifoodbag* web application in cloud. We also present the necessary concepts and technologies related to this thesis. Sarwarul wrote the sections defining cloud computing, characteristics of cloud computing, different implementation model of cloud computing, and an explanation of a scalable architecture. While Sabrina wrote the section on security issues regarding cloud computing.
- **Chapter 3** gives a system breakdown and analysis of the *ifoodbag* cloud architecture. In this chapter, we perform a step-by-step phase granular breakdown of the architecture proposed to implement the *ifoodbag* web application in cloud environment. We discuss the technologies mentioned in the design, the advantages and disadvantages of each of the relevant technologies, and alternative approaches. The proposed architecture is split into seven different parts based on functionality: (i) the name resolving process of the *ifoodbag* web application, (ii) distributing load among application servers, (iii) the HTTP connection between a user and *ifoodbag* application server, (iv) communication between the management node and other nodes (e.g. application server(s)), (v) caching web data, (vi) distributed database, and (vii) cloud storage. Sarwarul wrote the parts of this chapter that cover the first four parts, while Sabrina wrote the material covering the other three

parts. Sarwarul also wrote the sections concerning *ifoodbag*'s cloud network security and how to address DDoS attacks on the cloud network.

- **Chapter 4** presents our proposed security architecture for the *ifoodbag* Web application. We present the details of the technology that we select to secure the proposed cloud architecture. We describe the reasons for each choice. Both of us wrote the initial parts of Chapter 4. The remainder of this chapter presents details of our implementation (i.e. the specifics of the software used, including the version of the software used). Sarwarul wrote the details of the implementation regarding the first four parts of the design, while Sabrina wrote the details of the implementation regarding the other three parts.
- **Chapter 5** presents details of our implementation.
- **Chapter 6** presents some experimental results and our analysis of these results.
- **Chapter 7** describes how well the thesis goal was achieved, summarizes our conclusions, offers some reflections on this thesis project, and suggests some future work. The final three chapters were written jointly.

2 Background

This chapter presents a definition of cloud computing, the essential characteristics of cloud computing, different types of clouds, and some of the security issues relevant to cloud computing. The idea is to provide the reader with a clear understanding of cloud computing and its security issues. This information will be essential to the subsequent chapters and is necessary to understand the goal of this thesis project.

2.1 What is Cloud computing?

Cloud computing, often referred as just the *Cloud*, is a new buzzword in the IT world. However, the concept of cloud computing is not very new, as the concept dates back to the 1950s [6]. At that time academia, as well as industry, used terminals to connect to (often remote) mainframe computers. These terminals initially had no computing capabilities. The idea was to share resources (e.g. CPU time) of these costly mainframe computers among multiple users [6], and thus to make the use of a mainframe computer more cost effective.

Cloud computing can be considered as a service provided by a service provider. The user of this service does not need to know or worry about how the service (e.g. network, storage, application) is provided or maintained. Instead, the user is only concerned that the service is available whenever the user needs this service.

The United States of America's National Institute of Standards and Technology (NIST) [7] defines cloud computing as:

“... a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”[8]

The cloud computing approach relieves companies from needing to have their own data centers. It avoids purchase, management, and updating costs for hardware, cooling systems, storage, and power supplies. As a result, the use of the cloud computing enables companies to quickly startup a new business and to scale their business efficiently (Section 2.6 gives more discussion of how using the cloud helps business).

Figure 2-1 shows that cloud services (e.g. network, storage, application) reside inside a cloud network. Cloud users can access the various different cloud services from heterogeneous client platforms (e.g. smart phones, laptops, other computers in the same or another cloud, etc.), without knowing the exact location of the services. Additionally, the cloud service user need not know the processes to develop, manage, or maintain the services.

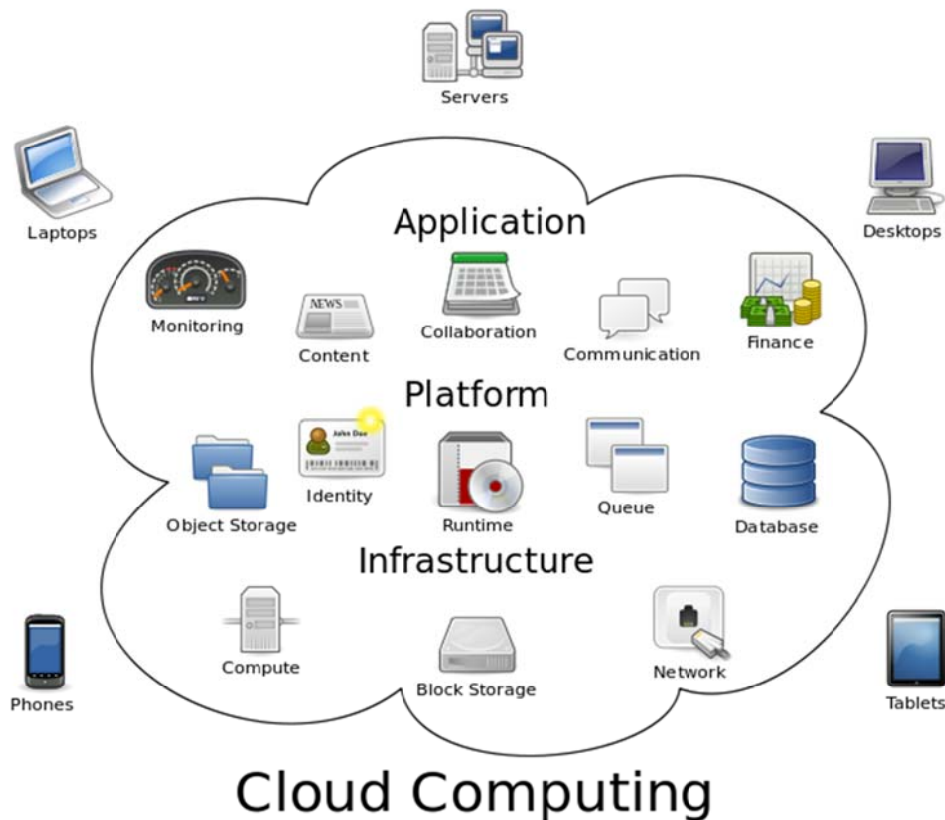


Figure 2-1: Cloud computing logical diagram [9]

2.2 Characteristics of Cloud Computing

NIST has identified "five essential characteristics" of cloud computing [8]: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. Each of these is described in more detail in the following subsections.

2.2.1 On-demand self-service

On-demand self-service enables cloud users to provision and release different cloud services (e.g. network, storage, applications) unilaterally, without interacting with the cloud service providers every time.

On-demand self-service replaces the lengthy process of provisioning and releasing services via a service provider. A cloud service provider can allow its users to handle their own services on demand. This process benefits both cloud users *and* cloud service providers. The simplified provisioning and releasing process helps the cloud users to make their business process agile. At the same time from the cloud service provider's point of view, enabling users to perform self-service reduces their workload and automates some of the tasks that would otherwise be need to be performed by the cloud service providers. This saves service providers time and money and allows them to focus on their strategy and high-valued responsibilities.

2.2.2 Broad network access

Users can be at any geographic location. If the users have network connectivity via their internet service providers and a client application (e.g. a web browser), then they will be able

to utilize cloud services. The cloud typically supports a wide range of client platforms, such as Windows, Linux, and Apple's OS X operating systems. As most smartphone browsers are supported it possible for nearly any user with Internet connectivity to utilize cloud based services.

2.2.3 Resource pooling

A cloud service provider can make their infrastructure capable of providing services simultaneously to multiple customers. To do this, the computing resources and servers are treated as a pool of resources from which multiple users can be assigned different physical and virtual resources. This is a multi-tenant model. From the pool of resources, multiple cloud users can dynamically provision and release resources according to their own needs. In such a system, the cloud user does not have knowledge or control over the exact physical location of a resource or know which specific resource is assigned to them.

2.2.4 Rapid elasticity

Cloud services can be provisioned and released in an elastic way. This means that at any moment in time cloud users can rapidly acquire additional resources or release previously acquired resource. This elasticity enables the cloud users to rapidly scale up or scale down their IT capabilities in order to match the changes in their business requirements.

2.2.5 Measured service

Usage of resources by the cloud users can be measured, controlled, and reported transparently to both the cloud service provider and the users of the services [10]. Using a metering capability cloud services providers and users can optimize their resource usage. Because the resources are being charged for as a function of usage (as measured in duration of a resource's assignment), this helps ensure maximum utilization of the resources that are managed by the cloud service provider as users will return idle resources to the pool.

2.3 Three ways to provide cloud based services

The three building blocks of cloud computing, as defined by NIST [8], are:

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a Service (IaaS)

These building blocks are also referred as cloud service models [11]. Figure 2-2 shows each of these alternative cloud service models. The following subsections further describe each of these models.

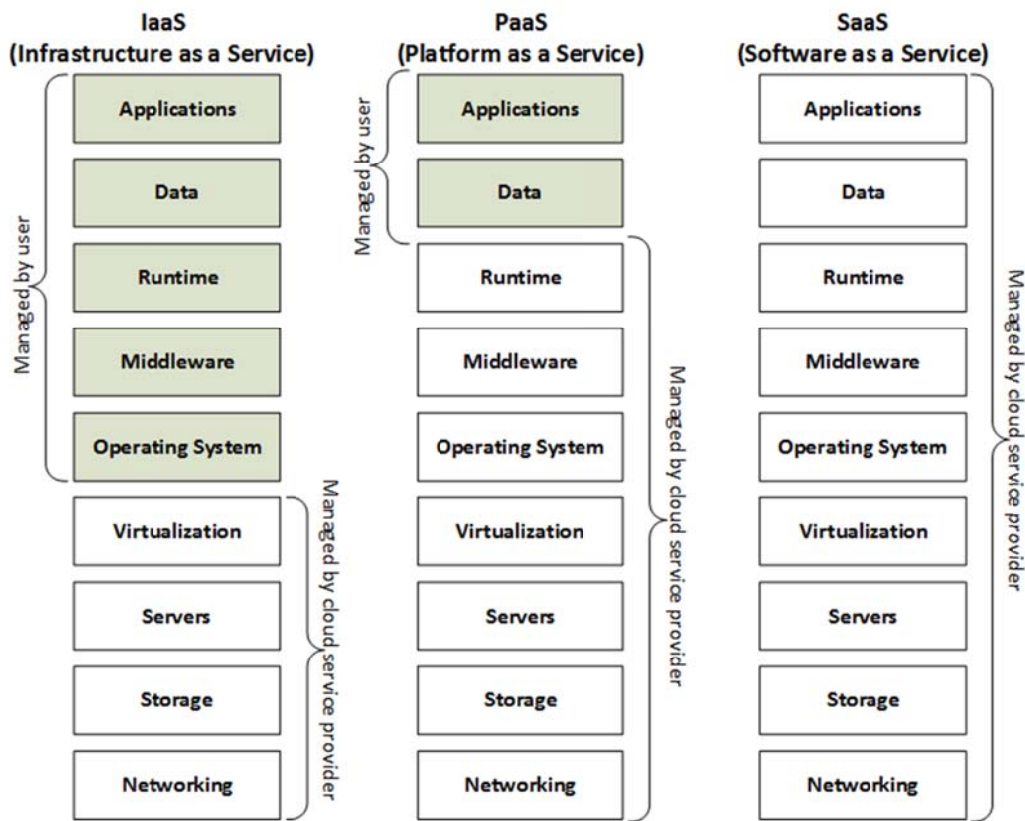


Figure 2-2: Cloud service model (Collected and edited from [12],[13])

2.3.1 Software as a Service (SaaS)

The software that is delivered as a service is typically an end user application, which is provided to the users on demand through a network connection. Users do not need to purchase or install the application; rather they simply utilize a network connection to the service provider and a web browser. SaaS makes applications available to users at any time and from anywhere. Instead of purchasing a complete product, the user simply pays per usage for the functionality that they want. The greater the use, the more the user pays. When the user stops using the application, then the user stops paying for it. The charges for usage need not be linear, thus there might be lower per unit of usage price for high volume users. Moreover, users do not need to care about installation, maintenance, and updates of this application. In this way, even a small company is able to use standard enterprise applications. Microsoft Office365 [14] is an example of SaaS.

2.3.2 Platform as a Service (PaaS)

In PaaS the cloud users get virtual servers with the necessary libraries, services, programming languages, and tools installed and configured by the cloud service provider. Users simply host their application on this readymade environment. Users do not need to manage or control the underlying infrastructure (i.e. network, servers, operating systems, storage). PaaS can be very helpful for software development organizations, as they frequently need a platform to host and test their software product. However, creating and maintaining the hosting environment can be a time consuming process and many not be cost effective. Using PaaS enables a software organization to avoid the cost and activities of supporting the development and maintenance of a testing environment.

2.3.3 Infrastructure as a Service (IaaS)

IaaS provides users with a wider variety of features than SaaS or PaaS. In IaaS a cloud service provider provides the user with storage, network connectivity, and other necessary computing resources. The user then uses these resources to set up a complete environment with their own choice of operating systems and applications according to their own needs. In IaaS the user has control over the operating system, storage, and deployed applications. However, the users do not manage or control the underlying cloud infrastructure.

2.4 Cloud Deployment Models

According to NIST cloud computing can be deployed in four different ways [8]:

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

2.4.1 Private cloud

A private cloud is dedicated to one organization. It is suitable for information that requires a high level of security [15]. A private cloud is not a new idea, as it was first described by Douglas Parkhill [16] in his 1996 book: “The Challenge of the Computer Utility”. A private cloud can be considered a dynamically provisioned datacenter, which delivers services to a certain business organization.

A private cloud may be on or off the premises of the organization it is utilized by. Based on this choice, there are two different implementations of a private cloud: (i) in-house and (ii) hosted.

Having an in-house private cloud requires one to buy, develop, maintain, and support the cloud environment within the organization’s own infrastructure. This can be costly for some organizations.

In contrast, a hosted private cloud is hosted by a service provider at his or her own site and then managed by that provider for a single customer. A hosted private cloud does not utilize a shared infrastructure. The network connection between the user and the service provider in a hosted private cloud can be over a private network connection or a tunnel over the internet.

2.4.2 Public cloud

A public cloud consists of a datacenter owned by a service provider, who manages this infrastructure. A public cloud is hosted at the service provider’s site and has the following characteristics:

- It supports multiple customers;
- Often utilizes shared infrastructure;
- Supports connectivity over the internet; and
- Is best suited for information that is not sensitive.

A public cloud can be less expensive than a private cloud [2], but the downside is that all of the data in the public cloud is beyond the organization’s firewall.

2.4.3 Hybrid cloud

A hybrid cloud is an emerging cloud deployment model and it is gradually attracting interest. According to a survey authored by North Bridge Venture Partners [17], within the next five years hybrid clouds will be the emphasis of 52 percent of the respondents’ cloud strategies, while the current figure is 36 percent [18]. According to another survey by

Coleman Parkes Research, 69% of organizations in Asia Pacific and Japan intended to adopt a hybrid cloud delivery model [19].

A hybrid cloud infrastructure is a composition of two or more distinct cloud infrastructures. The aims of a hybrid cloud are to provide the most appropriate solution to an organization, by combining the advantages of both the public and private cloud approaches. An organization can move some data to the public cloud, especially that data that is not so business critical. However, the organization can keep their business critical data, which makes the company unique, in an in-house or hosted private cloud.

Some companies have a conservative approach to new technology and consequently they are reluctant to shift to the cloud. A hybrid cloud offers them a good opportunity to make a slow start at a low cost by shifting a small part of their data and computing infrastructure into a cloud.

2.4.4 Community cloud

A community cloud is shared by a group of organizations, who share common computing concerns [20], such as application performance requirements. For example, consider an application whose different modules are developed, managed, and supported by different organizations. In order to facilitate the integration of the different modules of this application and to do so quickly, the separate organizations could deploy their modules into a shared cloud, i.e., a community cloud.

A community cloud can reside on premise or off-premise. The hosting organizations or third party cloud service providers can manage this community cloud. A community cloud provides its users with the flavor of a public cloud, while offering the security and privacy features of private cloud [20].

2.5 Scalable model of the *ifoogbag* web application in cloud

Another Master's thesis project by Iqbal Hossain and Iqbal Hossain has designed a dynamically scalable model for implementing the *ifoodbag* web application in cloud [3]. Figure 2-3 shows their proposed system model, which utilizes an IaaS model. This is generic model can be implemented as a private, public, or even hybrid cloud.

The complete model consists of four major tiers: load balancing, application, caching, and database. A brief explanation of these four tiers is:

Load balancing tier: In this tier a SQUID [21] proxy server is used as load balancer. After resolving names to IP address using a DNS server, packets sent by the end user will first reach the load balancer. The task of these load balancers is to distribute incoming requests over the available application servers.

Application tier: In this tier, there are a number of application servers. The number of active application servers will change based on a policy configured in the management node/nodes. The idea is to scale up or down the application tier as necessary to meet the organization's performance goals.

Caching tier: The caching tier consists of a distributed object caching system based upon *Memcached* [22]. The purpose of this tier is to speed up the *ifoodbag* web application servers by reducing the unnecessary load on the database tier.

Database tier: A distributed database realizes this tier. There are multiple slave databases, which replicate a master database. A snapshot backup of database will be stored in cloud storage. In the event of a failure of the database tier, this snapshot of the database provides service continuity.

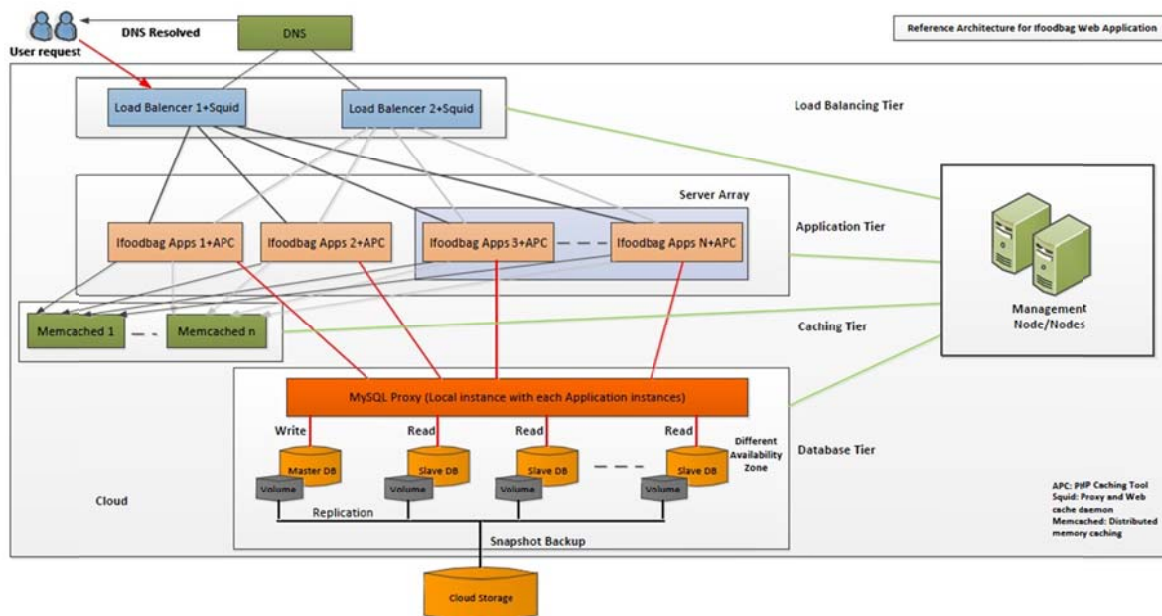


Figure 2-3: A dynamically scalable model for *ifoodbag* in cloud [3] (Appears here with the permission of the authors.)

2.6 How can using a cloud help a business?

It is a challenge for organizations to make the best use of their IT infrastructure. Cloud computing can help organizations to do so. Additionally, cloud computing can help organizations scale up or down their IT capabilities according to changes in their business requirements.

To understand how cloud computing can help an organization to optimize its IT capabilities; we can consider a small, newly established company: *ifoodbag*. The aim of this company is to provide services to its customers through a web based application developed by its own developers. The firm is not ready to purchase or maintain the hardware necessary to host its web application.

In this situation, the use of an IaaS model cloud enables *ifoodbag* to host its web-based solution in a cloud service provider's virtual machines. Using this cloud, *ifoodbag* pays only for their actual amount of use. This means that *ifoodbag* spends only for the resources that they use, which optimizes their business development costs. As a result *ifoodbag* benefits from their use of cloud computing. Furthermore, if they are successful they will need to scale up their services to meet the demands of their customers. Utilizing a cloud based solution means that they can grow exactly at the rate at which they need to grow, avoiding the risk of over or under dimensioning their service infrastructure. This scaling should help them to retain customers due to always being able to provide a consistent level of performance.

Cloud can also help large organizations. Consider an imaginary organization XYZ AB, which produces consumer products. XYZ AB has a research department where the researchers are distributed over multiple geographic locations. Collaboration among these researchers is a problem for XYZ AB. The company is also looking for a cost-effective storage and computing solution, as occasionally the research department needs additional resources (e.g. storage, memory, and processing) to store, process, and analyze a large amount of statistical data.

A cloud could provide a cost-effective solution to XYZ AB. When the company requires additional computing and/or storage, they simply request more from their cloud provider, on a

pay per use basis. Thus, they are able to move all of their statistical data analysis and data storage into the cloud provided by their cloud service provider. In this way XYZ AB can reduce their costs by avoiding the need to maintain additional resources (which are only occasionally used), and hence they can invest more in research.

For efficient collaboration among researchers distributed over different geographic locations, XYZ AB can use cloud based online collaborative solutions (e.g. Microsoft SharePoint Online) [23]. With all the benefits of efficient collaboration [24], researchers are expected to achieve better research results. These results will ultimately help XYZ AB to improve its products and/or services.

In summary, cloud computing can help both large and small organizations to optimize their business costs and to improve their products and/or services.

2.7 Security issues in cloud

Cloud computing is a growing area of concern in the IT security community because cloud architectures are literally popping up all over. Public clouds are available from Google[25], Amazon[26], Microsoft[27], Oracle[28], Eucalyptus[29], and many other vendors. This section is concerned with discovering the vulnerabilities of cloud computing and finding appropriate security solutions. This section will also discuss what early cloud adopters and developers have done as they became more concerned with security. While there is no “ultimate security” solution, security experts will try to minimize the potential for security threats as much as possible. Although they have tried to minimize security risk as much as possible, cloud computing still possesses many security risks. Some of these security risks are well known and some of them are new.

As described in section 1.1 confidentiality, integrity, and availability (CIA) are the three key aspects of security. Ensuring confidentiality means that no one can read our data unless we want them to read it, integrity ensures that no one can modify our data without the modifications being detected, and availability means that we can access our data at any time. Cloud computing also needs to deal with security risk/threats just as any other service. These attacks can subvert one or more of the three key aspects of security.

In this section, we will discuss a number of potential security attacks on cloud, especially: denial of service (DoS) attacks, authentication attacks, man-in-the middle, wrapping attacks, malware-injection attacks, flooding attacks, and browser attacks. We will also discuss accountability checking problems. We will first categorize these attacks based on the CIA model, and then we will discuss the root causes of these attacks and possible solutions.

2.7.1 Confidentiality related attacks

The first category of attacks that we will consider is attacks on confidentiality. Loss of confidentiality for a web-based service can destroy the trust which *ifoodbag* customers place in the company and could lead to financial losses for both the company and the customers. For these and other reasons, it is essential that the system preserve confidentiality. In the case of a cloud-based service, the attacks we consider are malware injection and data stealing.

2.7.1.1 Malware Injection Attack

The use of a malware injection attack method is spreading very rapidly and many websites have been affected. The objective of the attack can be to spread malware to anyone who utilizes the web server or to place malware into the web server in a direct attack on the service.

2.7.1.1.1 Examples of malware injection attacks

Normally a malware injection attack is done via a compromised FTP server. A virus attempts to sniff FTP passwords and sends these passwords (and the user name) back to the attacker. The attacker then uses this FTP user name and password to access the website in order to add malicious iframe coding to the site's web page. These web pages are used to infect visitors who browse to this website.[30]

In a cloud-based system, a web client's request is executed based on authentication and authorization. During this authorization and authentication process a large amount of metadata is exchanged between the web server and web browser. An attacker can take advantage of this metadata. In another form of malware injection attack, an adversary attempts to inject malicious service or code[31]. In this case, the injected malicious service or code appears as a valid instance of services running in the cloud. If the attacker is successful, then the cloud service will be vulnerable to eavesdropping and deadlocks, the later forces a legitimate user to wait until the completion of a job, which was not generated by the user. This type of attack is also known as a meta-data spoofing attack [32].

2.7.1.1.2 Methods of protect against malware Injection attack

When a cloud customer opens an account in the cloud, the cloud provider creates an image of the customer's virtual machine (VM) in the image repository system of the cloud. Some researchers have suggested that one should exploit the integrity protection offered at the hardware level, because if the hardware implements the trusted computing model then it is very difficult for an attacker to intrude at the IaaS level[31]. Use of the trusted computing model prevents unsigned code from being executed by the processor.

One of the approaches to detecting changes in files on the web server suggested by many experts is to exploit the File Allocation Table (FAT) system architecture. This technique is straightforward and its supported by virtually all existing operating systems [33]. In this approach you exploit knowledge of the code (of the OS and applications) that a customer is going to run based upon information from the FAT. This information can be compared with the previous instances of applications that have already been executed by the customer's VM to determine the validity and integrity of the new instance[31].

Another approach is to store the OS type of the customer when the customer first opens an account. Since the IaaS cloud is completely OS platform independent, it is possible to check if the new instance of the VM that is to be run is the same type of OS before launching a VM instance in the cloud[34].

2.7.1.2 Data Stealing

Data stealing is one of the most common approaches to breach a user account. Often the user account and password are stolen. As a result, stealing and destroying of confidential data can hamper the storage integrity and security of the cloud. The providers face the first strike of such kind of problem[31].

To protect against data stealing the customer will receive an e-mail about the resource usage and duration of the session at the end of each session. A special number (which acts as a numeric challenge) is sent in the same email. This number is used during the next login. By doing this, the customer will be aware of their usage & charges and due to the need to input a new numeric challenge every time they access the system it will be possible to detect if someone else has used the account in the meantime[35]. Note that if an attacker is able to get a copy of the e-mail, then they have access to the numeric challenge and if they also have the customer's account name and password, they can login. If the attacker blocks the new e-mail message sent at the end of this new session, then the customer will not have the correct

challenge for their next login and hence will detect the attacker's usage of their account. While if the attacker does not block the e-mail from their session, then the customer will be informed about the attacker's usage of their account.

2.7.2 Integrity related attacks

We will only consider one type of integrity related attack: a XML signature wrapping attack (also known as a XML rewriting attack). Wrapping attacks aim to inject a fake element into a message structure so that the message seems to have a valid signature, as a result the malicious element will be processed by the application logic. Using this method an attacker can make an arbitrary web service request while the request is authenticated as coming from a legitimate user [36].

When a user makes a SOAP web service request of the web server running on a VM through a browser, the user signs the security header of this SOAP message. An attacker captures this message as it passes the browser to the server and replaces the contents of the message with their malicious payload, but copies the signature [35]. Unfortunately, unless the message body was also signed – the web service will see a request that appears to be legitimate and will execute it. Using this method it is easy for the adversary to run malicious code in the cloud and interrupt the normal functioning of the cloud server [31].

To increase the security during the message passing between the web browser and web server when using a SOAP message, a timestamp can be added to the SOAP header. This timestamp is added in order to protect against an adversary who can intrude in the TLS layer.

2.7.3 Accountability Check

Because the customer will be charged based on their usage of resources, one attack is to simply use lots of resources (for example to store and distribute malicious content or use lots of cycles running malicious code) as if you were the legitimate user and generate a high bill for the customer. Since the customer will not be aware of such an attack until the provider charges the customer, the customer can be left with a very large bill. This can lead to various problems since the provider believes that the customer used these resources, while the customer believes that the provider is charging them for resources that they did not use[31].

Several methods can be used to protect against an accountability problem. One approach is for the provider to: (1) check the identity of the user before launching any instance of a customer's VM, (2) securely record resource usage records, (3) perform auditing of all such records, and (4) collect sufficient evidence concerning the usage in order to resolve potential future accounting disputes[35]. Note that the audit should be carried out by a neutral third party and have the following properties: (1) completeness, (2) accuracy, and (3) verifiability[35].

2.7.4 Availability related attacks

With respect to cloud computing there are two main availability related attacks: denial of service (DoS) and flooding. We will describe each of these in the paragraphs below

2.7.4.1 DoS attacks

In a typical DoS attack, a malicious party floods the machine or network with traffic, causing the service to respond slowly even to make the service inaccessible to its legitimate users. Another means of accomplishing this attack is to cause the service crash[37].

Some security experts have argued that Cloud computing is more exposed to DoS attack[38]. Since cloud computing resources are shared by many users this approach is more vulnerable to a DoS attack and such an attack can cause much more damage[31]. When the operating system of cloud computing node detects the high workload on the flooded service

then it requests more computational power (more virtual machines) in order to cope with additional workload. As each server has limited capacity and it takes time to allocate additional resources, the legitimate users will experience a negative impact on the service's availability. Additionally, the customer will be charged for these additional resources, hence inflicting a financial penalty on this customer. Note that the attacker need not flood all of the servers that provide a certain service, but might flood only a single server, in order to reduce the availability on the targeted service [39].

Consider the following two real-world incidents:

1. A Georgian blogger with multiple accounts on Twitter, Facebook, Live Journal, Googles Blogger, and Youtube was the target of a DoS that took down Twitter's entire site for several hours and slowed down this whole service[40].
2. During October 2009, Amazon's cloud customer Bitbucket experienced a 19-hour outage during a distributed DoS attack[41]. According to one of Bitbucket's operators, the company was attacked with a "flood of UDP [user datagram protocol] packets coming into our IP [internet protocol], basically eating away all bandwidth;" the attack introduced increase latency in delivering documents stored in Bitbucket's elastic block storage[EBS][41].

There are some counter measures against DoS attacks against a cloud, these countermeasures utilize several techniques: authentication, authorization, filtering, throttling and QoS (Quality of service)[42].

According to some security experts, one of the most popular countermeasures to protect against a DoS attack is to use an intrusion detection system (IDS). IDS will be loaded into each cloud server and these IDS systems will exchange information. When a specific cloud server is under attack the IDS alerts the whole set of IDSs. In this way, a DoS attack can be detected and if appropriate actions are taken the negative impact of this attack can be prevented [35].

Another recommendation is to ensure that the cloud provider restricts dynamic utilization of resources to set specific levels in order to counter internal DoS attacks[38]. The service level agreement (SLA) between the cloud provider and the customer should stipulate that the provider cloud provider should identify all DoS or DDoS attack methods, and have established measures (which are audited and verified) to mitigate such attacks[38].

2.7.4.2 Flooding attack:

A flooding attack attempts to cause a failure in a computer system or other data processing entity by providing more input than the entity can process properly[43]. All the computational servers in a cloud system work in a service specific manner and there may be internal (to the cloud) communication between these nodes. Whenever a server is overloaded or the server reaches its limits, then the server needs to transfer some of its load to a another server offering the same specific service [35]. By sharing with another server, the overloaded server offloads itself. In order to make the cloud more efficient and execute requests faster, this sharing approach is widely used to distribute load over a set of servers all providing the same service.

In a flooding attack the adversary creates bogus requests to the cloud service. As the server will first check the authenticity of the request before processing the request, the attacker's illegitimate requests must be checked to determine their authenticity, thus consuming CPU, memory, and network resources. Legitimate service request can starve waiting while server is busy processing the bogus request. As a result, the server will offload some of its load to another server. If the adversary is successful in engaging the whole cloud's

resource they can effectively compromise the availability of the all of the services that are running on this cloud[31].

A flooding attack can be prevented by organizing all the servers in the cloud system as groups of servers. Each group of servers is designated for a specific type of job. Each group of servers is utilized for a specific type of service. In this approach, all the servers within a group will communicate among themselves through message passing. When a specific server is overloaded, a new server will be deployed in the group and the name server, which has a complete record of the current states of all of the servers, will update the set of destinations for the specific requests to include the newly included server[31].

The hypervisor [44]that supports the VMs can use introspection to check if any unauthorized code is disrupting the usual computation. The hypervisor can also perform scheduling over servers in a group. In this way the flooding attack can be mitigated to some extent (if the hypervisor is locally breached, then further analysis and efforts will be required to secure the hypervisor.)[35].

3 System breakdown and analysis

We split the proposed architecture for implementing the *ifoodbag* web application in a cloud, into seven different parts based on functionality: (1) communication between *ifoodbag* user and DNS server, (2) distributing load among application servers, (3) HTTP connection between user and *ifoodbag* application server, (4) communication between the management node and other nodes (e.g. application server), (5) caching web data, (6) distributed database, and (7) cloud storage. Below we discuss all of these parts one by one. We also discuss about *ifoodbag*'s cloud network security.

3.1 Name resolving process of ifoodbag web application

An *ifoodbag* user uses a web browser to access the web application. The user enters the URL of the application (e.g. <https://www.ifoodbag.com>) in their browser. The browser queries a domain name system (DNS) server to resolve the host name into an Internet Protocol (IP) address. This section discusses DNS, how it works, security issues regarding DNS, and available solutions to improve DNS security.

3.1.1 What is the job of a DNS server?

DNS is an application layer protocol. This means that DNS works in the application layer of the TCP/IP protocol suite. The main function of DNS is to map user-friendly hostnames into IP addresses. DNS enables users to indicate the network interface to a computer (e.g. personal computer, server) by names, instead of needing to know the interface's IP addresses.

Figure 3-1 shows, in a very simplified form, how a host (Host A) can learn the IP address of the network interface of another host (Host B) with the help of DNS. Host A asks the DNS server for the IP-address of Host B. The DNS server provides Host A with the IP address of an interface to Host B. Given this IP address, Host A can now send IP packets to Host B. Further details of DNS can be found in [45].

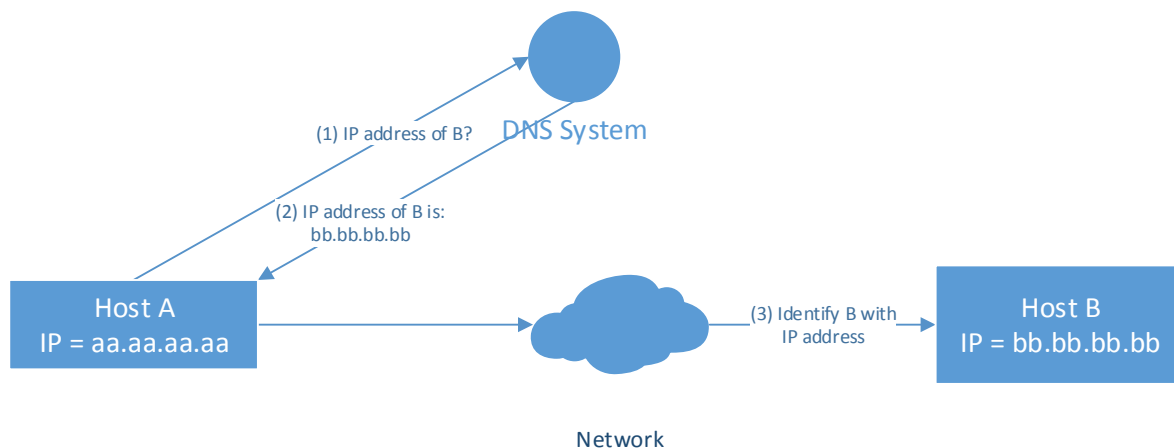


Figure 3-1: Host A using DNS to identify Host B in Internet (The idea for this figure is based upon [46].)

3.1.2 How DNS works

Every interface of each host in a network needs to be configured with an appropriate IP address, in order to communicate with other hosts in the, same or another, network. However, it is inconvenient for human being to remember the IP addresses of all the hosts they want to communicate with. Since human beings are good at remembering names it is desirable to utilize a simple user-friendly scheme for hostnames. A directory is used to keep track of

different hosts and their associated IP address. DNS provides a distributed hierarchical caching directory service.

Figure 3-2 depicts the hierarchical tree structure of DNS. In this tree, a *hostname* (e.g. www.ifoodbag.com) is a leaf, while the nodes above it in the hierarchy (e.g. .com, ifoodbag.com) form a *domain*. The DNS server that is responsible for hostnames and addresses within a certain domain is an *authoritative name server* for that domain.

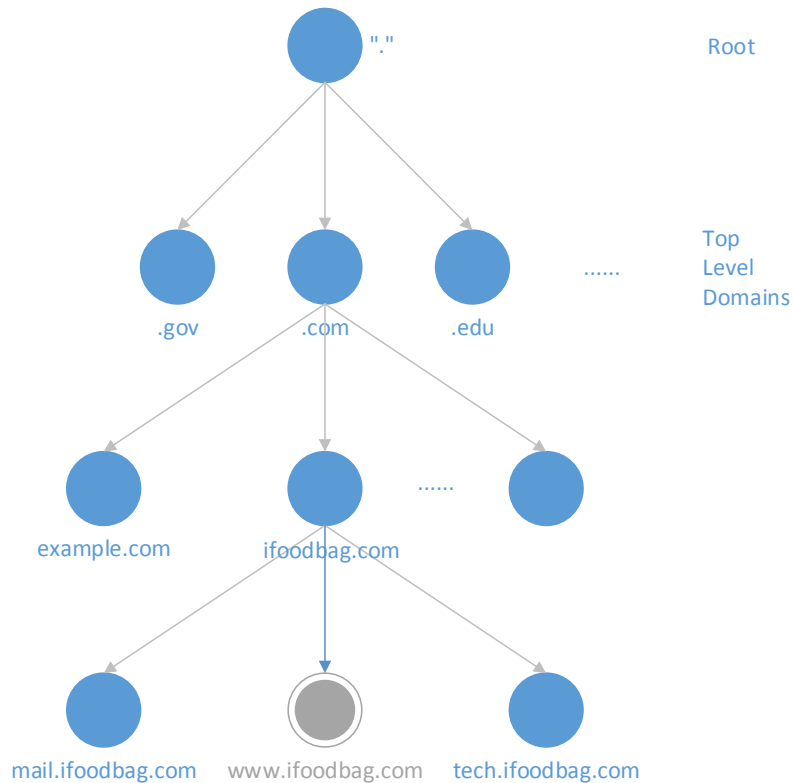


Figure 3-2: DNS hierarchical tree structure

When a host asks for the IP address corresponding to a certain hostname, a search logically starts from the top of the tree (i.e. the root of the DNS tree)*. The root DNS servers keep information about the name servers responsible for top-level domains (TLDs) (e.g. .com, .org). The authoritative name servers for top-level domains maintain information about the name server that is responsible for next level domains (e.g. ifoodbag.com). The authoritative name server of a domain is responsible for the hostname to (and from) IP address mapping information for the hostnames (e.g. www.ifoodbag.com, mail.ifoodbag.com) of that particular domain. These leaf DNS server also cache hostname-IP mapping information for hosts in other domains. It is this caching that leads to both DNS's good performance and to a number of attacks on DNS servers.

3.1.3 Iterative and Recursive Queries

A DNS query can be iterative and recursive. In the case of a recursive query, after receiving a request for resolution of a hostname (i.e., the request for an IP address associated with this hostname), a single DNS server continues the lookup process until it successfully

* Note that we say here that the lookup starts from the root. However, in practice this is not a common case since most internet service providers provide their own DNS servers that their customers generally use. For this reason, most customers' computers will contact one of their ISP's name servers for name resolution. For this reason the actual DNS performance is largely dependent upon these DNS servers and the root servers are only infrequently queried by an end user's computer.

resolves the name or the lookup process fails. After resolving the IP address for the hostname that was in the DNS query, the DNS server returns this IP address (or addresses) in a response to the host that made the request. If the lookup fails, i.e., the DNS server cannot resolve the name, then the DNS server reports this failure to the host that made the request. In summary, a recursive DNS server performs the hostname resolution process on behalf of the host that made the request.

When a host queries a DNS server to resolve a hostname in iterative way, the DNS server simply checks if it has the requested information. If it does have this information, then it provides it. Otherwise it refers the host that send the query to another DNS server (i.e. an authoritative DNS server at a lower level of the DNS tree structure), which may have the information.

Usually hosts perform recursive queries, so that the DNS server performs the complete name resolving process. In contrast, iterative queries are used by a DNS server to communicate with other DNS servers (e.g. a root DNS server or an authoritative DNS server for TLDs) in order to resolve the query. Each DNS query specifies whether an iterative or recursive lookup should be performed.

3.1.4 DNS name resolving process

Figure 3-3 presents a DNS name resolving process that consists of recursive and iterative queries. The following steps are performed in this process:

1. A DNS client, also called a resolver^{*}, sends recursive query to a name server to request an IP address (or several IP addresses) for a given hostname (in this case “www.ifofoodbag.com”).
2. The name server checks its cache. If it has the requested information, it immediately replies to the resolver. However, if this server does not have this information, then it sends iterative queries to a root DNS server it knows about.
3. The root DNS server does not know the IP address for www.ifofoodbag.com, but it knows the authoritative name server for top-level domains (TLDs). These top-level DNS servers know an authoritative name server for the .com domain. The server replies to the recursive name server with an IP address of the .com domain’s authoritative name server.
4. The recursive name server communicates directly with the .com domain’s authoritative name server.
5. This server knows the authoritative name server for the *ifofoodbag.com* domain. So it replies to the recursive name server with this authoritative name server’s IP address.
6. Now the recursive name server contacts the authoritative name server for the *ifofoodbag.com* domain, which knows the mappings for hostnames in this domain.
7. The authoritative name server for the *ifofoodbag.com* domain replies with the IP address (or addresses) of the web server.
8. At this point, the recursive name resolving process is finished. Now the name server replies to the resolver by telling it the IP address (or addresses) associated with the hostname (i.e. www.ifofoodbag.com) in its request. The resolver now caches this information for a certain period of time. The DNS response indicates how long this answer is expected to be valid; hence, the information can be cached up to this point in time – after which the information should be removed from the cache.

^{*} A resolver is a program that resolves hostnames to IP addresses (or the reverse) by communicating with appropriate name servers.

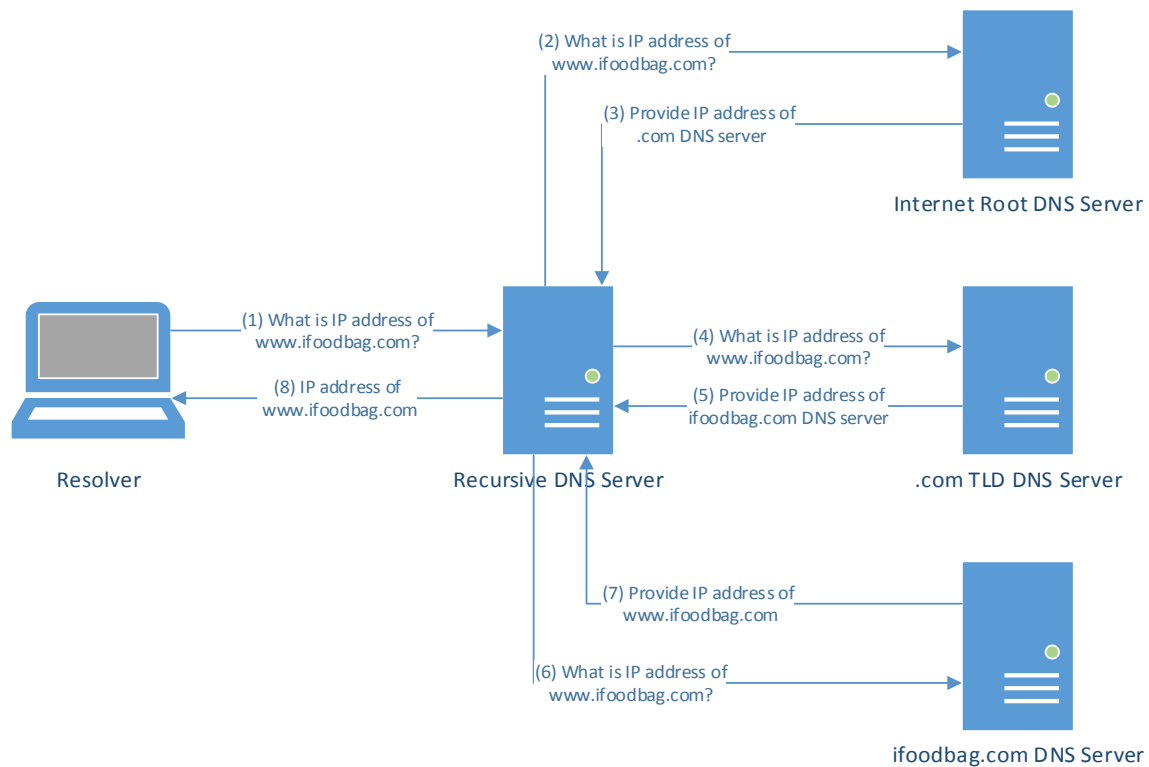


Figure 3-3: Recursive and iterative DNS queries

3.1.5 Importance of DNS security

DNS was designed when the internet was quite young. During this early period of time security was not a major concern, as internet access was limited to defense organizations, governmental agencies, and academia. This closed environment was reasonably secure, as only trusted entities were given access to the network, therefore security issues were frequently **not** considered when designing early internet protocols, including DNS.

In contrast, today the internet is used worldwide. A highly diverse population is using internet-based services in their daily life. With this wider range of usage, security risks have also increased. Security has become a major concern in the contemporary internet world. As a result it has become necessary to design security measures to replace (or supplement) the earlier fundamental internet protocols, which generally did not include any security mechanisms.

In addition to IP, UDP, and TCP; DNS is one of the most widely used protocols in the internet. Due to the absence of security measures in DNS itself, it is vulnerable to attacks. Criminals and malicious users have learned to create DNS DoS [47], to perform IP spoofing to redirect DNS queries to return answers containing the IP addresses of their own servers (which can contain malware), and to introduce many other hazards for internet users. The next subsection discusses these security issues.

3.1.6 DNS security issues and vulnerabilities

DNS security issues can be categorized into DNS server attacks (where the attacks are aimed directly at the DNS server) and DNS protocol attacks. DNS protocol attacks exploit vulnerabilities in the DNS protocol. Both of these types of attacks are discussed in detail in the following paragraphs.

3.1.6.1 DNS server attacks

An application that provides DNS service is commonly called DNS software or name server software. Bind is the most widely used name server software [48]. A server that runs this name server software is called a DNS server or a *name server*. An attacker can directly target a name server. Such an attack can be of the following two types [46]:

1. **Manipulating vulnerabilities in DNS software:** Name server may have vulnerabilities, bugs, and flaws. Example of vulnerabilities in the BIND name server software can be found in [49]. Attackers seek to exploit these bugs and flaws in name server software. Additionally, there might be other services running on the name server, so an attacker may take advantage of the bugs in those services also while attacking the name server.

Most of the bugs and flaws found in name server software are fixed as quickly as possible, by releasing updates and patches. Name server administrators can keep their system up to date by installing these updates, and thus protect their server from known threats.

2. **DoS attack:** A name server always attempts to reply to the queries it receives, without authenticating the queries. An attacker can take advantage of this to perform distributed denial of service (DDoS) attack on a name server. Performing a DoS attack causes the name server to be unavailable for a period of time. During this time the attacker can use a compromised name server to reply to name resolving queries, in place of the original DNS server. The compromised name server can provide false responses to these queries (for example, to redirect the users to the attacker's own site). Detail on DDoS DNS attack are given in [47].

3.1.6.2 DNS protocol attacks

The DNS protocol possesses several vulnerabilities, as it does not contain any security measures. Attackers can target these vulnerabilities in the DNS protocol and in implementations of the protocol in order to perform attacks. Some of these vulnerabilities are discussed below:

3.1.6.2.1 Zone File Compromise

A DNS zone file contains records (i.e. name and IP-address mapping information) for domain names and subdomains in a name server. An attacker with (local or remote) access to a name server can modify these zone files and change these records.

Security measures to prevent this type of attack include restricting access to the name server to only authorized people, monitoring access to name server, physically securing the computer, and limiting remote access to name server to specific authorized machines [50].

3.1.6.2.2 Zone Information Leakage

The zone file contains information about the network resources (e.g. servers, PCs, and printers) in a certain domain. Reading a zone file an attacker might learn the list of important servers and hence target those servers in an attack.

Sometimes, due to convenience, firewall and network access control policies are imposed on a block of IP address, instead of individual IP address. In this case, it is helpful for an attacker to learn the IP address allocation scheme used in a network and to know if there are any unallocated IP addresses. If there is an unallocated IP address, then the attacker can setup a false name server in the target network with one of these unallocated IP addresses. All of these tasks are easier if the attacker can read the DNS zone file since the zone file contains information about the IP addresses associated with all of the hostnames in a network.

Information from the zone file can leak if zone transfer is enabled for all machines in name server configurations. In this case, the attacker can get the entire zone file via the zone transfer process^{*}. Details on information leakage through DNS can be found in [52].

3.1.6.2.3 Cache poisoning

During the name resolving process a name server queries other name servers (e.g. root DNS, DNS responsible for TLDs) to get information, after resolving a name, the name server caches the information (i.e. the mapping) for a period of time. The next time the name server receives a query to resolve the same name it retrieves the information from its cache (if it is available) rather than performing the lookup process.

The goal of this caching mechanism is to improve the performance of the overall DNS infrastructure, but it also creates a security hole in the name resolving system. Attacks exploiting this security hole are known as *DNS cache poisoning*.

The main idea behind a DNS cache poisoning attack is to pollute a name server's cache with false information. This attack is possible because a DNS server never checks the validity of a response it receives from another name server during iterative queries nor does the name server check if the received information is relevant to its iterative query. Attackers exploit this failure to validate responses by sending false information (e.g. another IP address) to the name server querying for a certain name. After receiving this false information the name server caches it and will now provide this false information from cache for all subsequent queries for the same name (until the cache entry expires).

Usually attackers use a mechanism called *DNS spoofing* in order to perform *cache poisoning*. DNS spoofing is a process where the attacker configures a name server to reply to DNS queries actually sent to another (real) name server. The response packet sent by the attacker's name server contains the real name server's IP address in the source-address field, as the receiving name server checks the source IP address of the response packet. However, this check is not sufficient. The name server uses identity numbers to match queries and responses. So learn the identifier number of a query, the attacker performs *DNS ID hacking*. A detailed explanation of DNS ID hacking can be found in [53] and [54]. In the absence of knowing the correct identification number the attacker can carry out a brute force attack by sending all possible identification numbers[†].

3.1.7 Available solutions

Implementing the DNS Security Extension (DNSSEC) [55] can solve most of the problems of DNS mentioned in the sections above. Details of DNSSEC are explained in the next section.

Due to backward compatibility issues of DNSSEC with the existing DNS implementations, it will take time to implement DNSSEC worldwide. Until that time, name server administrators can take some measures to improve the security of DNS. Measures to improve security and protect against attacks on a name server have been extensively discussed in [56]. Here we list some of these measures:

1. Bugs and security flaws found in name server software that have been corrected are distributed as patch releases or updates. Name server administrators should keep their system up to date by installing newly released patches and updates.
2. Forbid recursive queries to prevent spoofing.

^{*} The term zone transfer refers to a process used to copy a DNS zone file from a primary DNS server to a secondary DNS server [51].

[†] Note that since this number is carried in a 16-bit field, hence there are only 2^{16} possible values.

3. Do not configure all DNS servers on the same subnet, in order to avoid a single point failure*. If possible, do not place these servers behind the same router or connected to the same leased line.
4. Restrict zone transfer from slave name servers, thus zone transfers should only be provided by the master name server. Scott Rose and Anastase Nakassis [58] discusses some measures to minimize DNS zone information leakage.
5. Configure different name servers for internal and external hosts. In that case if the external DNS is hacked, the internal DNS server will keep providing service to the internal hosts.

3.1.8 DNS Security Extension (DNSSEC)

DNSSEC [55] does not encrypt DNS data, so it does **not** provide any *confidentiality*, rather it signs the DNS data using asymmetric cryptography (i.e. public-private key based cryptography). A private key is used to digitally sign a DNS resource record in a zone or domain, and the associated public key is published with this signed information. The public key is used to validate the signatures on the data. The use of asymmetric cryptography provides *data integrity* and *origin authentication* in the DNS name resolving process.

3.1.8.1 How does DNSSEC work?

DNSSEC uses a digital signature [59] and asymmetric cryptographic keys to check the authenticity of the DNS responses [60]. Records in a zone file are individually signed, with a DNSSEC private key, so that individual records can be added, modified, or deleted in the zone file without resigning the entire zone file. Signatures generated for the records in a zone file reside inside the zone file itself, as new resource records called Resource Record Signature (RRSIG) records. The corresponding RRSIG record is returned along with the response to a name resolution query. To validate if the response is authentic; the associated RRSIG record is verified using a public key called the DNSKEY [60]. The operation of DNSSEC is illustrated in Figure 3-4 and consists of the following operations:

1. After receiving a request from a resolver to provide the IP address of a name (for example `www.ifoodbag.com`), a recursive DNS server follows the process previously explained in section 3.1.4 and the resolver will receive an IP address from the authoritative DNS server for the `ifoodbag.com` domain.
2. The recursive DNS server sent not only the IP address of `www.ifoodbag.com`, but also sent the digital signature of the record along with the associated public key. The recursive DNS server verifies the digital signature using the public key and to ensure that no one has modified the resource records during transmission. Thus, the integrity of the data is checked. The recursive DNS server also checks that the digital signature is still valid, whether the current date is between the start date and end date of the specified validity period.
3. However, if this was all then anyone could sign zone data with a private key and provide the associated public key! Therefore, in order to perform origin authentication, the recursive DNS server asks the DNS server responsible for `.com` domain if the public key it has received for the `ifoodbag.com` domain is correct.
4. The DNS server responsible for the `.com` domain checks if its delegation signer (DS) [61] information indicates that the public key has been provided by the `ifoodbag.com` provider. The DNS server then provides the signed DS record along with its public key to the recursive DNS server that sent the query.

* Note that at least one secondary name server is supposed to be operated by another party that is not in the same geographic area, not powered by the same source or electrical supply, and not connected to the same part of the network as the primary server. See more in [57].

5. Now the recursive DNS server queries the root DNS server in the same way in order to verify the public key received in the previous step for the .com domain.
6. The root DNS server replies with a signed DS record along with the public key. The recursive DNS server is manually configured with the public key of the root DNS server as a trusted key. Using this public key the recursive DNS server can verify the public key received from the root DNS server.
7. Once all of these checks are complete, the recursive DNS server provides the resolved IP address of www.ifofoodbag.com to the resolver.

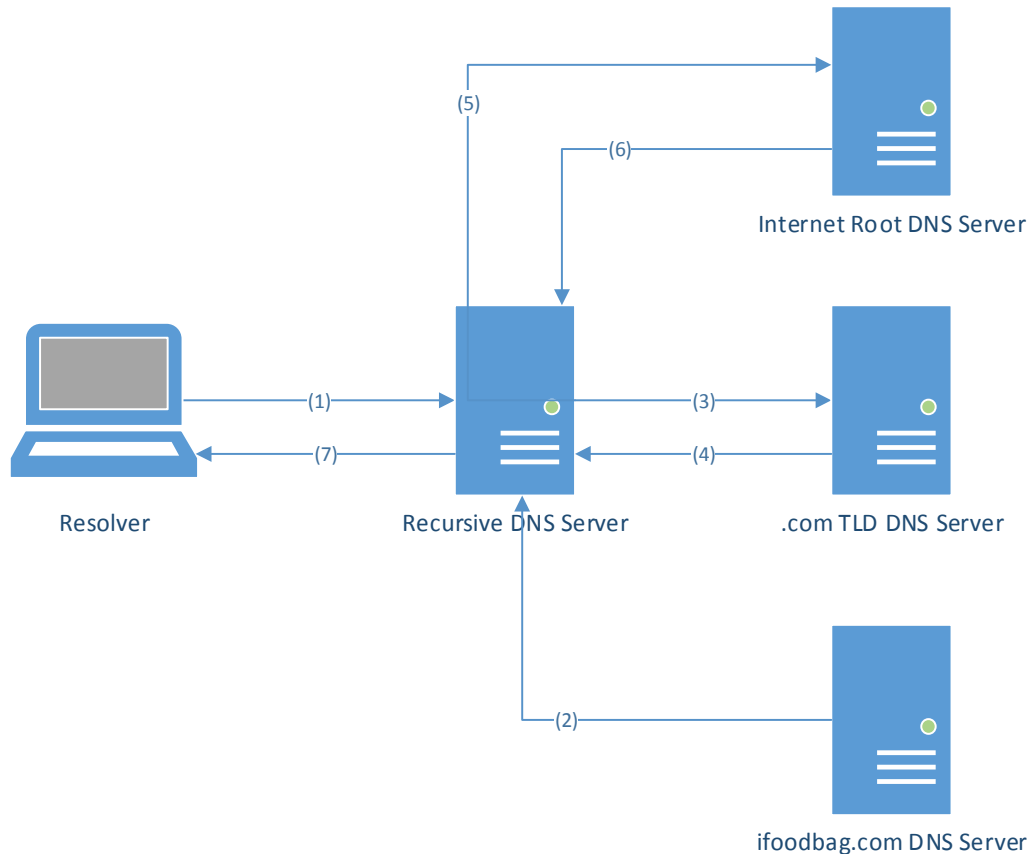


Figure 3-4: DNSSEC work process (concept taken from [62])

3.1.8.2 Issues raised by DNSSEC

DNSSEC consumes more computing resources than the simple (insecure) DNS name resolution process. This additional processing and communication can have a negative impact on the overall performance of name resolution system. Furthermore, DNSSEC uses asymmetric key cryptography, which requires complex mathematical operations. These cryptographic operations consume extra memory and processing power in the name servers. In addition to this, the digitally signed records, DNSSEC resource record sets are relatively large; thus transferring this additional information consumes more network bandwidth as do the additional messages for origin authentication.

DNSSEC can be susceptible to DoS attack, as with DNSSEC a small DNS query can generate a much larger response. Moreover, DNSSEC consumes a relatively large amount of computational resources (e.g. CPU overhead, memory, bandwidth consumption, etc.) in comparison to an ordinary DNS query. For these reasons a type of DoS attack, called a DNS amplification attack or DNS reflection attack is discussed in [63]. A more detail discussion about performance and security issues raised by DNSSEC can be found in [64]–[67].

3.2 Distributing load among application servers

A load balancing mechanism has been incorporated in the *ifoodbag* scalable cloud architecture. Below we discuss the concept of load balancing, why it is necessary to balance load, and several different available solutions for load balancing.

3.2.1 Why load balancing?

Response time* is an important concern for any web application, especially for e-commerce sites. Campbell and Alstad have performed research [67] to identify elements which affect a web application's response rate. They have formulated a relationship among these elements as shown in the following equation:

$$R = \left(\frac{\text{Payload}}{\text{Bandwidth}} \right) + RTT + \left(\frac{\text{AppTurns}(RTT)}{\text{Concurrency}} \right) + C_s + C_c$$

Equation 3-1: Web application response rate

The terms used in the above equation are described below:

R	Response time
Payload	Total number of bytes sent to the user's browser, including the webpage and all resource files (e.g. CSS files, JavaScript files, images).
Bandwidth	Minimal bandwidth, in bits per second, through the network connection between the user and the server.
Round Trip Time (RTT)	Amount of time required for data packet to traverse from user's browser to the server and back.
AppTurns	Total number of components needed for rendering the page. Components include image files, CSS files, etc.
Concurrency	Number of concurrent requests a browser makes for resources.
C _s	Computation time on server side.
C _c	Computation time on client side.

A low value for response time will improve the user's perceived performance of the web application. Conversely, a high value will degrade the user's perceived performance. For this reason the goal is to improve performance to a sufficient degree that the user's perception is that the service is *responsive* (i.e., they get a response within the time that they expect).

A load balancing mechanism can help improve web application performance and make the application more scalable. Rather than using a single application server, multiple application servers can be used to provide service to users.

There are multiple ways to split the load among multiple application servers [68]:

1. Splitting static and dynamic contents across multiple application servers.
2. Distributing responsibilities among multiple (identically configured) servers.

In the design for the *ifoodbag* web application in the cloud a program called *Squid* has been used for load balancing. Squid [21] is an open source application mainly known for its use as a caching web proxy. Squid can cache and reuse frequently requested website contents, thus Squid reduces the load on the actual web application server by reducing the number of user requests that must actually be served by the web server. Squid can also be used for load balancing **without** caching and reusing frequently requested contents. Detailed information about load balancing with Squid is presented in the next section.

* For a web application, response time is characterized by how quickly the application responds to the user's input.

3.2.2 Using squid for load balancing

While Squid is primarily thought of as a proxy server, Squid can also be configured to balance load among multiple application servers in a round robin manner for each HTTP* GET/POST request†. These GET and POST requests typically originate from a user's web browser; however they could also come from applications.

The *ifoodbag* web application server utilizes a collection of images, style sheets (CSS), JavaScripts, and php scripts. Two application servers could be configured with identical configurations and resource files. When Squid receives a HTTP request it might ask server₁ to execute the server side operations (e.g. user authentication), while it might ask server₂ to provide the information to the web browser so that it can render the web page. The load balancer can alternately ask each of the servers to provide CSS and images until the whole web page is delivered to the user's web browser [68]. In this way Squid can distribute tasks among multiple application servers and thus reduce the apparent response time and improve the user's *perceived* performance.

Squid also supports weighted round robin scheduling in order to make it possible to process more requests using application servers with a more powerful configuration (i.e. a larger amount of memory and higher processing speed than other application servers), while assigning fewer requests to application servers with more limited resources.

Squid can detect if an application server is unavailable. If so, it promptly communicates with a backup or secondary application server. In this way squid reduces the server failover time [40].

Figure 3-5 depicts a sample network diagram using Squid for balancing load among multiple application servers.

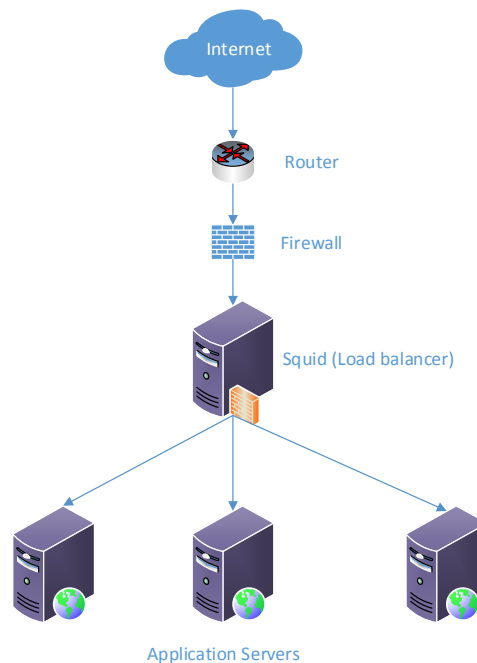


Figure 3-5: Load balancing with Squid

* For further details of HTTP, see section 3.3.1.

† GET requests data from a specified resource and POST submits data to a specified resource for processing [69].

3.2.3 Policies to improve squid security

Eric Galarneau has done extensive work describing security considerations when using Squid and possible actions to improve Squid's security. We summarize some points from his paper [70]:

1. Squid was primarily developed for UNIX based systems. However, it is available for many other operating systems. Security guidelines should be followed to setup and configure the host operating system for Squid. It is best to install Squid on a dedicated machine and only the necessary services should be installed on this machine. Some organizations provide guidelines to securing different operating systems, for example, CERT provides guidelines to secure UNIX based systems [71]. System administrators should consider these guidelines when configuring their system.
2. The Squid service should **not** run as the *root user*. Instead one should create a sandbox user specifically dedicated to Squid with an invalid shell (e.g. Noshell [72]) assigned to it. This sandbox user will have very limited access to the system's files.
3. Squid requires directories to store cache data locally. Ensure that only the Squid user (i.e. sandbox user) has read, write, and execute access to these directories.
4. When installing Squid, the integrity of the source file should be verified using the MD5 checksum.
5. Change the default port (TCP port number 3128) for Squid to another available port number in the high range, so that port scanners cannot easily detect the proxy server.
6. Use the authentication feature provided by Squid. Different methods of authentication are supported by Squid, for details see [73].
7. Turn on logging in Squid. There can be two problems when enabling this logging: (1) performance degradation and (2) handling large log files. Squid's peering capabilities can be used to improve its performance. In order to better deal with log files Squid can be configured to save the current log file after it exceeds a defined size, archive the previous log, and open a new log file for writing.

3.2.4 Alternative approaches for load balancing with Squid

There are other cost-effective solutions to distribute load among multiple servers. Below we present some of these solutions.

3.2.4.1 Round-robin DNS

Round-robin DNS is a load balancing technique where a DNS server balances the load among multiple servers with different IP addresses. All of the servers have the same hostname and provide the same service. A DNS server can be configured with the IP addresses of multiple identical servers each of which can perform the same function (e.g. multiple identical web servers). When the DNS server is asked for an IP address of the web server, the DNS server provides the IP addresses in a round-robin fashion. Round-robin DNS is often used to balance load among servers that are distributed in different geographical locations [74].

Round-robin DNS is very easy to implement and it is a very economical way of balancing load for a small to medium sized organization. However, round-robin DNS has some limitations:

1. Round-robin DNS does **not** provide fail-over. If a server becomes unavailable or goes down, the IP address of that particular server needs to be manually removed from the round-robin DNS server's zone record.
2. For round-robin DNS to work properly the time-to-live (TTL) of the records should be set to a very low value. Otherwise, until the TTL expires, a round-robin DNS server will reply to all the requesting users with the IP address of a single server based upon

the entry in its cache, which will hamper the round-robin technique for all machines which query this DNS server.

3.2.4.2 HAProxy and other open source load balancing platforms

HAProxy is an UNIX based open-source load balancing solution designed to provide high availability, load balancing, and proxy functionality[75]. The official website of HAProxy [75] claims that not a single vulnerability has been identified in HAProxy solution in the last 10 years.

There are several other open source and free to use load balancing solutions, such as: Core Balance [76], Crossroads [77], Distributor [78], Zen Load Balancer [79], and Octopus Load Balancer [80].

3.3 HTTP connection between a user's browser and an *ifoodbag* application server

The user's web browser creates a HTTP connection to an *ifoodbag* application server. Below we discuss the HTTP protocol, its security issues and available options to resolve the security issues of HTTP.

3.3.1 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is an application layer protocol in the TCP/IP protocol suite. HTTP has been used for client-server communication in the World Wide Web [81] for data communication since 1990 [82]. In order to improve the performance of HTTP, the Internet Engineering Task Force (IETF) [83] defined HTTP protocol version 1.1 [41]. This version is currently being used (it is also called HTTP/1.1).

HTTP is one of the most widely used internet protocols. We use the HTTP protocol every day to browse web pages in the internet. In order to browse a website or access a web application, we provide our desired web page or web application's Uniform (or universal) Resource Locator (URL) * to a HTTP *client*. This HTTP client is commonly a *web browser* (e.g. Chrome [85] or Internet Explorer [86]). The web browser sends an HTTP request to the web server. The web server replies with an HTTP response. This response may contain the requested content or indicate why the server is not able to process the request.

3.3.2 Security issues with HTTP

The prime security issues raised by HTTP protocol are related to the fact that HTTP messages exchanged between a HTTP client and a HTTP server are sent in plain text. As a result anyone with access to the network along the path that these messages pass can intercept the HTTP packet and can read its content. In many cases this message may contain sensitive information, such as a user name and password! Additionally, HTTP does not verify the identity of the web server (i.e. if this is the correct server to which the HTTP message was to be sent). As a result of these weaknesses for commercial data transactions, such as online banking or e-commerce data transactions, HTTP is unsuitable.

3.3.3 Solutions to improve security in HTTP

To integrate security into HTTP some solutions have been developed. Below we discuss two different technologies to make HTTP traffic secure: HTTP Secure (HTTPS) and Secure HTTP (S-HTTP). In addition, some security considerations regarding HTTP/1.1 have been discussed in [87].

* The URL is used to identify a network resource [84].

3.3.3.1 HTTP Secure (HTTPS)

HTTP Secure (HTTPS) is not a protocol itself, but rather it is a combination of HTTP with SSL/TLS* protocol. To secure HTTP communication, HTTPS layers HTTP on top of the SSL/TLS protocol in order to use its security services (which provide confidentiality and authentication)[88].

Using SSL/TLS's confidentiality service, HTTPS encrypts all the HTTP messages communicated between a HTTP client and a HTTP server. As a result of this encryption no unauthorized person can read the content - even if they sniff the packets. In this way data confidentiality is ensured.

Using SSL/TLS's authentication service, HTTPS verifies the identification of the HTTP server, which the client wants to connect. It does this by verifying the digital certificate of the server has been signed by a trusted certificate authority (CA). In this way a user can be assured that they are communicating with the correct web site.

Optionally the identity of the HTTP client can also be verified using digital certificates. This allows a HTTP client to authenticate itself to the web server by using a digital certificate (instead of or in addition to a traditional password based authentication system).

3.3.3.2 Secure Hypertext Transfer Protocol (S-HTTP)

Secure HTTP (S-HTTP) has been developed to secure the communication between a HTTP client and a HTTP server. This protocol was developed in order to establish secure commercial transactions for a wide range of applications (e.g. online transactions, online banking). S-HTTP is an extension of HTTP. It is compatible with HTTP and can work concurrently with HTTP, but S-HTTP is **not** compatible with HTTPS.

S-HTTP does not require a public key or digital certificate for the HTTP client, as it supports *only symmetric key* operation modes. The major difference between HTTPS and S-HTTP is that HTTPS creates a secure communication **channel** between a client and a server for preserving confidentiality and to provide authentication, while S-HTTP aims to secure *individual messages* communicated between a client and a server. More information about S-HTTP can be found in [89] and [90]. Currently, HTTPS is more commonly used than S-HTTP, due to the support for HTTPS by major companies, such as Microsoft [48].

3.4 Communication between the management node and other nodes

In the proposed design of the *ifoodbag* web application for execution in a cloud environment, there is a special node called the *Management Node*. This node stores all the policies needed to make the solution *elastic* and *dynamically scalable*. This node controls the application servers and maintains communication with the load balancers, distributed database, and the caching module. It is obviously important to ensure the security of the communication between this management node and the other nodes in the different tiers of the proposed architecture. Below we discuss how to secure the communications between the management node and other nodes using virtual private network (VPN) technology.

3.4.1 What is VPN?

An organization can have geographically distributed branches and datacenters, each of which has its own separate local networks and may use one of many internet service

* Transport Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL) are security protocols that provide confidentiality and authentication through public key cryptography and use of one or more digital certificates.

providers. One approach to maintain privacy and confidentiality of the data traffic between these individual networks is to setup leased lines. However, this can be very expensive. Instead we can create a VPN overtop of the internet connectivity between the sites, thus providing a very simple and cost-effective solution, which also provides scalability.

A VPN connects a group of computers, located in discrete networks, that communicate over a public network, such as the internet [91]. A VPN can be composed of separate networks interconnected by VPN tunnels. Connections between these networks are visible, but the actual contents of the traffic flowing through these tunnels are not visible to others.

Organizations create (site-to-site) VPN to connect remote branch office networks and remote datacenters to the corporate network. Additionally, individuals can use a VPN for (remote) access to network resources, even when they are not physically residing on the same physical network – but are using an untrusted public network to reach the remote network. Figure 3-6 depicts an example of using a VPN for remote-access.

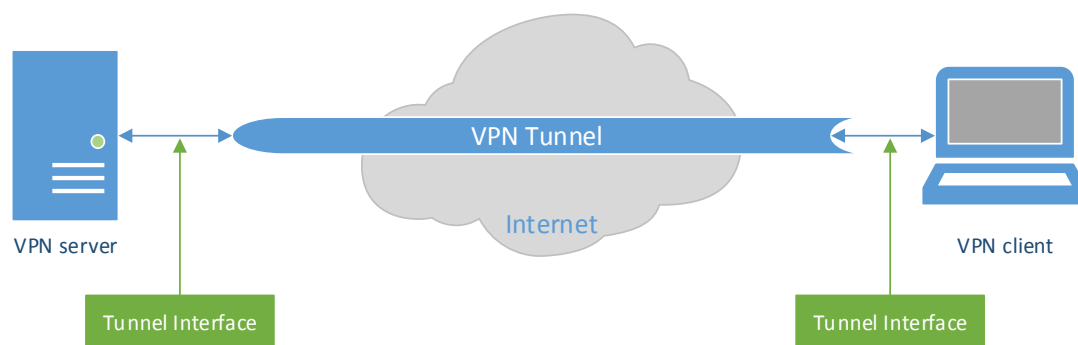


Figure 3-6: Remote-access VPN

3.4.2 VPN Tunnels

A VPN tunnel is a virtual tunnel created with the aid of IP packet *encapsulation*. IP packet encapsulation places an IP packet into another packet. This encapsulation is done in order to convey the encrypted contents of the inner packet via a public network (e.g. internet). Nodes (i.e. computer or network) at both ends of the tunnel create a *tunnel interface* to realize the end of the tunnel. Encapsulating an IP packet can be done by using standard encapsulation or a special tunneling protocol [92]. Different types tunneling protocols have been described in detail in [92]. Here we present a brief summary of these different tunneling protocols.

3.4.2.1 Point-to-Point Tunneling Protocol (PPTP)

PPTP [93] is a VPN tunneling protocol which allows encrypting and then encapsulating an IP datagram following a new IP header for transmission via an IP network (e.g. internet). PPTP can be used for both remote access or site-to-site VPNs. PPTP provides data confidentiality through encryption. However, PPTP does **not** provide data integrity or data origin authentication [92].

3.4.2.2 Layer 2 Tunneling Protocol (L2TP)

L2TP [94] is a *link layer* protocol used in the TCP/IP protocol suite. L2TP does **not** provide data confidentiality itself. Instead, it relies on another encryption protocol. Often L2TP is implemented with IPsec to provide data privacy and confidentiality [92]. Details of IPsec will be presented in section 3.4.2.4.

3.4.2.3 Secure Socket Tunneling Protocol (SSTP)

SSTP uses the HTTPS protocol over TCP port 443 to transport point-to-point protocol (PPP)* or L2TP/IPsec traffic. SSTP encapsulates PPP frames in IP datagrams and then encrypts the traffic using the Secure Sockets Layer (SSL) 3.0 channel of the HTTPS protocol. In this way a SSTP VPN connection provides confidentiality, integrity, and data origin authentication [40]. More details about SSTP are given in [95].

3.4.2.4 Internet Protocol security (IPsec)

IPsec [96] provides security services at the IP layer of the TCP/IP protocol suite, in both IPv4 and IPv6 network environments. Microsoft states that “IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption) and, replay protection” [97]. It should be noted that IPsec is a mandatory part of all IPv6 implementations.

3.4.3 VPN Implementation

A VPN can be implemented using any of the methods listed below:

1. **Using built-in functionality in operating system:** If the purpose of VPN is remotely access a single computer, then using the VPN software built into operating system can be a good option. For example, several versions of Windows and Linux servers have built in VPN server functionality [98] [99] [100].
2. **Software firewalls:** Microsoft’s ISA Server [101], Check Point [102], and Symantec Enterprise Firewall [103] provide VPN gateway functionality.
3. **Dedicated VPN router or VPN concentrator:** A dedicated VPN router (e.g. from Cisco [104], Juniper [105], AEP Networks [106], etc.) can be used to create a reliable and robust site-to-site or remote-access VPN tunnel.
4. **Third-party hosted VPN provider:** Some companies (e.g. Avaya Inc. [107]) provide solutions to organizations to set up remote-access and site-to-site VPN connections. These commercial solutions are proprietary to the solution providers.
5. **Open source solution:** There are several open source applications available (e.g. OpenVPN [108]). These are widely used to provide not only a cost effective, but also a scalable VPN solution. In the next section, we will present further details of OpenVPN.

3.4.4 OpenVPN

OpenVPN [108] is an open source software application to implement remote-access or site-to-site VPN connections. Some features of OpenVPN are:

1. To secure the network traffic and keep it private, OpenVPN uses encryption, authentication, and certification features of the OpenSSL [109] library [110].
2. OpenVPN supports authentication using both secret key cryptography[†] as well as public key cryptography[‡]. For public key cryptography based authentication, OpenVPN uses a custom security protocol, which provides the SSL/TLS connection a reliable transport layer [58]. The custom protocol uses SSL/TLS for secure key exchange [113]. OpenVPN protocol is explained further in [114].
3. OpenVPN uses the User Datagram Protocol (UDP) as its transport protocol and by default it uses UDP port 1194. OpenVPN can also be configured to use TCP as its transport protocol [58].

* PPP is a data link layer protocol used to create direct secure connection between two networking nodes [9].

† A pre-shared secret key is used for authentication. Details can be found in [111].

‡ A pair of public-private key is used for authentication. Details can be found in [112].

4. OpenVPN can perform Network Address Translation (NAT) [115] and firewall traversal.
OpenVPN is free to use as it is published under the GNU General Public License (GPL) [116] as stated in [113].

OpenVPN works in two different modes: (1) TAP and, (2) TUN. In TAP mode it is possible to transport any network protocols (e.g. IPv4, IPv6), while in TUN mode only IPv4 can be transported. TAP mode can be used to create a bridged network where the VPN nodes can have IP addresses from the local DHCP server. But in TUN mode it is not possible. TUN mode has a lower traffic overhead and it does not normally transport broadcast traffic [117].

3.5 Caching web data

Caching web data from the web application of *ifoodbag* is provided by Memcached. In this section we discuss the general idea of caching and memcached, the main principles of memcached, how memcached works, the advantages and disadvantages of memcached, and security in memcached.

3.5.1 What is a cache?

Caching pronounced “cashing” [118] is the process of storing data in a cache [9]. A cache is a temporary storage area, frequently a reserved section of main memory or a storage device [119]. We give a very simple example here to make the concept of a cache clearer. Whenever any user requests data via their web browser this data is stored in a subdirectory (which serves as a cache) under the directory used by your browser [118]. Later when this data is needed again, the browser can access the recently cached webpages. If the content is still available and the current time is within the time period during which this content is still valid, then the browser will utilize the cached copy rather than make another HTTP request to the original server [118]. Caching saves users time (hence the user’s perceives the service as more responsive) and it also saves the network the burden of retransmitting the same content – hence reducing the amount of network traffic [120].

3.5.1.1 What is Memcached?

In 2003, Memcached was developed by Brad Fitzpatrick for LiveJournal [121]. Today memcached is used by Netlog [122], Facebook [123], Flickr [124], Wikipedia [121], Twitter [125], YouTube [126], and Zynga [127]. These services use memcached as part of their web server platform.

Memcached is a high performance, distributed memory object caching system [121]. It is free, open source, and generic in nature [22]. One of the main uses of memcached is to speed up dynamic web applications by reducing the load on a database [128]. Memcached is implemented as an *in-memory key-value store* for small chunks of arbitrary data (strings and other small objects) resulting from database calls, API calls, or page rendering [22].

The design of memcached is simple but powerful. Since the design of memcached is simple it promotes quick deployment, ease of development, and solves many problems of large data caches [22].

3.5.1.2 Main principles of Memcached

The main principles of Memcached are:

1. Memcached is distributed (it works well on one machine or one hundred machines [129]).
2. Network access is fast (memcached servers are close to other application servers) [121].

3. Memcached does not provide persistency (this means if your server goes down, data that was stored in memcached will be lost)[129].
4. No replication (data is spread across N servers, but no key exists on more than one server)[129].
5. No authentication in a shared environment[121].
6. A single entry cannot be larger than 1MB (but it is often possible to compress large entities to fit into 1MB)[129].
7. Keys are limited to 250 characters[121].
8. No active clean up (memcached only cleans up when more space required, when it deletes the Least Recently Used (LRU) item first)[129].

3.5.2 How does Memcached work?

Memcached uses client-server architecture. Jeremy Leishman, Ben Robinson, and Josh Taylor have explained in detail how memcached works on both the server and client sides. In the following paragraphs we have summarized the major points of their paper[130].

3.5.2.1 Server Instances

Wherever free memory is available a number of Memcached server instances are running throughout a network. Each instance of a Memcached server listens on a specified port and IP address. Each machine has its own locally allocated memory. Therefore, an application that is utilizing Memcached can use all the remaining memory. It may also be advantageous to run multiple instances of Memcached on the same machine in such case where the total memory of a machine is greater than the amount that the kernel makes available to a single process. These multiple instances can be handled by listening on different ports.

3.5.2.2 Client Read process

We can use Memcache on the client side just like any other cache. When an application decides which object it needs to access it simply passes the object ID or similar identifier through a hash function, then it checks this hash against Memcache to see if the object is available. If the object is not available, then it fetches the object from the server and places it in the cache.

3.5.2.3 Client Write process

Before the client write process, the object is first fetched from the server or cache. When an object is updated the object is saved to the server and then saved in the cache. Since memcached does not support transactions, it makes sense to pull the object from the server, update it, and save to both the server and to the cache. By doing this we maintain the integrity of the data. However, the lack of persistence necessitates an extra write to the server to ensure that the updated object is preserved.

3.5.2.4 Hashing and keys

The web application maintains a hash table. This hash table contains information about what information is stored in each instance of Memcached. Before the web application accesses a database, an object request is sent to the appropriate server instance of the distributed cache.

Memcached uses a set of keys to look up cached results in the hash table. Storage keys are evenly spread out across the multiple nodes running Memcached. An application uses the hash table to determine which server to send the request to.

This hashing and key system is implemented in two layers. The first layer tells the application which server stores the key. A second layer identifies the serialized object that needs to be retrieved. This two layer system ensures that only the server with the appropriate data will be asked to retrieve an object. This layered architecture is shown in Figure 3-7. The steps that are followed are:

- Step 1 The application requests keys foo, bar, and baz. Key hash values are calculated to determine which memcached server should receive the requests.
- Step 2 The memcached client sends parallel requests to all relevant memcached servers.
- Step 3 The memcached servers send responses to the client.
- Step 4 The memcached client aggregates the responses for the application.

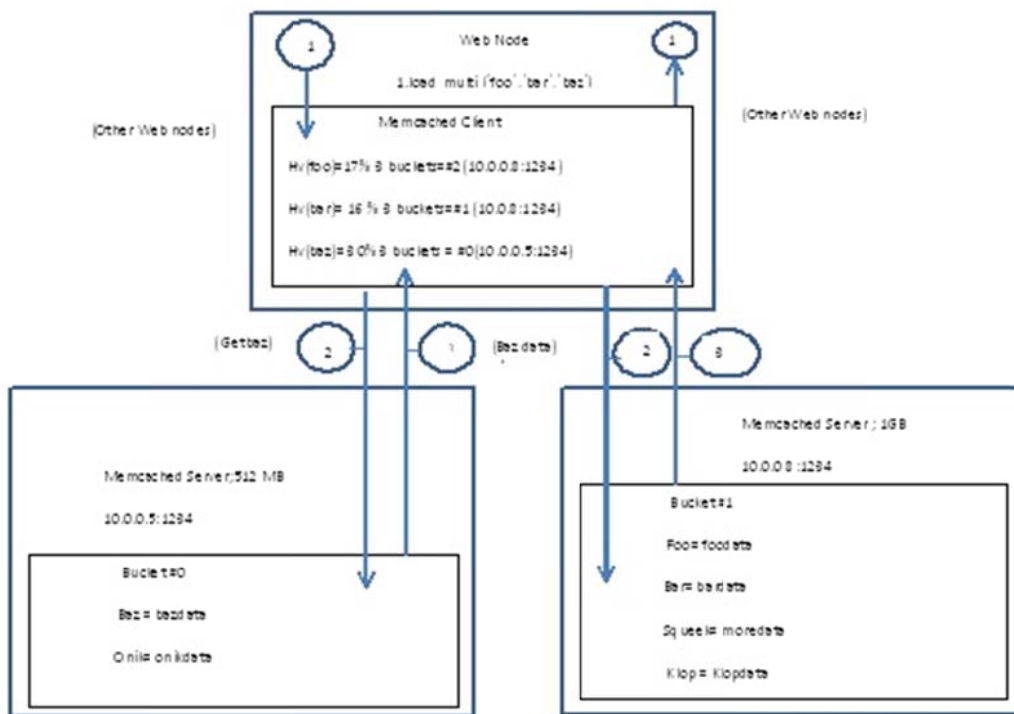


Figure 3-7: Memcached client is responsible for sending requests to the correct servers

If there is key collision, simply we will lose data. It means we will end up with a fake object or we may overwrite something in the cache with something totally different. A well designed application can prevent such condition. Every individual application is responsible for defining and using its own key creating structure. So, once we start running key collisions we need to recheck the code again.

3.5.2.5 Independence

If a server fails inside memcached, the remaining active servers function normally because each memcached server instance operates completely independently. Requests to the failed server cannot be resolved. However, the client can be configured to route around the failed machine to go directly to the source of the data. When a memcached server instance fails, all of the data within that instance will be lost.

3.5.2.6 Expiration of cached entries

Memcached implements the least recently used (LRU) mechanism to decide which cache entries should be discarded. This causes the least recently used item to be discarded first when there is a need to store a new item. Over the long term, this means that the most frequently

used items will remain in the cache and the least frequently used items will be replaced by potentially more frequently used items.

3.5.3 Advantages of Memcached

There are two main advantages of memcached: its low cost and its cross database management system (DBMS) support. Each of these will be discussed in a paragraph below.

3.5.3.1 Low cost

One of the main advantages with Memcached is its low cost. As stated earlier Memcached is open source software, so it is free to download and use. Therefore, no license fees are required. Memcached only requires some amount of RAM and some CPU cycles to operate. Additionally, there is no limitation on how many memcached server instances can be created[130].

3.5.3.2 Cross-DBMS

Cross-DBMS support is another advantage of Memcached, as Memcached was not developed for a particular platform (such as MySQL, PostgreSQL, MSSQL, or any other database). Therefore, we can use any database platform for the backend storage of our application and memcached can easily be integrated into our application to help us to optimize our architecture, while gaining the advantages of object caching[130].

3.5.4 Disadvantages of Memcached

There are three main disadvantages of memcached: its poor documentation, the volatility of the contents of the cache(s), and its security. Each of these will be discussed in a paragraph below.

3.5.4.1 Documentation

Memcached is open-source. Unfortunately it is not very well documented. There is a user community which voluntarily gives support, but when a developer runs into real trouble there are not many options[130].

3.5.4.2 Volatility

Memcached is a volatile storage system. This means that if a memcached server instance crashes, all the data that was stored within it will be lost. This can have potentially negative effects on our application. Even if the instance immediately returns to operation and runs normally, the end users have to start their session all over again[130].

3.5.5 Security of Memcached

One of the main drawbacks of the memcached is its security. Memcached does **not** provide any form of security (either authentication or encryption)[131]. So access to the memcached server should be protected by placing them in the same *private* zone of our application deployment environment[131]

We can further secure Memcached by using two common mechanisms:

Firewall We can start by firewalling everything. This firewall will only allow the data that we want to pass through it[132].

SASL We can use memcached in a hostile network by using Simple Authentication and Security Layer (SASL)[133]. In order to deploy memcached with SASL we will need: (1) a memcached server with SASL support and (2) a client that supports SASL[134].

SASL is a standard for adding authentication support to connection-based protocols.[1]. The newer version of Memcached which is 1.4.3 supports SASL.[1].

To configure SASL we need to do the following administrative operations[134].

```
# Create a user for memcached  
Saslpasswd2 -a memcached -c cacheuser
```

3.6 Cloud storage

In this section, we will discuss cloud storage, types of cloud storage, characteristic of cloud storage, and the advantages and disadvantages of cloud storage.

3.6.1 What is cloud storage?

The term “cloud storage” is primarily used for storage that is remotely located, typically called an online space. This storage can be realized using physical hard drives, USB flash drives, optical or magnetic tape drives, etc. [135]

Cloud storage provides a user with the ability to back up data stored on a server, typically this server is hosted by a cloud service provider [136]. An online storage facility uses a network of virtual storage servers and generally includes tools to manage the virtual storage space. Although data in cloud storage is located remotely, cloud storage can provide high security [135].

Due to the online backup facility that cloud storage provides, it becomes very easy to reach the data, a user only needs a connection to the internet and a user interface provided by the cloud storage vendor [136]. In contrast, data which is only stored in a single physical space can be lost or damaged (increasing risk), while cloud storage provides a data abstraction which is not limited to a single physical space, hence cloud data is safer [136]. Cloud storage technology can be used to provide redundant storage, thus ensuring the contents are not lost, even if a catastrophic situation occurs and one of the datacenters goes down or is destroyed [136]. In order to realize a redundant storage facility a minimum of three storage spaces should be available at any time, in order to provide at least two highly secure physical spaces together with cloud interface for the user to access their data, enabling the data to remain available and accessible at high speed [136].

3.6.2 Types of Cloud storage

There are mainly three types of cloud storage systems: private cloud storage, public cloud storage, and hybrid cloud storage:

Private cloud storage This type of cloud storage is primarily designed for one person or a single company. Storage comes in two formats: on-premises and externally hosted. Both types work well, but were mainly designed for businesses rather than for individuals. Private cloud storage offers greater administrative control and the customer can design the system based on their needs [137].

Public Cloud storage Any authorized person can access a public cloud storage facility over a network. This type of cloud storage requires (and offers) very little administrative control. In a public cloud we get the same security as private

cloud, but maintaining the public cloud is much easier than the private cloud[137].

Hybrid cloud

A hybrid cloud is a combination of both public and private clouds. In this type of cloud storage we can actually customize our features and insert additional applications based on our needs. In hybrid cloud storage a user can keep very important information within the private cloud, while storing less important information in a public cloud[137].

3.6.3 Characteristics of Cloud storage

According to Kodukula and Sasidhar some of the key cloud storage characteristics are [138]:

Manageability	Cost is crucial while considering the software development process, and that is why companies prefer a quality solution with low cost. Local storage cost may be lower in cost compared to remote cloud storage, but the management cost of local storage is higher in cost and remote cloud provides greater long-term benefits that can overcome the overall costs of local storage. Cloud storage is in demand because of the automatic scaling it provides, thus cloud storage will play a vital role in the future.
Access Method	Unlike traditional storage which can be accessed using multiple access methods, cloud storage uses RESTful APLs, Amazon Simple Storage Service (Amazon S3), WindowsAzure™, and Mezeo Cloud Storage Platform. Cloud storage also uses common access methods to provide immediate integration, such as file transfer protocols (FTP) and block based protocols, e.g. iSCSI.
Performance	Performance is a key factor when transferring data, as data needs to be reached quickly and in the correct format. TCP is normally used for bulk data transfer, but is not suitable for accessing chunks of data; therefore, TCP is not recommended for cloud storage. As a result, cloud storage frequently uses Aspera's Fast and Secure Protocol (FASP™) protocol together with UDP.
Multi-tenancy	Multi-tenancy means that data is available to several users. This is the main functionality provided by cloud storage.
Scalability	Scalability is another key quality provided by cloud storage. Cloud storage not only provides scalability in functionality but also in terms of bandwidth (load scaling). Another feature that cloud storage can provide is geographic distribution of data, thus data can be delivered to the user from the nearest copy of the data and data can be distributed appropriately using migration techniques.
Control	Cloud storage provides increased control for the user over both the data and the cost of storing and accessing this data. Cloud storage can use Reduced Redundancy Storage (RRS) to reduce the cost of storing less critical data.
Efficiency	Achieving efficiency requires using the best possible method when utilizing the available resources. For cloud storage to be efficient, it must store more data. Cloud storage can provide data reduction by compressing the original data in order to save space.
Cost	Cloud storage's main purpose is to reduce the <i>total</i> cost of storage, which includes the capital cost of purchasing storage devices, powering these devices, repair costs, and storage management cost.

3.6.4 Advantages and disadvantages of Cloud storage

This subsection examines the advantages and disadvantages of cloud storage.

3.6.4.1 Advantages of Cloud storage

Cloud storage has many advantages over local (individual) and other network connected online storage alternatives. According to Bruce Street Cloud Drive the advantages of cloud storage over traditional storage are [139]:

- Cloud storage is cheap and provides unlimited data storage facility, e.g. AmazonS3.
- This type of scalable data storage is cheaper than electronic data storage.
- No need for installation.
- No need for replacements.
- Cloud storage providers provide their own backup and recovery system.
- No physical presence, so there is no need for maintaining special environmental conditions.
- No need to hire additional personnel.
- No need for electrical power.
- No extra cost for physical data storage space.

3.6.4.2 Disadvantages of Cloud storage

Cloud storage also has some disadvantages. According to Bruce Street Cloud Drive some of the disadvantages of cloud storage are [139]:

- Performance is always bounded by the available network bandwidth.
- Cloud storage primarily relies on Internet connectivity; hence the availability of data could be a major issue.
- Internet throughput may negatively affect performance.
- Cloud storage uses specific networking protocols; hence compatibility issues may arise when dealing with the file access via a normal LAN.
- Cloud storage often requires *ad hoc* programs in order to provide compatibility between different protocols.
- There is no common industry standard protocol for cloud storage.
- Different interfaces are needed for different cloud storage data.
- The selection of a suitable protocol is difficult.

3.6.5 Traditional versus Cloud storage

Cloud storage is different from traditional storage in that traditional storage provides multiple options for storage[138]. For example, in memory storage as cache or RAM disks, local block devices for direct access storage, network attached block devices for logical unit number storage, and file systems (such as NFS & CIFS file servers) for storage area network or network attached storage. Additionally, FIFO structures can be used for messaging queues[138]. In contrast, in cloud storage we must use the different providers' techniques for memory (e.g. Amazon's elastic cache for memory), EC2database AMIs or Amazon simple DB for structural storage, for messaging queues we might use Amazon's simple queue service (SQS), and Amazon's Elastic Block Store (EBS) for snapshots of memory, file systems, or backups[138]

3.6.6 Reliability and Security factors about cloud storage

The two most important concerns about data are reliability and security. Data needs to be secure and the storage provider needs to be trustworthy and reliable. In order to secure data while it is being transferred different techniques have been developed. Encryption is used for this purpose together with an authentication and authorization process [140]. In order access the files or data the end user needs to provide the appropriate key. Authentication is done by requiring the user or applications to provide the appropriate credentials for each type of access. After authenticating the user or application, the storage system must check that this user or application is authorized to access the data in the requested manner.

With all these protection measures there is still a risk that hackers may steal the protected data either directly from the storage cloud or while the data is in plain text form (i.e., when it is vulnerable)[140]. Therefore cloud storage provider put lots of efforts into ensuring that the data is protected at all times [140]. Reliability is especially important for cloud storage of

data, as an unreliable cloud storage would be a liability[140]. Data needs to be stable and error-free, hence cloud storage systems utilize a variety of redundancy techniques[140].

3.7 Distributed database

This section describes what a distributed database is. This will complete the background necessary for the reader to understand the components of the system that is to be secured.

3.7.1 What is distributed Database?

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network[141]. A DDB can be located in multiple computers that are located in same physical location [142]. A centralized distributed database management system (DDBMS) controls the distributed database. This DDBMS periodically synchronizes all the data, so that if multiple users access the same data the data will be consistent. The DDBMS ensures that updates and deletions performed on the data at one location are automatically performed elsewhere[143].

Two processes (duplication and replication) ensure that the data in the DDB is up to date. Replication uses software to monitor changes inside the DDB. Once it identifies changes, the replication process makes all of the separate databases look same. This process can be complex, time consuming, and requires a lot of computer resources. These costs depend on the size of the distributed database[142].

In contrast, duplication is less complicated. In this process, one database acts as the master database and then duplicates are made of that database. During the duplication process, only the master database is allowed to be changed, thus ensuring that local data will never be overwritten with inconsistent data. Both duplication and replication ensure that all data are current and up to date in the distributed locations[142]

There are a variety of distributed database design technologies, such as: local autonomy, synchronous and asynchronous distributed database technologies. Depending on the sensitivity or confidentiality of the data these technologies may or may not be used. An important practical issue is the investment that a business is willing to make in order to achieve data security, consistency, and integrity[142].

3.7.2 Types of distributed databases

There are mainly two types of distributed databases: homogenous DDBs and heterogeneous DDBs. The following paragraphs will discuss both type of database based on the Oracle8i distributed database system[144].

3.7.2.1 Homogenous distributed database

A homogeneous DDB is a network of two or more databases that are placed in one or more machines. Figure 3-8 shows a distributed system that interconnects three databases: HQ, MFG, and SALES. An application can access or modify the data at the same time in several databases within a single distributed environment. For example, if we want to connect to the database MFG but we want to access data in the database HQ, then we need to create a synonym on MFG for the remote DEPT table to allow us to issue a query of the form:

```
select * FROM dept;
```

In this way, the distributed system gives the appearance of a native data access. Users on MFG do not have to know that the data they access actually resides in a remote database.

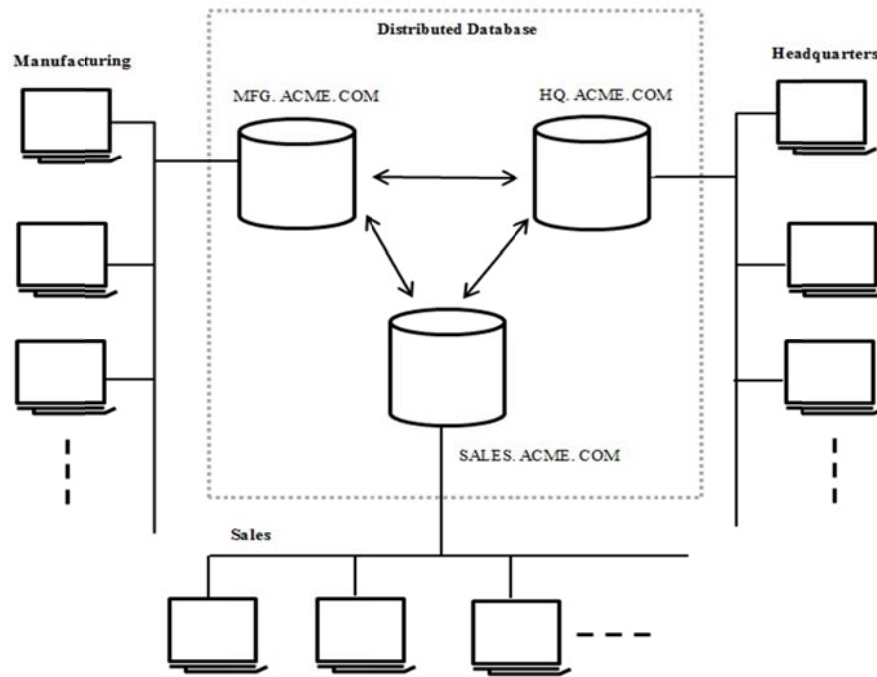


Figure 3-8: Homogenous Distributed database (142)

3.7.2.2 Heterogeneous database

If at least one of the databases is a non-Oracle system, then the result is a heterogeneous DDB. A heterogeneous DDB appears as a single, local, Oracle database to the application. The local Oracle database server hides the distribution and heterogeneity of the data.

The Oracle database server uses the non-Oracle system using the Oracle8i Heterogeneous service in conjunction with an agent. If the user accesses the non-Oracle database using an Oracle Transparent Gateway, then the agent is a system-specific application. We can also use generic connectivity to access the non-Oracle data database as long as this database supports ODBC or OLDB protocols.

3.7.3 Heterogeneous versus Homogenous DDB

Table 3-1 summarizes the differences between a heterogeneous and homogenous DDB.

Table 3-1: Heterogeneous versus Homogeneous DDB [145]

Homogeneous	Heterogeneous
<ol style="list-style-type: none"> 1. All sites have identical software. 2. Aware of each other and agree to cooperate for processing user request 3. Appears to user as single system 	<ol style="list-style-type: none"> 1. Different site use different schemes and software 2. Sites may not aware of each other and may provide only limited facilities for cooperation in transaction processing

3.7.4 Approaches to DDBs

There are two principle approaches to store a relation *r* in a distributed database system: replication and fragmentation.

In data replication data is replicated across multiple nodes. Data replication can improve performance by coping data to a location near the user. This replication can also improve reliability; as if one node fails, then another node with a copy of the data can continue

processing the database transaction. The advantages of data replication are better access and reliability and potentially improved performance.

According to Holowczak[146], data fragmentation can be done by 3 different ways:

- Horizontal** row in a table are split up across multiple nodes
- Vertical** a column in a table is split across multiple nodes
- Mixed** a combination of both horizontal and vertical methods

According to [145], the advantages of these three fragmentation types are:

- Horizontal**
 - Allow parallel processing on fragments of a relation
 - Allow a relation to be split, so that tuples will be located where they are accessed most often
- Vertical**
 - Allow parallel processing of a relation
 - Allow a tuple to be split so that each part of the tuple is stored where it is most frequently accessed and efficient joining of vertical fragments is allowed based upon a tuple-ID attribute
- Mixed** Fragment may be successfully fragmented to an arbitrary depth

3.7.5 Advantages and disadvantages of DDBs

Just as any other system, a DDB has advantages and disadvantages. In this section we will examine the advantages and disadvantages of DDBs.

According to[147], the advantages of a DDB are:

- Faster data access Often a given end user will only work with a subset of company's data. If this data is stored and accessed locally, then the database can deliver faster data than a remotely located database.
- Faster data processing Since the distributed database is located over several places so it is possible to process data at several sites. Doing this spreads the work load enabling the data to be processed faster.
- Availability A DDB is less dangerous than a single DB, as the latter is a potential single point failure. When using a DDB if one computer fails, processing can shift to other instances – enabling the workload to be immediately distributed to others.
- Modular growth In a DDB, new sites can be added without effecting the operations of other sites. This modular growth helps a company to expand easily and rapidly.
- Reduce operating cost DDBs can reduce operating costs as it is more cost effective to add a new database server to a network rather than update a large mainframe.

According to [148], the disadvantages of a DDB are:

Complexity	A DDB is more complex than a centralized database. In DDBs replication of data add some additional complexity. If the software does not handle the replication properly, thus will negatively affect availability, reliability, and performance.
Complex database design	Depending upon the design of the DDB, data fragmentation, allocation of fragments to specific sites, and replication need to be considered. This makes the design of the database more complex.
Cost	Since a DDB is distributed over multiple computers these computers each require maintenance. DBMS also requires extra hardware to establish network connections between the sites. There are also additional labor costs due to the need to manage and maintain each local database.
Security	In a centralized database it is much easier to control data since all the data are located in the same place. But in a DDB we must control the access to all of the replicated data that is placed in multiple locations and we have to secure the network between the different sites and the users.

Additionally, the remote database together with infrastructure needs to be secured by encrypting the traffic between the individual database servers and between the remote sites[149].

3.7.6 Security Weakness of distributed database

Security in a DDB could be more difficult to maintain than for a centralized database. The reason for this are [150]:

Multiple entry points	A DDB has many entry points to the system. In order to address this problem, each node needs to be secure physically and logically.
Encryption keys	Encryption requires that different parts of the system exchange secret keys. These keys are used to encrypt network traffic between these nodes. The more these keys are distributed, the greater the risk that their security is compromised.
Corrupted node	If a node is compromised (due to viruses or direct attacks by hackers), then the whole system could be vulnerable.

3.7.7 Security Components in a DDB

The general term “security” means to protect something against threats. The term “information security” means to protect information from unauthorized access, modification, or inappropriate use of data. The basic security of a DDB requires protecting the data from unauthorized users or malicious software (virus, worm, Trojan horse, etc.)

The four main security components of a distributed system are: (1) authentication, (2) authorization, (3) access control, and (4) encryption. These are described in more detail by Gupta et al. [151] as:

Authentication	Usually a “smart token” (hardware device, such as a smart card) creates a response to a challenge. This response is sent to an authentication server that is connected to the network.
Authorization	Following authentication, an authorization decision is made concerning whether the user should be allowed access to specific resources.
Encryption	All communication is encrypted using a cryptographic algorithm, such as RSA, PGP, or AES. Frequently a public and private key system is used, sometimes in conjunction with a certificate authority.
Access control	Access control can be implemented via access matrices, access lists, capabilities list, or tokens. These lists define the access permissions for specific resources or classes of resources for each user.

3.8 Addressing DDoS attack

To understand the importance of studying a distributed denial of service (DDoS) attack, consider the following:

- A study [152] conducted by Neustar Enterprise Services has discovered that :
“*In 2012, over 1 in 5 UK Organizations Hit by DDoS Attacks*”
- An advanced threat protection solutions provider Arbor Networks warns:
“*DDoS attacks scaling up alarmingly!*” [153]
- A report from Security firm Prolexic states that:
“*Attacks became bigger and more frequent in the first three months of 2013.*” [154]

After reading all the quotes above the necessity of studying DDoS attack and preparing the *ifoodbag* cloud network to prevent DDoS attacks should be clear. For this reason in this section we will discuss different types of DDoS attacks and how to prevent DDoS attacks in a cloud environment. We will also propose a guideline for *ifoodbag* to prepare its network to prevent DDoS attacks.

3.8.1 What is a DDOS attack

At first consider two definitions of a DDoS attack:

- National Institute of Standards and Technology (NIST) definition of DoS attack:
“*... An attack that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources.*” [155]
- Wikipedia says:
“*A denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack) is an attempt to make a machine or network resource unavailable to its intended users.*” [156]

A denial of service (DoS) attack is a type of attack that is primarily focused on interrupting *availability* of services of one or more victim system(s). A DDoS attack is a variant of a DoS attack. To perform a DDoS attack on a target, the attacker utilizes multiple computer systems in order to make it more difficult to detect or mitigate an attack. The computer system involved in a DDoS attack can be located in geographically distributed areas. It is possible that users of the computer systems participating in the DDoS attack do not even know that they are participating in a DDoS attack! Exploiting these many computers is accomplished by exploiting the flow or limitations of a protocol, network structure, or application that the participant computer systems are normally using. In this way, a DoS or

DDoS attack can target one or more primary victim(s) (i.e. the target computer system that will be attacked) and some secondary victims (i.e. the participant computer systems utilized to carry out the attack). Performing DDoS attack exploiting peer-to-peer system (P2P) is discussed in [157]–[159]. Whatever method is used, the purpose of a DoS or DDoS attack is ultimately to make a service or a network resource (e.g. a website, network storage, network access, physical memory, CPU) unavailable to its legitimate authorized users.

3.8.2 Different types of DDOS attacks

A very typical example of a DoS or DDoS attack is to flood a target server with packets and thus overload the server's network bandwidth and other resources (e.g. physical memory, processor, storage), so that the users of the server cannot connect to it or get the service that they expect from this server. Several other attack methods are presented in the following paragraphs.

3.8.2.1 SYN Flood Attack

Many operating systems have a limit on the number of concurrent half-open TCP connections (i.e. pending connections) on a particular port. Attackers exploit this limit to perform a SYN Flood attack [160]. A SYN Flood attack is performed by sending an enormous number of TCP SYNchronize (**SYN**) packets (often with false source IP address) to a server. After the server receives a TCP SYNchronize packet, it replies with a TCP SYNchronize-ACKnowledgement (**SYN-ACK**) packet and then waits to receive an ACKnowledge (**ACK**) packet. However, since the machine at the source IP address did not wish to open a TCP connection to this IP destination and destination port, it does not reply to the SYN-ACK packet. In this way the attacker exhausts the half-open connections limit of the target server's operating system. As a consequence the server cannot respond to legitimate SYN packets, until the half-open connections are timed out. [155], [161]

Some mitigation techniques against SYN Flood attack have been described in [153], [158] [160] (e.g. filtering, using firewall, reducing SYN-RECEIVED timer).

3.8.2.2 Smurf Attack

A smurf attack is a kind of amplifier attack. If broadcasting is enabled in a network device, then an attacker can exploit this to use the network as an amplifier (also called a 'smurf amplifier') and perform a DoS attack on a target. To perform a smurf attack, the attacker continuously sends ICMP PING requests to the broadcast address of the smurf amplifier network containing as their source IP address the IP address of the target. All of the hosts in the amplifier network reply to the PING requests and as a consequence the target starts receiving a potentially enormous number of ICMP PING response packets. These packets very quickly consume the target's network bandwidth, preventing legitimate users from accessing that target's resources. [155], [161], [162]

3.8.2.3 ICMP Flood Attack

An ICMP Flood Attack does exactly what it says. An attacker, from a network with a greater bandwidth than the target's network, sends an enormous number of ICMP echo requests which the target has to respond to. In this way the attacker slows down the victim's network with a flood of ICMP packets. [161], [163]

3.8.2.4 Ping of Death

The default size of a PING packet is 32 bytes, but can be adjusted to be larger. However, many operating systems cannot handle a PING packet larger than 65535 bytes. Attackers exploit this by performing a "Ping of Death" attack by sending PING packets larger than

65535 bytes, through fragmentation, to a target. Since the operating system of the target cannot handle such a large packet, it will experience buffer overflow and could even crash.

Preventing this attack can be done by using a firewall to check what the PING packet size would be after reassembling the fragmented PING packets. Another method is to increase the size of the memory buffer to larger than 65535 bytes in order to avoid buffer overflow. [164]

3.8.2.5 Teardrops

TEARDROPS is an old type of DoS attack. It exploits a weakness in early TCP implementations. Every network device fragments packets larger than the output link's maximum transmission unit (MTU). The receiver of these packets reassembles the fragmented packets. To perform a teardrop attack, the attacker sends fragmented IP packets with overlapping data in order to exploit the fact that the TCP/IP stack in operating systems (e.g. Windows 95, Windows 98) did not know how to handle such packets and the operating system would crash. However, today most operating system can correctly reassemble these fragments. [161], [165]

3.8.3 Review of work related to prevention, detection, and mitigation of DDoS attack

Many studies have been made of how to handle DDoS attacks and many elegant algorithms have been suggested. Some research deals with attack prevention and/or detection, some focus about how to filter DDoS attack and some research considers attack trace back. Here we discuss different research papers. For each of the following research papers we point out the proposed or deployed method and the scope of the method.

- “CBF: A Packet Filtering Method for DDoS Attack Defense in Cloud Environment” [166]

In this paper, incoming traffic packets are filtered based upon an attribute called confidence. The filter runs during two periods, one is a non-attack period and other is an attack period. The assumption behind this method is that mostly legitimate packets are processing during a non-attack period, thus establishing a nominal profile. A packet during the attack period is scored with the help of the nominal profile and decision is being made either to discard or accept the packet. The attractive feature of the proposed Confidence-Based Filtering (CBF) method is that it can distinguish between flash crowds (i.e. a lot of traffic to a Web site for a popular event) and denial-of-service attack.

- “PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial of Service Attacks” [167]

The basic idea in PacketScore is Conditional Legitimate Probability (CLP). CLP specifies the likelihood of a packet being a legitimate one by considering the attribute values it contains in the TCP and IP headers. Packets are discarded by comparing the CLP of each packet with a dynamic threshold. CLP is formed by comparing traffic characteristics during the attack period with the legitimate traffic characteristics. Packets are scored by using Bayes' Theorem [168]. High accurate filtering and ease in deployment make PacketScore suitable for cloud environment. But it cannot differentiate flash crowd and attack. Also it has to perform expensive operations to score packet, which directs to low processing efficiency in discarding packets.

- “ALPi: A DDoS Defense System for High-Speed Networks” [169]

ALPi follows the concept of PacketScore with some optimization. Implementation complexity is reduced and performance is increased. Simple score computation is based on a Leaky-bucket (LB) [170] overflow control scheme which makes ALPi possible for high-speed implementation. Another scoring scheme named attribute-

value-variation (AV) investigates the disparity of the current packet attribute values, and enhances the accuracy of detecting and distinguishing attacks. Combination of LB and AV lessen the memory requirement and implementation complexity significantly and also improve the accuracies in DDoS attack detection and packet discrimination.

- “Defense against Spoofed IP Traffic Using Hop-Count Filtering” [171]

In this Hop-Count Filtering (HCF) approach where an accurate IP-to-hop-count (IP2HC) mapping table is created to detect and discard spoofed IP packets. Hop-count information can be derived from TTL field of the IP header of a packet. An Internet server can distinguish the spoofed IP packets from the legitimate packet by IP2HC mapping. HCF is easy to setup. Their experiment shows that HCF can identify approximately 90% of spoofed IP packets.

- “Controlling IP Spoofing through Inter-domain Packet Filters” [172]

This paper proposes an inter-domain packet filter (IDPF) architecture. The IDPF is able to mitigate the limit of IP spoofing in the cloud. IDPFs are built from the information that is contained in BGP route updates and are positioned in the network border routers. Rules are set on IDPF that it does not filter out packets with legitimate source addresses and it does not require global routing information. IDPFs can work proactively to limit the IP spoofing capability of DDoS attackers. IDPF can also limit the affect by localizing the source of an attack packet to a tiny number of candidate networks.

- “A packet marking approach to protect cloud environment against DDoS attacks” [173]

Distributed Denial of Service (DDoS) can attack a cloud web services through the use of HTTP and XML. This new form of attack is called HX-DoS attack. It is a merging of HTTP and XML messages that are sent to flood and tear down the communication channel of the service provider of cloud. In this paper to distinguish between the legitimate and illegitimate messages a rule-set based detection technique CLASSIE is used and another technique called modulo-marking method to in order to avoid the spoofing attack. The positive side of this approach is that it has less false positive rate and the rate of detection and filtering of DDoS attacks is high.

- “Securing Cloud Servers against Flooding Based DDOS Attacks” [161]

In this paper Average Distance estimation technique is used to detection of DDoS attack in the cloud. Distance is calculated from the TTL field of an IP header. An exponential smoothing technique is used to forecast the distance mean-value of a packet in the next time period. Mean absolute deviation (MAD) determines whether the distance is normal or not. The technique they suggest is based on MMSE (Minimum Mean Square Error) linear predictor to support efficient traffic arrival rate prediction for disjoint traffic. The proposed solution has high detection rate and low false positive rate but this is implemented on a simulator with 100 nodes not in the real cloud.

- “Security Issues in Cloud Computing Solution of DDOS and Introducing Two-Tier CAPTCHA” [174]

In this paper a preventive measure two-tier CAPTCHA is proposed against Denial of service attack. An attacker generally averts legitimate users of cloud services from using their desired resources by flood the network. Graphical Turing Tests as an authentication method is widely used to make a distinction between human users from robots. This paper suggests using a CAPTCHA (Completely Automated Public Turing Tests to Tell Computers and Humans Apart) method named Two-Tier CAPTCHA. An alphanumeric CAPTCHA code with image is produced and tells the user to logon.

When first tier is passed, a query related to that CAPTCHA code is asked in the second tier to interact with the service. Combination of more difficulties makes the bot-programs difficult to guess, thus enhance the probability of preventing a bot program to perform DDoS attack.

- “Defense of DDoS Attack for Cloud Computing” [175]

An approach SOA-Based Traceback Approach (SBTA) that uses a Service Oriented Architecture (SOA) to trace-back and find the source of DDoS attacks is proposed in this paper. Along with SBTA a Cloud-Filter is combined with a defensive system to trace and recognize the source of attack messages in most cases. Another feature is that it can decrease the numbers of attack packets needed to rebuild the attack path. It also has a high detection rate with low false positives.

- “New Framework to Detect and Prevent Denial of Service Attack in Cloud Computing Environment” [176]

In this approach they propose a integrated solution framework for DDoS attack prevention, attack detection and attack trace back. The suggested approaches are statistical method Covariance Matrix to detect attack, TTL (Time-to-Live) value counting method to find out attack source, and using a Honeypot method to prevent an attack. This paper only proposes a solution; it does not provide any implementation.

- “Scalable Cloud Defenses for Detection, Analysis and Mitigation of DDoS Attacks” [177]

This paper also establishes some countermeasure to Detection, Analysis and Mitigation of DDoS Attacks. As resources become unavailable during DDoS attack, to guarantee resource availability for different stakeholders of the cloud they propose some solutions, such as: early and rapid detection of DDoS attack (It requires scalable inter-Cloud Data Correlation Analysis), delay as much as possible the effects of the DDoS attack by increasing the number of gateways under attack so that they can carry on handling traffic, rapid migration of virtual machines from attacked physical machines to non-attacked ones, maintainability to migrate VM during attack by guaranteeing network bandwidth, and to end the DDoS attack as soon as possible (requires DDoS traffic re-routing in collaboration with the traffic divert). This paper just proposes a solution without providing any experimental result.

4 Secure system design

In this chapter we provide a number of guidelines to improve the security of *ifoodbag*'s cloud implementation. The guidelines are initially presented in sections ordered based upon the flow of packets from a user to the *ifoodbag* infrastructure.

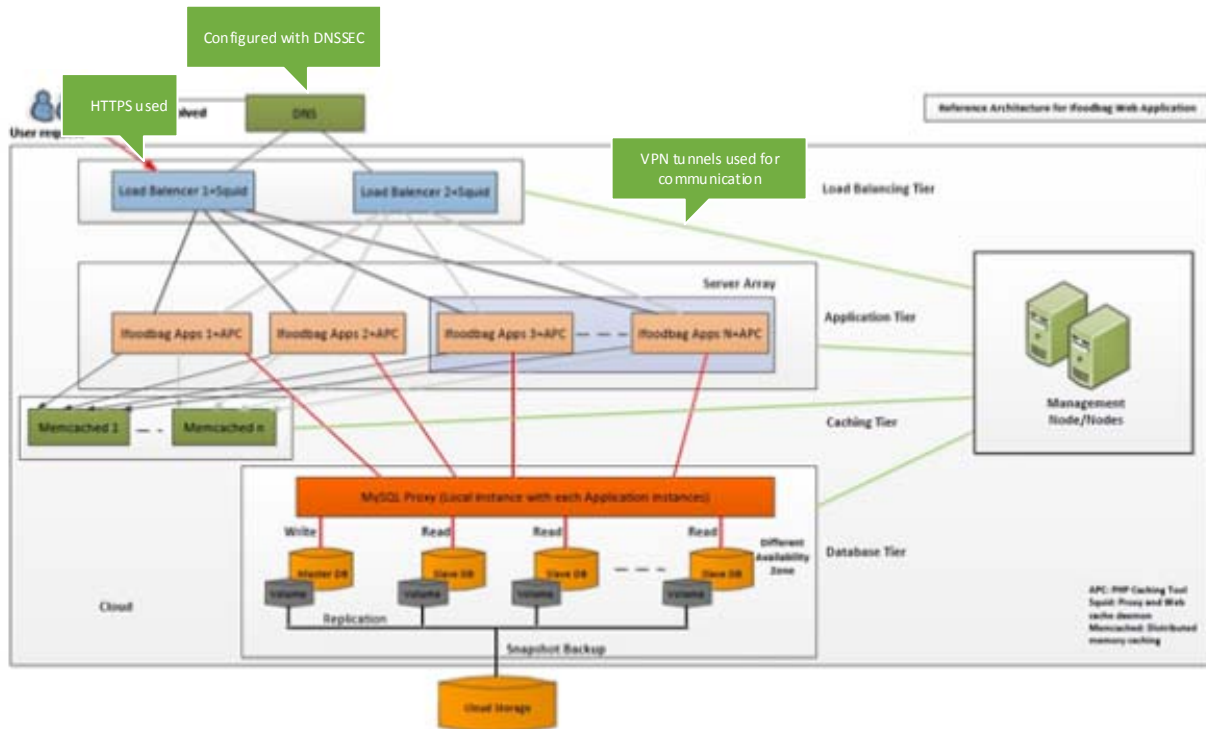


Figure 4-1: System design to secure ifoodbag cloud architecture

4.1 Secure name resolving through DNSSEC

To secure the name resolving process, we recommend *ifoodbag* use DNSSEC at its authoritative name server (i.e. the authoritative name server for the *ifoodbag.com* domain). However, increased security always comes at a cost. There have been several research projects that have measured the cost of DNSSEC, for different DNS resolver implementations (e.g. BIND [178], NSD [179], UNBOUND [180], and Microsoft DNS). BIND is the most widely used name server software [48], hence most researches have focused on it.

The comparative performance studies of DNS and DNSSEC give an idea about how the use of DNSSEC can affect the user's experience of a web application (e.g., *ifoodbag*'s web application). Additionally, these prior results give insights into selecting the name server application that is best suited to implement DNSSEC.

Daniel Migault, Cédric Girard, and Maryline Laurent have empirically investigated the performance of three different name server implementations: BIND, UNBOUND, and NSD*, for both DNS and DNSSEC [181]. As we recommend *ifoodbag* use DNSSEC for its authoritative name server, we only focus on observations regarding an authoritative name server.

* NSD is an authoritative only, open source name server

Migault, Girard, and Laurent noticed that:

- To process name resolve queries DNSSEC generates greater CPU load than DNS. For BIND, if DNS supports X number of queries per second (at maximum CPU load), then DNSSEC supports only 79% of X number of queries per second. For NSD, DNSSEC can support 83% of X number of queries per second. In other words, at a certain CPU load a DNSSEC server supports ~20% fewer queries per second than a DNS server.

NSD can handle more queries per second (for a given CPU load) than BIND. For DNSSEC, BIND at a maximum CPU load supports 43% fewer queries per second than NSD. For DNS, BIND at maximum CPU load supports 41% fewer queries per second than NSD. In other words, NSD can handle 2.3 more queries per second than BIND with DNS or DNSSEC.

- Authoritative Server Processing Time or server response time is the time authoritative servers take to receive a query, process it, and send response. This processing time is crucial as it directly impacts the performance seen by the end users. A DNSSEC authoritative server takes 10% longer response time for BIND and 20% longer response time for NSD than the response time of a DNS authoritative server. For DNS, the BIND response time is twice that of NSD. For DNSSEC, the NSD response time is 45% of BIND's response time. In both cases NSD's response time is always quicker than BIND.

For an authoritative server they observed that in terms of latency^{*}, without load considerations, NSD name server application always outperformed BIND, for both DNS and DNSSEC. For an authoritative name server, they have not stated exactly how the performance differs between DNS and DNSSEC. However, Figure 3(a) in [181] clearly shows that DNSSEC has higher latency than DNS for both BIND and NSD, which indicates that there is some performance degradation for DNSSEC.

Ager, Dreger, and Feldmann [63] focused only on BIND. They observed that DNSSEC utilizes twice the amount of memory and 1.1 to 2 times more CPU resources than DNS. However, they state that these amounts of CPU and memory usage should not be a problem for modern computer systems as these systems are generally equipped with lots of memory and a high performance processor.

4.2 Secure browsing through HTTPS

Ifoodbag is a web application. So in a typical use case, an *ifoodbag* user will access and browse the *ifoodbag* web application with the aid of a web browser using the HTTP protocol. Earlier in section 3.3 we noted that HTTP is *not* a secure protocol and that either HTTPS or S-HTTP should be used to provide improved security. We recommend *ifoodbag* use HTTPS to secure all of the web communication with its users. *Ifoodbag* could also use S-HTTP, but as this is rarely used in today's internet community this is not a recommended solution.

HTTPS uses public key cryptography, which relies on complex mathematical operations to implement encryption, decryption, digital signatures, etc. These operations may require more computing resources (e.g. physical memory and CPU resources) as well as can take some processing time, which will create a delay in the client-server request-response process. However, the question is how large this overhead is and if the delay is acceptable to the user, i.e, if the user experiences a satisfactory web browsing experience. We examined existing research regarding this performance.

In [182] Arthur Goldberg, Robert Buff, and Andrew Schmitt mention that transferring data using HTTPS has little performance penalty. They worked with Microsoft's Internet

* Latency = query initiated by client goes to authoritative name server + authoritative name server resolve the query and send response back to the client + client receives the response

Information Server (IIS) and Netscape Enterprise web server applications. They observed that the server side encryption process in HTTPS increases the response time by at most 22% for Netscape and 17% for Microsoft IIS. They have also observed that data transfer rate decreases as the level of encryption increases (e.g. changing a stream cipher RC4 key length from 40-bit to 128-bit decreases the throughput). However, they consider that their observed delay is acceptable due to the security that HTTPS provides.

In another study [183] of the security cost of HTTPS, Xubin He observed that HTTPS requires more system resources on the client side than standard HTTP. This is because server certificate verification, SSL encryption, etc. are handled in the client and these functions are **not** required for HTTP. He also observed that the server response time for HTTPS is always greater than for HTTP. For HTTP, the server response time increases rapidly as the number of concurrent clients increase. Once the server is saturated in terms of throughput, HTTPS has a 33% throughput reduction in comparison with HTTP in terms of number of the number of new connections per second that can be handled by the server.

Based on our study of the literature we conclude that for an e-commerce business site such as *ifoodbag*, where security is more important than performance, the increased response time of HTTPS is both acceptable and prudent.

4.3 Secure Cloud Storage

In cloud storage, data resides on multiple third party servers. Cloud storage providers' claim that they provide appropriate security, but no one really believes them [184]. Security and reliability are two biggest concern about cloud storage [185]. Data traveling through network or stored in the cloud storage in plain text is a serious security threat [184].

According to Vormetric [186] there are complex data security challenge in the cloud.

- The need of protect confidential business or regulatory data.
- Multiple clients are sharing the same cloud service.
- Data mobility and legal issues related to government rules as the EU Data Privacy Directive.
- Lack of standards of how cloud service providers securely recycle disk space and erase existing data.
- Auditing, reporting and compliance concerns
- Third party insider who doesn't work with company and can have visibility and control over data.

In order to maintain and insecure secure data most system uses different techniques and combined them. These include Encryption, Authentication and Authorization[185].

Encryption: We can encrypt our data by using any different cryptographic algorithm. Though it is possible to decrypt, encrypted file but most of the hackers do not have access to the amount of computer processing power that requires decrypting the information[185].

Authentication: User name and password is required[185].

Authorization: Usually client will provide the list of the people who have the power to access the stored information. There are many companies who apply multiple level of authorization power. For example: a general employee may have limited access to data that stored over cloud while HR manager have more extensive access on data[185]

Companies like ifoodbag where they have decided to place their data in the cloud yet they want to make sure their data is safe they have to consider above techniques to ensure their data security. We recommend ifoodbag to encrypt their data by using AES cryptographic algorithm and also follow the Swedish Data Protection law and other laws and regulations.

4.4 Method to Secure a DDB in the Cloud

Ifoodbag is an e-commerce based company, where customer chooses products online. Ifoodbag will manage their own database in which they store product information, customer information, and employee information. From a general point of view we know that all of this information is crucial for any e-commerce business, thus we have to protect the data stored in this database.

As ifoodbag is a small e-business company, we have to keep in mind how many customers', employees', and products' information will be stored in this database and how this information can be managed in the most secure way without taking too much time, resources, or cost. Considering ifoodbag's size and capability we have decided to secure ifoodbag's database by enforcing policy based security in a distributed database and storing encrypted data in database.

In [187], Neera Batra and Manpreet Singh discuss multilevel policy based security in a distributed database. According to these authors, a distributed database system's functions include distributed query management, distributed transaction processing, distributed meta data management, and enforcing security and integrity across the multiple nodes. Database security is the protection of the data stored inside the database from unauthorized access, malicious attacks, or accidental mistakes made by authorized individual. The most important issues in this security are authentication, identification, and access control. The database provides various layers and types of information security such as access control, auditing, authentication, and encryption. The five major security requirements for database management systems are: multi-level access control, confidentiality, reliability, integrity, and recovery. A complete data security solution must meet the following three requirements: (1) secrecy or confidentiality refers to the protection of data against unauthorized disclosure, (2) integrity refers to the prevention of unauthorized and improper data modification, and (3) availability refers to prevention and recovery from hardware or software error.

We recommended ifoodbag to encrypt sensitive data and store in database. *ifoodbag* will follow or set their own data policy to distinguish between sensitive and non sensitive data. For this encryption, we have used the AES encryption algorithm. The reason behind choosing this algorithm is that AES is more secure than DES[188]. Today DES is breakable through brute force attacks because of its small key size (56 bits). On the other hand AES can utilize keys of 128, 192, or 256 bits (a 128 bit key is already very difficult to break)[188].

4.5 Secure Memcached

One of the main drawbacks with memcached is its security. Memcached does not provide any form of security (Either authentication or encryption) [131]. Memcached is mainly built for speed, rather than for security. This means that anyone who can discover our memcache can read from it or write to it. If we want to secure Memcached we have to secure it by ourselves. Even though memcached does not come with built-in security, it is widely used by some of the most famous websites, including Facebook, Twitter, Github, etc. [189]

First, we examine why we should use memcached and then we will look at what are the most common security features that we can enable in order to secure memcached. Memcached has some features [190] that motivate us to choose it for ifoodbag's web caching:

Easy Scalability	Memcached needs minimal configuration to add a new node, where no special interconnect requires.
Minimal impact on node failure	When a particular node does fail, it has almost no impact on the overall cache other than reducing the available memory.
Multiple available clients	Memcached client API's supported by various language like PHP, C++, Java, Python, Ruby, Perl, .NET, Erlang, ColdFusion and many more.
Cross-platform	Memcached is available for wide variety of platform including Linux, BSD and Windows.
Architecture flexibility	Memcache have no restriction on all nodes to have a uniform cache size. The nodes with less physical memory can be set up to contribute 512MB of our cluster. And other nodes may have 2GB of memory for the Memcached instance. We can also run more than one instance of Memcached on a single node.
Multi-fetch	Instead of querying them in a loop one by one, we can request values for more than one key at once. If we query one by one it takes lot of network round trips.
Constant time function	It is a single key or hundred it takes same amount of time to perform an operation in memory.

We can secure Memcached by using two common features:

Firewall We can always start with Firewall. It will only allow the data that we want to pass through[132]. Even if we say we want to only save public information not private information, still we will want our memcached daemon protected and not open to any DoS attack. If we want to control access to any kind of data it's always advisable to use external protection like iptable or firewall rules[191].The reason behind, memcached doesn't come with any built-in authentication and it does not require any sort of user name or password[191].

We can always start with Firewall. It will only allow the data that we want to pass through[132]. Even if we say we want to only save public information not private information, still we will want our memcached daemon protected and not open to any DoS attack. If we want to control access to any kind of data it's always advisable to use external protection like iptable or firewall rules[191].The reason behind, memcached doesn't come with any built-in authentication and it does not require any sort of user name or password[191].

SASL Usually deployment of memcached is within trusted network. But if we want to place memcached in hostile network or in case administrator would like to have control over clients in those cases we can use SASL to secure memcached [192]. Latest Version of memcached support SASL authentication. This SASL framework provide authentication and give data security[193].

SASL adds authentication support to connection based protocols, for example: Memcached client binary protocol in our case. The way SASL works is that there is function (API) call that is added to the client that provides functionality for identifying and authenticating a given user. SASL enable protection during the entire connection. Once a user is identified and

authenticated, SASL provides a security layer between the client protocol and connection[194].

Previously Memcached had no authentication layer. Later binary protocol and SASL was added[194] Anyone was able to read the memcached data if they can gain the access over memcached. Therefore it was actually web developer's job to make sure that, their network was locked and they also design secure application. Then the binary protocol was added to memcached for more efficient and compact client-server connection[194]. This actually made possible to add SASL support also. With SASL support, function was added to the server and forced client connection to authenticate itself before the whole connection can continue[194].This requires a client, which sent authentication information before using SASL. At the beginning only spymemcached client (java) was supported. But now libmemcached also has SASL support. Basically SASL uses some sort of storage where is stores user credential. For this kind of storage it can use LDAP or SQL database[194].

Basically user credential will be stored in a database file on the server that memcached will be running on. When a user wants to connect to the memcached server, it will not allow the user to connect to the server unless the user provides their credential. And those given credential must match the user credential that already stored in SASL database [194].

From the discussion above, we observe that memcached gives better performance than security. However, in any e-commerce business security is the highest priority provided that we can achieve sufficient performance. Ifoodbag's main concern is also security. From this discussion, we can see that, even though memcached does not come with built-in security, we can still configure an appropriate security feature . In this case, we recommend firewall or SASL authentication for securing memcached.

5 Implementation

Based on the desired functionality we have split the proposed design to implement ifoodbag's web application in a cloud environment into seven different modules (see Chapter 3). Then we identify potential security threats for each of these modules. Next, we study available mechanisms to prevent the identified security threats. We wanted to know how implementing the security mechanisms could affect the ifoodbag web application's performance. To learn this we have applied two different approaches: (1) studied work done by other researchers and (2) implemented some of these mechanisms and observed their performance empirically. This chapter presents the implementation process and our empirical observations by splitting the investigation into the process of secure browsing (using the ifoodbag web application), securing the communication between the nodes within the system, and security for the database. For each of these we consider the motivation for this part of the solution and describe an experimental setup to measure the performance of this part of the solution.

5.1 Secure browsing of ifoodbag web application

From the background presented in Chapter 2, we see that the HTTP protocol (used together with a browser application to realize the World Wide Web) does not have any security mechanism. We found two alternative mechanisms to provide security in HTTP communication: (1) HTTPS and (2) S-HTTP. HTTPS is widely used in today's internet world, thus we will focus on it. HTTPS provides security for HTTP communication by the use of the Secured Socket Layer (SSL). SSL is a cryptographic protocol that uses public-key cryptography for confidentiality, integrity, and certificates for authentication. These mechanisms involve complex mathematical operations. These mathematical operations might add undesirable overhead to the overall communication process performed by HTTP protocol. We aim to measure this overhead.

Figure 5-1 shows the experiment setup to measure the performance overhead for HTTPS. Section 6.2.3 describes the results of this test using the experimental setup listed in

Table 5-1.

In our experiment, we download content to an HTTP-client (running in a client machine) from a web server and measure the total time required to perform this operation. We start counting time when we send a request to the webserver and stop counting time when we receive all of the content from the server. In our experiment we consider only server response-time (i.e. time requires to receives content from server) as the matrix, while comparing performance of HTTP and HTTPS protocol. We do not observe usage of other computing resources (e.g., processor, memory, storage) and count them as matrices to measure performance.

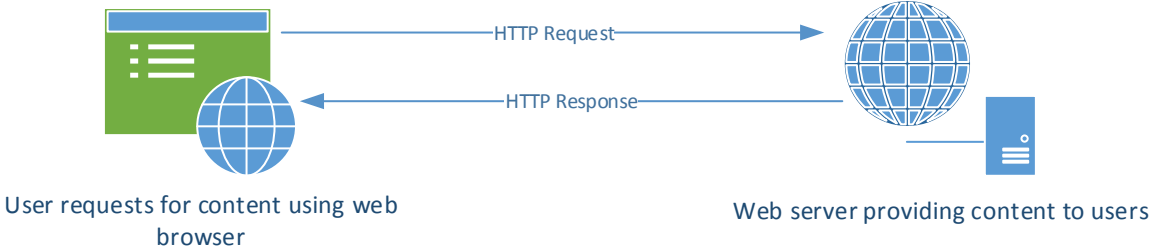


Figure 5-1: HTTP & HTTPS experiment setup

Table 5-1: HTTP and HTTPS experimental configuration

Network	Client and Server both reside in the same Local Area Network (LAN). LAN connection speed 1.0 Gbps. Did not perform any other network transaction (i.e. upload, download, web-site browsing) to leave the full bandwidth available for the experiment traffic.
Web client	As web client we could use any web browser (e.g. Internet Explorer, Google Chrome). Instead we have used WebClient class library [195] from .Net Framework 4.5. The reason was ease of measuring request/response time easily and accurately.
Web server	IIS 7.5.7600.16385
Client machine	Operating System: Windows 8 (64-bit) RAM: 8.00 GB Processor: Intel(R) Core(TM) i7 CPU M 620 @ 2.67GHz 2.66 GHz, x64-based processor
Server machine	Operating System: Windows Server 2008 R2 Enterprise (64-bit) RAM: 1.48 GB Processor: Intel(R) Core(TM) i7 CPU M 620 @ 2.67GHz 2.66 GHz, x64-based processor
Tools	We have written a tool to send data to and receive data from a web server as well as measure time required for this process. The tool writes the time in a text file. The tool was written with the C# programming language and Microsoft .Net framework 4.5 was used. We have used Microsoft's Visual studio 2012 as an Integrated Development Environment (IDE). Cached data was cleared from the system before each session. The source code for this tool is presented in Appendix D. The prime reason for selecting C# programming language to develop the tool was the experience of the author with this programming language. However, another programming language such as Java or Python could have been used.
Plots	To create plots from the collected data we have used Microsoft's Office Excel 2010.

5.2 Secure Communication between Nodes

In the proposed design to implement ifoodbag's web application in a cloud environment, nodes in different tiers are connected to each other. Management nodes are connected to virtual machine (VM) instances in the application tier (See Figure 5-2). A management node uses this connection to execute a policy for scaling up and down the number of nodes in the application tier. Additionally, the nodes in the application tier have connections with nodes in the database, caching, and load balancing tiers. In the proposed design for the ifoodbag cloud, all of these connections are unprotected. Based upon our study we suggest that a VPN tunnel can be used to protect all of these communication channels (see Section 3.4). However, using a VPN might add undesired overhead to the normal communication process between nodes, which could negatively affect the overall application's performance and user experience. In order to assess this effect we performed an experiment to measure the overhead for using a VPN tunnel.

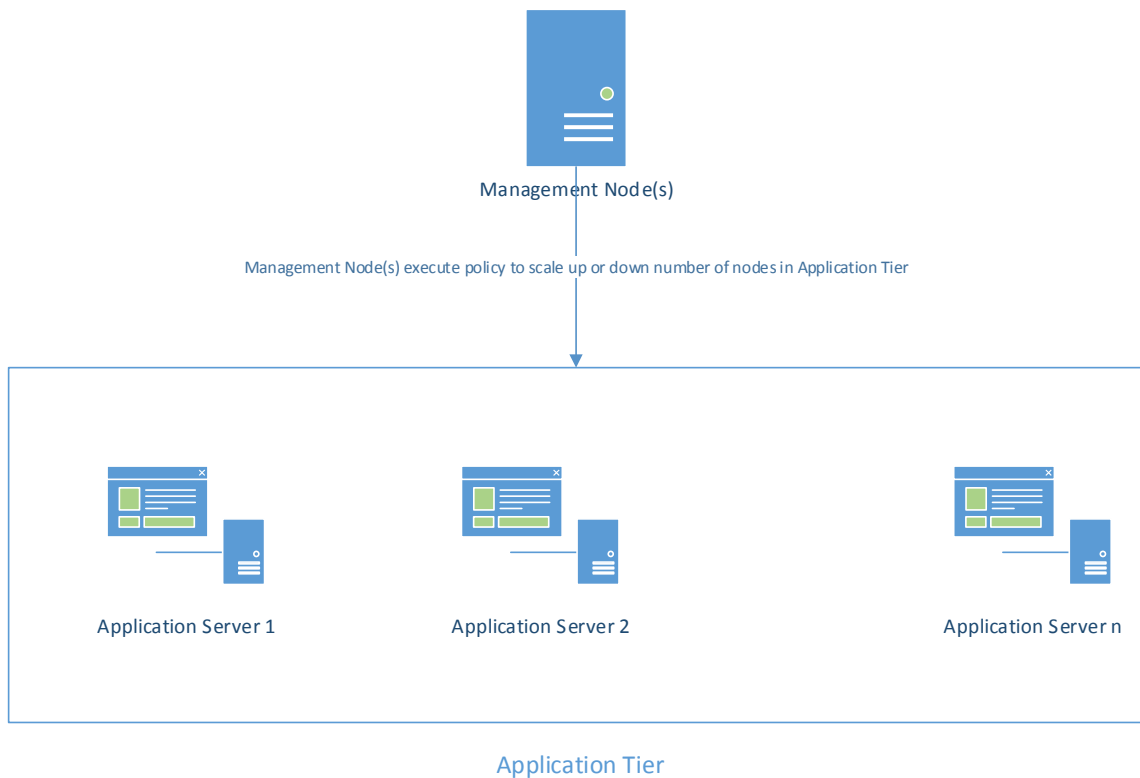


Figure 5-2: Management Node(s) execute policy to scale up or down number of nodes in Application tier

Figure 5-3 below depicts the experiment setup.



Figure 5-3: VPN Experiment scenario

We connect two nodes through a VPN tunnel and then from one of these nodes we send SQL SELECT database queries to the other node, which acts as a database server. We measure the time from sending a query until receiving a response. Section 6.2.4 describes the results of this test using the experimental setup listed in Table 5-2. In our experiment we consider only the time, required to receive query result from database server, as the matrix while comparing performance of different secure connection mechanisms between nodes. We do not observe usage of other computing resources (e.g., processor, memory, storage) and count them as matrices to measure performance.

Table 5-2: VPN experimental configuration

Server machine	Operating System: Windows Server 2008 R2 Enterprise (64-bit) RAM: 1.48 GB Processor: Intel(R) Core(TM) i7 CPU M 620 @ 2.67GHz 2.66 GHz, x64-based processor
Client machine	Operating System: Windows 8 (64-bit) RAM: 8.00 GB Processor: Intel(R) Core(TM) i7 CPU M 620 @ 2.67GHz 2.66 GHz, x64-based processor
Network	Client and Server both reside in the same Local Area Network (LAN). LAN connection speed 1.0 Gbps. Did not perform any other network transaction (i.e. upload, download, web-site browsing) to leave the full bandwidth available for the experiment traffic.
Database	Microsoft SQL Server 2008 R2
VPN application	We have measured data for OpenVPN (version 2.1.3) and IPSec VPN tunnel. We have followed [196] to configure OpenVPN for our experiment.
Tools	To run the SQL query from a remote client we could use any database administrative tool (e.g. Toad [197], Navicat [198]). Instead we have used our own tool. This tool start counting time just after sending a SQL SELECT query and stop just after getting query result from Database server. The reason for using our own tool was to measure the request/response time easily. The tool was written in the C# programming language and Microsoft .Net framework 4.5 was used. We have used Microsoft's Visual studio 2012 as an Integrated Development Environment (IDE). The code for tool is presented in Appendix C.

5.3 Secure information in database

According to the proposed cloud design, ifoodbag will store all the data in a distributed database. Part of this data will be business related, which ifoodbag would not like to make public. For example, customers information, employees salary data. Ifoobdag can encrypt these data in the database to keep data confidentiality. Encrypting data while writing in the database and decrypting data while reading from database might add overhead in the data read/write process. So we perform experiment to check if there is any overhead. We use AES encryption in our experiment, as it is well studied and more secure than other algorithms (e.g. DES).

Figure 5-4 shows the experiment setup. In order to avoid any limitations due to the network our client and server were running in same local machine. We have used Windows, Apache, MySQL, and PHP (WAMP) as our server and Toad as our Client. First we stored plain text data and calculated the query time and then we encrypt and decrypt data and calculate the time of for the query including the time to do encryption and decryption of data. Section 6.2.6 describes the results of this test using the experimental setup listed in Table 5-3.

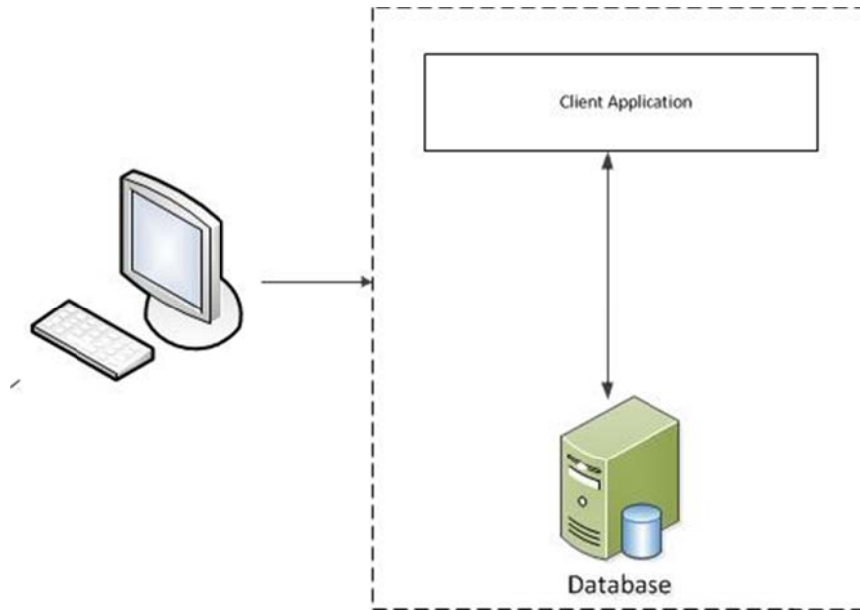


Figure 5-4: Database experiment scenario

Table 5-3: Database experimental configuration

Network	The client and server both resided in the same local machine, hence there was no limitation or delay due to a network connection.
Server Machine	Operating system: Windows 7 Ultimate. System type: (32bit). Ram 2.00GB. Processor: Intel(R) Core(TM)2 Duo CPU P7550 @ 2.26 GHz 2.26 GHz
Client Machine	Same as server machine
Server Software	For server software WAMP Server (32 Bits & PHP 5.3) 2.2E [199] was used.
Client Software	For client software TOAD for MySQL 7.0 [197] was used.
Tools	A tool was written in order to encrypt and decrypt data in database as well as measure the time required when using plain text data and encrypted and decrypted data. The tool writes the time in a temporary file. The tool was written with C# programming language. Microsoft visual studio C# 2010 Express was used to run the code and integrate the environment. The source code for this tool is presented in Appendix I.

5.4 Secure information in Cloud Storage

Ifodbag have decided to use cloud storage to store their file, images, financial record and other data. As we know in cloud storage, data resides on multiple third party servers. Cloud storage providers' claim that they provide appropriate security, but no one really believes them [184]. Ifodbag send all their data as a plain text file it will lead to serious data security threats. Since those data are business related, ifodbag want to make sure their data are safe while traveling over network and also while reside on multiple third party server. Ifodbag can encrypt those file before send them and make sure their data confidentiality and integrity. The file that will travel over network will have different size. And each different type of file size takes different encryption and decryption time. We use two type of different cryptographic algorithm: AES and 3DES (they are well studied and more popular than other algorithm) to see the time difference between them.

Figure 5-5 shows the experiment setup. We run our experiment through network in order to get real time data. For this experiment we used Windows and Microsoft visual c# 2010 express .At first we connected two node over network. Then we encrypted a file push and over network and stored in our desired location. After that we pull the file and decrypt and read it in client side. In our experiment we consider only the time, required to receive and send encrypted file over the network. We do not observe usage of other computing resources (e.g., processor, memory, storage) and count them as matrices to measure performance. Section 6.2.7 describe the result of this experiment and our experimental setup listed in Table 5-4.



Figure 5-5: Cloud storage experiment scenario

Table 5-4: Cloud storage experiment configuration

Network	Client and Server both reside in the same Local Area Network (LAN). And we used wireless LAN for this experiment. LAN connection speed 54MBPs. Did not perform any other network transaction (i.e. upload, download, web-site browsing) to leave the full bandwidth available for the experiment traffic.
Client machine	Operating System: Windows 7 Professional (64-bit) RAM: 3.00 GB Processor: Pentium(R) DualCore CPU T4200@2.00GHz 2.00GHz
Server machine	Operating system: Windows 7 Ultimate. System type: (32bit). Ram 2.00GB. Processor: Intel(R) Core(TM)2 Duo CPU P7550 @ 2.26 GHz 2.26 GHz
Tools	We have written a tool to send file to and receive file from server as well as measure time required for this process. The tool writes the time in a text file. The tool was written with the C# programming language. and. We have used Microsoft's Visual C# 2010 Express. The source code for this tool is presented in Appendix L.
Plots	To create plots from the collected data we have used Microsoft's Office Excel 2010.

6 Results and evaluation

The chapter begins by revisiting some of the security issues given the proposed ifoodbag cloud design and then gives our recommendations for some security mechanisms that should be adopted to address each of the issues. Some of our recommendations are based on the research works previously performed by other researchers. While for some other recommendations are based on our empirical observations. We compare our empirical observation with other research works in case we find any other similar research has been done. We also mention the reasons for recommending a particular security mechanism and rejecting other alternative options. Next a set of security guidelines are given. The chapter ends with a set of general guidelines that we suggest should be adopted by ifoodbag.

6.1 Revisit design issues

In this section, we briefly review the security issues we identified in the cloud architecture proposed for ifoodbag web application.

The first security issue we identified is in the name resolving process. As the DNS protocol was not designed with any security mechanisms, an attacker can exploit this lack of security in the DNS protocol to fool ifoodbag's customers into using the attacker's web service. An attacker may attack the DNS server itself or alter DNS messages between the user (in this case an ifoodbag customer) and the DNS server. In this way, instead of original ifoodbag web server, the user will be connected to a server of the attacker's choice.

Another problem we identified concerned web browsing. Typically, a user will use an HTTP client (i.e. a web browser) to connect to the ifoodbag web application. HTTP is a plain text protocol and it does not include an appropriate security mechanism to protect the user's communication session, thus an attacker can read and modify the messages sent between the user's HTTP client and the web server.

In the proposed architecture for the ifoodbag web service, nodes in different tiers (e.g. VM instances, database servers, and load balancers) are connected to a management node. This management node executes the policies to manage the number of VM instances (see Figure 6-1). The VM instances in *application tier* are connected to and communicate with nodes in the *caching tier*, as well as nodes in the *database tier*, in order to store and retrieve information (e.g. order information, billing information, and user information). All of this communication between nodes in the different tiers is crucial to provide the proper service to ifoodbag's customers; hence the communication channel between these nodes needs to be secured.

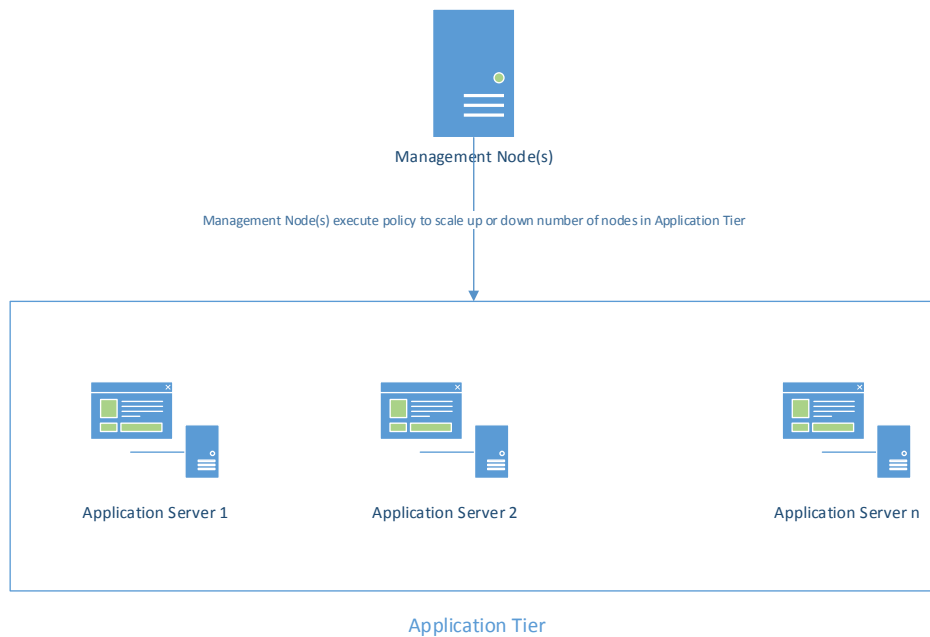


Figure 6-1: Management Node(s) execute policy to scale up or down number of nodes in application tier

In the proposed cloud architecture load-balancers play an important role to uniformly distribute load among available application servers making the cloud system scalable. Given this important role of the load-balancers, the security of these load-balancers also needs to be improved.

In the proposed cloud architecture, a distributed database is used. Ifoodbag stores their information in this database. The information that will be stored in this cloud-based distributed database ranges from highly sensitive data to non-sensitive data. For any company, maintaining the confidentiality and integrity of data is always top priority, thus maintaining data security is important. We recommend that ifoodbag encrypt their data using AES encryption, hence encrypted data is stored in the distributed database.

In the proposed design Memcached is used for caching. Memcached provides good performance, but does not come with built-in security. We suggest that ifoodbag configure Memcached with SASL authentication. This allows them to place the Memcached cache even in a hostile network. Fortunately, the latest version of Memcached provides this feature.

As ifoodbag will store their data in cloud storage, their data will reside in multiple third party servers. Data travelling through network or stored in cloud storage as plain text would lead to serious security threats, hence we recommend that ifoodbag use AES cryptography to encrypt and decrypt data to be stored in cloud storage.

6.2 Recommendations

We split our recommendations into seven sets of recommendations based upon the division of the solution into seven parts. Each of these will be the topic of one of the following subsections

6.2.1 For Secure Name Resolving

The importance of security in name resolving process and, security issues in current DNS protocol were discussed in earlier (see Section 3.1, Chapter 3). For securing the name resolving process we recommend ifoodbag use DNSSEC with its domain's authoritative name server. We recommend using the NSD name server software for implementing DNSSEC.

We make this suggestion since we have found that DNSSEC is the most prominent and commonly used mechanism for securing the name resolving process. We did not find another solution that could secure the DNS name resolving process.

The reason for selecting the NSD name server software from among the many alternatives is based on our study of [181] where a performance comparison of NSD with other name server software is given. This research observed that NSD performs better than BIND and UNBOUND.

In our studies we have found that implementing DNSSEC *will* add overhead to the DNS service. In addition to the added overhead to the individual DNS queries and responses, DNSSEC also consumes more computing resources (e.g. memory and CPU time). Slower name resolving can slow down browsing; however we recommend ifoodbag use DNSSEC due the increased security it provides. Fortunately, browsers and hosts do DNS caching and it is likely that the DNS responses will have a long validity; hence the practical effect of DNSSEC's extra overhead and additional processing will be small. Additionally, according to Moore's law [200] "*the overall processing power of computers will double in every two years*", hence we believe that computers will be sufficiently capable that DNSSEC can be used without user observable performance loss.

One concern is that if ifoodbag simply uses DNSSEC for its domain's authoritative name server this does not ensure that the overall name resolving process is secure! This is because many name servers (both caching and authoritative name servers) in today's internet world are still not using DNSSEC. As a result, clients might use a compromised caching name server leading to a security threat in the name resolving process. Therefore, it is essential for internet users and administrators to be aware of the importance of secure name resolving and to start using DNSSEC by default.

6.2.2 For Securing Load Balancers

An Ifoodbag network administrator will set some policies in the management node to add or remove VM instances in the application tier based on number of concurrent users. The management node executes these pre-set policies to scale up or scale down the cloud environment in order to make optimal use of available computing resources (and to avoid unnecessary expense for the company). The load balancer uniformly distributes user requests among the available application servers (i.e. nodes in application tier). Due to limitations in computing resource (i.e. memory, CPU, storage) all the nodes in application tier can only handle a certain number of concurrent user requests. If the load balancer is compromised, then some nodes will not receive any user requests, while other nodes will be overloaded with user requests (i.e. when the number of concurrent user requests exceed the node's handling capacity). It is possible that the overloaded server will not be able to respond to all the user requests in time, which will cause a bad user experience. The network bandwidth of the overloaded node can be fully occupied, thus the server will become inaccessible at a certain point in this traffic load. The unavailability of this server will cause unavailability of ifoodbag web service to the user, which as a consequence might violate the service level agreement (SLA). Additionally, unless traffic is prioritized the management nodes might not be able to communicate with these overloaded servers. It is clear that in the ifoodbag cloud load balancers play an important role in making the system scalable and hence it is important to secure these load balancers.

We recommend that ifoodbag follow the security policies prescribed in section 3.2.3 of this report. In addition to using squid, ifoodbag might consider the other mechanisms we have presented in section 3.2.4.

6.2.3 For Secure Browsing

We recommend ifoodbag use HTTPS for the secure browsing of ifoodbag’s web application. HTTPS uses SSL, which is a public key based security protocol. SSL uses certificates for authentication purposes and encryption for message privacy and confidentiality. Ifoodbag could use only authentication of server for users browsing the website or accessing publicly available information. However, for logging-in registered customers, ifoodbag should use strong two-factor authentication. In two-factor authentication mode, a user has to provide the server a valid client certificate issued by a trusted certificate authority, along with a valid username-password pair. In this way authentication is based on ‘something the user knows’ (i.e. password) as well as ‘something the user has’ (i.e. a certificate). In this way, the client and server can *mutually* authenticate each other. Mutual authentication allows the user to know that they are really communicating with the ifoodbag web server and allows the ifoodbag web server that to know that it is communicating with a specific ifoodbag customer. Ifoodbag can also use the security guidelines provided by a web server software provider or developer. For example, Microsoft has security guidance for IIS [201].

HTTPS provides transport layer security with cryptographic protocol SSL or its descendent Transport Layer Security (TLS). This cryptographic operation adds some overhead to HTTP browsing.

As described in section 5.1 and as shown in Figure 6-2 we use experiments to assess the effect of using HTTPS and compare it to use HTTP. In our experiments, we download content to an HTTP-client (running in a client machine) from a web server and measure the total time required to perform this operation. The time counting starts when we send a request to webserver and stops when we receive all of the content from the server. The same experiment is being repeated multiple times (27 times for a large file and 18 times for a small file). We wanted to observe if the overhead for HTTPS varies with content size, so we have performed the experiment for two different sized files: one file was 3.13MB and another one was 390MB in size (i.e., much bigger than the first file). The files’ extension was ‘.mov’.

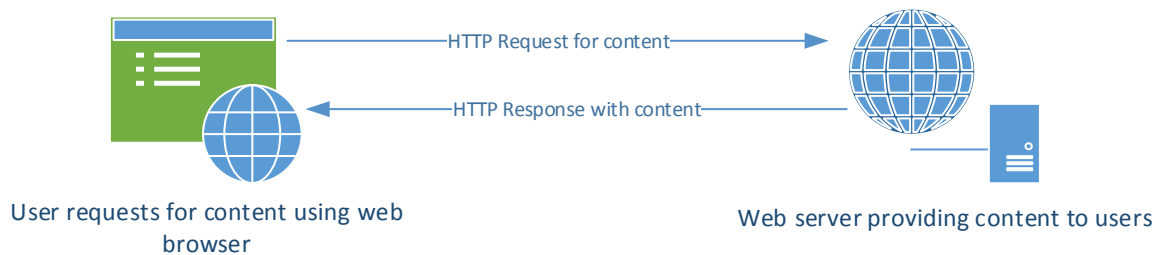


Figure 6-2: HTTP & HTTPS experiment setup

We plot our experimental results in Figure 6-3. To make the figure clearer we have plotted only a subset of the total results (the complete set of data can be found in Appendices E and F. In these plots, the X-axis represents different iterations of the test and the Y-axis represents time (T) in microseconds.

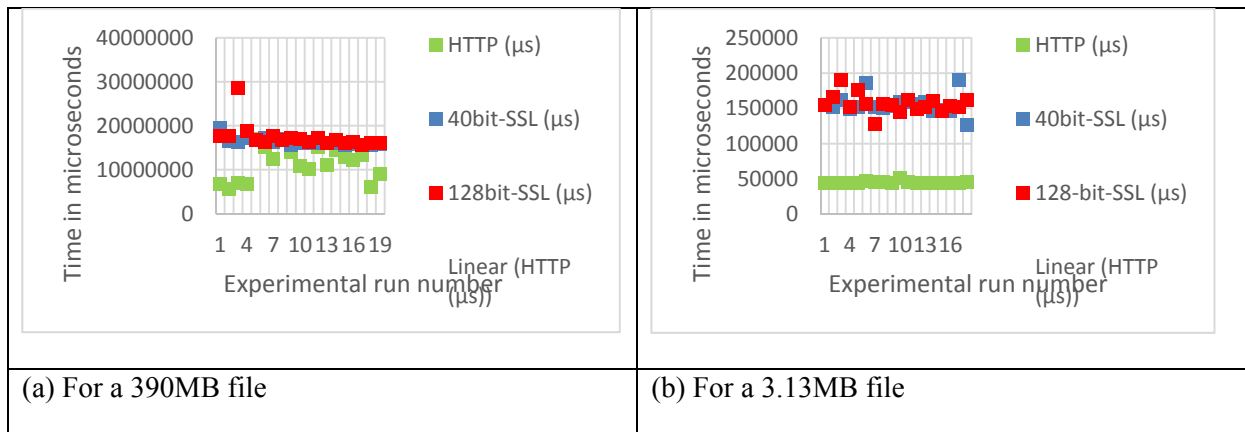


Figure 6-3: Download time comparison for HTTP and HTTPS (40bit & 128bit): (a) for a 390MB file and (b) for a 3.13MB file

In the figures above (Figure 6-3) we can clearly see that a data transfer using HTTPS takes longer time than when using HTTP.

128 bit encryption is more secure than 40 bit encryption. Besides more than 99% of modern web browsers support 128 bit encryption [202]. So in our analysis henceforth we only focus on HTTPS with 128bit SSL and write just HTTPS.

We calculate average* data-transfer time for HTTP and HTTPS and present this data in Table 6-1.

Table 6-1: HTTP and HTTPS data transfer times

File size	HTTP	HTTPS	HTTPS overhead
3.13MB	45 millisecond	156 millisecond	$156 / 45 \sim 3$ times
390 MB	11 second	17 second	$17 / 11 = 1.5$ times

From the average data transfer time we see that for a 3.13MB sized file that HTTP is three times faster than HTTPS. However, for a 390MB file HTTP is only 1.5 times faster than HTTPS. This indicates that with increasing amounts of data HTTPS performs proportionally better. Our observation in this experiment is aligned with the experiment in [182] and [183] where they have observed little performance penalty with HTTPS.

Analyzing our collected data we observe that, in our experiment required time for creating a single HTTPS session took about 24 millisecond. After creating secured session HTTPS took 130 millisecond for transferring per byte data. We also observe that, in our experiment, per byte data transfer using normal HTTP protocol took about 82 millisecond. While implementing secure cloud environment, ifoodbag can use this experiment data (e.g. secured session creation time, per unit data transfer time) for anticipating the cost of security in their system, based on the size and type of data they are going to secure.

From our analysis of the experiment data† we can see that the overhead of HTTPS consists of two main parts: (1) initial additional overhead to establish the session key and (2) the additional per SSL/TLS segment processing time. As a result, even though most users' individual interactions with the web service will be small – the initial overhead only occurs once per session‡.

* Average time calculation is done by the arithmetic mean formula: Average = sum of all data / total number of data.

† Detail analysis available in Appendix L.

‡ Note that if the user transfers a very large amount of data it may be necessary to incur additional key exchange overhead due to the need to re-key. However, in practice this effect is not expected to be visible to the user.

Reducing browsing time, by delivering content to the user as quickly as possible, is desirable for any web application. While security is also an important concern, specially in e-commerce. But security comes with cost. Nowadays some big IT organizations (such as Google, Amazon, and Facebook) are using HTTPS for secure browsing of their website. Given the magnitude of the overhead and the fact that others have adopted HTTPS we recommend that ifoodbag use this mechanism for secure browsing of their ifoodbag web application.

6.2.4 For Secure Communication between Nodes

To secure the communication between nodes (e.g. between the web server and database server) ifoodbag can use IPSec. IPSec is an IETF standard for securing Internet Protocol (IP) communications. IPSec provides data confidentiality (through encryption) and authentication (through a cryptographic hash). IPSec works either in transport mode or in tunnel mode [203]. Tunnel mode is for use between a host (e.g. a computer) and a security gateway. For example, a host can securely connect to a corporate network via the internet using tunnel mode. Transport mode secures a connection between two hosts. Ifoodbag can use IPSec in transport mode to secure connection between its nodes.

The concept of securing a connection between nodes (e.g. webserver and database server) using IPSec is well described by Microsoft in [204]. Microsoft states that IPSec adds additional overhead to establish & maintain secure connections and can introduce increased network latency; hence, they do not recommend using IPSec for securing all traffic in a network.

Alternatively, ifoodbag can also use higher layer VPN tunnel to secure the communication between nodes. Such a VPN tunnel provides end-to-end security between nodes using higher layer protocols, rather than operating at the IP layer as IPSec does.

We investigated the performance penalty of a VPN tunnel and IPSec empirically. From the available alternative VPN software implementations, we selected the OpenVPN implementation, as this is one of the most widely used open source implementations. OpenVPN uses the OpenSSL library [205] to provide the necessary cryptographic functions [206]. We have used an SSL certificate for authentication. However, we could use a pre-shared key or username password based authentication. For IPSec we have used a pre-shared key for authentication, but could also use IKE (Internet Key Exchange) X.509 certificates [207] instead. For ease of use we have used pre-shared key.

For our experiments, we set a VPN tunnel between two nodes and then sent a SQL SELECT query from one node to another node (which is acting as a database server). We count the time from sending a query to the server until we receive a response from it. Figure 6-4 depicts our experiment scenario. We set OpenVPN in TUN mode. Appendix A presents how we setup an IPSec policy for our experiment. Detailed information about setting up a firewall and IPSec policies are available in [208].



Figure 6-4: VPN Experiment scenario

In our experiment we have observed the following things:

1. Securing the communication between nodes through both OpenVPN tunnel and IPSec imposes overhead which is reflected in the maximum data-transfer rate (see Figure 6-5 for a plot of this in terms of the query time). Appendix G contains detail data from this experiment.

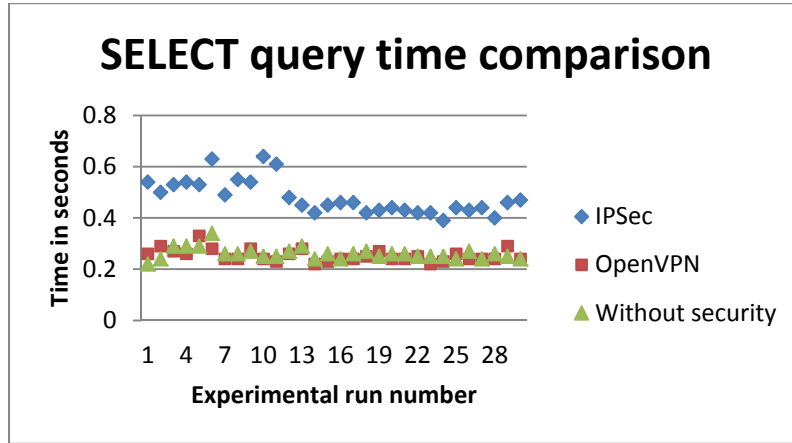


Figure 6-5: SELECT query time comparison among IPSec, OpenVPN, and normal traffic (i.e. without any security mechanism applied)

2. We have also observed that the query time depends on the data size. In Figure 6-6 below, we see that for both IPSec and OpenVPN, the time required to send and receive the SQL SELECT query result is higher for a bigger dataset (i.e. larger query result). Appendix H contains detail data from this experiment.

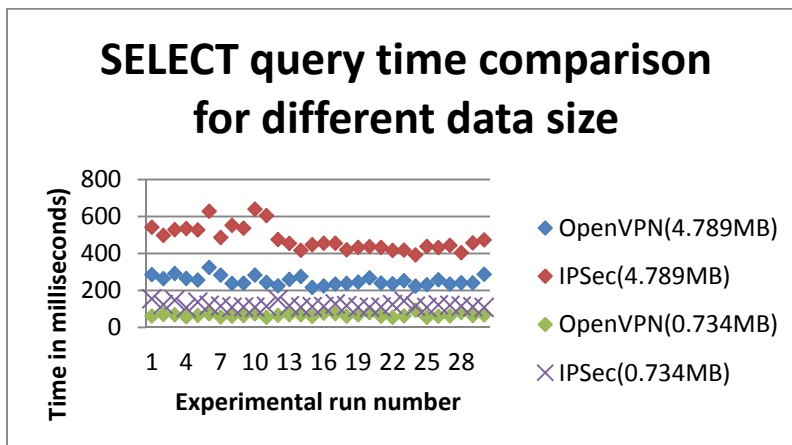


Figure 6-6: SQL SELECT query time for different data sizes

3. OpenVPN performs much faster than IPSec*. For OpenVPN the average time required to send a SQL SELECT query and receive result is 0.27 second, which is almost two times faster than IPSec (average time for IPSec is 0.48 second). We can see that OpenVPN does not add much overhead, as average time for OpenVPN is very close to average time required without a VPN tunnel, which is 0.25 second.

From the calculation of our collected experiment data we found single session creation time for OpenVPN is 32.93 millisecond and for IPSec it is 53.99 millisecond. We also calculate that, after creating secured session, per byte data transfer time for OpenVPN is 0.042

* Which is also supported by [206]. But we did not find any explicit performance comparison between IPSec and OpenVPN.

millisecond while it is 0.078 millisecond for IPsec. Detail of the calculation is presented in Appendix M (

4. Appendix M. Analysis of data in appendix H).

In our experiment data we observe that, regardless of the file size, per byte data transfer time for IPsec is twice than OpenVPN. We also observe that data transfer time for both IPsec and OpenVPN can be split into two main parts: (1) time required to create secured session and, (2) time required to transfer the actual content. So for transferring content of any size there will be an initial session creation time. Afterwards as the amount of the transferred data (using that session) increase, per byte data transfer rate will gradually decrease. We present our calculations in Appendix M (

Appendix M. Analysis of data in appendix H).

Now how it affects ifoodbag! Ifoodbag can have consistent secure connection between its nodes, whose session length reasonably will not be very short. Once a secured session is created, data can be transferred through that connection seamlessly*. So as a consequence, for ifoodbag per byte data transfer time through secured connection will not make a noticeable performance penalty. While implementing secure cloud environment, ifoodbag can use our experiment data (e.g. secured session creation time, per unit data transfer time) for anticipating the cost of security in their system, based on the size and type of data they are going to secure.

From our experimental observations we recommend ifoodbag use OpenVPN; as it provides security for node-to-node communication *without* incurring a significant performance penalty. Besides, OpenVPN is open source, so it is free to use. It is platform independent, so can be used in different (e.g. Windows, Linux) environments. We have also found OpenVPN very easy to configure.

6.2.5 Memcache

One of the main drawbacks with memcached is its poor security. Memcached does not provide any form of security (as it provides neither authentication nor encryption.) Memcached is built primarily for speed, not for security. However, we can configure memcached security by ourselves. Even though memcached does not provide built-in security but it does have many excellent features, such as scalability, multiple available clients, cross platform, multi-fetch, etc. These many features were the main reasons why we chose memcached for use with the ifoodbag cloud architecture.

Since our main concern is providing the best possible secure design for ifoodbag, we choose two mechanisms to secure memcached. As we discussed earlier in section 4.5, we can secure memcached with firewall. This is a very simple mechanism to apply. Usually deployment of memcached is *within* a trusted network. However, if we deploy memcached in an untrusted network or if an administrator wants greater control over clients, then we can use memcached together with SASL. We discussed in section 4.5 how to deploy memcached with SASL.

6.2.6 Distributed database

We mentioned earlier that, ifoodbag will store all their data in a database and part of that data will be business related data that is sensitive data. For ifoodbag this data is very important and they want to maintain confidentiality and integrity for all of their data. We recommended in section 4.4 that ifoodbag apply a multilevel security policy for their distributed database in order to improve the overall security of the database. A database security policy defines authorization, authentication, and access control policies. These policies should be designed to prevent unauthorized access, malicious attacks, or accidental mistakes. For storing highly sensitive data, we recommend ifoodbag to store only encrypted data in the database. For this encryption we recommend they use the Advanced Encryption Standard (AES) [209] algorithm. Another widely used encryption algorithm is DES[210]. We recommend AES over DES because of AES's larger key size (with keys of 128, 192, or 256bits). Currently 128 bits are considered practically unbreakable. On the other hand, DES has only a 56 bit key, making DES less secure than AES. In fact, DES can be broken in practice; hence, the United States of America's National Institute of Standards and Technology recommends that firms not use DES.

* There can be default session timeout. In that case after a certain period of time session need to be recreated. This session timeout can be adjusted while configuring secure connection for optimum output.

We performed data encryption using AES for two different size of database TABLES. We performed this experiment using a single machine, where both the client and server were running in same machine. In order to avoid any delay or limitations due to networking, we did not use any network connection for this experiment. The source code for this experiment, to measure AES encryption/ decryption time presented in Appendix I. The raw data from the experiment are in Appendices J and K.

For this experiment, we used two identical databases. We named them as Big_database and Small_database. These names are based on number of rows and columns in the table of the database. We first created a Small_database called **TESTDB**. Under TESTDB we created a table called Product_Info. In our Product_info table we created 3 columns and 100 rows, and then we record 100 Test data entries in this table. An example of the table is shown in Table 0-1.

Table 0-1: Table structure of small_database

Product_Info		
ID (int 11)	Name (VarChar 50)	Price (VarChar 100)
1	Chicken	100
2	Chicken	100
3	Chicken	100
4	Chicken	100

In our experiment for small table we have observed the following things:

1. From Table 0-2 we can see that the size of the plain data and encrypted data (encrypted using the AES algorithm) are the same.

Table 0-2: Table record information for small_database

Schema for product_info (Small table)		
Number of records	Total size of the table (Unencrpted)	Total size of the table (encrypted)
100	16384 bytes	16384 bytes

2. As shown in Table 0-3, the time difference between storing plain data and encrypted data in database is only *slightly* longer for storing the data with encryption. Here we can see clearly that, Insert Query for encrypted data takes more time than plain data. We observed that overhead for storing data with Encryption is 0.6%. We also see that size of the table does not play any role in this case.

Table 0-3: Data insert time with and without AES encryption for small_database

Table size	Insert with encryption	Insert without encryption
16384	244915 millisecond 244.915 second	243458 millisecond 243.458 second

- As shown in Table 0-4 the observed time difference between performing a Select Query with encryption and a select query without encryption is also small, although the Select Query with encryption takes slightly more time than a select query without encryption. Overhead for retrieving encrypted data is 1.66%.

Table 0-4: SELECT data from small_database with and without AES decryption

Table size	Select with encryption	Select without encryption
16384	162741 millisecond 162.741 second	160084 millisecond 160.084 second

- We observed that for small_database insert time with AES is 0.2391 milliseconds per block (16 byte). And 0.01494 millisecond per byte.
- Small_database select time with AES is 0.1586 milliseconds per block (16 byte) and 0.0099 milliseconds per byte.*

Now we perform the similar operations with our Big_database TESTDB, with 4 columns and 100 rows in our Product_table. An example of some entries in this table is shown in Table 0-5.

Table 0-5: Table structure of big_database

Product_info			
ID (int 11)	Name (VarChar 50)	Price (int 100)	Image (varchar500)
1	Chicken	100	XXXXXXXXXX
2	Chicken	100	XXXXXXXXXXXXXXXXXX
3	Chicken	100	XXXXXXXXXX

In our experiment with Big table we observed the following:

- From the Table 0-6 we can again see that, size of the total table is the same with or without encryption.

Table 0-6: Table data record information for big_database

Schema for product_info (Small table)		
Number of records	Total size of the table (Unencrypted)	Total size of the table (encrypted)
100	65536 bytes	65536 bytes

- Table 0-7 shows the observed time difference between an Insert Query with encryption and select query without encryption. Again the Insert Query with encryption takes slightly more time than the Insert Query without encryption.

Table 0-7: Data insert time with and without AES encryption for big_database

* Data analysis is available in Appendix S.

Table size	Insert with encryption	Insert without encryption
65536 byte	1755442millisecond 1755.442 second	218839 millisecond 218.839 second

3. Table 0-8 shows the observed time difference between a Select Query with encryption and select query without encryption. Again the Select Query with encryption takes slightly more time than a Select Query without encryption.

Table 0-8: Data select time with and without AES decryption for big_database

Table size	Select with encryption	Select without encryption
65536 byte	164713 millisecond 164.713 second	161951 millisecond 161.951 second

4. We observed that for Big_database insert time with AES is 0.4285 milliseconds per block (16 byte). And 0.0267 millisecond per byte.
5. Big_database select time with AES is 0.0402 milliseconds per block (16 byte) and 0.0025 milliseconds per byte.

Summary of the experiment:

1. This experiment utilized a client and server both running in the same machine. The time for Select and Insert Queries might increase if the client and server were connected via a network.
2. For our data, we stored (text and images). Storing multimedia (such as video clips) with much larger objects would likely increase the query times, but this has not been evaluated.
3. In both the large and the small databases, the size of the database was independent of whether the data was encrypted or not. Similarly the time to perform an Insert or a Select query was only slightly longer for the cases of encrypted data.

From our experiment we observe that it takes some additional time to encrypt and decrypt the data. Since ifoodbag is a small company and the initial size of their database will be relatively small (although with time this database will grow). The time to perform encryption and decryption per entry is very small and the information stored in the database is one crucial for the company, hence protection of the data inside this database is a primary concern, therefore we recommend that the additional time cost of encrypt and decryption is worthwhile and should be done.

6.2.7 Cloud Storage

Today nearly all companies have adopted the trend to store their data in the cloud. This trend is motivated because cloud storage gives the company access to data at any time & anywhere and cloud storage enables the company to outsource the effort to provider highly reliable and safe data storage. The requirements of *ifoodbag* are no different from these other companies.

In cloud storage, data resides on multiple third party servers. While cloud storage providers claim that they provide appropriate security, there is always the possibility of a security breach. Moreover, data traveling through a network or stored in cloud storage as plain text enables a potentially serious security threat. In order to ensure confidentiality and integrity of data in cloud storage, we have proposed that *ifoodbag* use data encryption for both stored and transmitted data.

We performed data encryption by using AES and 3DES cryptographic method. And we have run this experiment over network connection. For this experiment we choose two different types of files. We named them as Big File (4MB) and small file (29KB). For both file size we have encrypt and decrypt them by using AES and 3DES and compare the time. Raw data for this experiment are presented in Appendix P and Appendix Q .

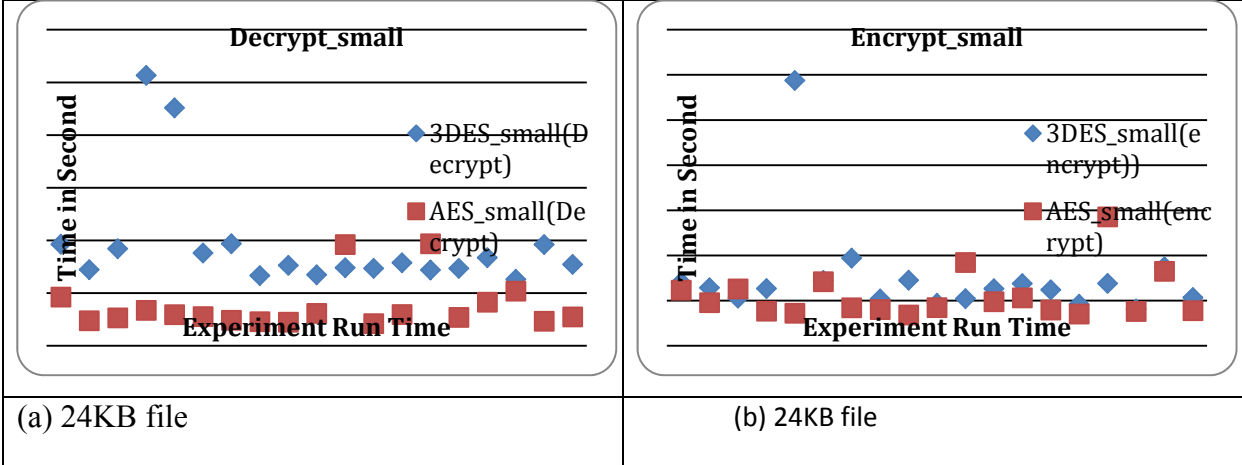


Figure 0-1: 24KB file Encryption/Decryption for AES and 3DES

- At first we encrypt and decrypt a small file by using two different encryption algorithm AES and 3DES. We run this experiment 100 times. We observed from this experiment is AES encryption and decryption is faster than 3DES encryption and decryption. For 24KB file AES encryption takes 35, 9206 miliseconds to encrypt per block (16 byte). It means it takes 2, 2450 milisecond to transfer one byte. Now at the other hand for the same file size 3DES took 24, 4854 milisecond to encrypt per block (8 byte). Which means it takes 3, 0606 milisecond to transfer one byte. Now if we observe per byte encryption time, 3DES takes more time than AES.
- For 24KB file AES decryption takes 48, 1905 miliseconds to decrypt per block (16 byte). It means it takes 3, 0119 milisecond to transfer one byte. At the other hand for the same file size 3DES took 62, 7341 milisecond to encrypt per block (8 byte). Which means it takes 7,8417 milisecond to transfer one byte. Now if we observe per byte decryption time 3DES takes more time than AES.

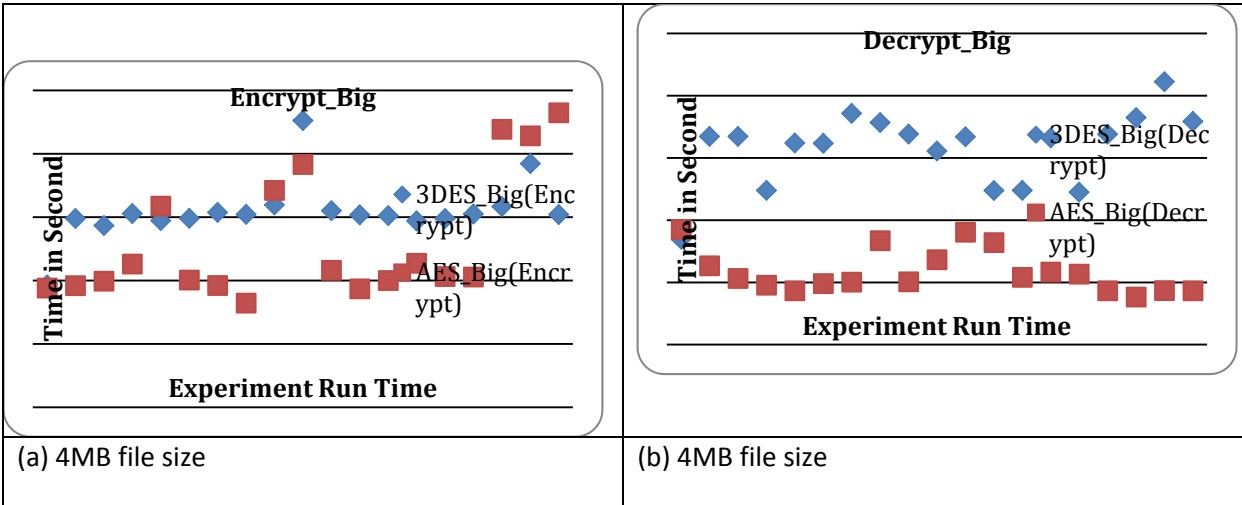


Figure 0-2: 4MB file Encryption/Decryption for AES and 3DES

Now we used 4MB file for storing and retrieving encrypted data in cloud storage by using AES and 3DES algorithm.

1. For this big file 3DES also takes more time than AES. We observed average time for AES to encrypt this file is 5242,861 milisecond and for 3DES is 6187,814 milisecond. We also calculate per block encryption time for both algorithms. AES takes 19,999 milisecond to encrypt per block (16 byte) which is 1, 2499 milisecond per byte. On the other hand 3DES takes 11,8023 milisecond to transfer per block (8 byte) which is 1,4752 milisecond per byte. It means 3DES takes little more time than AES to encrypt per byte data.
2. Average time for AES to decrypt this file is 5852, 79 milisecond and for 3DES is 15800, 18 milisecond. We also calculate per block decryption time for both algorithms. AES takes 22, 3266 milisecond to decrypt per block data (16 byte), which is 1, 3954 milisecond for one byte. 3DES takes 60, 2719 milisecond to decrypt per block (8byte), which is 3, 7670 milisecond per byte. For decrypting 3DES is taking more time than AES.

We have also run this experiment in local machine to avoid the network related delay. But we observed that in both case (with/ without network) there is significant time difference between encryption and decryption. Usually Encryption and decryption time for symmetric algorithm are often same. But the exception could happen while encrypting/decrypting multiple blocks[211]. And also if the file has been randomly selected from the disk and encrypt it then it will take several milliseconds to read each chunk of the file. So generally this file will be cached in memory while writing. In this case reading this data from the disk will not suffer the penalty[211].

We calculate average* encryption/decryption time for AES and 3DES and present this data in the table below.

* Average time calculation is done by the arithmetic mean formula: Average = sum of all data / total nuber of data. A detailed analysis is presented in Appendix R. AES Block size =128 bit (16 byte); 3DES Block size = 64 bit (8 byte).

Table 0-9: AES and 3DES encryption/decryption time for 4MB file

Cryptographic Algorithm	Encryption Time	Decryption Time
AES	5,242861 second	5,85279 second
3DES	6,187814 second	15,80018 second
Overhead time	$6,187814/5,242861 = 1.18$	$15,80018/5,85279 = 2.67 \sim 3$ times

From the above discussion we observe that in both file size AES encryption/ decryption is faster than 3DES. For 4MB file AES encryption is 1.18 times faster than 3DES. And for the same file size AES decryption is almost 3 times faster than 3DES.

Ifoodbag will store their data over cloud. Since data traveling through network is a serious threat we recommended ifoodbag to encrypt their data and send over network. In this experiment we have performed data encryption/decryption by using two cryptographic algorithm AES and 3DES. From our observation above we saw that AES gives better performance than 3DES and AES is also very well studied. Other researcher also have measured the performance for AES,DES and 3DES. And according to their observation on different cryptographic algorithm ,it shows that AES gives better performance over DES and 3DES[212].Therefore we recommend ifoodbag to use AES encryption algorithm. Also in our observation we calculate per block data encryption/decryption time. When ifoodbag will encrypt their data and send to the cloud storage they can use this experiment data for measuring the cost of security based on the file size.

6.3 Security guidelines

Table 0-10 summarizes our recommendations to ifoodbag with respect to the security issues that we have considered in the different areas (as described above).

Table 0-10: Summary of security issues and recommendations

Areas	Recommendations
Load balancer	Follow security policies presented in section 3.2.
Browsing web application	Use HTTPS for secure browsing.
Client-server authentication	Use server authentication for browsing publicly available information in the website. However, for logging-in registered customers, who can order services and perform financial transactions, use strong two-factor authentication.
Communication between nodes	Use VPN tunnels to secure the communication between nodes in the same or different tiers.
Name resolving	Use DNSSEC for ifoodbag.com domain's authoritative name server. Use authoritative-only name server software.
Distributed Database	Use AES encryption to encrypt and decrypt data.
Storage data protection	Use Advanced encryption standard (AES) for stored data
Memcached	Use firewall or Memcached with SASL

6.4 General guidelines

We recommend ifoodbag utilize redundancy in its cloud network. For example, there should be at least one secondary name server for ifoodbag's domain. There should be redundant management nodes that are periodically synchronized with the primary

management node, thus if the primary management node fails, the backup management node can be used to provide seamless service to users. Similarly, redundancy is necessary for the distributed database server. There should be extra copy of each virtual machine template (from which new VM instances can be created) to facilitate instantiating new VMs as necessary. Additionally, there should be sufficient redundancy of network connections. For example, node in different tiers should be interconnected through two (or more) network connections using different network interfaces and routed via different network providers. Overall, the design should be such that if any interface is down then communication can continue through another network interface, i.e., single network interface and network link failures should not cause system failures.

Using updated software versions for all of the software and operating system can help to improve the overall security of the complete system. However, before updating software it is good to check the release notes and test the new release in a test environment, rather than installing the updated directly into the production environment. This implies that there is a need to design a secure test environment that largely duplicates the production environment in order to perform this testing. Fortunately, using a cloud based solution facilitates this temporary need for a larger infrastructure, as the test environment can be returned to the cloud's pool of resources as soon as testing is completed.

7 Conclusions and Future Work

This chapter discusses about how well the goal of this thesis was achieved. It also presents further research possibilities to build upon this thesis work. We also present reflections on the social, economic, sustainability, and ethical aspects of this thesis project at the end of this chapter.

7.1 Conclusions

The goal of this thesis project was to identify potential threats, risk factors, and vulnerable points of the design of ifoodbag's cloud based web application. This proposed design was done by two other Master's thesis project students: Iqbal Hossain and Iqbal Hossain [3]. The main aim of their design was to provide dynamic scaling of ifoodbag's web application running in a cloud environment. Our aim was to propose countermeasures to the identified threats, risks, and vulnerabilities of the proposed system design along with a set of security guidelines concerning network security in order to enhance the security for their proposed system.

We have achieved our goal as we have presented a set of security guidelines that ifoodbag should follow in order to secure its cloud environment. These guidelines are based on both our empirical observations and the results of previous research. The security issues that we identified in the proposed design drove the development of these guidelines. We studied all of the identified security issues and examined possible countermeasures. From the potential set of countermeasures, we have chosen the best solution for ifoodbag in developing our final set of security guidelines. We also proposed some general security guidelines for ifoodbag based upon our own observation of ifoodbag's cloud based web application design.

During this thesis project, we have learned a great deal more about a number of different security mechanisms and their implementations. In this thesis project, we worked not only with a cloud environment but also with several different security mechanisms, which we had not known about earlier. This thesis project was a good experience for both of us.

7.2 Future work

Due to the limited duration of this project, it was not possible to perform all of the tasks that remain in the scope of this thesis project. Additionally, the observations made in the course of this thesis project suggest some areas for further research. We plan to carry on some of these tasks in near future. Other thesis students and researcher may want to explore one or more of these topics.

In this thesis we presented a summary of different methods to identify and prevent DDoS attacks in a cloud network. We presented the methods proposed in different research papers. However, there is a clear need for empirical testing of the proposed mechanisms in order to find an optimal solution for ifoodbag to prevent and detect DDoS attacks against its network. Further research is needed to design a new mechanism to overcome limitations of the existing methods.

We recommend that ifoodbag use DNSSEC in its name server software. We also recommended that ifoodbag use the NSD name server software for its DNSSEC implementation. Our recommendation is based the currently available research results. Further study should investigate the performance of other (authoritative-only) name server software (e.g. VantioTM [213], Knot DNS [214]).

Experiments with different numbers of concurrent users should be done to see how HTTP and HTTPS deal with different numbers of concurrent users.

In this thesis, we recommended that ifoodbag encrypt their sensitive data in the database by using AES encryption. In the future, a number of different cryptographic algorithms should be compared to identify the most suitable algorithm for a database with the actual pattern of queries that ifoodbag's web application experience. However, since we worked with the proposed design of the ifoodbag cloud and not with a working instance of this system we were not able to know what the actual pattern of queries is.

We did not carry out experiments with Memcached. In future experiments should be done to evaluate security solutions in this area.

We recommend ifoodbag to use AES cryptographic algorithm while storing data in cloud storage. In this experiment we have considered AES and 3DES as our cryptography algorithm and we compare the time taken between this two known algorithms. In future we can use other algorithm such as Blowfish, RC2, and RC6 and measure the performance. Other parameters such as CPU time, memory consumption, and battery power need to be considered in future.

We did not empirically investigate the overall effect of implementing all of the recommended security mechanisms in the different modules of ifoodbag's cloud. For example, the application server communicates with a database server through VPN tunnel and the data is again encrypted before storing it in the database server. A future investigation should explore if there is a way to combine these separate operations in order to reduce the total overhead.

We performed VPN experiments with connections only between two nodes. We did not observe the effect of implementing a VPN tunnel for every connection in the overall network configuration of the ifoodbag architecture. We also did not check if a VPN server's performance has an impact on the number of concurrent clients connected to it. Both of these types of future investigations should be done.

7.3 Reflections

In this section we explore a number of social, economic, legal and ethical aspects of this thesis project.

7.3.1 Social

The set of security guidelines proposed in this thesis could be followed by any organization to improve the security of their own cloud environment. This improved security should protect the confidentiality and integrity of both the organization's data and its customers' data. Providing greater protection of this data will help the company to preserve their customer's personal privacy.

Having reliable data about their customer's past purchases may enable to company to both provide more efficient purchasing and distribution of food, but could also provide input to their suppliers that could reduce the amount of waste and loss of food. This information could be used to suggest new, interesting, and nutritious meals to ifoodbag's customers, potentially providing gains in public health. When there is a problem with tainted food, ifoodbag can potentially contact their customers to warn them of a problem.

7.3.2 Economic

Information security is an important concern for any business. E-commerce companies* such as ifoodbag can use cloud-based solutions to reduce their business startup time, to make their business more scalable, and to optimize their IT costs†. Today security remains the biggest concern for companies who are considering adopting a cloud environment, especially as concerns sensitive business critical data (e.g. customer information, product prices, employee salaries). A small startup company who wants to have its own private cloud environment often cannot use enterprise security solutions due to financial limitations. Hiring a security specialist or trying out different network security devices lie beyond such a firm's capacity. The set of guidelines provided in this thesis can help these companies focus on addressing the most urgent problems – that have well-known solution. Following these guidelines can help the company to improve the security of their cloud based IT infrastructure. Following these guideline can financially benefit the companies, both because these guidelines are publicly available, hence they do not have to pay for them, and because these security guidelines recommends the most suitable solutions from both a performance and cost point of view for the different modules of the ifoodbag cloud environment. Other companies can follow these recommendations to realize a solution appropriate to their own cloud environment. These guidelines should help companies improve their security, while optimizing their expenditures to provide overall security for their cloud environment.

As the profit margins for businesses who act as intermediaries in the sales and distribution of food are low, earning large amounts of money depends upon large volumes of sales each with a small profit. Increasing this sales volume requires that the company be able to scale its IT infrastructure to match their sales volume. Failure to provide appropriate security can lead to both loss of customer loyalty (and its accompanying loss in sales and revenue) and severe financial problems if attackers are successful in preventing the orderly operation of the business.

7.3.3 Legal and Ethical issues

In this thesis, we have used only publicly available information sources. We did not reveal any commercial information. Applications used for performing experiments and developing tools were either under GNU General Public License or with a valid license from the vendor. We created our own tools for performing the experiments in this thesis. The experimental results were not fabricated. All of the experimental data are available to others on request.

We did not consider issues regarding personal integrity with regard to the data mining that is possible with the data that the business will be able to collect from its customers. However, we believe that the set of security guidelines that we have proposed will increase the privacy of the customers with respect to avoiding the unwanted disclosure of their data to other parties that are not party to the transactions that they want to make. It will be up to ifoodbag to protect their customers' data when the company interacts with other parties, for example only providing aggregated data – thus avoiding disclosing information about individual customers. However, recent research results provide continuing evidence that it is very hard to provide anonymized data. In this thesis, we did not explore the question of any requirements to disclose business or customer information to governmental authorities.

* Businesses who provide services through internet.

† More benefits have been discussed in section 2.6

References

- [1] Matt Bishop, *Introduction to Computer Security*, 1st ed. Pearson Education, 2004.
- [2] C.P. Pfleeger and S.L Pfleeger. "Security in Computing / 1.3 The Meaning of Computer Security." [Online]. Available: <http://flylib.com/books/en/4.269.1.16/1/>. [Accessed: 02-May-2013].
- [3] Iqbal Hossain and Iqbal Hossain, "Dynamically scalable model for implementing web application in cloud," Master's Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, (expected) 2014.
- [4] Seyyed Khandani, "Engineering-Design-Process." saylor.org, Aug-2005.
- [5] "Ex Post Facto Study: SAGE Research Methods." [Online]. Available: <http://srmo.sagepub.com/view/encyc-of-research-design/n145.xml>. [Accessed: 20-Dec-2013].
- [6] C. Strachey, "Time Sharing in Large Fast Computers," presented at the Proceedings of the International Conference on Information processing, 1959, vol. B.2.19, pp. 336–341.
- [7] "National Institute of Standards and Technology." [Online]. Available: <http://www.nist.gov/index.html>. [Accessed: 02-May-2013].
- [8] "NIST SP 800-145, The NIST Definition of Cloud Computing - SP800-145.pdf." [Online]. Available: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. [Accessed: 02-May-2013].
- [9] "File:Cloud computing.svg - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/File:Cloud_computing.svg. [Accessed: 02-May-2013].
- [10] "Essential characteristics of Cloud Computing - Essential characteristics of Cloud Computing.pdf." [Online]. Available: <http://www.isaca.org/Groups/Professional-English/cloud-computing/GroupDocuments/Essential%20characteristics%20of%20Cloud%20Computing.pdf>. [Accessed: 02-May-2013].
- [11] "Cloud computing service models." [Online]. Available: <http://www.ibm.com/developerworks/cloud/library/cl-cloudservicemodels/?cmp=dw&cpb=dweld&ct=dwnew&cr=dwnen&ccy=zz&csr=021011>. [Accessed: 02-May-2013].
- [12] "SAAS, PAAS and IAAS – Making Cloud Computing Less Cloudy | CIO Research Center." [Online]. Available: <http://cioresearchcenter.com/2010/12/107/>. [Accessed: 02-May-2013].
- [13] The BPM freak, "IaaS, PaaS, SaaS – a pictorial representation" [Online]. Available: <http://thebpmfreak.wordpress.com/2012/09/28/iaas-paas-saas-a-pictorial-representation/>. [Accessed: 02-May-2013].
- [14] Microsoft, "Microsoft Office - Microsoft Word, Outlook & Excel - Office.com." [Online]. Available: <http://office.microsoft.com/en-001/>. [Accessed: 02-May-2013].
- [15] "What is Cloud Computing? - Intec." [Online]. Available: <http://www.intec.co.uk/software-solutions/cloud-it/what-is-cloud-computing/>. [Accessed: 02-May-2013].
- [16] Douglas F. Parkhill, *The Challenge of the Computer Utility*. Addison-Wesley, 1966.
- [17] "North Bridge Venture Partners." [Online]. Available: <http://www.northbridge.com/>. [Accessed: 19-Aug-2013].
- [18] "2012 Future of Cloud Computing Survey Exposes Hottest Trends in Cloud Adoption | Business Wire." [Online]. Available: <http://www.businesswire.com/news/home/20120620006697/en/2012-Future-Cloud-Computing-Survey-Exposes-Hottest>. [Accessed: 02-May-2013].
- [19] "Survey: 69% APAC businesses intend to adopt hybrid cloud | Asia Cloud Forum." [Online]. Available: <http://www.asiacloudforum.com/content/survey-69-apac-businesses-intend-adopt-hybrid-cloud>. [Accessed: 02-May-2013].
- [20] "What is community cloud? - Definition from WhatIs.com." [Online]. Available: <http://searchcloudstorage.techtarget.com/definition/community-cloud>. [Accessed: 02-May-2013].
- [21] "squid: Optimising Web Delivery." [Online]. Available: <http://www.squid-cache.org/Intro/>. [Accessed: 02-May-2013].

- [22] “memcached - a distributed memory object caching system.” [Online]. Available: <http://memcached.org/>. [Accessed: 02-May-2013].
- [23] Microsoft, “Microsoft SharePoint Online.” [Online]. Available: <https://partner.microsoft.com/global/productssolutions/productsonlineservices/osofficesharepointonline>. [Accessed: 02-May-2013].
- [24] P. Loan-Clarke, “Benefits of Research Collaboration,” 2002. [Online]. Available: http://www.rer.emich.edu/module9/i4_benefits.html. [Accessed: 10-Jun-2013].
- [25] Google, “Google Cloud Platform — Cloud Platform.” [Online]. Available: <https://cloud.google.com/>. [Accessed: 21-Jul-2013].
- [26] Amazon, “Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more.” [Online]. Available: <http://www.amazon.com/>. [Accessed: 21-Jul-2013].
- [27] Microsoft, “SkyDrive - Microsoft Windows.” [Online]. Available: <http://windows.microsoft.com/en-us/skydrive/download>. [Accessed: 21-Jul-2013].
- [28] Oracle, “Oracle Managed Cloud Services.” [Online]. Available: <http://www.oracle.com/us/solutions/cloud/managed-cloud-services/overview/index.html>. [Accessed: 21-Jul-2013].
- [29] “Build AWS-compatible Private Clouds with Eucalyptus | Open Source. AWS-compatible. Private Clouds.” [Online]. Available: <http://www.eucalyptus.com/>. [Accessed: 21-Jul-2013].
- [30] “Malware Injection Attack - Knowledgebase - RockSoft Server Consultancy.” [Online]. Available: <http://rocksoft.com.my/knowledgebase.php?action=displayarticle&id=754>. [Accessed: 02-May-2013].
- [31] B.Meena, Krishnaveer Abhishek Challa, “Cloud Computing Security Issues with Possible Solutions.”, *International Journal on Computer Science and Technology (IJCST)*, Vol. 3, Issue 1, Jan.-March 2012. Available: <http://www.ijcst.com/vol31/2/bmeena.pdf>. [Accessed: 02-May-2013].
- [32] K. Zunnurhain and Susan V. Vrbsky, “Security Attacks and Solutions in Clouds”, *Proceedings of the 1st international conference on cloud computing, 2010*, pp. 145–156.
- [33] “A Detailed Analysis of the Issues and Solutions for Securing Data in Cloud,” *IOSR Journal of Computer Engineering (IOSRJCE)*, vol. 4, no. 5, pp. 11–18, Oct. 2012.
- [34] K. Zunnurhain and Susan V. Vrbsky, “Security Attacks and solutions in Clouds,” The University of Alabama, Tuscaloosa.
- [35] “Security Attacks and Solutions in cloud.” .
- [36] “Researchers demo cloud security issue with Amazon AWS attack - Computerworld.” [Online]. Available: http://www.computerworld.com/s/article/9221208/Researchers_demo_cloud_security_issue_with_Amazon_AWS_attack. [Accessed: 02-May-2013].
- [37] “What is a denial of service attack?” [Online]. Available: <https://www.paloaltonetworks.com/resources/learning-center/what-is-a-denial-of-service-attack-dos.html>. [Accessed: 21-Jul-2013].
- [38] Neomi Antedomenico, “Optimizing security of Cloud Computing within the DOD,” Naval Postgraduate School, Monterey, California.
- [39] A. Singh and M. Shrivastava, “Overview of Attacks on Cloud Computing,” *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 1, no. 4, Apr. 2012.
- [40] “Twitter, Facebook attack targeted one user | InSecurity Complex - CNET News.” [Online]. Available: http://news.cnet.com/8301-27080_3-10305200-245.html. [Accessed: 21-Jul-2013].
- [41] “DDoS Attack Hits Amazon Cloud Customer Hard.” [Online]. Available: <http://www.thewhir.com/web-hosting-news/ddos-attack-hits-amazon-cloud-customer-hard>. [Accessed: 02-May-2013].
- [42] “Security and Privacy in Cloud Computing - 600.412.lecture02.pdf.” [Online]. Available: <http://www.cs.jhu.edu/~ragib/sp10/cs412/lectures/600.412.lecture02.pdf>. [Accessed: 02-May-2013].
- [43] “Flooding Attack.” [Online]. Available: <http://www.javvin.com/networksecurity/FloodingAttack.html>. [Accessed: 02-May-2013].

- [44] “What is hypervisor? - Definition from WhatIs.com.” [Online]. Available: <http://searchservvirtualization.techtarget.com/definition/hypervisor>. [Accessed: 26-Jan-2014].
- [45] “Domain Name System - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Domain_Name_System. [Accessed: 23-Jul-2013].
- [46] “Security Issues with DNS,” *SANS Institute InfoSec Reading Room*. [Online]. Available: http://www.sans.org/reading_room/whitepapers/dns/security-issues-dns_1069. [Accessed: 23-Jul-2013].
- [47] “Anatomy of a DNS DDoS Amplification Attack,” *WatchGuard*. [Online]. Available: <http://www.watchguard.com/infocenter/editorial/41649.asp>. [Accessed: 23-Jul-2013].
- [48] “The most widely used name server software: BIND,” *Internet Systems Consortium*. [Online]. Available: <http://www.isc.org/downloads/bind/>. [Accessed: 23-Jul-2013].
- [49] “CERT® Advisory CA-2001-02 Multiple Vulnerabilities in BIND,” *CERT*. [Online]. Available: <http://www.cert.org/advisories/CA-2001-02.html>. [Accessed: 23-Jul-2013].
- [50] “DNS Security (Part 1): Issues in DNS Security,” *WindowsSecurity*. [Online]. Available: http://www.windowsecurity.com/articles-tutorials/misc_network_security/DNS-Security-Part-1.html. [Accessed: 23-Jul-2013].
- [51] “Explanation of a DNS Zone Transfer,” *Microsoft Support*. [Online]. Available: <http://support.microsoft.com/kb/164017>. [Accessed: 23-Jul-2013].
- [52] S. Rose, R. Chandramouli, and A. Nakassis, “Information Leakage through the Domain Name System,” in *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology, 2009*, pp. 16–21.
- [53] “DNS ID PREDICTION AND EXPLOITATION,” *Tripod*. [Online]. Available: <http://c0vertl.tripod.com/text/dnsattack.txt>. [Accessed: 23-Jul-2013].
- [54] “Security Issues with DNS - security-issues-dns_1069.”
- [55] “Domain Name System Security Extensions,” *IETF*. [Online]. Available: <http://www.ietf.org/rfc/rfc2535.txt>. [Accessed: 23-Jul-2013].
- [56] “Securing an Internet Name Server,” *VeriSign*. [Online]. Available: http://www.linuxsecurity.com/resource_files/server_security/securing_an_internet_name_server.pdf. [Accessed: 23-Jul-2013].
- [57] R. Elz, R. Bush, S. Bradner, and M. Patton, “Selection and Operation of Secondary DNS Servers,” *RFC 2182*, Jul-1997. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2182.txt>.
- [58] S. Rose and A. Nakassis, “Minimizing information leakage in the DNS,” 2008, vol. 22, pp. 22–25.
- [59] “digital signature (electronic signature),” *SearchSecurity*. [Online]. Available: <http://searchsecurity.techtarget.com/definition/digital-signature>. [Accessed: 23-Jul-2013].
- [60] “Step-by-Step: Demonstrate DNSSEC in a Test Lab,” *Step-by-Step: Demonstrate DNSSEC in a Test Lab*. [Online]. Available: <http://technet.microsoft.com/en-us/library/hh831411.aspx>. [Accessed: 23-Jul-2013].
- [61] “Delegation Signer (DS) Resource Record (RR),” Dec-2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3658.txt>. [Accessed: 23-Jul-2013].
- [62] “DNSSEC course.” [Online]. Available: <http://www.dnsseccourse.nl/en/player.html>. [Accessed: 23-Jun-2013].
- [63] G. Lindsay, “DNSSEC and DNS Amplification Attacks,” *Security TechCenter*. [Online]. Available: <http://technet.microsoft.com/en-us/security/hh972393.aspx>. [Accessed: 24-Jul-2013].
- [64] J. Bau and C. M. John, “A Security Evaluation of DNSSEC with NSEC3.,” 2010, p. 115.
- [65] R. Curtmola, Aniello Del Sorbo, and Giuseppe Ateniese, “On the performance and analysis of DNS security extensions.,” *Springer Berlin Heidelberg*, pp. 288–303, 2005.
- [66] B. Ager, H. Dreger, and A. Feldmann, “Exploring the Overhead of DNSSEC.,” Apr. 2005.
- [67] “Scaling Strategies and Tactics for Dynamic Web Applications - 8150.pdf.”
- [68] “HOWTO: Load balance HTTP with Linux and Squid,” *PARKER SAMP*. [Online]. Available: <http://parkersamp.com/2010/11/howto-load-balance-http-with-linux-and-squid/>. [Accessed: 23-Jul-2013].

- [69] “HTTP Methods: GET vs. POST,” *w3schools.com*. [Online]. Available: http://www.w3schools.com/tags/ref_httpmethods.asp. [Accessed: 23-Jul-2013].
- [70] “Security considerations with Squid proxy server - security-considerations-squid-proxy-server_1048.” .
- [71] “UNIX Configuration Guidelines,” *CERT® Coordination Center*. [Online]. Available: http://www.cert.org/tech_tips/unix_configuration_guidelines.html. [Accessed: 23-Jul-2013].
- [72] “Noshell v1.2,” *ticalc.org*. [Online]. Available: <http://www.ticalc.org/archives/files/fileinfo/400/40005.html>. [Accessed: 23-Jul-2013].
- [73] “HTTP Authentication,” *IETF*, Jun-1999. [Online]. Available: <http://www.ietf.org/rfc/rfc2617.txt>. [Accessed: 23-Jul-2013].
- [74] “round robin DNS,” *WEBOPEDIA*. [Online]. Available: http://www.webopedia.com/TERM/R/Round_Robin_DNS.html. [Accessed: 23-Jul-2013].
- [75] “HAProxy,” *HAProxy*. [Online]. Available: <http://haproxy.1wt.eu/>. [Accessed: 23-Jul-2013].
- [76] “Core Balance,” *Core Balance*. [Online]. Available: <http://core-balance.sourceforge.net/>. [Accessed: 23-Jul-2013].
- [77] “Welcome to Crossroads,” *Crossroads*. [Online]. Available: <http://crossroads.e-tunity.com/>. [Accessed: 23-Jul-2013].
- [78] “Distributor,” *Distributor*. [Online]. Available: <http://distributor.sourceforge.net/>. [Accessed: 23-Jul-2013].
- [79] “ZENLOADBALANCER,” *ZENLOADBALANCER*. [Online]. Available: <http://www.zenloadbalancer.org/web/>. [Accessed: 23-Jul-2013].
- [80] “Octopus Load Balancer,” *sourceforge*. [Online]. Available: <http://sourceforge.net/projects/octopuslb/>. [Accessed: 23-Jul-2013].
- [81] “World Wide Web,” *Wikipedia, the free encyclopedia*. 22-Jul-2013.
- [82] “Hypertext Transfer Protocol -- HTTP/1.1,” *IETF*. [Online]. Available: <http://www.ietf.org/rfc/rfc2616.txt>. [Accessed: 23-Jul-2013].
- [83] “The Internet Engineering Task Force (IETF),” *IETF*. [Online]. Available: <http://www.ietf.org/>. [Accessed: 23-Jul-2013].
- [84] “URL,” *About.com Wireless / Networking*. [Online]. Available: <http://compnetworking.about.com/od/internetaccessbestuses/g/bldef-url.htm>. [Accessed: 23-Jul-2013].
- [85] “Get a fast, free web browser,” *Chrome*. [Online]. Available: <https://www.google.com/intl/en/chrome/browser/>. [Accessed: 23-Jul-2013].
- [86] “Fast and fluid for Windows 7,” *Windows*. [Online]. Available: Fast and fluid for Windows 7. [Accessed: 23-Jul-2013].
- [87] “Security Considerations,” *w3*. [Online]. Available: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html>. [Accessed: 23-Jul-2013].
- [88] “HTTP Secure,” *Wikipedia, the free encyclopedia*. 17-Jul-2013.
- [89] “Secure Hypertext Transfer Protocol,” *Wikipedia, the free encyclopedia*. 27-May-2013.
- [90] “The Secure HyperText Transfer Protocol,” *IETF*, Aug-1999. [Online]. Available: <http://tools.ietf.org/html/rfc2660>. [Accessed: 23-Jul-2013].
- [91] “Virtual private network,” *Wikipedia, the free encyclopedia*. 18-Jul-2013.
- [92] *Microsoft*, “VPN Tunneling Protocols,” *Microsoft Technet*. .
- [93] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn, ‘Point-to-Point Tunneling Protocol (PPTP)’, *Internet Request for Comments*, vol. RFC 2637 (Informational), July 1999, Available at <http://www.rfc-editor.org/rfc/rfc2637.txt>. [Accessed: 23-Jul-2013].
- [94] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter, ‘Layer Two Tunneling Protocol “L2TP”’, *Internet Request for Comments*, vol. RFC 2661 (Proposed Standard), August 1999, Available at <http://www.rfc-editor.org/rfc/rfc2661.txt>. [Accessed: 23-Jul-2013].
- [95] “The Secure Socket Tunneling Protocol,” *Microsoft Technet*. [Online]. Available: <http://technet.microsoft.com/en-us/magazine/2007.06.cableguy.aspx>. [Accessed: 23-Jul-2013].

- [96] S. Kent and K. Seo, 'Security Architecture for the Internet Protocol', *Internet Request for Comments*, vol. RFC 4301 (Proposed Standard), December 2005, Available at <http://www.rfc-editor.org/rfc/rfc4301.txt>. [Accessed: 23-Jul-2013].
- [97] "IPsec," *Networking and Access Technologies*. [Online]. Available: <http://technet.microsoft.com/en-us/network/bb531150.aspx>. [Accessed: 23-Jul-2013].
- [98] "How to Setup a VPN (PPTP) Server on Debian Linux." [Online]. Available: <http://www.howtogeek.com/51237/setting-up-a-vpn-pptp-server-on-debian/>. [Accessed: 23-Jul-2013].
- [99] "Remote Access," *Networking and Access Technologies*. [Online]. Available: <http://technet.microsoft.com/en-us/network/dd420463.aspx>. [Accessed: 23-Jul-2013].
- [100] "Configure a Remote Access VPN Server," *Windows Server*. [Online]. Available: <http://technet.microsoft.com/en-us/library/cc725734%28v=ws.10%29.aspx>. [Accessed: 23-Jul-2013].
- [101] "Using an ISA Server virtual private network," *Microsoft Technet*. [Online]. Available: <http://technet.microsoft.com/en-us/library/cc722838.aspx>. [Accessed: 23-Jul-2013].
- [102] "Check Point." [Online]. Available: <http://www.checkpoint.com/>. [Accessed: 23-Jul-2013].
- [103] "Configuration of a Symantec Enterprise Firewall/VPN client tunnel when the firewall/VPN server is subjected to external NAT," *Symantec*, 20-Jan-2002. [Online]. Available: <http://www.symantec.com/business/support/index?page=content&id=TECH80190>. [Accessed: 23-Jul-2013].
- [104] "Cisco RV180 VPN Router," *Cisco*. [Online]. Available: <http://www.cisco.com/en/US/products/ps11995/index.html>. [Accessed: 23-Jul-2013].
- [105] "Juniper," *Juniper*. [Online]. Available: <http://www.juniper.net/us/en/products-services/security/sa-series/>. [Accessed: 23-Jul-2013].
- [106] "AEP Series E Remote IPsec VPN," *Retec Technology (UK) Ltd*. [Online]. Available: http://www.retec-technology.co.uk/Prod_ser_E_rem.html. [Accessed: 23-Jul-2013].
- [107] "Business Communications Solutions from Avaya." [Online]. Available: <http://www.avaya.com/usa/>. [Accessed: 23-Jul-2013].
- [108] "OPENVPN," *OPENVPN*. [Online]. Available: <http://openvpn.net/>. [Accessed: 23-Jul-2013].
- [109] "Welcome to the OpenSSL Project," *OpenSSL*. [Online]. Available: <http://www.openssl.org/>. [Accessed: 23-Jul-2013].
- [110] "OpenVPN Protocol (OpenVPN)," *WIRESHARK*, 27-Jun-2013. [Online]. Available: <http://wiki.wireshark.org/OpenVPN>. [Accessed: 23-Jul-2013].
- [111] "What is secret-key cryptography?," *RSA Laboratories*. [Online]. Available: <http://www.rsa.com/rsalabs/node.asp?id=2166>. [Accessed: 23-Jul-2013].
- [112] "What is public-key cryptography?," *RSA Laboratories*. [Online]. Available: <http://www.rsa.com/rsalabs/node.asp?id=2165>. [Accessed: 23-Jul-2013].
- [113] "OpenVPN," *Wikipedia, the free encyclopedia*. 25-Jun-2013.
- [114] "Security Overview," *OPENVPN*. [Online]. Available: <http://openvpn.net/index.php/open-source/documentation/security-overview.html>. [Accessed: 23-Jul-2013].
- [115] "The IP Network Address Translator (NAT)," *IETF*, May-1994. [Online]. Available: <http://www.ietf.org/rfc/rfc1631.txt>. [Accessed: 23-Jul-2013].
- [116] "gnu.org." [Online]. Available: <https://gnu.org/licenses/gpl.html>. [Accessed: 23-Jul-2013].
- [117] "BridgingAndRouting – OpenVPN Community." [Online]. Available: <https://community.openvpn.net/openvpn/wiki/BridgingAndRouting>. [Accessed: 07-Jan-2014].
- [118] "What is caching? - Definition from WhatIs.com." [Online]. Available: <http://whatis.techtarget.com/definition/caching>. [Accessed: 30-Jul-2013].
- [119] "What is cache?" [Online]. Available: <http://www.computerhope.com/jargon/c/cache.htm>. [Accessed: 30-Jul-2013].
- [120] "What is cache? - Definition from WhatIs.com." [Online]. Available: <http://searchstorage.techtarget.com/definition/cache>. [Accessed: 30-Jul-2013].
- [121] "Introduction to memcached." [Online]. Available: http://www.slideshare.net/oemebamo/introduction-to-memcached?from_search=3. [Accessed: 30-Jul-2013].

- [122] “Database Sharding at Netlog.” [Online]. Available: <http://www.slideshare.net/oemebamo/database-sharding-at-netlog-presentation>. [Accessed: 30-Jul-2013].
- [123] “Open Source - Facebook Developers.” [Online]. Available: <https://developers.facebook.com/opensource/>. [Accessed: 30-Jul-2013].
- [124] “High Scalability - High Scalability - Flickr Architecture.” [Online]. Available: <http://highscalability.com/flickr-architecture>. [Accessed: 30-Jul-2013].
- [125] “Caching with Twemcache | Twitter Blog.” [Online]. Available: <https://blog.twitter.com/2012/caching-twemcache>. [Accessed: 30-Jul-2013].
- [126] “Creating the YouTubeActivityViewer application using the PHP client library, memcache and jQuery - YouTube — Google Developers.” [Online]. Available: https://developers.google.com/youtube/articles/youtube_api_activity_php. [Accessed: 30-Jul-2013].
- [127] Zynga Engineering. “Building a scalable game server” [Online]. Available: <http://code.zynga.com/2011/07/building-a-scalable-game-server/>. [Accessed: 30-Jul-2013].
- [128] “Memcached: What is it and what does it do?” [Online]. Available: <http://www.slideshare.net/brianlmoon/memcached-what-is-it-and-what-does-it-do-2097515>. [Accessed: 30-Jul-2013].
- [129] “memcached.” [Online]. Available: <http://www.slideshare.net/hectcastro/memcached-2536880>. [Accessed: 30-Jul-2013].
- [130] J. L. Josh Taylor, “Memcached.” .
- [131] “Applying memcached to increase site performance.” [Online]. Available: <http://www.ibm.com/developerworks/xml/library/os-memcached/index.html>. [Accessed: 30-Jul-2013].
- [132] “Memcached Security.” [Online]. Available: <http://dustin.sallings.org/2010/08/08/memcached-security.html>. [Accessed: 30-Jul-2013].
- [133] A. Melnikov and K. Zeilenga, ‘Simple Authentication and Security Layer (SASL)’, *Internet Request for Comments*, vol. RFC 4422 (Proposed Standard), June 2006, Available at <http://www.rfc-editor.org/rfc/rfc4422.txt>. [Accessed: 30-Jul-2013].
- [134] “SASLHowto - memcached - HOWTO use SASL auth. - Memcached - Google Project Hosting.” [Online]. Available: <http://code.google.com/p/memcached/wiki/SASLHowto>. [Accessed: 30-Jul-2013].
- [135] “Cloud Storage Definition - What is Cloud Storage.” [Online]. Available: <http://mp3.about.com/od/glossary/g/Cloud-Storage-Definition-What-Is-Cloud-Storage.htm>. [Accessed: 30-Nov-2013].
- [136] “Cloud Lexicon | ProfitBricks.” [Online]. Available: <http://www.profitbricks.co.uk/cloud-lexicon>. [Accessed: 30-Nov-2013].
- [137] “Different Cloud Storage Types.” [Online]. Available: <http://www.cloudstoragebest.com/cloud-storage-types/>. [Accessed: 30-Nov-2013].
- [138] Subramanyam Kodukula, Talasila Sasidhar, “A Generalized Cloud storage Architecture with Backup Technology for any Cloud Storage Providers,” vol. 2.
- [139] “www.clouddrive.com.au - www.clouddrive.com.au-WhitePaper.pdf.” .
- [140] “HowStuffWorks ‘Concerns About Cloud Storage.’” [Online]. Available: <http://computer.howstuffworks.com/cloud-computing/cloud-storage3.htm>. [Accessed: 30-Nov-2013].
- [141] Johan Gamper, “Distributed Database Chapter 1:Introduction.” .
- [142] “Distributed database.” [Online]. Available: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Distributed_database.html. [Accessed: 30-Nov-2013].
- [143] “What is distributed database? - Definition from WhatIs.com.” [Online]. Available: <http://searchoracle.techtarget.com/definition/distributed-database>. [Accessed: 30-Nov-2013].
- [144] “Distributed Database Concepts.” [Online]. Available: http://docs.oracle.com/cd/A87860_01/doc/server.817/a76960/ds_conce.htm#12187. [Accessed: 30-Nov-2013].
- [145] “Database system concepts.” .

- [146] Richard Holowczak, "Database Management Systems II - Prof. Holowczak." [Online]. Available: <http://cisnet.baruch.cuny.edu/holowczak/classes/9440/distributed/>. [Accessed: 01-Dec-2013].
- [147] "Client / Server Application: Distributed Database Management System: Pros and Cons." [Online]. Available: <http://b0402131.blogspot.se/2008/04/distributed-database-management-system.html>. [Accessed: 01-Dec-2013].
- [148] "What are the Advantages and Disadvantages of Distributed Database Management System?" [Online]. Available: <http://ecomputernotes.com/database-system/adv-database/advantages-and-disadvantages-of-ddbms>. [Accessed: 01-Dec-2013].
- [149] "Pros and Cons of Distributed Databases." [Online]. Available: <http://www.articlesbase.com/programming-articles/pros-and-cons-of-distributed-databases-935835.html>. [Accessed: 01-Dec-2013].
- [150] "Teach-ICT OCR A2 ICT G063 Syllabus: distributed database." [Online]. Available: http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/334_applications_ict/distributed_database/miniweb/pg7.htm. [Accessed: 01-Dec-2013].
- [151] V.K. Gupta, Sheetlani Jitendra, Gupta Dhiraj, Concurrency Control and Security issues of Distributed Databases, Research Journal of Engineering, Vol. 1(2), 70-73, August 2012 DOI: 10.ISCA-JEngS-2012-047. Available: <http://www.isca.in/IJES/Archive/v1i2/10.ISCA-JEngS-2012-047.pdf>
- [152] "Neustar | Promote & Protect Your Brand via Neustar Enterprise Services." [Online]. Available: <http://www.neustar.biz/enterprise>. [Accessed: 24-Nov-2013].
- [153] "DDoS attacks scaling up alarmingly, says Arbor - Security - News & Features - ITP.net." [Online]. Available: <http://www.itp.net/595509-ddos-attacks-scaling-up-alarmingly-says-arbor#.UpIYvhBRyKJ>. [Accessed: 24-Nov-2013].
- [154] C. Dewey, "Only 2 countries have been hit with DDoS attacks every day since May. The U.S. is one." *Washington Post*. [Online]. Available: <http://www.washingtonpost.com/blogs/the-switch/wp/2013/10/26/only-2-countries-have-been-hit-with-ddos-attacks-every-day-since-may-the-u-s-is-one/>. [Accessed: 24-Nov-2013].
- [155] "Computer Security Incident Handling Guide." National Institute of Standards and Technology, Jan-2004.
- [156] "Denial-of-service attack," *Wikipedia, the free encyclopedia*. 23-Nov-2013.
- [157] Naoum Naoumov and Keith Ross. 2006. Exploiting P2P systems for DDoS attacks. In Proceedings of the 1st international conference on Scalable information systems (InfoScale '06). ACM, New York, NY, USA, , Article 47 . DOI=10.1145/1146847.1146894 <http://doi.acm.org/10.1145/1146847.1146894>.
- [158] El Defrawy, Karim, Minas Gjoka, and Athina Markopoulou, "BotTorrent: Misusing BitTorrent to Launch DDoS Attacks," in *Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, 2007, pp. 1–6.
- [159] "P2P Networks Hijacked for DDoS Attacks | Netcraft." [Online]. Available: http://news.netcraft.com/archives/2007/05/23/p2p_networks_hijacked_for_ddos_attacks.html. [Accessed: 24-Nov-2013].
- [160] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," Aug-2007. [Online]. Available: <http://tools.ietf.org/html/rfc4987>. [Accessed: 24-Nov-2013].
- [161] Niraj Suresh Katkamwar, Atharva Girish Puranik, and Purva Deshpande, "Securing Cloud Servers against Flooding Based DDoS Attacks," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 1, pp. 50 – 55, Nov. 2012.
- [162] "Smurf Attack explained - DDoS Protection." [Online]. Available: <http://www.ddosprotection.net/smurf-attack-explained/>. [Accessed: 24-Nov-2013].
- [163] DDoSProtection.Net, "ICMP Flood attack explained," *DDoS Protection*. [Online]. Available: <http://www.ddosprotection.net/icmp-flood-attack-explained/>. [Accessed: 24-Nov-2013].
- [164] DDoSProtection.Net, "Ping Of Death (POD) attack explained," *DDoS Protection*. [Online]. Available: <http://www.ddosprotection.net/ping-of-death-attack-explained/>. [Accessed: 24-Nov-2013].

- [165] DDoSProtection.Net, “Teardrop attack explained (Legacy attack),” *DDoS Protection*. [Online]. Available: <http://www.ddosprotection.net/teardrop-attack-explained-legacy-attack/>. [Accessed: 24-Nov-2013].
- [166] Chen, Q., Lin, W., Dou, W., & Yu, S, “CBF: A packet filtering method for DDoS attack defense in cloud environment.,” in *Dependable, Autonomic and Secure Computing (DASC), 2011*, 2011, pp. 427 – 434.
- [167] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, “PacketScore: a statistics-based packet filtering scheme against distributed denial-of-service attacks.,” in *Dependable and Secure Computing*, 2006, pp. 141–155.
- [168] “Bayes’ Theorem: Introduction.” [Online]. Available: <http://www.trinity.edu/cbrown/bayesweb/>. [Accessed: 20-Dec-2013].
- [169] Paulo E. Ayres, Huizhong Sun, H. Jonathan Chao, Wing Cheong Lau, “ALPi: A DDoS Defense System for High-Speed Networks,” *Selected Areas in Communications, IEEE Journal*, vol. 24, pp. 1864–1876, Oct. 2006.
- [170] Microsoft, “The Leaky Bucket Buffer Model (Windows).” [Online]. Available: [http://msdn.microsoft.com/en-us/library/windows/desktop/dd206751\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd206751(v=vs.85).aspx). [Accessed: 20-Dec-2013].
- [171] Haining Wang, Cheng Jin, and Kang G. Shin. 2007. Defense against spoofed IP traffic using hop-count filtering. *IEEE/ACM Transactions on Networking (TON)*, Volume 15, Number 1, February 2007, pp. 40-53. DOI:10.1109/TNET.2006.890133 <http://dx.doi.org/10.1109/TNET.2006.890133>
- [172] Zhenhai Duan, Xin Yuan, and J. Chandrashekar, ‘Controlling IP Spoofing through Interdomain Packet Filters’, *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 1, pp. 22–36, January 2008, DOI:10.1109/TDSC.2007.70224.
- [173] E. Anitha and S. Malliga, “A packet marking approach to protect cloud environment against DDoS attacks,” in *Information Communication and Embedded Systems (ICICES)*, 2013, pp. 367–370.
- [174] Poonam Yadav and Sujata, “Security Issues in Cloud Computing Solution of DDOS and Introducing Two-Tier CAPTCHA,” *International Journal on Cloud Computing: Services & Architecture*, vol. 3, no. 3, pp. 25–40, June 2013.
- [175] L. Yang, T. Zhang, J. Song, J. S. Wang, and P. Chen, “Defense of DDoS attack for cloud computing,” presented at the In Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference, 2012, vol. 2, pp. 626–629, DOI: 10.1109/CSAE.2012.6272848.
- [176] M. N. Ismail, A. Aborujilah, S. Musa, and A. Shahzad, “New Framework to Detect and Prevent Denial of Service Attack in Cloud Computing Environment,” *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 4, pp. 226–237. Available: <http://www.cscjournals.org/csc/manuscript/Journals/IJCSS/volume6/Issue4/IJCSS-760.pdf>
- [177] Joseph Latanicki, Philippe Massonet, Syed Naqvi, Benny Rochwerger, Massimo Villari, “Scalable Cloud Defenses for Detection, Analysis and Mitigation of DDoS Attacks,” *IOS Press*, pp. 127–137, 2010.
- [178] “Internet Systems Consortium | BIND.” [Online]. Available: <http://www.isc.org/downloads/bind/>. [Accessed: 24-Nov-2013].
- [179] “nlnetlabs.nl :: Name Server Daemon (NSD) ::” [Online]. Available: <http://www.nlnetlabs.nl/projects/nsd/>. [Accessed: 24-Nov-2013].
- [180] “Unbound.” [Online]. Available: <http://unbound.net/>. [Accessed: 24-Nov-2013].
- [181] Daniel Migault, Cédric Girard, and Maryline Laurent, “A Performance view on DNSSEC migration,” in *Proceedings of 2010 International Conference on Network and Service Management (CNSM)*, 2010, pp. 469–474, DOI: 10.1109/CNSM.2010.5691275.
- [182] Arthur Goldberg, Robert Buff, and Andrew Schmitt, “A comparison of HTTP and HTTPS performance,” Computer Science Department, Courant Institute of Mathematical Science, New York University, 1998.
- [183] Xubin He, “A Performance Analysis of Secure HTTP Protocol,” Tennessee Tech. University, STAR Lab, Technical Report, 2003.

- [184] A. Kumar, B. G. Lee, H. Lee, and A. Kumari, 'Secure storage and access of data in cloud computing', presented at the 2012 International Conference on ICT Convergence (ICTC), Jeju Island, 2012, pp. 336–339, DOI:10.1109/ICTC.2012.6386854, Available at <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6386854>.
- [185] Johan Strickland, "How Cloud Storage Works." <http://computer.howstuffworks.com/cloud-computing/cloud-storage.htm>.
- [186] Vormetric, "Data security in the cloud", <http://www.vormetric.com/sites/default/files/wp-data-security-in-the-cloud.pdf>.
- [187] Neera Batra and Manpreet Singh, "Multilevel Policy Based Security in Distributed Database," in *Advances in Computing and Communications*, vol. 190, 2011, pp. 572–580.
- [188] Jim Reavis, "Goodbye DES, Hello AES.," NetworkWorldFusion 30 July 2001 [Online]. Available: <http://www.networkworld.com/research/2001/0730feat2.html>. [Accessed: 30-Dec-2013].
- [189] Nathan Malcolm, "Securing memcached servers." [Online]. Available: <https://coderwall.com/p/i2tcmg>. [Accessed: 30-Dec-2013].
- [190] Ankit Mathur, "Speed up Your Cloud with Memcached - LINUX For You." LINUX For You, 19 March 2012 [Online]. Available: <http://www.linuxforu.com/2012/03/speed-up-cloud-with-memcached/>. [Accessed: 30-Dec-2013].
- [191] Marius Ducea, "Securing Memcached - MDLog:/sysadmin." [Online]. Available: <http://www.ducea.com/2008/01/11/securing-memcached/>. 11 January 2008 [Accessed: 30-Dec-2013].
- [192] "SASLAAuthProtocol - memcached - SASL Authentication for Memcached - Memcached - Google Project Hosting." [Online]. Available: <https://code.google.com/p/memcached/wiki/SASLAAuthProtocol>. [Accessed: 30-Dec-2013].
- [193] Isode Ltd., "SASL (Simple Authentication and Security Layer)." [Online]. Available: <http://www.isode.com/products/sasl.html>. [Accessed: 30-Dec-2013].
- [194] Patrick Galbraith, "SASL memcached now available!", 7 November 2009.
- [195] Microsoft, "WebClient Class (System.Net)." [Online]. Available: [http://msdn.microsoft.com/en-us/library/system.net.webclient\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.net.webclient(v=vs.110).aspx). [Accessed: 18-Dec-2013].
- [196] "Easy_Windows_Guide – OpenVPN Community," *OPENVPN Community Wiki and Tracker*, Mar-2013. [Online]. Available: https://community.openvpn.net/openvpn/wiki/Easy_Windows_Guide. [Accessed: 18-Dec-2013].
- [197] "Toad for SQL Server - Toad World." [Online]. Available: <http://www.toadworld.com/products/toad-for-sql-server/default.aspx>. [Accessed: 23-Dec-2013].
- [198] "Navicat | DB Admin Tool for MySQL, MariaDB, SQL Server, SQLite, Oracle & PostgreSQL." [Online]. Available: <http://www.navicat.com/>. [Accessed: 23-Dec-2013].
- [199] "WampServer, the web development platform on Windows - Apache, MySQL, PHP." [Online]. Available: <http://www.wampserver.com/en/>. [Accessed: 30-Dec-2013].
- [200] "Moore's Law." [Online]. Available: <http://www.moorelaw.org/>. [Accessed: 28-Dec-2013].
- [201] Microsoft, "Security Guidance for IIS," *Microsoft Technet*, 26-Jan-2009. [Online]. Available: [http://technet.microsoft.com/en-us/library/dd450371\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/dd450371(v=ws.10).aspx). [Accessed: 18-Dec-2013].
- [202] GeoTrust, "The Truth About SSL Encryption Strengths," Document number: TB-ENC-0605, GeoTrust, Maidstone, Kent, UK. [Online]. Available: http://www.trustico.com/material/techpaper_encryption.pdf. [Accessed: 07-Jan-2014].
- [203] Eland Systems, "IPSEC." [Online]. Available: <http://www.elandsys.com/resources/ipsec/>. [Accessed: 28-Dec-2013].
- [204] Microsoft, "What Is IPsec?: Security Policy; Security Services." [Online]. Available: [http://technet.microsoft.com/en-us/library/cc776369\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc776369(v=ws.10).aspx). [Accessed: 28-Dec-2013].
- [205] "OpenSSL: The Open Source toolkit for SSL/TLS," *OpenSSL*. [Online]. Available: <http://www.openssl.org/>. [Accessed: 10-Jan-2014].

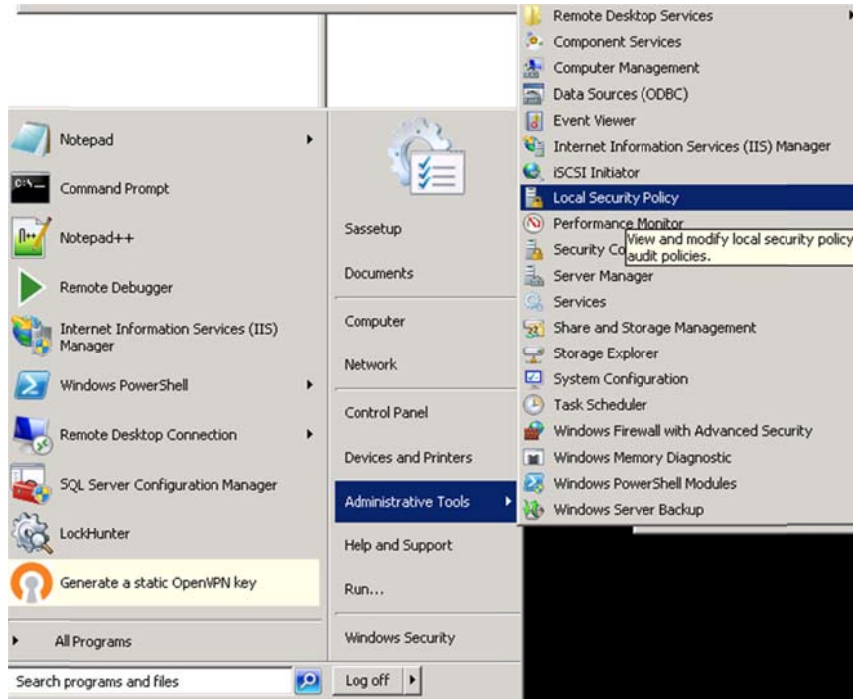
- [206] “PPTP vs L2TP vs OpenVPN.” [Online]. Available: <https://www.ivpn.net/knowledgebase/62/PPTP-vs-L2TP-vs-OpenVPN.html>. [Accessed: 28-Dec-2013].
- [207] B. Korver, “The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX,” *RFC 4945*, Aug-2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4945.txt>. [Accessed: 10-Jan-2014].
- [208] Microsoft, “Download Windows Firewall with Advanced Security: Step-by-Step Guide: Deploying Windows Firewall and IPsec Policies from Official Microsoft Download Center.” [Online]. Available: <http://www.microsoft.com/en-us/download/details.aspx?id=11698>. [Accessed: 28-Dec-2013].
- [209] “What is Advanced Encryption Standard (AES)? - Definition from WhatIs.com.” [Online]. Available: <http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard>. [Accessed: 30-Dec-2013].
- [210] “What is Data Encryption Standard (DES)? - Definition from WhatIs.com.” [Online]. Available: <http://searchsecurity.techtarget.com/definition/Data-Encryption-Standard>. [Accessed: 30-Dec-2013].
- [211] “Why does AES encryption take more time than decryption? - Information Security Stack Exchange.” [Online]. Available: <http://security.stackexchange.com/questions/38055/why-does-aes-encryption-take-more-time-than-decryption>. [Accessed: 26-Jan-2014].
- [212] Daa Salama Abd Elminaam, “Evaluating The Performane of symmetric Encryption Algorithm,” *International Journal of Network Security*, Vol.10, No.3, May 2010, pp. 216–222.
- [213] “Vantio AuthServe.” [Online]. Available: <http://nominum.com/infrastructure/engines/authoritative-dns/>. [Accessed: 06-Dec-2013].
- [214] “Knot DNS.” [Online]. Available: <https://www.knot-dns.cz/>. [Accessed: 06-Dec-2013].

Appendix A: How to set policy with IPsec

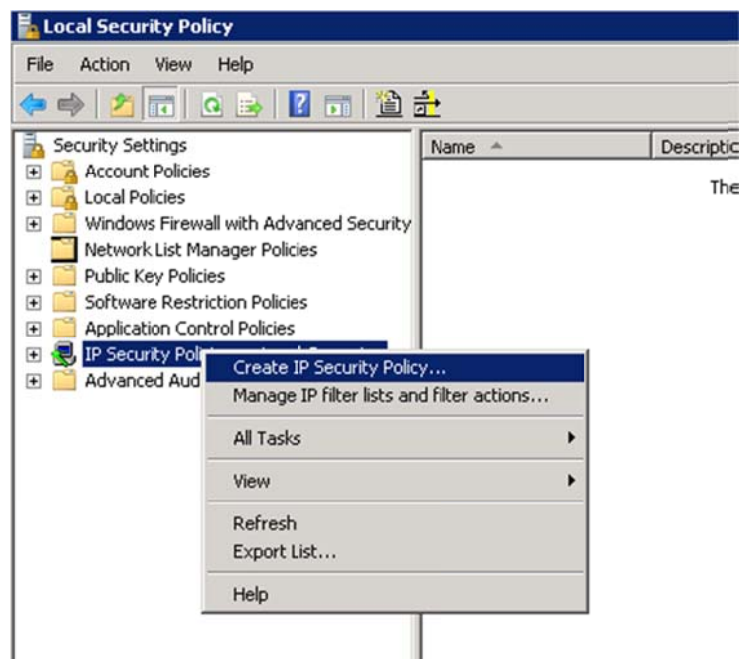
Here we show how we set host-to-host IPsec VPN tunnel with a very simple IP filter policy set.

Host1 settings

1. Open LocalSecurityPolicy tool in the server.



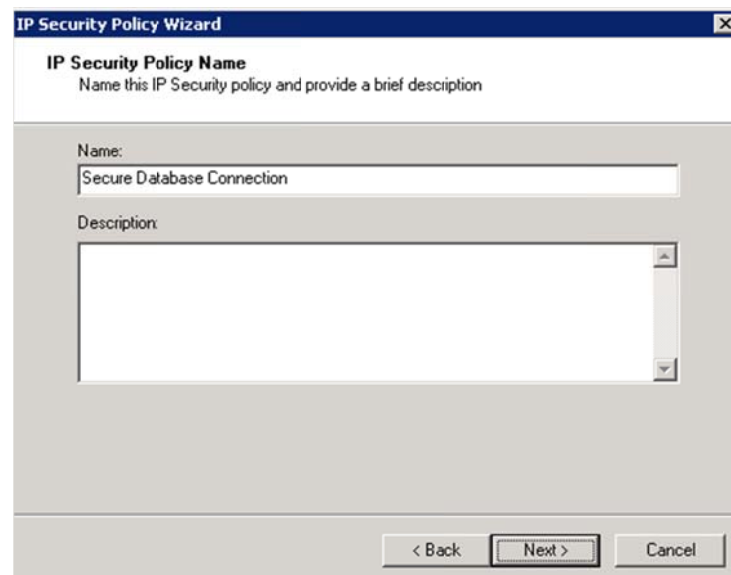
2. Right click 'IP Security Policies on Local Computer' and select 'Create IP Security Policy...'



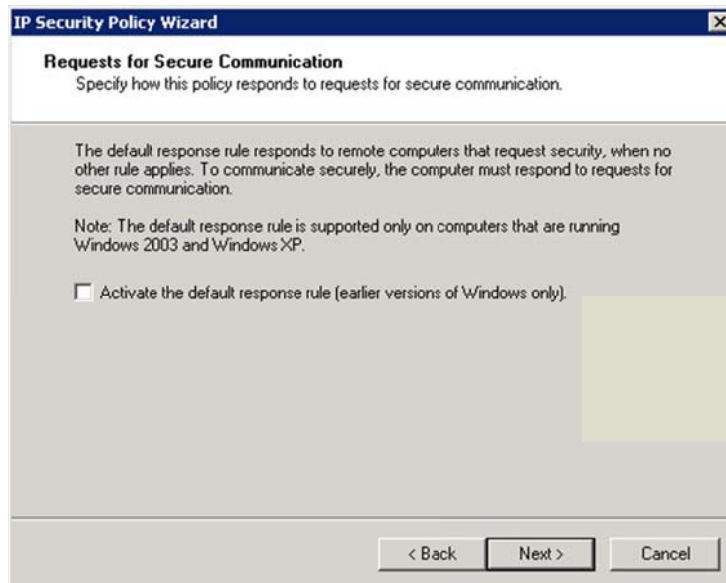
3. Click Next



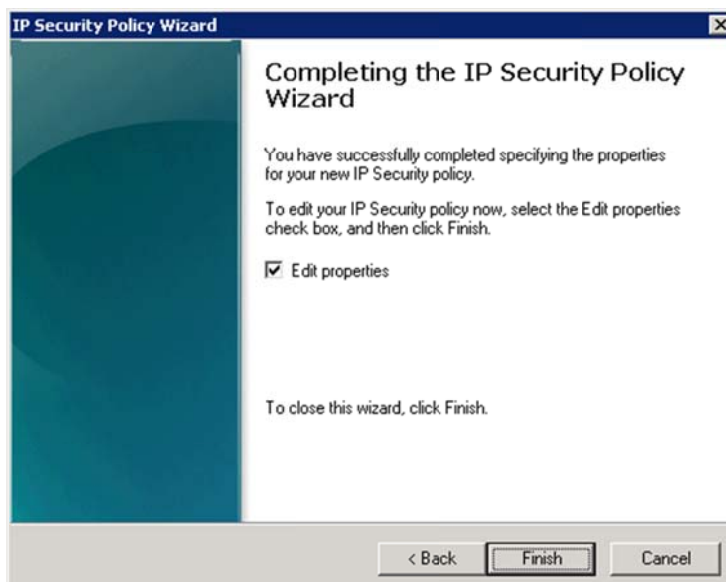
4. Provide a name for the new security policy.



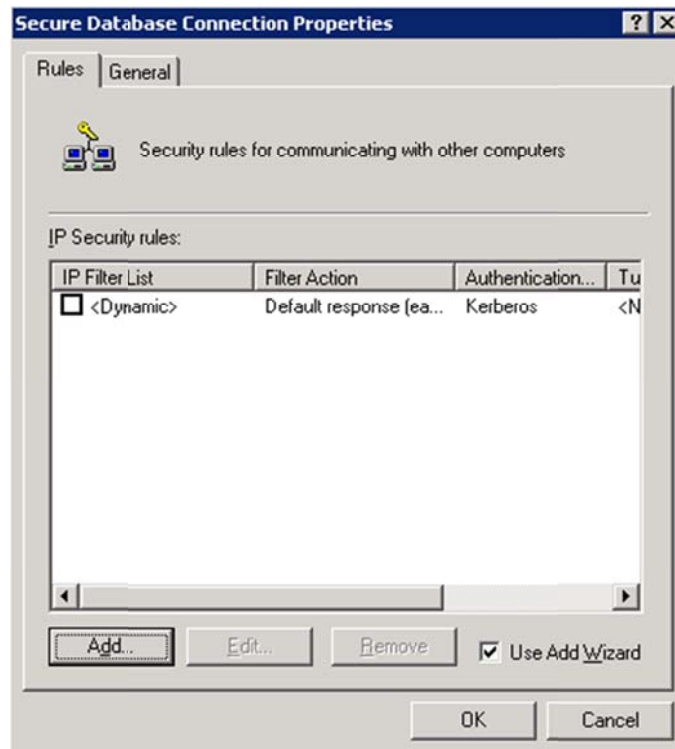
Do not check the box, select Next.



5. Click Finish after checking the 'Edit properties'.



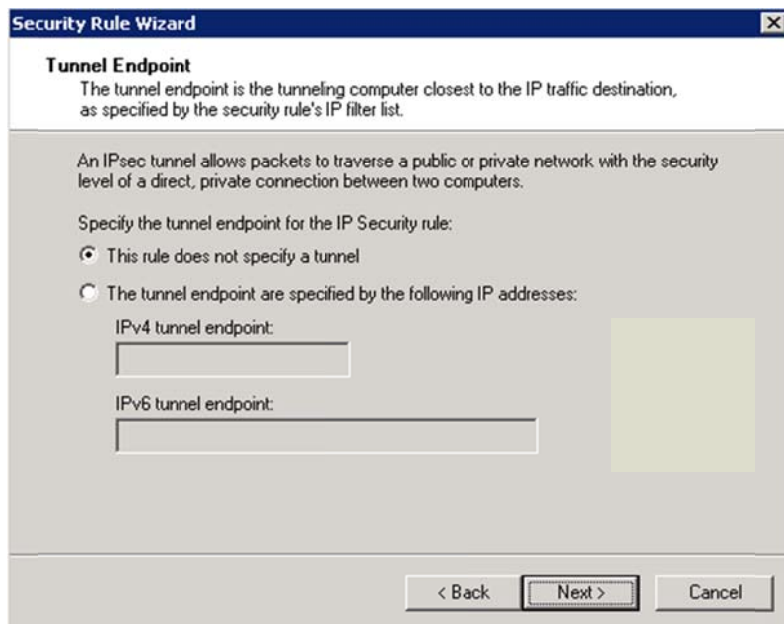
Click 'Add' button to add a new security rule.



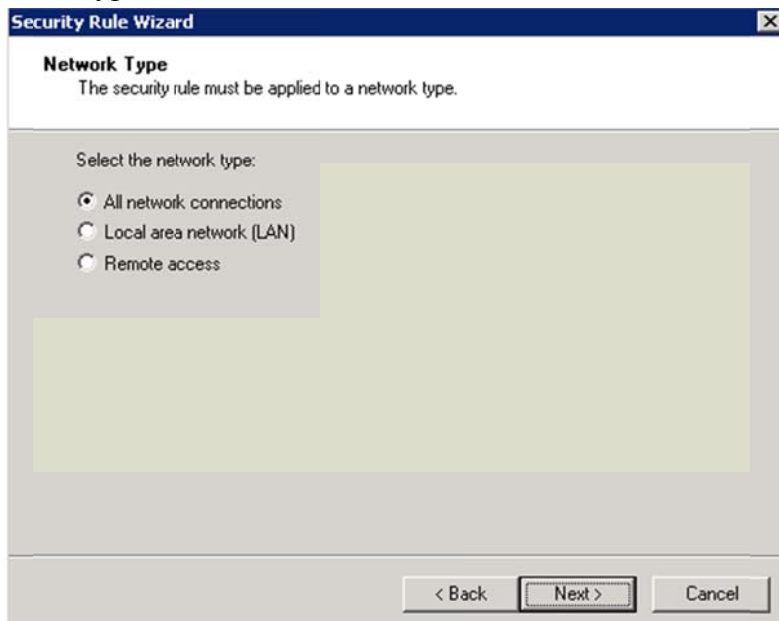
6. Click Next.



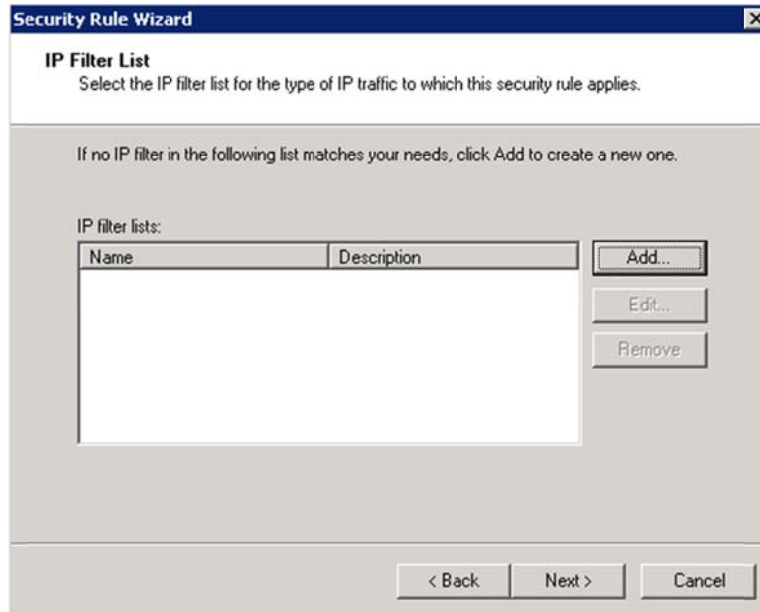
Select 'This rule does not specify a tunnel'.



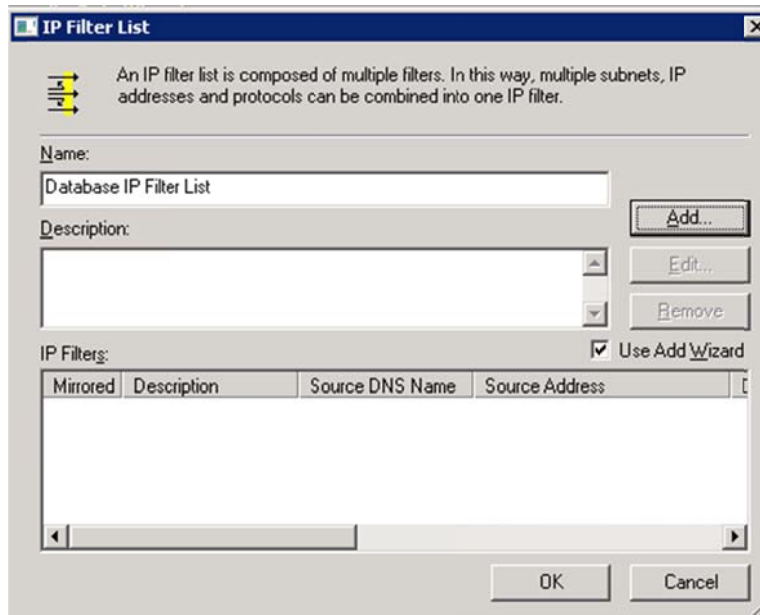
7. Select network type 'All network connections'.



8. Add a new IP filter list.



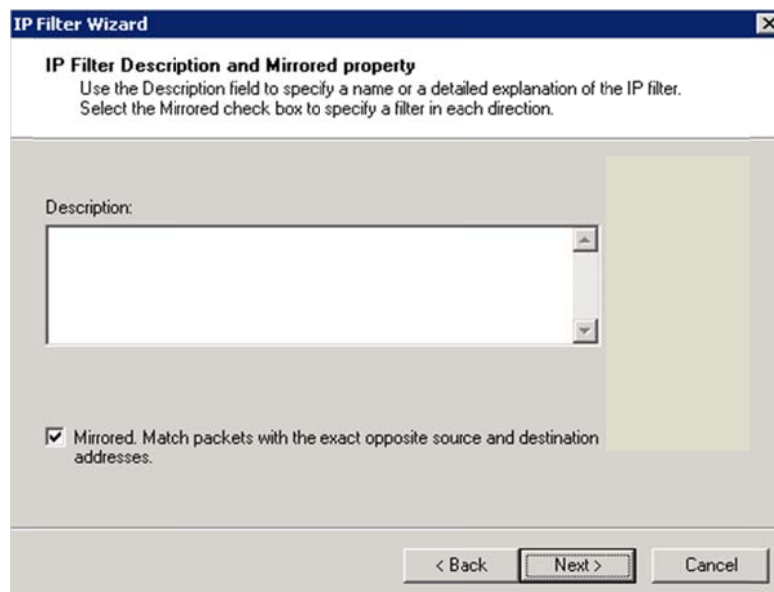
9. Click Add, to open 'IP Filter Wizard'.



10. Click Next.



11. Check the box.



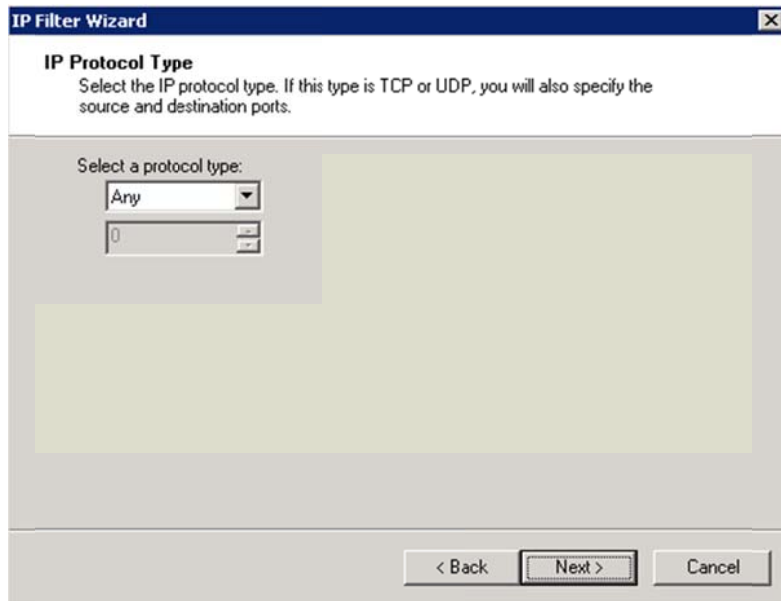
12. Select source address 'Any IP Address'.

The screenshot shows a dialog box titled "IP Filter Wizard" with a close button (X) in the top right corner. The main heading is "IP Traffic Source" with the instruction "Specify the source address of the IP traffic." Below this, there is a label "Source address:" followed by a dropdown menu. The dropdown menu is open, and "Any IP Address" is selected. The rest of the dialog box is a large, empty, light-colored rectangular area. At the bottom, there are three buttons: "< Back", "Next >" (which is highlighted with a dashed border), and "Cancel".

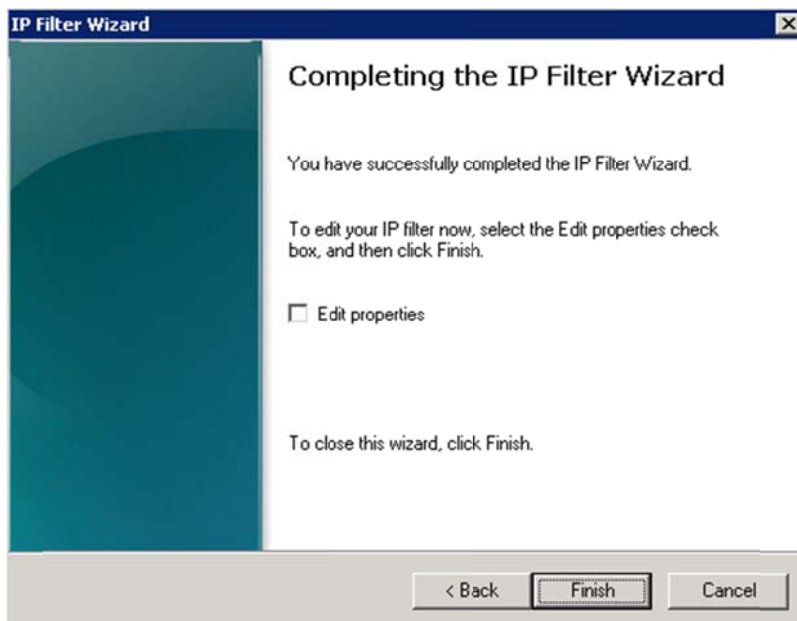
13. Select destination address 'Any IP Address'.

The screenshot shows a dialog box titled "IP Filter Wizard" with a close button (X) in the top right corner. The main heading is "IP Traffic Destination" with the instruction "Specify the destination address of the IP traffic." Below this, there is a label "Destination address:" followed by a dropdown menu. The dropdown menu is open, and "Any IP Address" is selected. The rest of the dialog box is a large, empty, light-colored rectangular area. At the bottom, there are three buttons: "< Back", "Next >" (which is highlighted with a dashed border), and "Cancel".

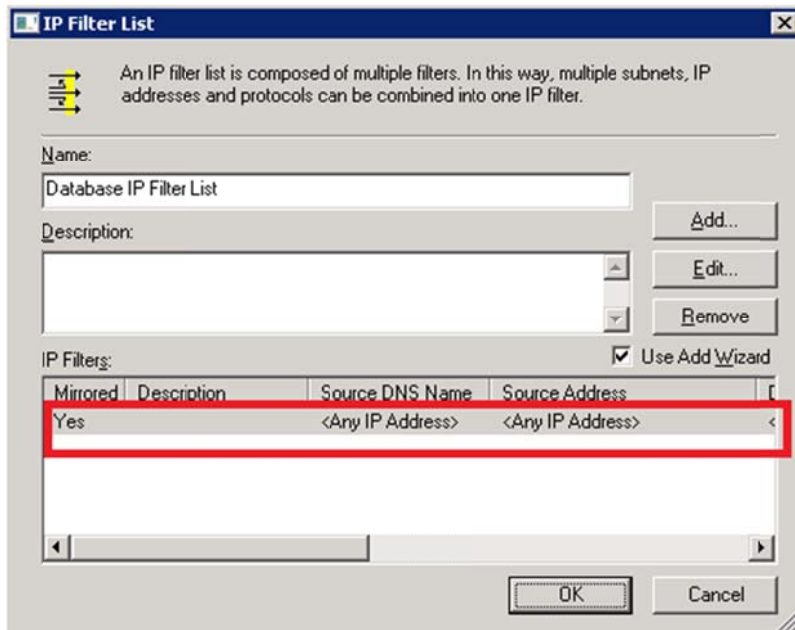
14. IP Protocol Type 'Any'.



15. Click Finish.



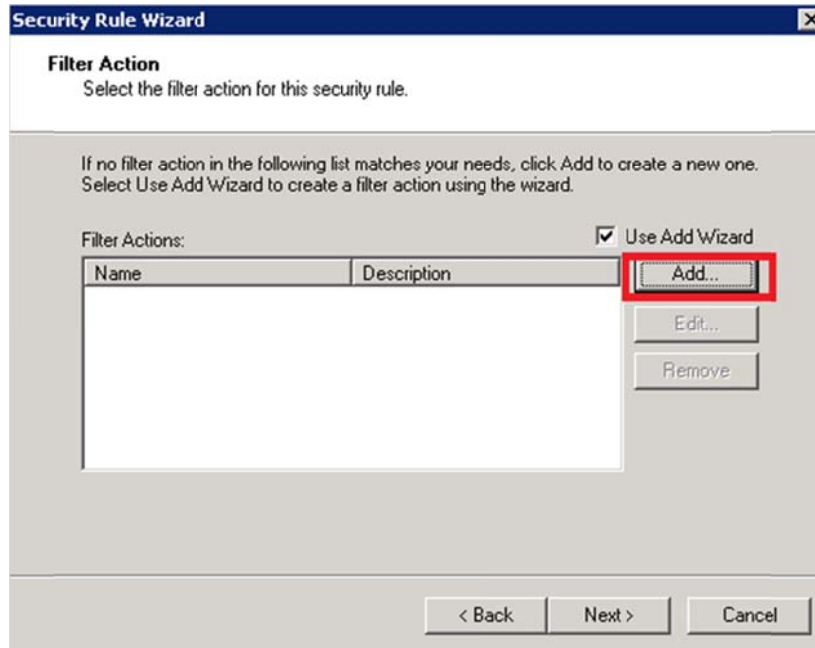
16. Click Ok.



17. Click Next.



18. Click Add.



19. Click Next.



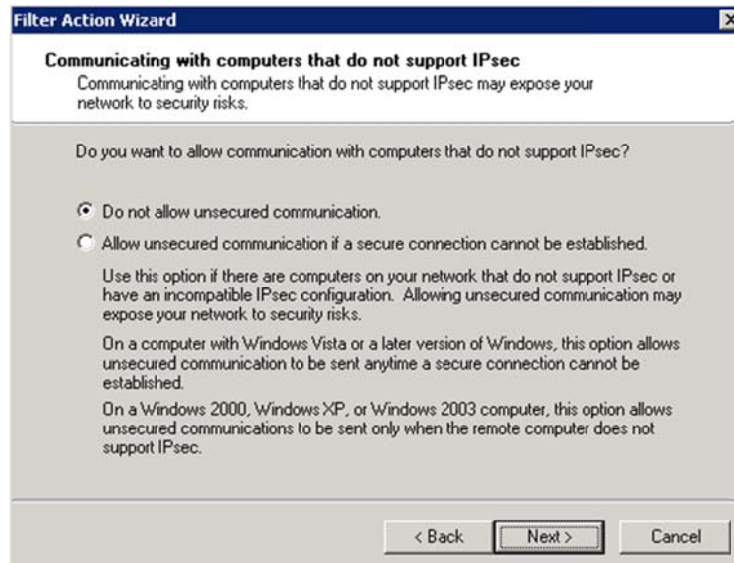
20. Provide a Filter Action Name.

The screenshot shows a dialog box titled "Filter Action Wizard" with a close button (X) in the top right corner. The main heading is "Filter Action Name" with the instruction "Name this filter action and provide a brief description." Below this, there are two input fields: "Name:" with the text "Require Security" entered, and "Description:" which is currently empty. At the bottom of the dialog, there are three buttons: "< Back", "Next >" (which is highlighted with a black border), and "Cancel".

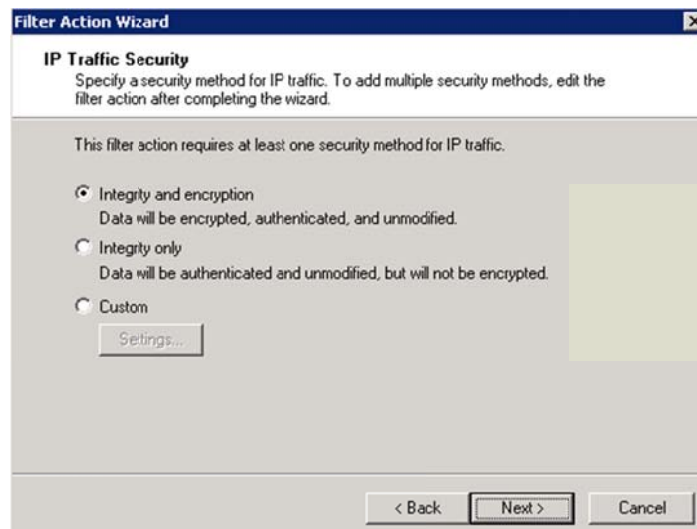
21. Select Negotiate security.

The screenshot shows a dialog box titled "Filter Action Wizard" with a close button (X) in the top right corner. The main heading is "Filter Action General Options" with the instruction "Set the filter action behavior." Below this, there are three radio button options: "Permit", "Block", and "Negotiate security". The "Negotiate security" option is selected, indicated by a filled radio button. At the bottom of the dialog, there are three buttons: "< Back", "Next >" (which is highlighted with a black border), and "Cancel".

22. Do not allow unsecured communication.



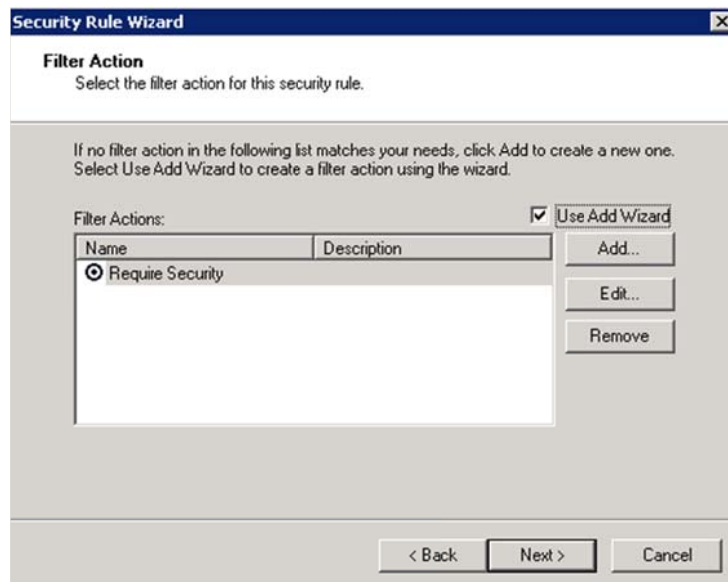
23. Integrity and encryption.



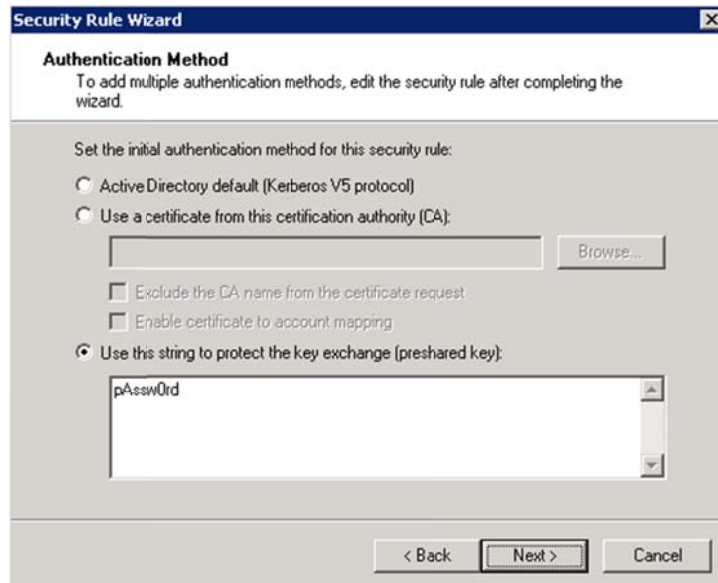
24. Click Finish.



25. Click Next.



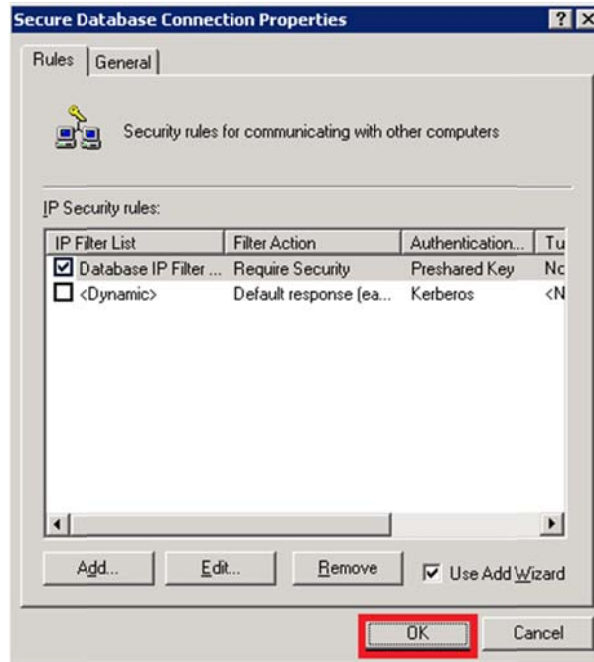
26. Use pre-shared password. For improved security, one can use a certificate.



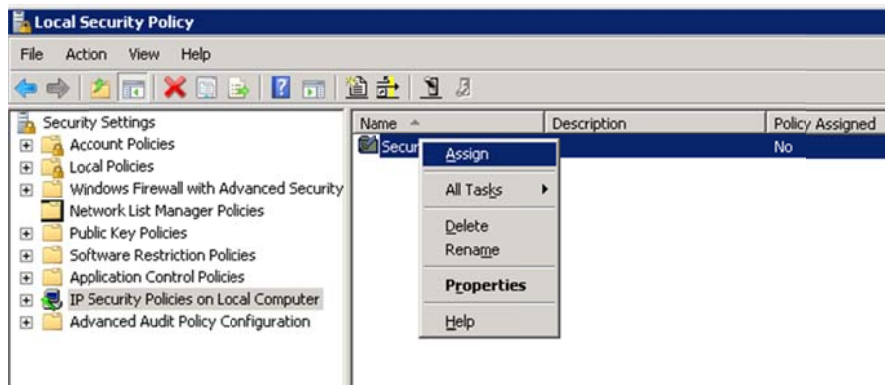
27. Click Finish.



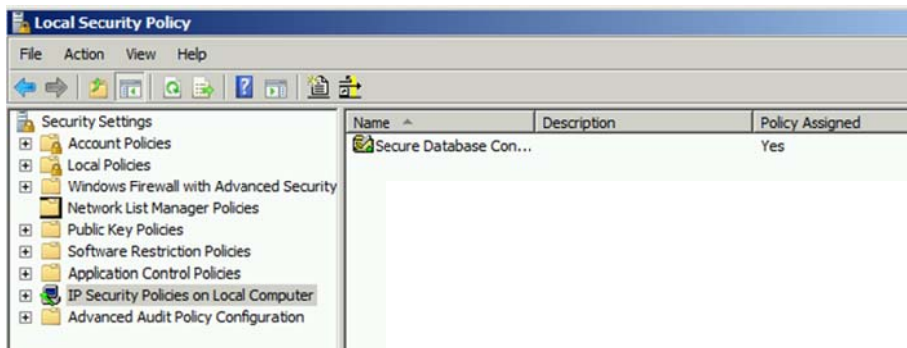
28. Click Ok.



29. In the 'LocalSecurityPolicy' tool, select 'Secure Database Connection' policy, right click and select Assign.



30. Now when the policy is assigned in server, no other machine can communicate with it without secure IPsec tunnel.



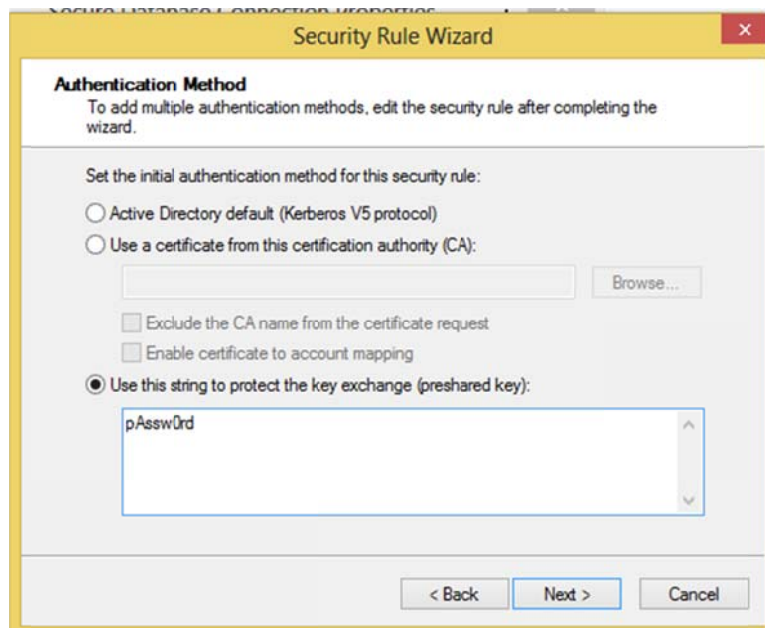
Host 2 settings

1. Open 'LocalSecurityPolicy' tool and create a new IP Security Policy.



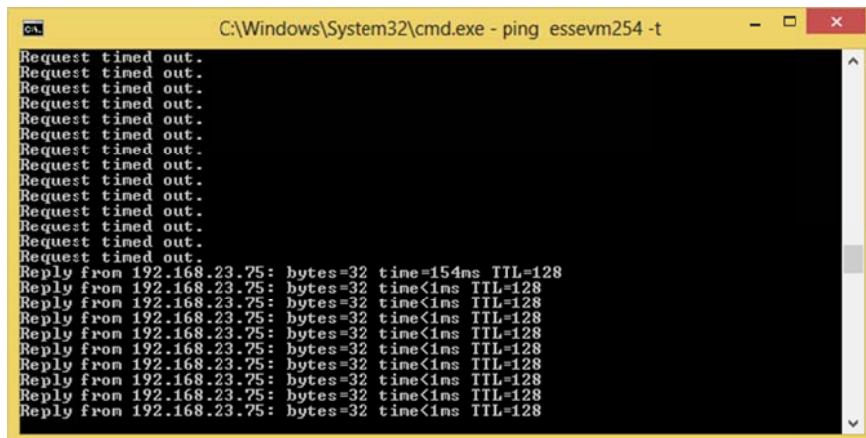
The screenshot shows the 'IP Security Policy Wizard' dialog box. The title bar reads 'IP Security Policy Wizard'. The main heading is 'IP Security Policy Name' with the instruction 'Name this IP Security policy and provide a brief description'. There are two input fields: 'Name:' with the text 'Secure Database Connection' and 'Description:' which is currently empty. At the bottom, there are three buttons: '< Back', 'Next >', and 'Cancel'.

2. Follow the same process presented for host1 configuration. Provide pre-shared secret string.



The screenshot shows the 'Security Rule Wizard' dialog box. The title bar reads 'Security Rule Wizard'. The main heading is 'Authentication Method' with the instruction 'To add multiple authentication methods, edit the security rule after completing the wizard.' There are three radio button options: 'Active Directory default (Kerberos V5 protocol)', 'Use a certificate from this certification authority (CA):', and 'Use this string to protect the key exchange (pre-shared key):'. The third option is selected. Below the 'Use a certificate...' option is a 'Browse...' button. Below the 'Use this string...' option is a text box containing 'pAssw0rd'. At the bottom, there are three buttons: '< Back', 'Next >', and 'Cancel'.

3. Assign this newly created policy in client machine. Now this machine (host2) will be able to communicate with host1, through IPsec VPN tunnel.

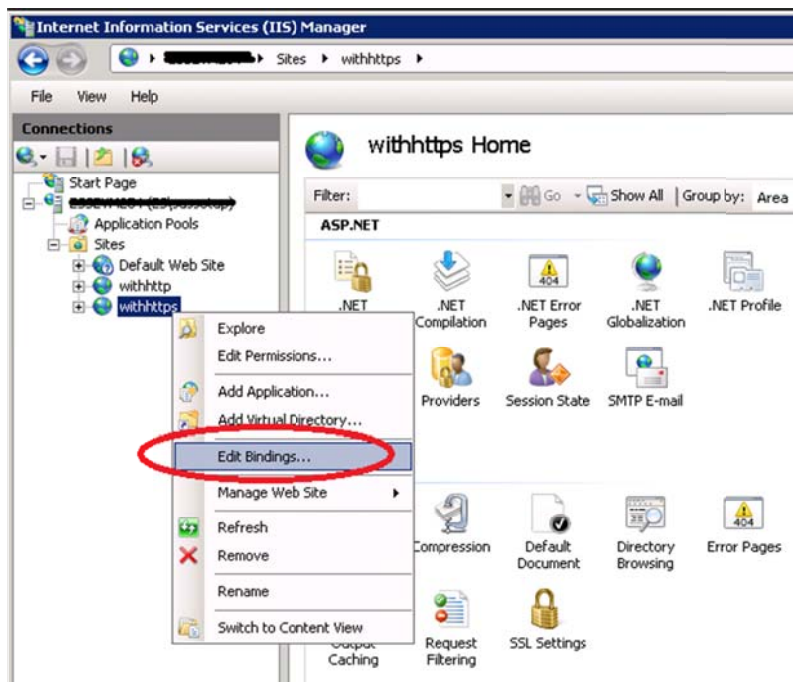


```
C:\Windows\System32\cmd.exe - ping essevm254 -t
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Reply from 192.168.23.75: bytes=32 time=154ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
Reply from 192.168.23.75: bytes=32 time<1ms TTL=128
```

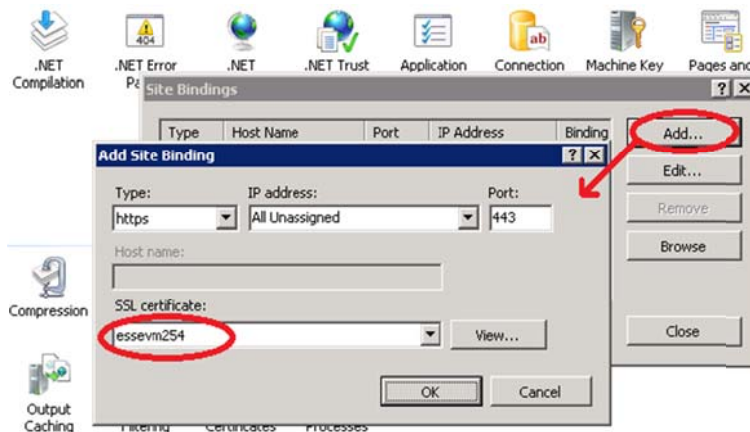
4. Policy set in this example works for any protocol. But can also be modified to filter IP packet for certain protocols (e.g. ICMP, TCP). For improved security, one can use a certificate instead of secret string.

Appendix B: Set HTTP in IIS Web Server

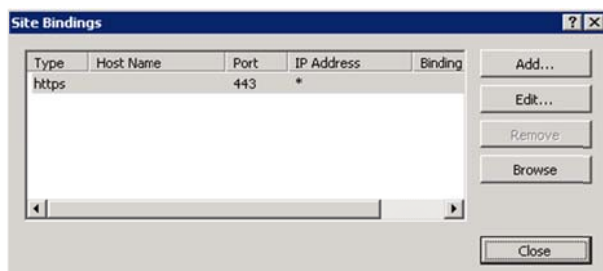
1. Open IIS Manager. Right click the website, select 'Edit Bindings...'



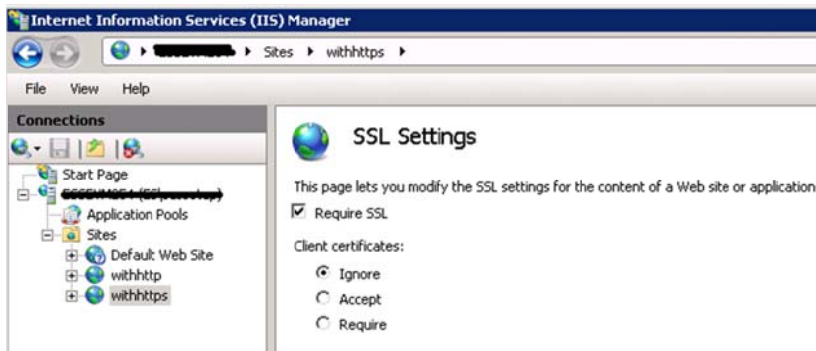
2. In *Site Bindings* click 'Add...'. In *Add Site Binding*, select Type: https. Default port for https is 443. Select the SSL certificate created for your domain, from the dropdown. You can use either a domain certificate or a self-signed certificate. The domain certificate should be signed by a root CA.



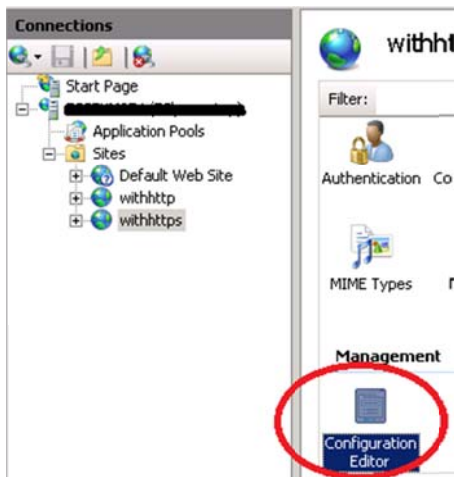
3. Click *Close* to close this window.



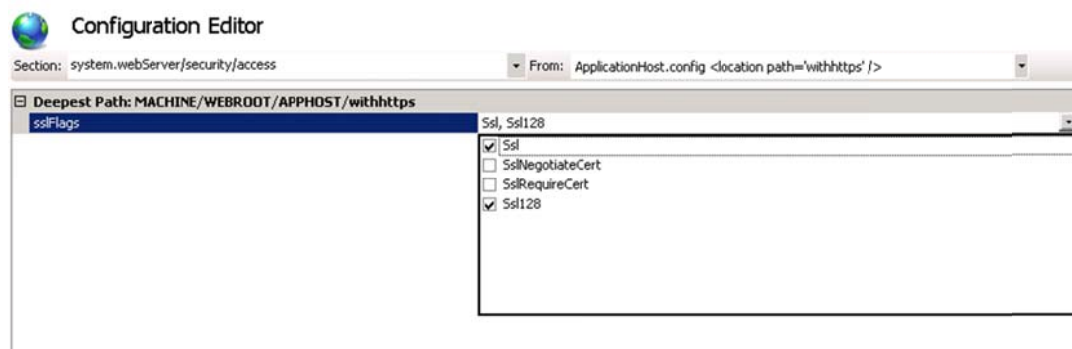
- Open SSL Settings for your website. Check *Require SSL* option. You can ignore Client certificate option. But if you want client-authentication, then you can select *Require* option. In that case client should have a valid SSL certificate signed by a valid CA, which the client must use to authenticate himself.



- Now you can browse your website securely with HTTPS.
- To set 128-bit SSL, open *Configuration Editor*.



- In section `system.webServer/security/access`, set sslFlag *Ssl128*. And then apply changes.



Appendix C: Code to measure time to get response for a SQL SELECT query through VPN Tunnel

```
public static void DatabaseQueryTimeWithVPN(bool withVPN)
{
    string vpn = string.Empty;
    if (withVPN)
    {
        vpn = "With_VPN";
    }
    else
    {
        vpn = "Without_VPN";
    }
    using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + vpn + DateTime.Now.ToString("yyyyMMddhhmmss") + @".txt",
true))
    {
        for (int i = 0; i < 101; i++)
        {
            try
            {
                string _connectionstring = @"Data
Source=server_name;Initial Catalog=database_name;User
ID=user_name;Password=password";
                string _sql = "SELECT * FROM [dbo].[Object]";
                SqlConnection _connection = new
SqlConnection(_connectionstring);
                SqlCommand _command = new SqlCommand(_sql,
_connection);

                Stopwatch sw = new Stopwatch();
                sw.Start();

                SqlDataAdapter _adapter = new SqlDataAdapter(_command);
                DataTable _table = new DataTable();
                _adapter.Fill(_table);

                sw.Stop();
                long microseconds = sw.ElapsedTicks /
(Stopwatch.Frequency / (1000L * 1000L));
                file.WriteLine(microseconds);
            }
            catch
            {
                throw;
            }
        }
    }
}
```


Appendix D: Code to measure time required to download certain content with HTTP and HTTPS

```
public static void DownloadFileWithHTTP(string file_name, string protocol,
string port)
{
    string url = protocol + @"://server_name:" + port + "/" +
file_name;
    string date = DateTime.Now.ToString("yyyyMMddhhmmss");
    string download_file = @"C:\Temp\";

    using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + "128bit" + protocol + file_name + date + @".txt", true))
    {
        for (int i = 0; i < 21; i++)
        {
            WebClient Client = new WebClient();
            Client.Headers.Add("Cache-Control", "no-cache");
            Stopwatch sw = new Stopwatch();
            sw.Start();
            Client.DownloadFile(url, download_file + date + i +
file_name);

            while (true)
            {
                if (File.Exists(download_file + date + i + file_name))
                {
                    break;
                }
            }
            sw.Stop();
            long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency
/ (1000L * 1000L));
            file.WriteLine(microseconds);
            Client.Dispose();
            System.Threading.Thread.Sleep(2000);
        }
    }
}
```


Appendix E: Download time comparison for 390MB file

HTTP (μ s)	40bit-SSL (μ s)	128bit-SSL (μ s)
6619888	19424229	17705437
5665637	16415765	17628397
7049345	16385790	28657104
6722050	17265591	18749434
16938904	16736338	16680060
15062397	17319341	16391779
12447988	16271539	17664477
16773799	16873728	16642464
13977538	15680192	17291940
10783901	16020279	16987831
10129006	16101393	16261636
15227629	16187010	17222976
11082007	16144726	16007760
14513519	16070624	16692974
12847325	15609038	15965587
12112063	15916124	16282130
13238979	15597212	15581126
6021303	15588578	15999261
9013063	15880652	16013349
8257191	15720894	16081402
8911434	16128582	15637327
13060831	16059672	15636593
6503861	16057336	16002641
6090674	15995457	15510446
9864397	15989264	15908172
7745550	16058579	15632266
9456673	16054261	15624307

Appendix F: Download time comparison for 3.13 MB file

HTTP (ms)	HTTPS (40bit-SSL) (ms)	HTTPS (128-bit-SSL) (ms)
44.4	154.82	154.81
44.03	152.41	166.1
44.97	162.17	190.56
44.96	149.83	152.06
44.18	152.63	175.64
47.15	186.6	156.38
45.33	152.52	128.23
45.39	150.88	156.65
44.69	152.93	154.24
51.12	159.25	144.65
45.48	158.91	161.43
45.02	156.71	149.14
44.7	159.39	151.88
44.68	146.77	159.87
44.45	147.65	146.17
44.77	145.87	153.27
44.61	190.17	151.77
45.5	126.54	162.25

Appendix G: SQL SELECT query time comparison for different VPN setup

ipsec-with- vpn (ms)	open-vpn- withvpn (ms)	without- vpn (ms)	IPSec-with- VPN (seconds)	Open-VPN- withVPN (seconds)	without-VPN (seconds)
542499	264456	220201	0.54	0.26	0.22
498138	292032	240466	0.5	0.29	0.24
529162	265498	285025	0.53	0.27	0.29
535560	257237	287882	0.54	0.26	0.29
527619	325463	290996	0.53	0.33	0.29
628570	283707	340699	0.63	0.28	0.34
486958	238674	257940	0.49	0.24	0.26
552691	239598	261221	0.55	0.24	0.26
537291	283773	268246	0.54	0.28	0.27
640124	243715	247224	0.64	0.24	0.25
605522	225317	253019	0.61	0.23	0.25
475507	260355	268410	0.48	0.26	0.27
454648	275642	294362	0.45	0.28	0.29
417593	216715	238874	0.42	0.22	0.24
447618	225165	257448	0.45	0.23	0.26
455745	235009	244404	0.46	0.24	0.24
455464	238878	260151	0.46	0.24	0.26
420402	245891	265660	0.42	0.25	0.27
433605	268831	249270	0.43	0.27	0.25
437959	240216	256531	0.44	0.24	0.26
433456	236971	255731	0.43	0.24	0.26
416928	253292	249677	0.42	0.25	0.25
418622	224169	249205	0.42	0.22	0.25
392940	231309	248959	0.39	0.23	0.25
438361	258923	238777	0.44	0.26	0.24
432830	235224	267288	0.43	0.24	0.27
444449	241921	241960	0.44	0.24	0.24
404964	241363	261750	0.4	0.24	0.26

457458	287673	247280	0.46	0.29	0.25
473775	244811	238416	0.47	0.24	0.24
459226	230502	252478	0.46	0.23	0.25
433959	234665	251870	0.43	0.23	0.25
425034	237319	260009	0.43	0.24	0.26
419331	225067	255516	0.42	0.23	0.26
392948	234249	250498	0.39	0.23	0.25
435950	227171	240628	0.44	0.23	0.24
412111	262062	263300	0.41	0.26	0.26
447693	224131	268760	0.45	0.22	0.27
444187	248484	261439	0.44	0.25	0.26
452084	250360	246344	0.45	0.25	0.25
459374	223156	261100	0.46	0.22	0.26
393309	245752	260278	0.39	0.25	0.26
408719	225175	245245	0.41	0.23	0.25
460758	240391	250151	0.46	0.24	0.25
429217	237371	252518	0.43	0.24	0.25
424170	236430	237887	0.42	0.24	0.24
470226	234145	251132	0.47	0.23	0.25
428279	237535	253394	0.43	0.24	0.25
410476	241150	266996	0.41	0.24	0.27
421197	238731	246860	0.42	0.24	0.25
400522	243599	247487	0.4	0.24	0.25
440069	239574	249984	0.44	0.24	0.25
426170	304971	248571	0.43	0.3	0.25
420002	259236	249205	0.42	0.26	0.25
436510	215632	244047	0.44	0.22	0.24
416689	235138	256050	0.42	0.24	0.26
452456	241709	265454	0.45	0.24	0.27
379916	241812	261822	0.38	0.24	0.26
427627	236804	248771	0.43	0.24	0.25
444216	258225	243298	0.44	0.26	0.24
441017	234572	243447	0.44	0.23	0.24

439130	234483	265675	0.44	0.23	0.27
451048	256233	262629	0.45	0.26	0.26
423616	239825	249666	0.42	0.24	0.25
451420	251121	245192	0.45	0.25	0.25
420335	267189	243221	0.42	0.27	0.24
454004	214463	254070	0.45	0.21	0.25
411791	232464	286898	0.41	0.23	0.29
463304	225509	247994	0.46	0.23	0.25
403114	224059	241330	0.4	0.22	0.24
451401	240687	253163	0.45	0.24	0.25
450725	220339	264219	0.45	0.22	0.26
436858	241593	244905	0.44	0.24	0.24
387106	232694	253263	0.39	0.23	0.25
451353	235159	253695	0.45	0.24	0.25
431619	232992	247663	0.43	0.23	0.25
432919	224119	259045	0.43	0.22	0.26
407610	226851	254580	0.41	0.23	0.25
435626	231151	257073	0.44	0.23	0.26
414602	226701	246399	0.41	0.23	0.25
422334	236668	247121	0.42	0.24	0.25
427789	224687	245325	0.43	0.22	0.25
425600	226601	264547	0.43	0.23	0.26
411528	229031	247156	0.41	0.23	0.25
453682	240930	258958	0.45	0.24	0.26
401120	234320	253269	0.4	0.23	0.25
408375	219455	249212	0.41	0.22	0.25
445874	239362	264323	0.45	0.24	0.26
416639	228631	249897	0.42	0.23	0.25
417969	228352	251777	0.42	0.23	0.25
449324	231873	253950	0.45	0.23	0.25
399127	220661	248240	0.4	0.22	0.25
403950	235152	251516	0.4	0.24	0.25
431953	246764	249933	0.43	0.25	0.25

427316	223550	248366	0.43	0.22	0.25
421893	237184	255176	0.42	0.24	0.26
457434	219999	242653	0.46	0.22	0.24
417518	234491	251372	0.42	0.23	0.25
403124	242826	268993	0.4	0.24	0.27

Appendix H: SQL SELECT query time (in milliseconds) comparison for different VPN setup and for different result sizes

4.789 MB		0.734 MB	
OpenVPN	IPSec	OpenVPN	IPSec
(ms)	(ms)	(ms)	(ms)
286.5	542.5	62.3	155.75
264.46	498.14	70.61	124.25
292.03	529.16	69.3	148.03
265.5	535.56	58.05	107.5
257.24	527.62	65.6	137.39
325.46	628.57	74.35	120.28
283.71	486.96	56.88	115.52
238.67	552.69	60.98	109.64
239.6	537.29	65.12	111.74
283.77	640.12	74.2	109.83
243.72	605.52	55.36	114.8
225.32	475.51	66.4	151.03
260.36	454.65	68.69	118.11
275.64	417.59	69.07	110
216.72	447.62	60.45	113.01
225.17	455.75	77.53	114.28
235.01	455.46	75.18	128.32
238.88	420.4	61.54	118.58
245.89	433.61	68.25	110.07
268.83	437.96	80.46	109.35
240.22	433.46	61.55	111.89
236.97	416.93	55.29	124.22
253.29	418.62	63.2	143.52
224.17	392.94	93.26	115.56
231.31	438.36	55.34	109.1
258.92	432.83	61.13	123.08
235.22	444.45	63.28	120.54
241.92	404.96	81.12	116.09
241.36	457.46	65.2	112.73
287.67	473.78	68.64	109.19

Appendix I: Code to measure AES encryption time

```
Using
    System;

using
    System.Collections.Generic;

using
    System.Linq;

using
    System.Text;

using
    System.Data.SqlClient;

using
    System.Diagnostics;

using
    System.Data;

using
    MySql.Data.MySqlClient;

using
    System.Threading;

using
    System.IO;

namespace
    test_thesis
{
    class Program
```

```

{
static void Main(string[] args)
{
DatabaseInsertQueryTime(
false); //true for encr, false for unencrp
//DatabaseSelectTime(true);
}

public static void DatabaseSelectTime(bool withEncryption)
{
string encrp = string.Empty;
if (withEncryption)
{
encrp =
"With_encryption";
using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\"
+ "Select_" + encrp + DateTime.Now.ToString("yyyyMMddhhmmss")
+ @".txt", true))
{
Stopwatch sw = new Stopwatch();
sw.Start();
try
{
 MySqlConnection conn;
 MySqlCommand cmd = new MySql.Data.MySqlClient.MySqlCommand();
 string myConnectionString;

myConnectionString =
"server=127.0.0.1;uid=root;" +
"pwd=;database=testdb;";

conn =
new MySql.Data.MySqlClient.MySqlConnection(myConnectionString);

```

```

conn.Open();

string Name = "Test";
int Price = 200;

cmd.Parameters.AddWithValue(
"@Name", Name);

cmd.Parameters.AddWithValue(
"@Price", Price);
string SQL = "SELECT Id, AES_DECRYPT(Name, 'usa2010'), AES_DECRYPT(Price,
'usa2010') from product_info";

cmd.Connection = conn;

cmd.CommandText = SQL;

MySqlDataReader rdr = cmd.ExecuteReader();

sw.Stop();

long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));

file.WriteLine(microseconds);

Thread.Sleep(1000);
}

catch
{
throw;
}
}
}

else
{

encrp =

"Without_encryption";
using (System.IO.StreamWriter file = new System.IO.StreamWriter(

```

```

@"C:\Temp\"

+ "Select_" + encrp + DateTime.Now.ToString("yyyyMMddhhmmss")
+ @".txt", true))

{

Stopwatch sw = new Stopwatch();

sw.Start();

try

{

SqlConnection conn;
SqlCommand cmd = new MySql.Data.MySqlClient.MySqlCommand();
string myConnectionString;

myConnectionString =

"server=127.0.0.1;uid=root;" +
"pwd=;database=testdb;";

conn =

new MySql.Data.MySqlClient.MySqlConnection(myConnectionString);

conn.Open();

string Name = "Test";
int Price = 200;

cmd.Parameters.AddWithValue(

"@Name", Name);

cmd.Parameters.AddWithValue(

"@Price", Price);
string SQL = "SELECT Id, Name, Price from product_info";

cmd.Connection = conn;

cmd.CommandText = SQL;

MySqlDataReader rdr = cmd.ExecuteReader();

sw.Stop();

long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));

```

```

file.WriteLine(microseconds);

Thread.Sleep(1000);
}
catch
{
throw;
}
}
}
}

public static void DatabaseInsertQueryTime(bool withEncryption)
{
string encrp = string.Empty;
if (withEncryption)
{
encrp =

"With_encryption";
using (System.IO.StreamWriter file = new System.IO.StreamWriter(

@"C:\Temp\"

+ "Insert_" + encrp + DateTime.Now.ToString("yyyyMMddhhmmss")
+ @".txt", true))
{
Stopwatch sw = new Stopwatch();

sw.Start();

for (int i = 0; i < 100; i++)
{
try
{

```



```

 MySqlConnection conn;
 MySqlCommand cmd = new MySql.Data.MySqlClient.MySqlCommand();
 string myConnectionString;

 myConnectionString =

 "server=127.0.0.1;uid=root;" +
 "pwd=;database=testdb;";

 conn =

 new MySql.Data.MySqlClient.MySqlConnection(myConnectionString);

 conn.Open();

 string Name = "Test";
 string Price = "200";
 string image = "x";

 cmd.Parameters.AddWithValue(

 "@Name", Name);

 cmd.Parameters.AddWithValue(

 "@Price", Price);

 cmd.Parameters.AddWithValue(

 "@image", image);
 string SQL = "INSERT into product_info(Id,Name,Price,image) VALUES (NULL,
 AES_ENCRYPT(@Name, 'usa2010'),AES_ENCRYPT(@Price,
 'usa2010'),AES_ENCRYPT(@image, 'usa2010'))"; //for normal insert

 cmd.Connection = conn;

 cmd.CommandText = SQL;

 cmd.ExecuteNonQuery();

 sw.Stop();

 long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
 1000L));

 file.WriteLine(microseconds);

 Thread.Sleep(1000);

 }

 catch

 {

```

```

throw;

}

}

}

}

else

{

encrp =

"Without_encryption";
using (System.IO.StreamWriter file = new System.IO.StreamWriter(

@"C:\Temp\"

+ "Insert_" + encrp + DateTime.Now.ToString("yyyyMMddhhmmss")
+ @".txt", true))

{

Stopwatch sw = new Stopwatch();

sw.Start();

for (int i = 0; i < 100; i++)

{

try

{

 MySqlConnection conn;
 MySqlCommand cmd = new MySql.Data.MySqlClient.MySqlCommand();
 string myConnectionString;

myConnectionString =

"server=127.0.0.1;uid=root;" +
"pwd=;database=testdb;";

conn =

new MySql.Data.MySqlClient.MySqlConnection(myConnectionString);

conn.Open();

string Name = "Test";
string Price = "200";

```

```

string image = "x";

cmd.Parameters.AddWithValue(
"@Name", Name);

cmd.Parameters.AddWithValue(
"@Price", Price);

cmd.Parameters.AddWithValue(
"@image", image);
string SQL = "insert into product_info(Id,Name,Price,image) values(NULL,
@Name, @Price, @image)";//for normal insert

cmd.Connection = conn;

cmd.CommandText = SQL;

cmd.ExecuteNonQuery();

sw.Stop();

long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));

file.WriteLine(microseconds);

Thread.Sleep(1000);
}

catch
{
throw;
}

}

}

}

}

}

}

```

Appendix J: Time for inserting data

We ran this experiment 100 times. Client and database server were residing in the same machine. So there was no network related delay. In this experiment in every run we got the same value.

Insert data with AES encryption big_database	Insert data without encryption big_database	Insert data with AES encryption small_database	Insert data without encryption small_database
1755442	218839	243458	244915

Appendix K: Time for selecting data

With AES decryption big_database	Without decryption big_database	With AES decryption small_database	Without decryption small_database
164713	161951	162741	160084

Appendix L: Analysis of data in appendices E & F

		Per byte data transfer time in μs (using median time)		
		Per byte data transfer time = median time/file size		
File size		HTTP	HTTPS (40bit-SSL)	HTTPS (128bit-SSL)
In Megabytes	In Bytes			
3.13MB file	3,282,043	0.013670601	0.046550	0.047082
390MB file	408,944,640	0.063719331	0.039271017	0.039764884
Ratios	HTTPS (40bit-SSL)/HTTP	HTTPS (128bit-SSL)/HTTP		
1.5MB file	3.405115061	1.011428272		
390MB file	0.616312451	1.012575848		
	File size (In Bytes)	HTTP	HTTP (40bit-SSL)	HTTP (128bit-SSL)
Data transfer time for initial 1.5MB (Median value)	1,572,864	44,867.50	152,779.00	154,525.00
Data transfer time for 390MB (Median value)	408,944,640	26,057,679.00	16,059,672.00	16,261,636.00
Data transfer time for subsequent (390MB - 1.5MB)	405,662,597	26,012,811.50	15,906,893.00	16,107,111.00
Per byte data transfer time for subsequent data	(time/file size)	0.064124	0.039212	0.039706
Required time to transfer 3.13Mb sized file at the data transfer rate of sub-sequent data		210458.552	128695.88	130315.7583
HTTPS session creation time			24083.12001	24209.24166

Appendix M. Analysis of data in appendix H

File size		Per byte data transfer time in millisecond using median time		
Megabytes	Bytes	OpenVPN	IPSec	
4.789	5021630.464	0.048750103	0.0907285	
0.734	769654.784	0.084973161	0.150119251	
File size (MB)	IPSec/OpenVPN Ratio			
4.789	1.861093523			
0.734	1.766666667			
File size		Average file transfer time in millisecond (Median)		
Megabytes	Bytes	OpenVPN	IPSec	
4.789	5021630.464	244.805	455.605	
0.734	769654.784	65.4	115.54	
Subsequent data	4251975.68	Required time to transfer subsequent data	179.405	340.065
Per byte data transfer rate for subsequent data			4.21933E-05	7.99781E-05
Ratio: per byte transfer time for first 0,734MB/ transfer rate for subsequent data	OpenVPN	IPSec		
	2013.900475	1877.00412		
It shows that first 0,734MB per byte data transfer time is almost two times higher than subsequent data transfer time.				
Required time to transfer 0,734MB sized file at the data transfer rate of sub-sequent data			32.47429593	61.55553884
Session creation time			32.92570407	53.98446116

For both 4,789MB and 0,734MB sized files per byte data transfer time for IPSec is almost two times higher than OpenVPN. We also calculate the session creation time. For OpenVPN it is 32.93 millisecond and for IPSec it is 53.99 millisecond.

		Required time to transfer subsequent data		
Subsequent data	4251975,68		179,405	340,065
Per byte data transfer rate for subsequent data			0,042193327	0,079978
Ratio: per byte transfer time for first 0,734MB/ transfer rate for subsequent data	OpenVPN	IPSec		
	2,013900475	1,87700412		

It shows that for first 0,734MB data, per byte data transfer time is almost two times higher than subsequent data transfer time (for both IPSec and OpenVPN). It is because to create a secured connection between nodes there is time required to create a secured session. Regardless the file size which can be consistent. As a result due to the initial session creation time for a small file per byte data transfer time gets higher than the time for a bigger file.

Appendix N. Storage security with AES and TripleDES

We have used the following C# code to perform experiment with AES and TripleDES for storage security.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Threading;
using System.IO;
using System.Security.Cryptography;
using System.Runtime.InteropServices;
using System.Configuration;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            move_without_security("no_security");
            AES_encrypt();
            AES_decrypt();
            tripleDES_encrypt_decrypt();
        }
        public static void move_without_security(String security)
        {
            string bigfileName = "big_file.doc";
            string smallfileName = "small_file.doc";
            string sourcePath = @"C:\Temp";
            string targetPath = @"C:\DestTemp";

            // Use Path class to manipulate file and directory paths.
            string sourceFileBig = System.IO.Path.Combine(sourcePath, bigfileName);
            string sourceFileSmall = System.IO.Path.Combine(sourcePath, smallfileName);
            string destFileBig = System.IO.Path.Combine(targetPath, bigfileName);
            string destFileSmall = System.IO.Path.Combine(targetPath, smallfileName);

            using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + security + "_big_" + DateTime.Now.ToString("yyyyMMddhhmmss") +
@"\.txt", true))
            {
                for (int i = 0; i < 100; i++)
                {
                    Stopwatch sw = new Stopwatch();
                    sw.Start();
                    System.IO.File.Copy(sourceFileBig, destFileBig, true);
                    while (System.IO.File.Exists(destFileBig))

```

```

        {
            break;
        }
        sw.Stop();
        System.IO.File.Delete(destFileBig);

        long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
        file.WriteLine(microseconds);
        Thread.Sleep(1000);
    }
}

using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + security + "_small_" + DateTime.Now.ToString("yyyyMMddhhmmss") +
@"\.txt", true))
{
    for (int i = 0; i < 100; i++)
    {
        Stopwatch sw = new Stopwatch();
        sw.Start();
        System.IO.File.Copy(sourceFileSmall, destFileSmall, true);
        while (System.IO.File.Exists(destFileSmall))
        {
            break;
        }
        sw.Stop();
        System.IO.File.Delete(destFileSmall);

        long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
        file.WriteLine(microseconds);
        Thread.Sleep(1000);
    }
}
}

public static void AES_encrypt()
{
    string bigfileName = "big_file.doc";
    string smallfileName = "small_file.doc";
    string sourcePath = @"C:\Temp";
    string targetPath = @"\\RIZVI-PC\experiment";

    // Use Path class to manipulate file and directory paths.
    string sourceFileBig = System.IO.Path.Combine(sourcePath, bigfileName);
    string sourceFileSmall = System.IO.Path.Combine(sourcePath, smallfileName);
    string destFileBig = System.IO.Path.Combine(targetPath, bigfileName);
    string destFileSmall = System.IO.Path.Combine(targetPath, smallfileName);

    using (System.IO.StreamWriter file = new System.IO.StreamWriter(

```

```

@"C:\Temp\" + "AES_encrypt" + "_small_" +
DateTime.Now.ToString("yyyyMMddhhmmss") + @".txt", true))
    {
        for (int i = 0; i < 100; i++)
        {
            Stopwatch sw = new Stopwatch();
            sw.Start();
            EncryptFile(sourceFileSmall, destFileSmall);
            while (System.IO.File.Exists(destFileSmall))
            {
                break;
            }
            sw.Stop();
            System.IO.File.Delete(destFileSmall);

            long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
            file.WriteLine(microseconds);
            Thread.Sleep(1000);
        }
    }
    using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + "AES_encrypt" + "_big_" + DateTime.Now.ToString("yyyyMMddhhmmss")
+ @".txt", true))
    {
        for (int i = 0; i < 100; i++)
        {
            Stopwatch sw = new Stopwatch();
            sw.Start();
            EncryptFile(sourceFileBig, destFileBig);
            while (System.IO.File.Exists(destFileBig))
            {
                break;
            }
            sw.Stop();
            System.IO.File.Delete(destFileBig);

            long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
            file.WriteLine(microseconds);
            Thread.Sleep(1000);
        }
    }
}

public static void AES_decrypt()
{
    string bigfileName = "big_file_en.doc";
    string smallfileName = "small_file_en.doc";
    string sourcePath = @"C:\Temp";

```

```

//string targetPath = @"\\RIZVI-PC\experiment";
string targetPath = @"C:\DestTemp";

// Use Path class to manipulate file and directory paths.
string sourceFileBig = System.IO.Path.Combine(sourcePath, bigfileName);
string sourceFileSmall = System.IO.Path.Combine(sourcePath, smallfileName);
string destFileBig = System.IO.Path.Combine(targetPath, bigfileName);
string destFileSmall = System.IO.Path.Combine(targetPath, smallfileName);

using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + "AES_decrypt" + "_small_" +
DateTime.Now.ToString("yyyyMMddhhmmss") + @".txt", true))
{
    for (int i = 0; i < 100; i++)
    {
        Stopwatch sw = new Stopwatch();
        sw.Start();
        DecryptFile(sourceFileSmall, destFileSmall);
        while (System.IO.File.Exists(destFileSmall))
        {
            break;
        }
        sw.Stop();
        System.IO.File.Delete(destFileSmall);

        long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
        file.WriteLine(microseconds);
        Thread.Sleep(1000);
    }
}
using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + "AES_decrypt" + "_big_" + DateTime.Now.ToString("yyyyMMddhhmmss")
+ @".txt", true))
{
    for (int i = 0; i < 100; i++)
    {
        Stopwatch sw = new Stopwatch();
        sw.Start();
        DecryptFile(sourceFileBig, destFileBig);
        while (System.IO.File.Exists(destFileBig))
        {
            break;
        }
        sw.Stop();
        System.IO.File.Delete(destFileBig);

        long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
        file.WriteLine(microseconds);
    }
}

```

```

        Thread.Sleep(1000);
    }
}

public static void tripleDES_encrypt_decrypt()
{
    string bigfileName = "big_file.doc";
    string smallfileName = "small_file.doc";
    string sourcePath = @"C:\Temp";
    string targetPath = @"\\RIZVI-PC\experiment";

    // Use Path class to manipulate file and directory paths.
    string sourceFileBig = System.IO.Path.Combine(sourcePath, bigfileName);
    string sourceFileSmall = System.IO.Path.Combine(sourcePath, smallfileName);
    string destFileBig = System.IO.Path.Combine(targetPath, bigfileName);
    string destFileSmall = System.IO.Path.Combine(targetPath, smallfileName);
    //Must be 64 bits, 8 bytes.
    string sSecretKey = null;

    // Get the key for the file to encrypt.
    // You can distribute this key to the user who will decrypt the file.
    sSecretKey = GenerateKey();

    // For additional security, pin the key.
    GCHandle gch = GCHandle.Alloc(sSecretKey, GCHandleType.Pinned);

    using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + "threeDES_encrypt_decrypt" + "_small_" +
DateTime.Now.ToString("yyyyMMddhhmmss") + @".txt", true))
    {
        for (int i = 0; i < 100; i++)
        {
            Stopwatch sw = new Stopwatch();
            sw.Start();
            // Encrypt the file.
            ThreeDESEncrypt(sourceFileSmall, destFileSmall, true);

            while (System.IO.File.Exists(destFileSmall))
            {
                break;
            }
            sw.Stop();

            long microseconds = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));

            sw.Start();
            ThreeDESDecrypt(destFileSmall, sourceFileSmall, true);
            while (System.IO.File.Exists(destFileSmall))

```



```

        {
            break;
        }
        sw.Stop();
        //System.IO.File.Delete(destFileSmall);
        long microsecondsDecr = sw.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
        file.WriteLine(microseconds + " " + microsecondsDecr);
        Thread.Sleep(1000);
    }
}
using (System.IO.StreamWriter file = new System.IO.StreamWriter(
@"C:\Temp\" + "threeDES_encrypt_decrypt" + "_big_" +
DateTime.Now.ToString("yyyyMMddhhmmss") + @".txt", true))
{
    for (int i = 0; i < 100; i++)
    {
        Stopwatch sw1 = new Stopwatch();
        sw1.Start();
        // Encrypt the file.
        ThreeDESEncrypt(sourceFileSmall, destFileSmall, true);

        while (System.IO.File.Exists(destFileBig))
        {
            break;
        }
        sw1.Stop();

        long microseconds = sw1.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));

        sw1.Start();
        ThreeDESDecrypt(destFileSmall, sourceFileSmall, true);
        while (System.IO.File.Exists(destFileBig))
        {
            break;
        }
        sw1.Stop();
        //System.IO.File.Delete(destFileBig);
        long microsecondsDecr = sw1.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
        file.WriteLine(microseconds + " " + microsecondsDecr);
        Thread.Sleep(1000);
    }
}
}

public static void EncryptFile(string inputFile, string outputFile)
{

```

```

try
{
    string password = @"myKey123"; // Your Key Here
    UnicodeEncoding UE = new UnicodeEncoding();
    byte[] key = UE.GetBytes(password);

    string cryptFile = outputFile;
    FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

    RijndaelManaged RMCrypto = new RijndaelManaged();

    CryptoStream cs = new CryptoStream(fsCrypt,
        RMCrypto.CreateEncryptor(key, key),
        CryptoStreamMode.Write);

    FileStream fsIn = new FileStream(inputFile, FileMode.Open);

    int data;
    while ((data = fsIn.ReadByte()) != -1)
        cs.WriteByte((byte)data);

    fsIn.Close();
    cs.Close();
    fsCrypt.Close();
}
catch
{
    Console.WriteLine("Encryption failed!", "Error");
}
}
///<summary>
/// Steve Lydford - 12/05/2008.
///
/// Decrypts a file using Rijndael algorithm.
///</summary>
///<param name="inputFile"></param>
///<param name="outputFile"></param>
public static void DecryptFile(string inputFile, string outputFile)
{
    {
        string password = @"myKey123"; // Your Key Here

        UnicodeEncoding UE = new UnicodeEncoding();
        byte[] key = UE.GetBytes(password);

        FileStream fsCrypt = new FileStream(inputFile, FileMode.Open);

        RijndaelManaged RMCrypto = new RijndaelManaged();
    }
}

```

```

        CryptoStream cs = new CryptoStream(fsCrypt,
            RMCrypto.CreateDecryptor(key, key),
            CryptoStreamMode.Read);

        FileStream fsOut = new FileStream(outputFile, FileMode.Create);

        int data;
        while ((data = cs.ReadByte()) != -1)
            fsOut.WriteByte((byte)data);

        fsOut.Close();
        cs.Close();
        fsCrypt.Close();
    }
}

public static void RSAEncrypt(string sInputFilename, string sOutputFilename, string
sKey)
{
    RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
    byte[] inputBytes = System.IO.File.ReadAllBytes(sInputFilename);

    byte[] encrBytes = RSA.Encrypt(inputBytes,true);
    File.WriteAllBytes(sOutputFilename,encrBytes);
}

public static void RSADecrypt(string sInputFilename, string sOutputFilename, string
sKey)
{
    RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
    byte[] inputBytes = System.IO.File.ReadAllBytes(sInputFilename);
    byte[] decrBytes = RSA.Decrypt(inputBytes, true);
    File.WriteAllBytes(sOutputFilename,decrBytes);
}
// Function to generate a 64-bit key.
public static string GenerateKey()
{
    // Create an instance of a symmetric algorithm. The key and the IV are generated
automatically.
    DESCryptoServiceProvider desCrypto = DESCryptoServiceProvider.Create() as
DESCryptoServiceProvider;

    // Use the automatically generated key for encryption.
    return ASCIIEncoding.ASCII.GetString(desCrypto.Key);
}

public static void ThreeDESEncrypt(string sInputFilename, string sOutputFilename, bool
useHashing)

```

```

{
    byte[] keyArray;
    byte[] toEncryptArray = System.IO.File.ReadAllBytes(sInputFilename);

    System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();
    // Get the key from config file

    string key = "sabrina ali tandra";
    //System.Windows.Forms.MessageBox.Show(key);
    //If hashing use get hashcode regards to your key
    if (useHashing)
    {
        MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
        keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
        //Always release the resources and flush data
        //of the Cryptographic service provide. Best Practice

        hashmd5.Clear();
    }
    else
        keyArray = UTF8Encoding.UTF8.GetBytes(key);

    TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
    //set the secret key for the tripleDES algorithm
    tdes.Key = keyArray;
    //mode of operation. there are other 4 modes. We choose ECB(Electronic code Book)
    tdes.Mode = CipherMode.ECB;
    //padding mode(if any extra byte added)
    tdes.Padding = PaddingMode.PKCS7;

    ICryptoTransform cTransform = tdes.CreateEncryptor();
    //transform the specified region of bytes array to resultArray
    byte[] resultArray = cTransform.TransformFinalBlock
        (toEncryptArray, 0, toEncryptArray.Length);
    //Release resources held by TripleDes Encryptor
    tdes.Clear();
    File.WriteAllBytes(sOutputFilename, resultArray);
}
public static void ThreeDESDecrypt(string sInputFilename, string sOutputFilename,
bool useHashing)
{
    byte[] keyArray;
    //get the byte code of the string

    byte[] toEncryptArray = System.IO.File.ReadAllBytes(sInputFilename);

    System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();
    //Get your key from config file to open the lock!
    string key = "sabrina ali tandra";

```

```

if (useHashing)
{
    //if hashing was used get the hash code with regards to your key
    MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
    keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
    //release any resource held by the MD5CryptoServiceProvider

    hashmd5.Clear();
}
else
{
    //if hashing was not implemented get the byte code of the key
    keyArray = UTF8Encoding.UTF8.GetBytes(key);
}

TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
//set the secret key for the tripleDES algorithm
tdes.Key = keyArray;
//mode of operation. there are other 4 modes.
//We choose ECB(Electronic code Book)

tdes.Mode = CipherMode.ECB;
//padding mode(if any extra byte added)
tdes.Padding = PaddingMode.PKCS7;

ICryptoTransform cTransform = tdes.CreateDecryptor();
byte[] resultArray = cTransform.TransformFinalBlock
    (toEncryptArray, 0, toEncryptArray.Length);
//Release resources held by TripleDes Encryptor
tdes.Clear();
File.WriteAllBytes(sOutputFilename, resultArray);
}
}
}

```

Appendix O. AES encryption-decryption time in the same machine

The following table contains time required to encrypt file and store it in the same machine as well as to read that file and decrypt it using AES cryptographic protocol. Time in the table below is in microseconds (μs).

29KB sized file		4MB sized file	
Encryption (μs)	Decryption (μs)	Encryption (μs)	Decryption (μs)
1462724	92700	8354182	754689
765674	7486	10322132	768015
133422	10442	7315201	717588
138319	15847	8767458	796359
133042	20372	9244208	801317
158607	8358	8715674	743078
123633	40316	10494561	763999
140224	10893	10400836	753809
149282	20001	5984092	752841
56534	19903	6793399	729157
151349	9818	6698820	771885
309927	34744	8148126	740608
56632	7686	9742898	785465
75292	7811	9971424	748344
124351	33689	9311425	771217
65638	9891	8568230	763948
85950	8247	9207283	756567
63583	17597	9461273	774078
609116	8153	11483141	716814
59482	8687	7148071	734637
52287	8364	8868827	728639
94683	10317	8569560	1013893
94566	8123	11906146	794713
106400	21473	8684342	726142
78908	20233	13436621	749176
765305	20716	10995309	750342
111644	7532	7997987	1008354
95232	8411	9893744	754139
94791	8547	9911379	770119
108480	13376	6390885	749110
100374	14688	6058626	723742
107351	7701	5725993	746245
64519	11417	5751751	1017240
53972	25077	7349389	771784
87083	25766	6701277	763084
49497	11214	6644249	745041
73836	22005	6711508	751608
63976	22828	6611468	737011
43213	11203	6511748	1040078
754228	8054	6457006	757560
123369	12050	6938072	749179
107363	11451	6181663	783830
70264	16585	6343417	721436
93050	10573	6038125	721173

91264	10715	5627264	1007771
95790	11298	5953908	761315
104905	7492	6033246	754482
80477	15175	5905844	742443
83849	11774	5377937	739498
92205	9075	5097727	756822
83489	24992	5863451	1058466
75906	14570	5854124	794519
111712	8862	5690237	769110
123186	8393	6152615	745561
107857	28969	6664390	782113
97856	7707	6227315	752339
111054	30368	5770664	1033526
129895	8061	5956709	751462
117286	8624	5690471	774784
80036	10827	6044818	785910
99647	21822	5283261	755121
57911	7730	5239049	745290
69884	25453	5435508	759516
46661	7615	5902145	751064
53460	11954	5794090	803751
423582	8880	6063655	746527
93415	10823	5854799	772273
69639	8995	5446528	759736
60701	24147	5685093	809104
428852	10690	5753069	735123
53652	21730	5421171	754311
49696	21039	5349482	787379
55931	18333	5377856	738785
96801	19535	6691515	733019
88075	11296	6232514	758182
140665	22099	6313139	757672
102519	22549	6370132	782180
65660	11014	6130039	781062
452650	27142	5653980	754092
85599	20703	5724477	756460
62575	10858	5664470	784737
57028	15680	6310425	747307
103654	20866	5791080	737035
87146	10482	6111928	759188
101642	20917	5709742	793240
48902	12178	6602330	728736
87535	7822	6533724	746959
237263	8017	5910315	739121
91582	26958	6601759	793159
86982	24954	5614790	777523
59844	21578	5762670	758779
106582	11000	5761184	796688
67170	8228	6709901	747725
59093	21553	5480410	750139
77037	7422	5684761	760987
45182	20783	6004414	763703
56190	20908	5713442	753802
43336	7416	5553742	763773
71263	7396	6067038	783975

57500	19129	7143748	756553
-------	-------	---------	--------

Appendix P. AES encryption-description time between networked hosts

The following table contains time required to encrypt file and store it in a remote host as well as to read that file from remote host and decrypt it using AES cryptographic protocol. Time in the table below is in microseconds.

29KB sized file		4MB sized file	
Encryption (μs)	Decryption (μs)	Encryption (μs)	Decryption (μs)
61679	92654	3765225	9252946
47758	47561	3838633	6337063
63110	53124	3981792	5323375
38408	67331	4521242	4780659
36051	59202	6352921	4327162
70999	55553	4015812	4912764
42062	48497	3843236	5028714
40142	45771	3287647	8359716
34211	45042	6844786	5052242
42378	61541	7665980	6822391
92260	192273	4326138	9043941
48935	42218	3739339	8205595
53119	59405	4000245	5408254
39611	193898	4549605	5835289
35320	54082	4127562	5670417
142724	83028	4111058	4328180
38020	103747	8774485	3830122
82502	46671	8570590	4349260
39020	547963	9298062	4334925
65949	154422	10214988	6583317
99040	48174	12617174	5034627
49778	136315	3912204	5385797
45269	144877	3379613	4894090
45696	36815	4014194	8152574
36715	253035	3652538	4417860
414952	56139	3583022	4232766
43250	63134	3444676	4344362
53021	46018	3677761	4620313
40451	50461	3891486	3721355
42562	42627	4087926	4460701
88772	41031	5829298	3632867
38794	122732	5339418	3821908
43912	42573	5432029	3900471
39881	88220	4280735	4233975
61289	41908	3305008	4388977
41369	40836	3863563	4908307
37043	55674	3614967	5262714
61964	52337	4086004	4973022
43714	46476	3514964	5308535
38086	87335	3065282	4195356
43604	396878	3168261	3830117
47720	54171	3250118	4801943
50370	63761	3496038	4769786
43617	66275	3725971	4590719

828419	47519	3417113	7116464
38960	59109	3334440	4075602
38284	299372	3417579	4760704
37174	66714	3305106	4183517
54508	47782	3338738	3842193
61582	68034	3385202	4794402
35433	48348	3437887	3893469
37502	48463	3242420	4279925
43489	77382	3252890	4700516
59513	50751	3398753	4646729
35401	47571	3422227	4091481
43348	50219	5039247	3519370
42217	53998	3432700	3566156
44925	50547	4709088	3833637
68374	53305	4834222	3587355
36248	48716	3843782	3234760
59567	57078	3650054	3193943
45591	69295	3319097	3667509
44310	50419	3454158	3304801
162918	54823	3350577	3287189
37748	47967	3413599	3380187
36615	1296732	3311907	3870531
40203	317873	3240620	3639048
43353	74999	3277652	3827708
38283	51745	3328480	3483988
40723	74665	3216052	3384074
44255	635302	3250528	3348038
66420	65359	3452739	3215443
40157	57254	3825976	3109156
39353	56631	3569463	3199420
55132	56551	3394973	3711963
44351	43985	4190554	3472490
40894	53906	3863572	3439735
38831	55499	3694443	3296424
45426	61800	5097150	3345776
40411	64221	4047337	3316590
38444	1247946	4159483	3314358
52404	199872	4144623	3311860
45165	64362	3726186	3238560
46089	200713	3982848	3409929
51129	758323	4459027	3758216
41414	130346	3814156	3247163
39880	517667	4635833	3373638
38732	123640	4035561	3329547
61426	173897	4483650	3407878
37383	138103	4351296	3219106
43208	143103	5715330	3530067
51999	174837	7479034	3461609
43576	118732	4215618	3458955
80254	277993	4781677	3465196
57239	156119	4057627	3176057
56950	153578	3794852	3429351
48068	510750	5114914	3600746
51150	121845	3332978	3741512
47224	418542	3317899	3707900

41850	158744	3432837	4455144
-------	--------	---------	---------

Appendix Q. TripleDES encryption-decryption time between networked hosts

The following table contains time required to encrypt file and store it in a remote host as well as to read that file from remote host and decrypt it using TripleDES. Time in the table below is in microseconds.

29KB sized file		4MB sized file	
Encryption (μs)	Decryption (μs)	Encryption (μs)	Decryption(μs)
70333	192969	3861896	8480260
64219	144536	5959316	16729711
52736	184336	5732541	16734524
63461	513226	6112895	12390029
293766	451700	5890296	16208446
72449	176276	5960466	16182323
97172	193760	6149777	18581673
51862	133322	6089744	17842379
72629	152886	6385312	16918848
47189	134904	9048187	15564266
52328	148148	6204458	16708357
63179	146847	6074299	12377160
68929	157667	6050436	12400045
62020	143867	5883715	16609882
46080	147018	5955180	12256169
69101	167344	6099507	16910061
40668	126242	6334260	18241947
87586	192129	7692492	21131017
53460	154487	6083685	17936403
39541	152434	5887615	11946385
73116	152594	5976445	12675801
35216	483772	6027389	12334256
265809	369566	6315009	18476119
39810	140788	5873525	20229790
105434	204232	5650929	19181833
62869	125978	5119167	17991365
43807	139803	5619249	11746853
41688	131692	6485053	20459054
79719	170396	6179555	12531945
67315	153620	5940943	12501685
108566	190799	4875003	12408070
68981	145131	5188493	17299811
64197	139631	4825625	14814131
68977	180494	4695675	10513875
81299	174940	5244046	10843056
59241	172605	4861003	10326151
58752	137976	4816807	16644473

54764	189669	4939389	14592029
90328	182626	4432113	10101036
54830	158332	4869947	14534626
68403	152252	4791499	10588599
74116	152339	4936562	10318782
79199	177764	4869682	15320314
45413	133718	5848425	14669517
92861	186102	4809118	15809110
46351	136642	5084075	15427494
115881	243056	5072715	14858951
48774	128397	4450173	14561958
62716	127481	4948815	14758597
59753	157343	5192521	10856596
59425	144619	4930486	12821470
41285	171549	6081569	13117045
70373	179099	6146570	20773590
45169	154649	5750038	12218175
56865	141593	5281283	11777933
86337	161200	5280351	16108211
47694	140706	5449938	17154451
46986	129610	5392714	16397289
73498	178675	5112656	14192411
55308	145851	5073074	16267846
53623	140448	5390609	15988910
57547	154741	5293412	11531253
67345	189862	5639237	16180198
67335	165951	6264837	17002809
47554	132545	5550074	11956530
60880	187420	5502901	16092683
62647	167825	5203798	11588249
63903	177489	5814559	16104561
63650	166148	5628522	16041407
70944	141148	5664870	12117637
66116	176406	5453637	11890673
62678	145317	5642231	12446839
57423	153730	5778769	16976381
65934	158460	5331147	16736482
55857	148131	5303269	16372689
65770	140782	5283578	16258112
91869	171486	5356489	11571958
57965	163863	6263908	18319020
63893	154309	6273620	13844207
44918	142441	6186494	12878655
61395	153789	6151373	18940248
49068	182359	6517170	17665009
102672	150583	6469264	12619370

80083	170793	6100428	12414613
64928	174319	5754428	11938819
52878	144872	5504464	11956167
66747	183295	5668987	13240078
34964	127280	6019641	13753145
74054	163615	6747043	22869987
60856	169682	6103298	12485160
75428	149761	6568653	18371706
46250	172529	6275030	13314084
46539	164663	6145367	17430494
83590	190002	5627378	16152812
52152	183460	5679803	18221091
52369	219769	5709508	17257984
109585	200940	5520618	12691581
61009	173419	6430077	16825436
76399	197183	6408529	13094586
79569	167835	6375499	16912735

Appendix R. Analysis of data in Appendix P and Q

Small file(KB)	Big file(MB)	Small file(Byte)	Big file(Byte)		
24	4	24576	4194304		
Block size(byte)					
AES	16				
3DES	8				
Per block time in Millisecond for AES		Small file(Byte)	Big file(Byte)		
Encryption		35,92067571	19,99992732		
Decryption		48,19059416	22,32662301		
Per block time in Millisecond for 3DES		Small file(Byte)	Big file(Byte)		
Encryption		24,48545436	11,80231817		
Decryption		62,73410088	60,2729195		
Per byte time in Millisecond for AES		Small file(Byte)	Big file(Byte)		
Encryption		2,245042232	1,249995457		
Decryption		3,011912135	1,395413938		
Per byte time in Millisecond for 3DES		Small file(Byte)	Big file(Byte)		
Encryption		3,060681795	1,475289771		
Decryption		7,84176261	3,767057469		

Appendix S. Analysis of data in Appendix J and K

		Small Table(Byte)	Big Table(Byte)
		16384	65536
Block size(byte)			
AES	16		
Per block time in Millisecond for AES		Small Table(Byte)	Big Table(Byte)
Encryption		0,239174805	0,428574707
Decryption		0,158926758	0,040213135
Per byte time in Millisecond for AES		Small file(Byte)	Big file(Byte)
Encryption		0,014948425	0,026785919
Decryption		0,009932922	0,002513321

Appendix T. Data Push and Pull time without security

24 KB File (μ s)	4 MB File (μ s)
49803	4482547
65255	3937065
47722	3852489
55204	4352998
66080	4751501
68841	3446578
57827	3180194
50101	3609307
54843	3293308
46219	4875510
42176	7277629
47299	10153265
53849	7220695
54910	4906060
45629	3693896
55285	3789783
56829	3571232
46411	2869818
52504	2850159
60879	2790749
63855	3086638
48619	2694082
58809	2801652
54506	2744821
52865	2968960
54054	2923529
61292	2821214
80247	2902039
54918	2814408
55136	2971082
51546	2956447
59782	2844639
57454	3160540
49634	2805218
53315	2876336
50657	2940087
55339	2798696
60027	2774400
50834	2931202

50005	2845413
44269	2863441
64707	2822749
41114	2789600
52403	2819767
57124	2789292
45016	2693350
40658	2769854
59647	2582967
49500	2890855
53972	3519018
78184	3850949
54919	2759021
43292	2843998
59857	2782217
52450	2794603
55113	2695109
64380	2768217
55482	2877848
51535	2914009
45983	2883025
102258	3076615
50262	3049290
44332	2831232
44047	2972763
48847	2777171
86407	2789335
59143	3071123
54996	2802890
53205	3049093
99098	2762018
99652	2914002
54051	2718219
48678	2910523
60294	3087852
52355	2646147
177838	3405327
53598	2850370
47084	2847569
49924	2768253
89009	2890836
55848	3065164
65173	2814745
51672	2910367
61503	2670353

52312	2910307
61779	2960014
48220	2703699
50730	2705577
57172	3247103
70862	2771115
49261	2783321
55322	2829447
51378	2731589
43069	2696324
66209	2611655
80145	2798316
104564	3297520
64003	2937992
58401	2970796

