# SECURITY AS A SERVICE

**A PROJECT REPORT**

***Submitted by***

**Gokularam.S**

**Thangaram.K**

**Sakthi Charanya.P**

*to the*

**FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**INFORMATION TECHNOLOGY**

**DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY**

**COLLEGE OF ENGINEERING, GUINDY**

**ANNA UNIVERSITY**

**CHENNAI 600 025**

# ANNA UNIVERSITY
# CHENNAI - 600 025
# BONA FIDE CERTIFICATE

Certified that this project report titled Security as a service is the bona fide work of Gokularam.S, Thangaram.K, Sakthi Charanya.P who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

**PLACE: Guindy**
**DATE: 30/03/2017**

**Dr. K.Vidya**
**Associate Professor**
**PROJECT GUIDE**
**DEPARTMENT OF IST, CEG**
**ANNA UNIVERSITY**
**CHENNAI 600025**

**COUNTERSIGNED**

**Dr. SASWATI MUKHERJEE**
**HEAD OF THE DEPARTMENT**
**DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY**
**COLLEGE OF ENGINEERING, GUINDY**
**ANNA UNIVERSITY**
**CHENNAI 600025**

# ABSTRACT

Cloud computing is becoming increasingly important for provision of services and storage of data in the Internet.the main focus is to provide security as a service for the users as and when needed. Our main contribution is a security architecture that provides a flexible security as a service model that a cloud provider can offer to its users.we will be using Nodejs and open stack for API creation and hosting the service.

# ACKNOWLEDGEMENT

We express our deep gratitude to our guide, **Dr.Vidya.K,Associate Professor** for guiding us through every phase of the project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us. We thank her for being easily approachable and quite thoughtful. Apart from adding her own input, she has encouraged us to think on our own and gave form to our thoughts. We owe her for harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this exent.

We are extremely grateful to **Dr. Saswathi Mukherjee, Head of the Department, Information Science and Technology, Anna University, Chennai 600025**, for extending the facilities of the department towards our project and for her unstinting support.

We express our thanks to the review committee members **Dr.P.Yogesh, Associate Professor** and **Dr. P.Geetha, Assistant Professor** for their valuable suggestions and critical reviews throughout the course of our project.

We also thank our parents, family and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Gokularam S Thangaram K Sakthi Charanya P

# TABLE OF CONTENTS

**ABSTRACT** iii
**LIST OF FIGURES** vi
**LIST OF SYMBOLS AND ABBREVIATIONS** vii

**1 INTRODUCTION 1**
1.1 OBJECTIVE 1
1.2 PROBLEM STATEMENT 1
1.3 ORGANIZATION OF THE REPORT 2

**2 RELATED WORKS 3**

**3 DESIGN 5**
3.1 MAIN ARCHITECTURE 5
3.2 Sequence Flow 5
3.3 Security Module 6
3.4 Log File 7

**4 IMPLEMENTATION 8**
4.1 Open Stack deployment 8
4.2 Service 8
4.3 Two-step authentication 9
4.4 Encryption 9
4.4.1 Key Setup 9
4.4.2 Plaintext Encryption 10
4.4.3 SubBytes 11
4.4.4 ShiftRows 11
4.4.5 MixColumns 12
4.4.6 AddRoundKey 13

**5 RESULTS AND ANALYSIS 14**
5.1 Security Service 14

**6 CONCLUSION 19**

**REFERENCES 20**

# LIST OF FIGURES

4.1 AES encryption flow 11
4.2 Shift rows 12
4.3 Mix columns 12
4.4 Add Round Key 13

5.1 Service Login page 14
5.2 Service request page 15
5.3 Client register page 16
5.4 Register QRcode generation page 17
5.5 QR code verification page 18

# LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| API | Application program interface |
| AES | Advanced Encryption Standard |
| IDS | Intrusion Detection System |
| CSP | Cloud Service Provider |
| IPS | Intrusion Prevention Service |

# CHAPTER 1

# INTRODUCTION

Cloud computing and storage provides users with capabilities to store and process their data in third-party data centers Organizations use the cloud in a variety of different service models and deployment models . Security concerns associated with cloud computing fall into two broad categories: security issues faced by cloud providers and security issues faced by their customers. The responsibility is shared, however. The provider must ensure that their infrastructure is secure and that their clients data and applications are protected, while the user must take measures to fortify their application and use strong passwords and authentication measures.

## 1.1 OBJECTIVE

Whenever an application is developed significant time is spent on thinking and deploying a suitable security for such application. Also deploying and maintaining such security takes time and work of a person. This can be simplified by building a service which provides user specific security, thus enabling one to focus much on content of the application rather than both content and security.

## 1.2 PROBLEM STATEMENT

Security measures taken from application side such as authentication mechanism, data security and so are very important that they cannot compromise on vulnerabilities which can be updated later. They have to be perfectly great from the first time they go live. Such works are often the same with variety of options. It though has some application dependent requirements also. So such security measures can be provided as a service to enable application developers have a secured application and secured database.

## 1.3 ORGANIZATION OF THE REPORT

The report is structured as follows.

**Chapter 2** introduces the existing work in the domain related to our project and their drawbacks.

**Chapter 3** explains the dessign and the interaction between various modules.

**Chapter 4** gives the implentation details and the work done.

**Chapter 5** shows the results obtained for various scenarios.

**Chapter 6** deals with the future work proposed to our project.

# CHAPTER 2

# RELATED WORKS

## Security-as-a-service for Microservices-Based Cloud Applications

By adding a new API primitive FlowTap for the network hypervisor, we build a flexible monitoring and policy enforcement infrastructure for network traffic to secure cloud applications. We demonstrate the effectiveness of our solution by deploying the Bro network monitor using FlowTap. Results show that our solution is flexible enough to support various kinds of monitoring scenarios and policies and it incurs minimal overhead

## Collaborative intrusion detection as a service in cloud computing environment in Progress in Informatics and Computing

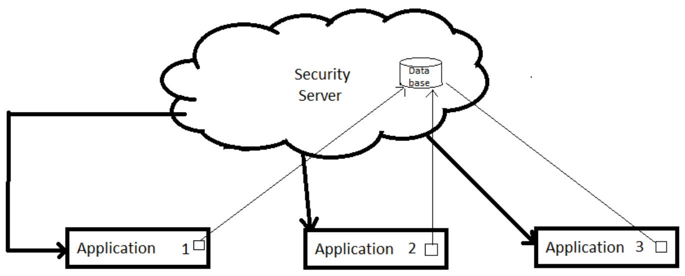
With the growing trends of cloud computing, the security issues in this area are growing at the same speed as its development. Some malicious intruders and other malware activities tend to find the inner vulnerabilities and spare no effort to control the administration or conduct the pure break-down service with curiosity or on purpose. Traditional defense systems such as firewall, intrusion detection and malware code system are still utilized in nowadays network scenes, but they may not support enough in cloud computing environment with old-fashioned architectures. Here we focus on intrusion detection system (IDS) to defend against intruders and other attacks. In this paper, we proposed a collaborative intrusion detection service and our goal is to make use of the state-of-the-art computing framework in cloud environment and to provide a rounded IDS service for both cloud providers and cloud tenants, while the collaborative architecture will help to respond to the attacks promptly. We set up our system prototype and discuss the empirical results on the preference. The experimental results demonstrate that our system does enhance the security when some network-based attacks happen and ensure that both cloud service providers and tenants are protected with satisfaction.

**Security modelling and analysis of a Self Cleansing Intrusion Tolerance technique in Information Assurance and Security**
Since security is increasingly the principal concern in the conception and implementation of software systems, it is very important that the security mechanisms are designed so as to protect the computer systems against cyber attacks. An Intrusion Tolerance Systems play a crucial role in maintaining the service continuity and enhancing the security compared with the traditional security. In this paper, we propose to combine a preventive maintenance with existing intrusion tolerance system to improve the system security. We use a semi-Markov process to model the system behavior. We quantitatively analyze the system security using the measures such as system availability, Mean Time To Security Failure and cost. The numerical analysis is presented to show the feasibility of the proposed approach.

# CHAPTER 3

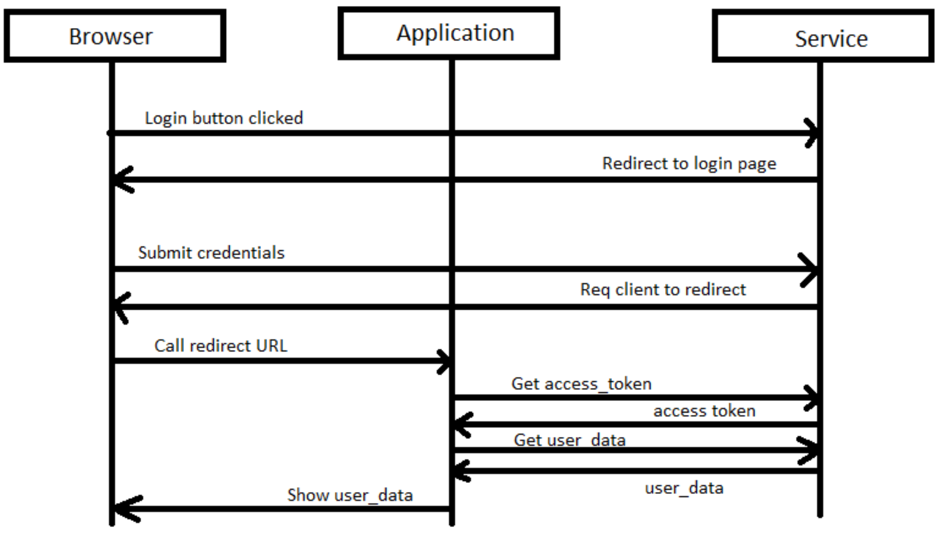
# DESIGN

## 3.1 MAIN ARCHITECTURE



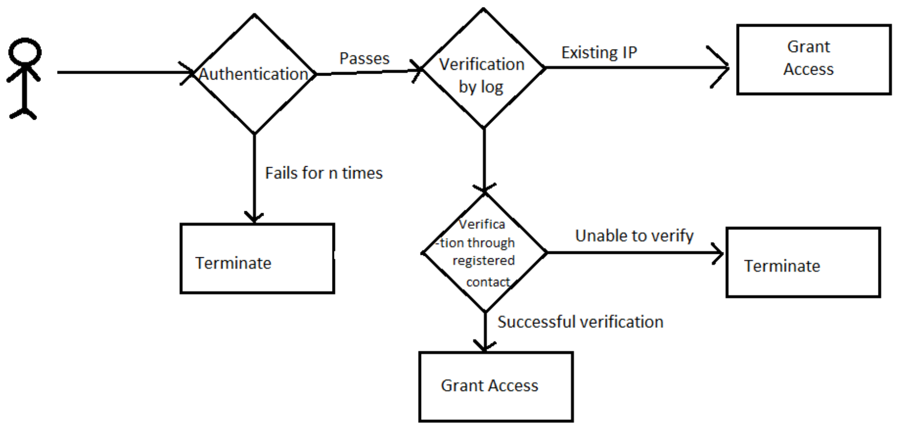
The overall system architecture depicts how the user gets the security as a service which is deployed in the cloud. The client application first gives the request which is maintained in a log and the log file checks for authentication and grants access to the service. The client application can be any application. Whenever the client wants to subscribe the service, an instance is created as a script and the script redirects to the database which contains the details about the log file. The authentication details are stored in encrypted format in service providers database.

## 3.2 Sequence Flow



The Application developer chooses the best security out of available securities from the service provider. Then developer specifies all the information needed from their users. After this service generates an application specific script which should be embedded in their code to make use of their specified service. Thereafter, whenever a user of the application, tries to access that application will be redirected to the service, where they get authenticated as specified. The data that are collected here are stored in the service database in encrypted format. After successful authentication, the application user will be redirected to the application page and application will be given an access token with which only they can get the user's data.

## 3.3 Security Module



The security model proposed above take the users request object and gets the IP out of it using the simple python code in combination with Flask.This IP address is matched with the already existing IP address and the port number in the log file.Attacks like spoofed IP and authentication breach can be stopped.Other attacks such as DOS is stopped by monitoring the specific request packets such as ICMP,UDP, TCP SYN.A predefined set of policies are defined by the security model by which the above attack can be compromised.

## 3.4 Log File

Applications installed on desktops may use local storage and local databases, as well as sending data Applications commonly write event log data to the file system or a database (SQL or NoSQL). Applications installed to remote storage. LOG ATTRIBUTES : Source address e.g. user's device/machine identifier, user's IP address User identity Type of event Log file verifies the IP addresses. If IP address exists already then it directly gives access to the service and any new IP address is identified then it verified by sending a OTP and the IP address gets registered. Then it will be given access.

# CHAPTER 4

# IMPLEMENTATION

## 4.1 Open Stack deployment

OpenStack is a free and open-source software platform for cloud computing, mostly deployed as an infrastructure-as-a-service.In this we have created a configuration file for personalization.Using its own inbuild services, we have created an Iaas architechture in which we are going to deploy our service. It requires either Ubuntu 16.04 or Centos 7. After installing we have to configure the OS and network settings. Then we have to clone the repository of openStack from which we install the application in OS. SSH keys also need to configured. This phase is the configuration of deployment host. For additional services, the files in conf.d provide examples showing the correct host groups to use. For getting IP address from remote client, we used the following implementation: from flask import request
@app.route(”method specify”)
def method()
$returnjsonify('client_ip' : request.remote_addr)$
scanning the vms that has been deployed in openstack periodically
import nmap
$nm = nmap.portscanner()$
$nm.scan(''provideip_range'')$
$forhostinnm.all_hosts():$
$nmapservers()//checkingallpossiblefilesfor$
$write_file(dict(''open_hosts'',''len(open_hosts)''))$

## 4.2 Service

Service is build as an API which serves the security to the clients according to their demand. Client gets authenticated to get on board. Once they are onboard they can request for security services available. Once they are clear of what exact security service they want, they get a script generated for client, which they have to integrate in their application to make use of it. This script generation is done by specified pre-defined script for each script and is integrated based on the client's choice.

## 4.3 Two-step authentication

Authentication implementation is done with two-step process. First, they provide registered credentials. Then, they will be having time-based one time password which will be stored in their phone through application like google authenticator. It is implemented using speakeasy. Speakeasy implements one-time passcode generators as standardized by the Initiative for Open Authentication (OATH). The HMAC-Based One-Time Password (HOTP) algorithm defined by RFC 4226 and the Time-Based One-time Password (TOTP) algorithm defined in RFC 6238 are supported.This is a three-step process:

1. Generate a secret
2. Show a QR code for the user to scan in
3. Authenticate the token for the first time

## 4.4 Encryption

### 4.4.1 Key Setup

1. Selecting two large primes at random : p, q
2. Typically 512 to 2048 bits
3. Computing their system modulus n=p*q
4. note (n)=(p-1)(q-1)
5. Selecting at random the encryption key e
6. where 1¡e¡(n), gcd(e,(n))=1
7. Meaning: there must be no numbers that divide neatly into e and
8. into (p-1)(q-1), except for 1.
9. Solve following equation to find decryption key d
10. e*d=1 mod (n) and 0dn
11. In other words, d is the unique number less than n that when multiplied by e gives you 1 modulo (n)
12. Publish public encryption key: PU=e,n
13. Keep secret private decryption key: PR=d,n

### 4.4.2 Plaintext Encryption

Below shows the overall algorithm of encryption strategy used. It consists of ten rounds. Each round consits of four processes namely SubBytes, ShiftRows, MixCoumns and AddRoundKey in each round except the last round where only three process SubBytes, ShiftRows and AddRoundKey takes place. In the

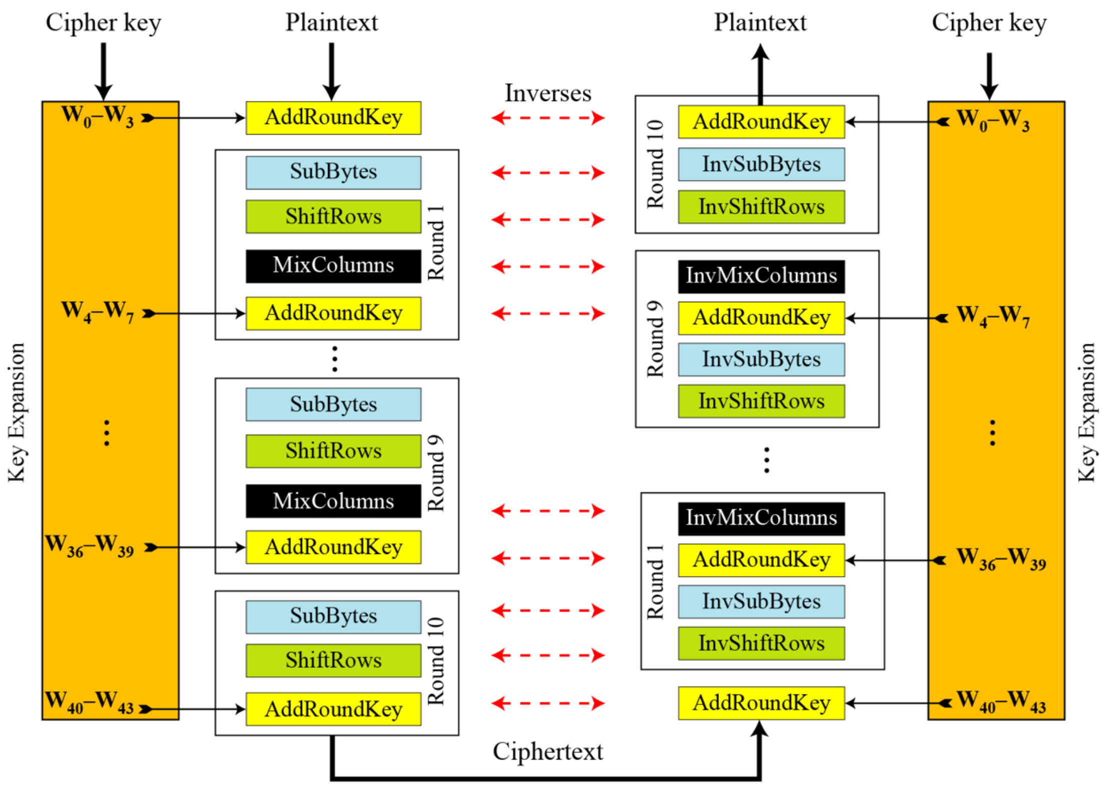


**Figure 4.1: AES encryption flow**

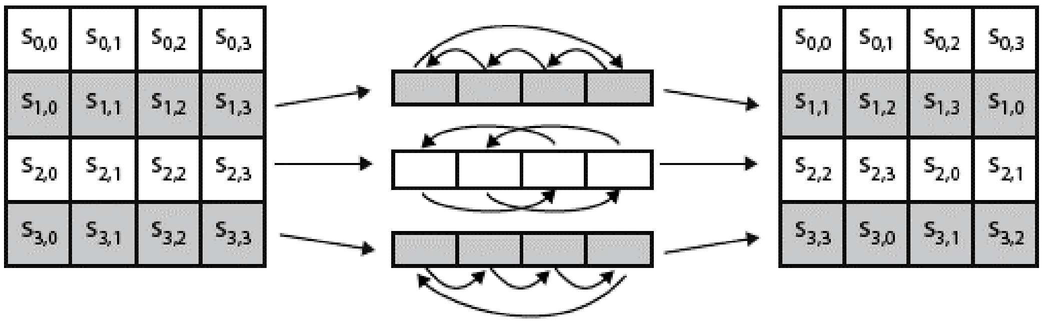
process of decryption all the process happens in reverse order of encryption.

### 4.4.3 SubBytes

A simple substitution of each byte to provide confussion. Each byte of state is replaced by byte indexed by row and column . A table is constructed for substitution of bytes and also inverse table for the same for decryption purpose.

### 4.4.4 ShiftRows

Shifting, which permutes the bytes. A circular byte shift in each row
1st row is unchanged



$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

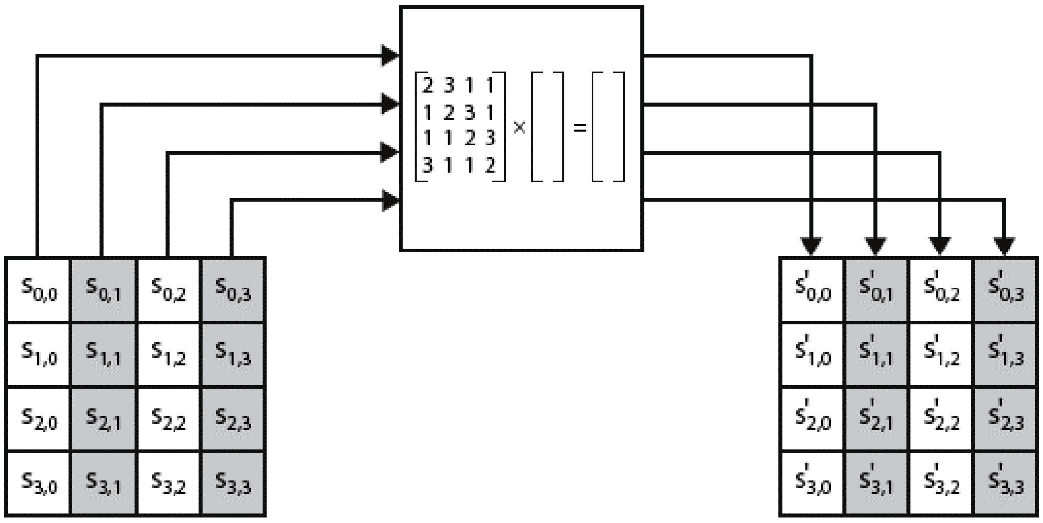
**Figure 4.2: Shift rows**

2nd row does 1 byte circular shift to left

3rd row does 2 byte circular shift to left

4th row does 3 byte circular shift to left

### 4.4.5 MixColumns



2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

**Figure 4.3: Mix columns**

A constant matrix is generated and inverse matrix for the same is also used for decryption. Then the block is used for multiplication with the current matrix.

In inverse mixColumns, the inverse matrix is used for multiplication.

### 4.4.6 AddRoundKey

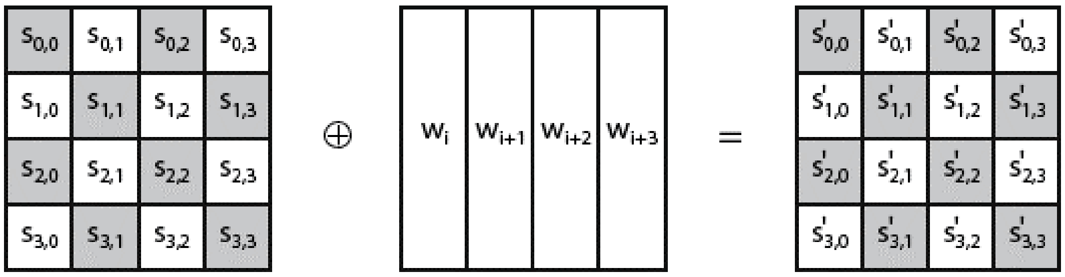

$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$\oplus$
$w_i$ $w_{i+1}$ $w_{i+2}$ $w_{i+3}$
$=$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

**Figure 4.4: Add Round Key**

AddRoundKey proceeds one column at a time.This xors a round key word with each state column matrix the operation is just like matrix addition. For inverse operation the same is xored as xor of a same number twice will produce zero and xor of zero and a number gives the same number.

# CHAPTER 5

# RESULTS AND ANALYSIS

This chapter brings about the screenshots involved in our project. These screenshots give a better understanding on the modules which were implemented in our project. Also the significance of DPDK in processing the packets is discussed.

## 5.1 Security Service

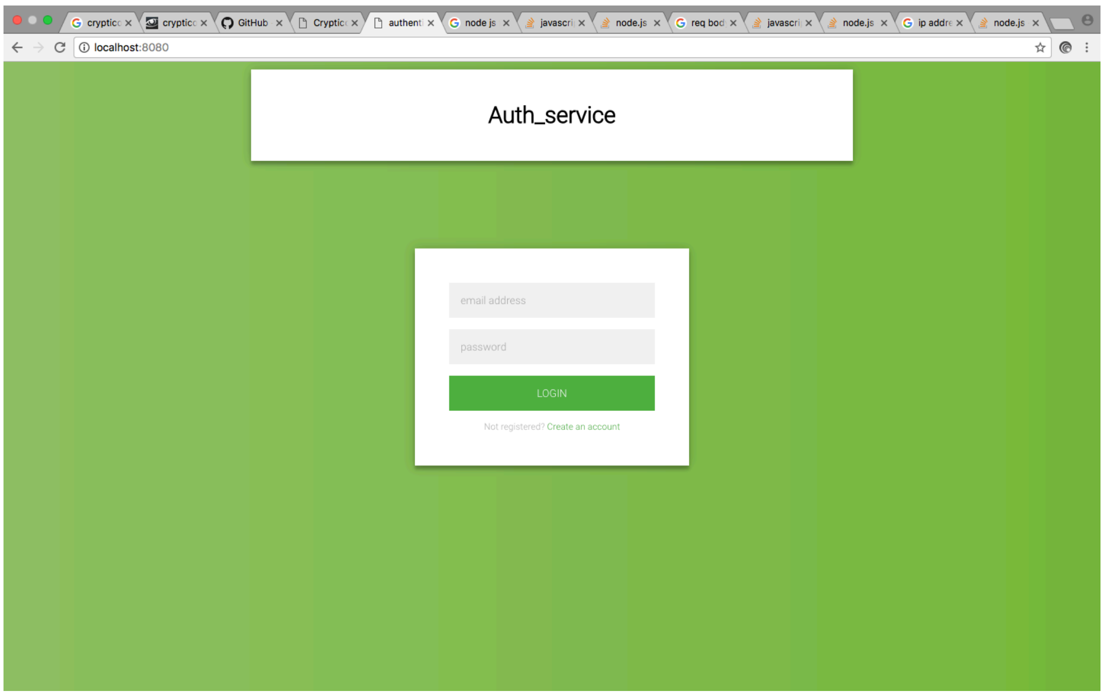
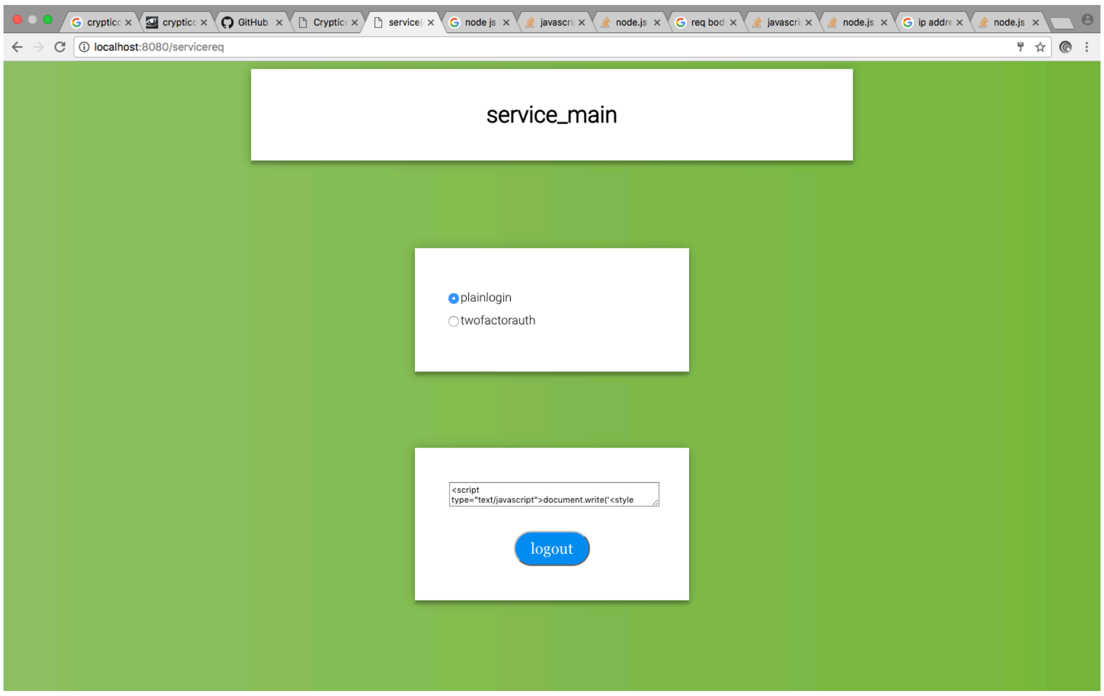


**Figure 5.1: Service Login page**

Figure 5.1 ,shows the UI for the service authentication page.



[✓ plainlogin]
twofactorauth

```
<script
type="text/javascript">document.write('<style
```

**Figure 5.2: Service request page**

Figure 5.2 ,shows the page where client can choose the services they would like to integrate.

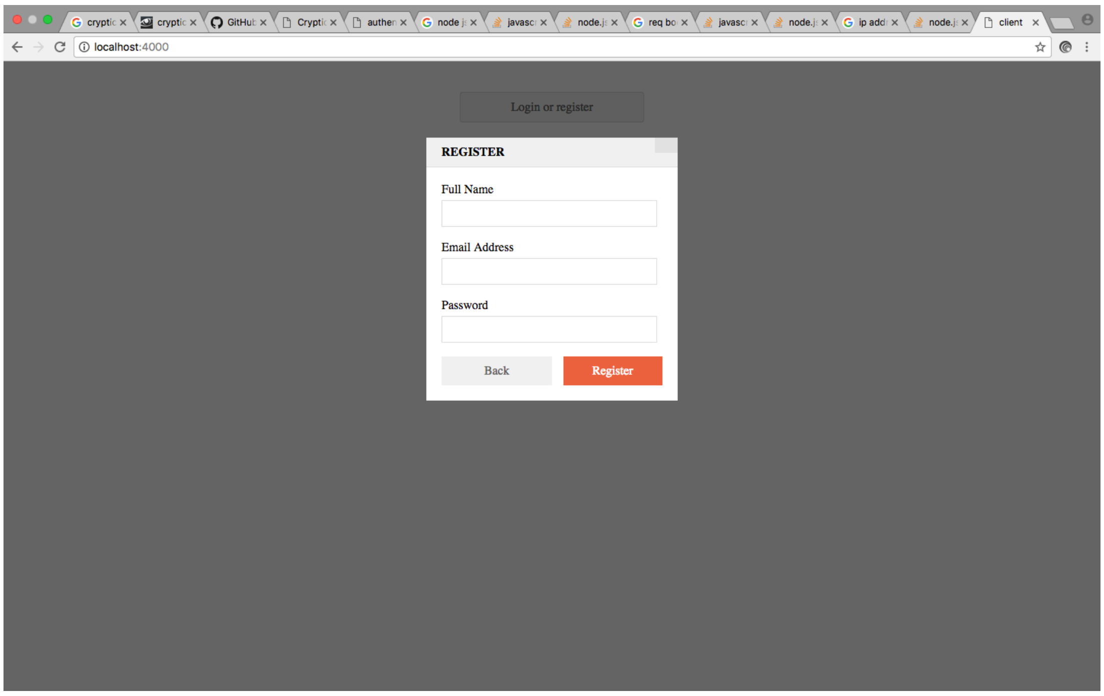


**Figure 5.3: Client register page**

Figure 5.3 ,shows the client page where users register.

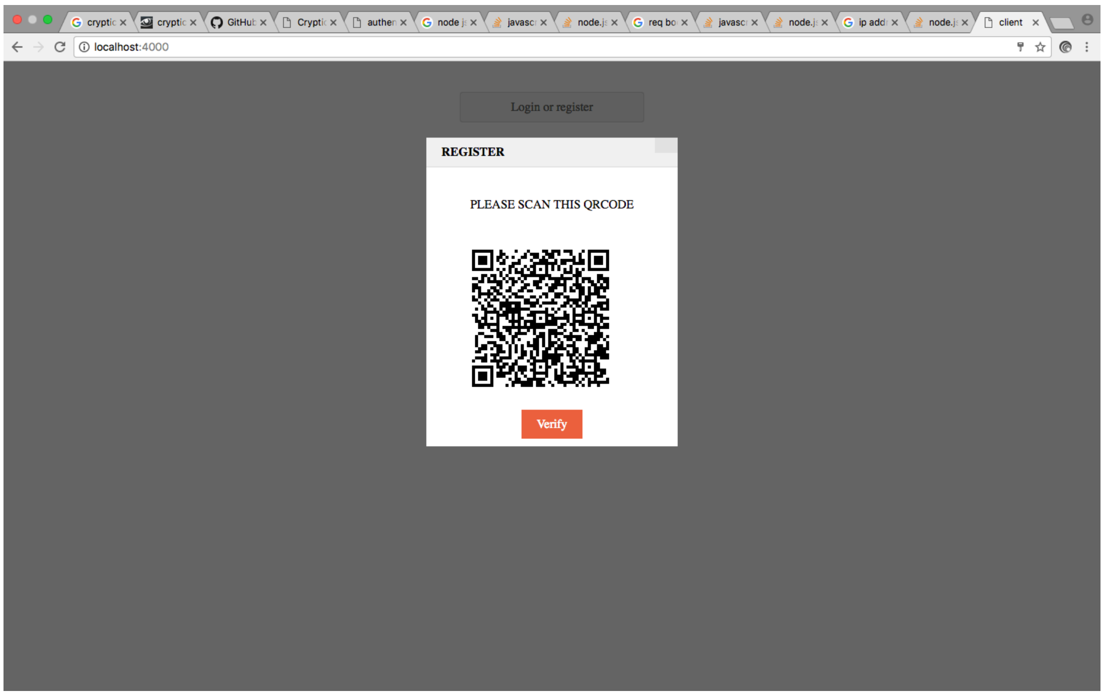


**Figure 5.4: Register QRcode generation page**

Figure 5.4 ,shows the QR code generated for that user which he has to scan with application like Google authenticator.

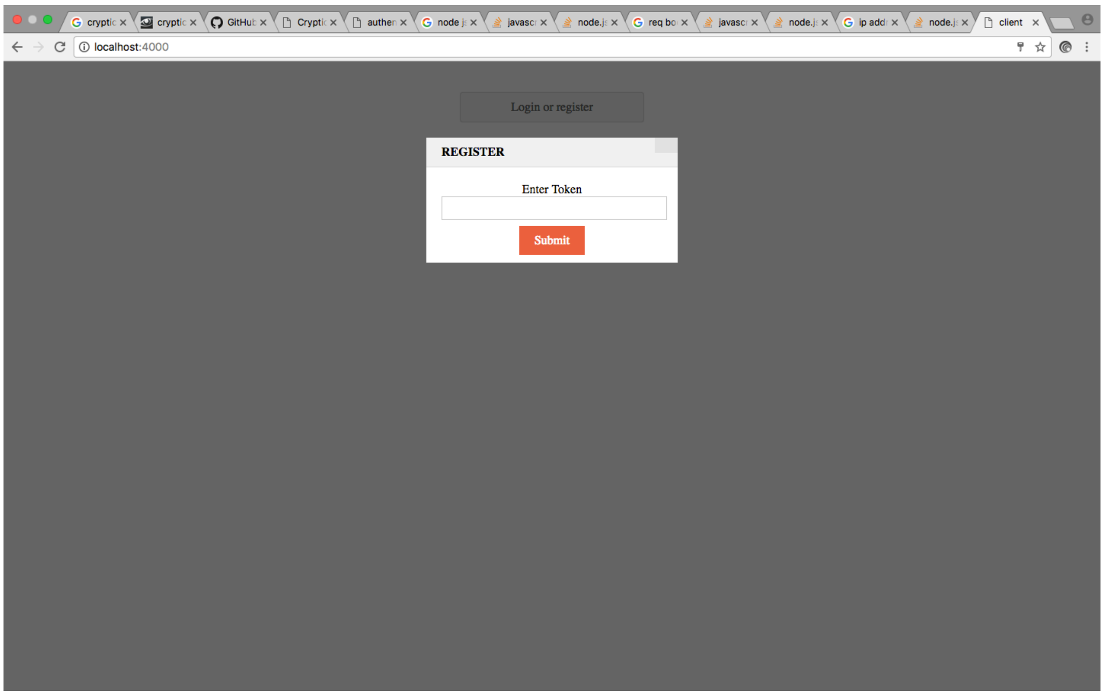


**Figure 5.5: QR code verification page**

Figure 5.5 ,shows the verification of user with Time based OTP which will be available in their app which is scanned at the time of authentication.

# CHAPTER 6

# CONCLUSION

A simple and effective means of service which helps in authentication integration for an application is made. This service helps in choosing an authentication mechanism which the application host wish to implement and has database which the host uses to store details of their users in encrypted format.

## FUTURE WORK

In this work, authentication mechanism and encryption techniques are provided in the service.The suspecious activities of users can be identified by monitoring their activities. A log file has to be maintained and processing has to be done inorder to identify the suspicious activity automatically and suspend that user from onboarding into website.

# REFERENCES

[1] Yugiong Sun, Susanta Nanda and Trent Jaeger, ”Security-as-a-service for Microservices-Based Cloud Applications” , IEEE 7th International Conference on Cloud Computing Technology and Science

[2] Iman El Mir, Dong Seong Kim and Abdelkrim Haqiq, ‘Collaborative intrusion detection as a service in cloud computing environment’, IEEE International Conference on Progress in Informatics and Computing (PIC)

[3] Hong Liang,Yufei Ge,Wengiao Wang and Lin Chen, ‘Security modelling and analysis of a Self Cleansing Intrusion Tolerance technique’, 11th International Conference on Information Assurance and Security (IAS)