Your Project Title

A project report submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

of

Bachelor of Technology

in

Information Technology

by

Your Name(s)

Reg. No. 170911xxx

Under the guidance of

Dr.XYZ (Name of Internal Guide)
Designation of Internal Guide
Department of I & CT

Manipal Institute of Technology
Manipal, India

Dr.ABC (Name of External Guide)
Designation of External Guide
Project Group Name/Department Name (if any)
Company name
Company Address

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

JUNE 2021

I dedicate my thesis to my friends and family.

DECLARATION

I hereby declare that this project work entitled **Title of your project** is original and has been carried out by me in the Department of Information and Communication Technology of Manipal Institute of Technology, Manipal, under the guidance of **your guide name**, **Guide's designation**, Department of Information and Communication Technology, M. I. T., Manipal. No part of this work has been submitted for the award of a degree or diploma either to this University or to any other Universities.

Place: Manipal

Date :DD-MM-YY

Name1

Name2

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this project entitled **Your Project Title** is a bonafide project work done by **Mr. ABC (Reg.No.:160911xxx)** at Manipal Institute of Technology, Manipal, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Information Technology.

Dr.XYZ	Dr.Smitha N Pai
Designation of guide	Professor & Head
Department of I & CT	Department of I & CT
Manipal Institute of Technology	Manipal Institute of Technology
Manipal, India	Manipal, India

Company Certificate

please attach the completion certificate obtained by the company in the letter head signed by the mentor/authorized person from the company. Please ensure that the duration with starting and ending date are specified in the ceritficate along with the title of the project.

NOTE: NOT applicable to the sudents interning at Dept./MIT .

ACKNOWLEDGEMENTS

Students may acknowledge the internal guide, exetrnal guide/mentor, project coordinator, HOD, Director etc.

ABSTRACT

The abstract is brief synopsis of the project work and should be written in 3 paragraphs. The first paragraph should introduce the area of the topic and give importance of the work / topic in the present day scenario, hence leading to the objective of the project work. The second paragraph may discuss briefly about the design/ methodology that was adopted in addition to the tools used. The third paragraph should discuss briefly the important results that were obtained and its significance. You may also discuss about the important conclusion(s) of the project work. (The abstract should fit in one page only)

[**Security and Privacy**]: Cryptography—key management, symmetric cryptography; Security Services—authentication, access control

Contents

List of Tables

List of Figures

ABBREVIATIONS

LDA : Latent Drichlet Allocation

API : Application Programming Interface

NOTATIONS

α : Smoothing factor for words

β : Smoothing factor for topics

Chapter 1

Introduction

1.1 Introduction

The HUMAN population is continually expanding, as is the demand for food. According to UN projections[1,] the human population will reach 9.7 billion in 2050, up from the present estimate of 7.2 billion. Given that the majority of population growth (about 8030 years) occurs in least developed nations, where food shortage is a major issue, reducing food waste in these countries should be a primary goal. Global yield losses are projected to range from 20detection methods necessitate hand inspection of plants by professionals. This method must be continuing and pricey for large farms, or it may be utterly out of reach for many small farmers in rural areas. As a result, in recent decades, many initiatives to automate sickness detection have been made. Hyperspectral images are one of the most effective methods. Hyperspectral images are used to monitor broad areas and are often captured by satellite or aerial imaging equipment. This technique has several drawbacks, including a high cost of equipment, high dimensionality, and a small number of samples, all of which make it unsuitable for machine learning analysis (ML). Because of recent advances in computer vision and the availability of affordable hardware, RGB image analytics are currently the most often utilised technique. Another reason to look at RGB

photos is that, because to the widespread usage of cellphones, these solutions can reach even the most remote locations. Traditional machine learning (ML) methodologies or the deep learning (DL) method can be used to analyse RGB photos. Image pre-processing and attribute extraction are used in traditional methods, which are then fed into one of the machine learning algorithms. Support vector machines (SVM), k-nearest neighbours (k-NN) and full connected neural networks (FCNN), decision trees, and random forests are all popular techniques. For image classification problems, researchers have turned nearly totally to DL techniques in recent years. Classic approaches almost always outperform when a small dataset is provided, and they can be used without the usage of hand-made features. In this study, we compare the DL approach to established ML algorithms for plant disease categorization.

Definition 1 *vvhffff*

1.2 Motivation

Shortcomings in the previous work / referenced paper, Brief importance of the work in the present context, Significance of the possible end result etc.

1.3 Objectives

1.4 Organization of Report

Organization of the project report (chapter wise)

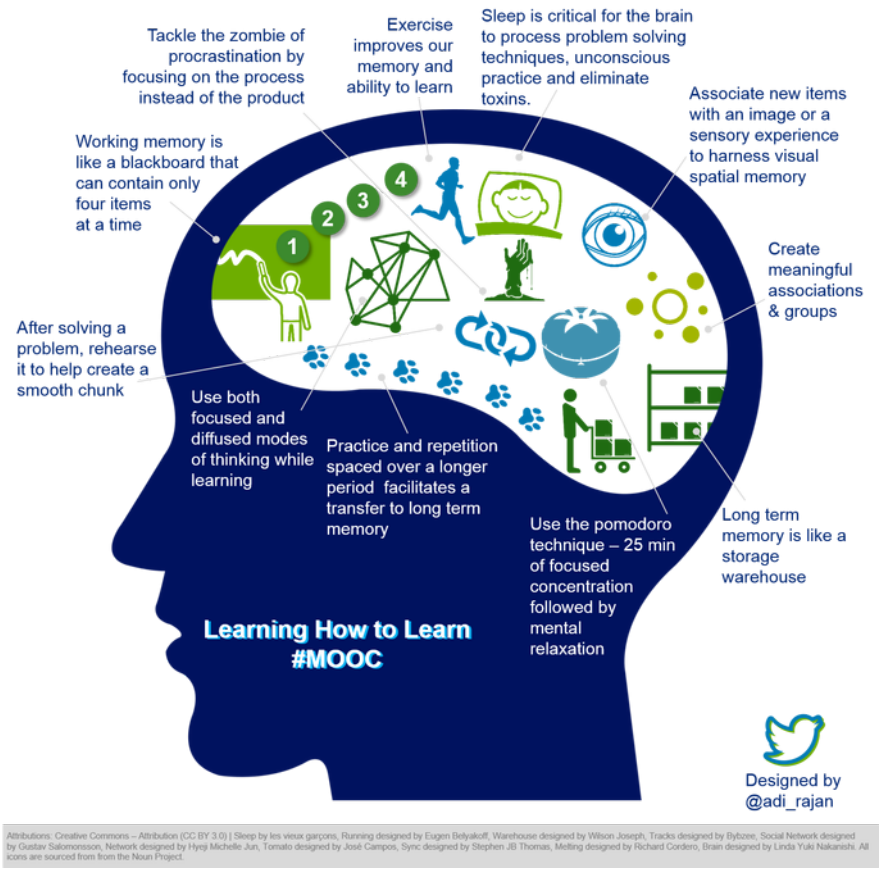

Figure 1.1: Learning how to learn

Chapter 2

Literature Survey

- Fungicide sprays, disease-specific chemical treatments, and vector management, for example, could give early indications of crop health and pesticide disease identification. This could aid in illness prevention and increased production. In this paper, the writers investigate, analyse, and acknowledge the need for a rapid, cost- effective, and reliable agricultural-progress monitoring sensor. They looked at current technologies like as spectroscopic and imaging methods, as well as volaceous profiling approaches to plant disease detection, in order to design ground-based sensors for monitoring plant health and disease in the field.

- A method for diagnosing image processing sickness was established after a thorough analysis of the authors' findings, which included double-stranded ribonucleic acid investigations, nucleic acid testing, and microscopy, among other plant diagnostic instruments.

- At the moment, computer vision is being utilised to identify plant diseases in a variety of ways. One method for identifying illnesses is to employ colour extraction services supplied by writers. In this study, the YcbCr, HSI, and CIELB colour models were successfully employed to recognise noise from a variety of sources, including camera flashes.

- The method for generating characteristics could be utilised to detect plant diseases. In recent tests, Patil and Bodhe used a threshold segmentation approach to estimate the area of sugarcane leaves and its triangular threshold for lesion zones, with an average accuracy of 98.60 percent.

- The texture extraction feature can be used to diagnose plant issues. Patil and Kumar created a method for detecting plant disease that uses textures such as inertia, homogeneity, and correlation to construct the images gray-level cooccurrence pattern.

- In maize leaves, they combined disease detection with colour extraction. All of these factors come together to produce a potent set of traits for improving image and classification quality. Some of the most frequent traditional character extraction methods were discussed by the authors. Due to the rapid rise of artificial intelligence research, the study focuses on the implementation of these approaches and strategies (AI).

- To recognise species of leaves, plagues, or illnesses, different neural network back- feed propagation algorithms use one input, one output, and a hidden layer. The authors recommended this model.

- A pest and disease control software model for agricultural crops has been developed.

- The researchers also use a new method for determining the location of damaged cotton leaves that enhances system accuracy by 95 percent. The forward-looking neural network and the particular swarm optimization (PSO) are used to demonstrate this.

- Vector Machine Support can also detect and differentiate between various plant ailments. This approach is used to diagnose sugar beet diseases,

and it has a success rate of 65 to 90 percent, depending on the kind and stage of the disease.

- Other methods that combine feature extraction and the Network Ensemble can be used to diagnose plant diseases (NNE). NNE generalises learning skills more broadly through training and the integration of a number of neural networks. Because of its 91 percent accuracy, this approach was widely utilised to diagnose tea leaf problems.

In biology, bioinformatics, biomedicine, robotics, and 3D technologies, among other fields of study, the authors have offered in-depth learning approaches to handle some of the most difficult topics. This work applies deep learning techniques for plant disease diagnostics by inventing and executing deep learning methodology. According to a thorough analysis of the most recent literature, the researchers did not study a comprehensive approach of evaluating leaf photos to diagnose plant ailments. The next parts go through our deep CNN recognition technique.

Chapter 3

Design and Methodology

3.1 System Analysis

3.1.1 What is a Waterfall Model?

The Waterfall Model is a phase-by-phase method for partitioning software development. Each step of the SDLC is designed to be used for specialised tasks. Winston Royce was the first to debut it in 1970.

3.1.2 Requirements:

The first stage entails determining what needs to be created, as well as the function and purpose of the structure. The specs of the product, as well as its

Devices	**Requirements**
Processor	Any Updated Processor
RAM	Min 4GB
Hard Disk	Min 100 GB

Table 3.1: Hardware Requirements

Type	System
Operating System	Any Updated Processor
Technology	Python3.6
IDE	PyCharm
Front End	PyQt5
Operating System	Windows

Table 3.2: Software Requirements

inputs and outputs, are checked and reported here.

3.1.3 System design

This step evaluates the requirements from the first phase and creates the system design. The system design aids in the selection of hardware and software, as well as the overall system architecture. The software code for the next level is currently being written.

3.1.4 Implementation

The system is built up of smaller programmes called devices, each with its own set of system design inputs, which are then integrated in the following phase. Unit testing is the process of creating and testing each unit's functionality.

3.1.5 Inclusion and Testing

All of the units developed during the implementation phase are incorporated into the system after each item has been tested. To uncover any problems or challenges, the application must be tested on a regular basis. Tests are conducted to ensure that the consumer has no difficulties installing the programme.

3.1.6 Deployement of the System

The product is deployed or supplied to the customer once functional and non-functional testing is completed.

After the system or component has been installed, changes are needed to improve or boost performance. These changes could be made in response to client requests or as a consequence of issues that arise while the system is in use. Ongoing software support and maintenance are offered to the customer.

3.2 Rights of Functioning

- Login as an Adminstrator
- Add a photo to your profile
- Detecting Plant Disease
- Precognition
- Training and Education
- Image Analysis
- Results of the Image

3.3 Non-Functional Requirements

3.3.1 What is Non Functional Requirement?

NON-FUNCTIONAL REQUIREMENT (NFR) is a quality indicator for software systems. They evaluate the software system's responsiveness, usability, security, portability, and other features, all of which are crucial to its success. A non-functional request would be, "How fast does the site charge?" Systems that do not meet user requirements may arise if non-functional criteria are

not used. Non-functional criteria can be used to constrain or limit the design of a system across multiple agile backlogs. If the number of visitors to the site exceeds 10,000, the website should load in three seconds. Non-functional needs are just as significant as functional requirements.

3.3.2 Examples of Non Functional Requirements

1. After the first successful login, users must update the previously set login password. In addition, the beginning must never be repeated.

2. Employees demanded that their pay information be updated. The security administrator should be notified about this attempt.

3. For each unsuccessful user attempt to access the data item, an audit trail is kept.

4. A website should be able to track the activity of 20 million people at the same time.

5. Is software portability required? As a result, switching between operating systems isn't a problem.

6. Information confidentiality, technological export limits, intellectual property rights, and other issues should all be looked into.

3.3.3 Benefits of Non Functional Requirements

- They offer a fantastic user experience and straightforward software deployment.

- They assist in the development of a software security policy.

3.3.4 Drawbacks of Non Functional Requirements

- The high-quality software subsystems are unaffected by any functional requirement.
- Extra attention is necessary throughout the software architecture / high-level design phase, which increases costs.
- When they're implemented, they normally don't result in the establishment of a new software subsystem.
- It's difficult to change non-functional items after the architectural phase is finished.

3.4 Implementation

The original data set was increased offline to create this data set. The original dataset is included in this project. This dataset contains roughly 87K pictures of healthy and damaged crop leaves divided into 38 classifications. To preserve the directory's structure, the complete data set is split into 80/20 training and validation ratios. A fresh directory with 33 test images will be released later.

Convolution covnets, or computational neural networks, are neural networks with shared parameters.

Layers are classified as follows:

1. The input layer
2. Convolutional Layer
3. The Layers' Activation
4. The layer of a swimming pool
5. A layer that is completely connected

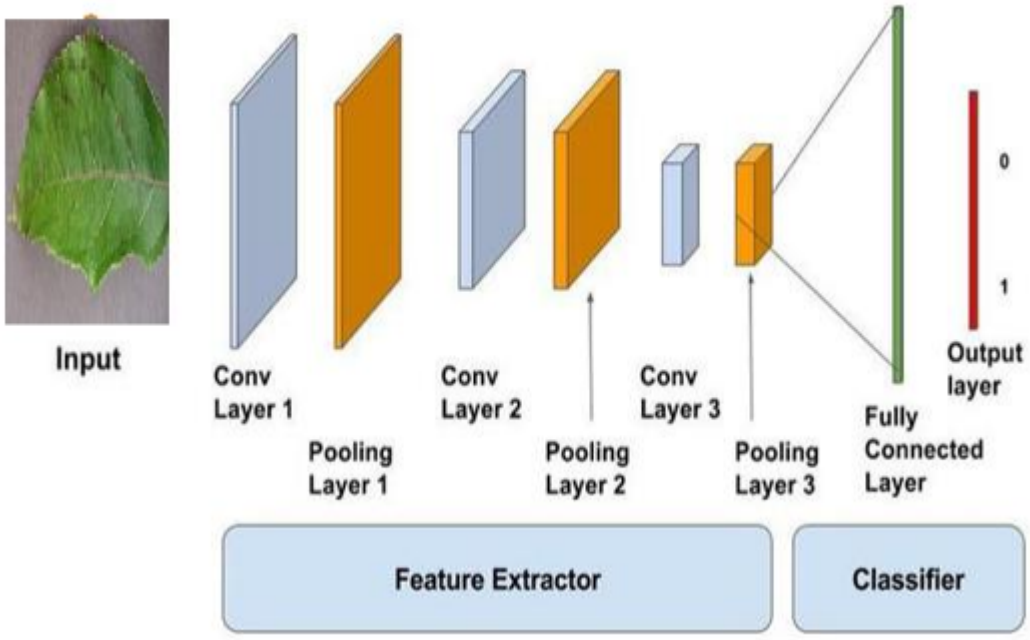

Figure 3.1: CNN Layer

The image's raw input is stored on this layer, which has a width of 32 pixels, a height of 32 pixels, and a depth of three pixels. The output volume is computed by computing the dot product between all filters and the image patch in the convolution layer. This layer's output volume will be 32 × 32 x 12 if we use a total of 12 filters.

This layer will apply an element-by-element activation function to the output of the convolution layer. Frequent activation functions include RELU:max(0,x),Sigmoid: $\frac{1}{1+e^{-x}}$

Chapter 4

Chapter Title

4.1 abvvv

discuss about the results obtained, you may inlcude sample screenshots of the results obtained

Chapter 5

Student Contribution

This chapter should include Individual contribution of each student. Applicable to GROUP PROJECTS ONLY.

5.1 Contribution of Student 1

Minimum of 2 pages

5.2 Contribution of Student 2

Minimum of 2 pages

Chapter 6

Conclusion and Future Scope

6.1 vvvvv

This chapter should include

- Brief summary of the work - Problem statement / objective, in brief, Work methodology adopted, in brief
- Conclusions - General conclusions, Significance of the results obtained
- Future scope of work.

Appendices

Appendix A

Code (if required)

A.1 Kerberos Protocol

```
MODULE main

VAR

--Creating agents which are to type agtype
agA : agtype;
agB : agtype;
agS : agtype;
agI : agtype;
Iactive: boolean;

--Assigning initial to values to all variables

ASSIGN
init(agA.state):=wait;
init(agB.state):=wait;
init(agS.state):=wait;
init(agI.state):=wait;
init(agA.count):=0;
init(agB.count):=0;
init(agS.count):=0;
init(agI.count):=0;
init(agA.authenticated):=FALSE;
init(agB.authenticated):=FALSE;
init(agI.authenticated):=FALSE;
init(agS.authenticated):=TRUE;
```

```
--Transitions for the variable indicating presence or absence of intruder

next(Iactive):=
case
!Iactive:{0,1};
Iactive & agI.state=receive4beta:{0,1};
1:Iactive;
esac;

--Transitions for agent A's state

next(agA.state):=
case
agA.state=wait: send1;
agS.state=send2 & agA.state=send1: receive2;
agA.state=receive2: send3alpha;
agB.state=send4alpha & agA.state=send3alpha: receive4alpha;
agA.state=receive4alpha: wait;
1:agA.state;
esac;

--Transitions for agent B's state

next(agB.state):=
case
agA.state=send3alpha & agB.state=wait: receive3alpha;
agI.state=send3beta & agB.state=wait & Iactive: receive3beta;
agI.state=send3beta & agB.state=send4alpha & Iactive: receive3beta;
agB.state=receive3alpha:send4alpha;
agB.state=receive3beta:send4beta;
agB.state=send4alpha:wait;
1:agB.state;
esac;

--Transitions for Server S's state

next(agS.state):=
case
agA.state=send1 & agS.state=wait: receive1;
agS.state=receive1:send2;
agS.state=send2:wait;
1:agS.state;
esac;
```

```
--Transitions for the Intruder's state

next(agI.state):=
case
agI.state=wait & agA.state=send3alpha & agB.state=wait & Iactive: receive3beta;
agI.state=receive3beta & Iactive: send3beta;
agI.state=send3beta & agB.state=send4beta & Iactive  : receive4beta;
agI.state=receive4beta & Iactive: wait;
1:agI.state;
esac;

--Transitions for Agent A's counter

next(agA.count):=
case
agA.state=send1|agA.state=receive2: agA.count;
agA.state=send3alpha & agA.count<1:agA.count+1;
agA.count=1 & agA.state=receive2: 0;
1:agA.count;
esac;

--Transitions for Agent B's counter

next(agB.count):=
case
agB.state=receive3beta & agB.count<2|agB.state=receive3alpha & agB.count<2: agB.count+1;
agB.state=send4alpha |agB.state=send4beta:agB.count;
agB.count=1 & agA.state=receive4alpha & !Iactive:0;
agB.count=2 & agA.state=send3alpha|agB.count=1 & agA.state=send3alpha: 0;
1:agB.count;
esac;

--Transitions for Agent I's counter

next(agI.count):=
case
agI.state=receive3beta & agI.count<2 & Iactive:agI.count+1;
agI.state=send3beta & Iactive:agI.count;
agI.state=receive4beta & agI.count<2 & Iactive: agI.count+1;
agI.count=2: 0;
1:agI.count;
esac;

--Transitions for variable indicating agent A's authentication
```

```
next(agA.authenticated):=
case
agA.state=receive4alpha :TRUE;
1:agA.authenticated;
esac;

--Transitions for variable indicating agent B's authentication

next(agB.authenticated):=
case
agB.state=send4alpha |agB.state=send4beta :TRUE;
1:agB.authenticated;
esac;

--Transitions for variable indicating agent B's authentication which
--indicates that it has received the fourth message

next(agI.authenticated):=
case
agI.state=receive4beta :TRUE;
1:agI.authenticated;
esac;

--Agent S always is authenticated so transitions to the false state do not occur

next(agS.authenticated):=
case
1:agS.authenticated;
esac;

--Specifications which detect the presence of replay attack

--Agent B should not receive more messages than what agent A has sent it
--SPEC AG!(agA.count < agB.count);

--Agent I should never receive the fourth message
SPEC AG!(agI.state=receive4beta);

--Module for each agent's type which includes the agent's state variable,
--its counter and its authentication variable

MODULE agtype
```

```
VAR
state: {wait, send1, receive1, send2,receive2,
send3alpha, send3beta, receive3alpha, receive3beta,
send4alpha,send4beta, receive4alpha, receive4beta };
count:{0,1,2};
authenticated:boolean;
```

A.2 Kerberos Protocol with Freshness Concept

```
MODULE main

VAR

--Creating agents which are to type agtype
agA : agtype;
agB : agtype;
agS : agtype;
agI : agtype;
Iactive: boolean;
Fresh:0..20;
Time:0..20;

--Assigning initial to values to all variables

ASSIGN
init(agA.state):=wait;
init(agB.state):=wait;
init(agS.state):=wait;
init(agI.state):=wait;
init(agA.count):=0;
init(agB.count):=0;
init(agS.count):=0;
init(agI.count):=0;
init(agA.authenticated):=FALSE;
init(agB.authenticated):=FALSE;
init(agI.authenticated):=FALSE;
init(agS.authenticated):=TRUE;
init(Fresh):=0;
init(Time):=0;
```

```
--Transitions for the variable indicating presence or absence of intruder

next(Iactive):=
case
!Iactive:{0,1};
Iactive & agI.state=receive4beta:{0,1};
1:Iactive;
esac;

--Transitions for agent A's state

next(agA.state):=
case
agA.state=wait: send1;
agS.state=send2 & agA.state=send1: receive2;
agA.state=receive2: send3alpha;
agB.state=send4alpha & agA.state=send3alpha: receive4alpha;
agA.state=receive4alpha: wait;
1:agA.state;
esac;

--Transitions for agent B's state

next(agB.state):=
case
agA.state=send3alpha & agB.state=wait & Fresh=0: receive3alpha;
agI.state=send3beta & agB.state=wait & Iactive & Fresh=0: receive3beta;
agI.state=send3beta & agB.state=send4alpha & Iactive & Fresh=0: receive3beta;
agB.state=receive3alpha:send4alpha;
agB.state=receive3beta:send4beta;
agB.state=send4alpha:wait;
1:agB.state;
esac;

--Transitions for Server S's state

next(agS.state):=
case
agA.state=send1 & agS.state=wait: receive1;
agS.state=receive1:send2;
agS.state=send2:wait;
1:agS.state;
esac;
```

```
--Transitions for the Intruder's state

next(agI.state):=
case
agI.state=wait & agA.state=send3alpha & agB.state=wait & Iactive: receive3beta;
agI.state=receive3beta & Iactive: send3beta;
agI.state=send3beta & agB.state=send4beta & Iactive  : receive4beta;
agI.state=receive4beta & Iactive: wait;
agI.state=send3beta & Time>2: wait;
1:agI.state;
esac;

--Transitions for Agent A's counter

next(agA.count):=
case
agA.state=send1|agA.state=receive2: agA.count;
agA.state=send3alpha & agA.count<1:agA.count+1;
agA.count=1 & agA.state=receive2: 0;
1:agA.count;
esac;

--Transitions for Agent B's counter

next(agB.count):=
case
agB.state=receive3beta & agB.count<2|agB.state=receive3alpha & agB.count<2: agB.count+1;
agB.state=send4alpha|agB.state=send4beta:agB.count;
agB.count=1 & agA.state=receive4alpha & !Iactive:0;
agB.count=2 & agA.state=send3alpha|agB.count=1 & agA.state=send3alpha: 0;
1:agB.count;
esac;

--Transitions for Agent I's counter

next(agI.count):=
case
agI.state=receive3beta & agI.count<2 & Iactive:agI.count+1;
agI.state=send3beta & Iactive:agI.count;
agI.state=receive4beta & agI.count<2 & Iactive: agI.count+1;
agI.count=2: 0;
1:agI.count;
esac;
```

```
--Transitions for variable indicating agent A's authentication

next(agA.authenticated):=
case
agA.state=receive4alpha :TRUE;
1:agA.authenticated;
esac;

--Transitions for variable indicating agent B's authentication

next(agB.authenticated):=
case
agB.state=send4alpha|agB.state=send4beta :TRUE;
1:agB.authenticated;
esac;

--Transitions for variable indicating agent B's authentication which
--indicates that it has received the fourth message

next(agI.authenticated):=
case
agI.state=receive4beta :TRUE;
1:agI.authenticated;
esac;

--Agent S always is authenticated so transitions to the false state do not occur

next(agS.authenticated):=
case
1:agS.authenticated;
esac;

--Transitions for the freshness variable

next(Fresh):=
case
agA.state=send3alpha & agB.state=wait:0;
Fresh<20:Fresh+1;
1:0;
esac;

--Transitions for the Intruder's timer variable

next(Time):=
```

```
case
agI.state=receive3beta:0;
Time<20:Time+1;
1:0;
esac;

--Specifications which detect the presence of replay attack

--Agent B should not receive more messages than what agent A has sent it
SPEC AG!(agA.count < agB.count);

--Agent I should never receive the fourth message
SPEC AG!(agI.state=receive4beta);

--Module for each agent's type which includes the agent's state variable,
--its counter and its authentication variable

MODULE agtype

VAR
state:{wait, send1, receive1, send2,receive2,
send3alpha, send3beta, receive3alpha, receive3beta,
send4alpha, send4beta, receive4alpha, receive4beta};
count:{0,1,2};
authenticated:boolean;
```

Appendix B

Trace Files

B.1 Replay Attack

```
-- specification AG !(agA.count < agB.count)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  agA.state = wait
  agA.count = 0
  agA.authenticated = 0
  agB.state = wait
  agB.count = 0
  agB.authenticated = 0
  agS.state = wait
  agS.count = 0
  agS.authenticated = 1
  agI.state = wait
  agI.count = 0
  agI.authenticated = 0
  Iactive = 0
-> Input: 1.2 <-
-> State: 1.2 <-
  agA.state = send1
  agS.count = 2
-> Input: 1.3 <-
-> State: 1.3 <-
  agS.state = receive1
-> Input: 1.4 <-
-> State: 1.4 <-
```

```
  agS.state = send2
-> Input: 1.5 <-
-> State: 1.5 <-
  agA.state = receive2
  agS.state = wait
-> Input: 1.6 <-
-> State: 1.6 <-
  agA.state = send3alpha
  Iactive = 1
-> Input: 1.7 <-
-> State: 1.7 <-
  agA.count = 1
  agB.state = receive3alpha
  agI.state = receive3beta
-> Input: 1.8 <-
-> State: 1.8 <-
  agB.state = send4alpha
  agB.count = 1
  agI.state = send3beta
  agI.count = 1
-> Input: 1.9 <-
-> State: 1.9 <-
  agA.state = receive4alpha
  agB.state = receive3beta
  agB.authenticated = 1
-> Input: 1.10 <-
-> State: 1.10 <-
  agA.state = wait
  agA.authenticated = 1
  agB.state = send4beta
  agB.count = 2
-- specification AG !(agI.state = receive4beta)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 2.1 <-
  agA.state = wait
  agA.count = 0
  agA.authenticated = 0
  agB.state = wait
  agB.count = 0
  agB.authenticated = 0
  agS.state = wait
  agS.count = 0
```

```
  agS.authenticated = 1
  agI.state = wait
  agI.count = 0
  agI.authenticated = 0
  Iactive = 0
-> Input: 2.2 <-
-> State: 2.2 <-
  agA.state = send1
  agS.count = 2
-> Input: 2.3 <-
-> State: 2.3 <-
  agS.state = receive1
-> Input: 2.4 <-
-> State: 2.4 <-
  agS.state = send2
-> Input: 2.5 <-
-> State: 2.5 <-
  agA.state = receive2
  agS.state = wait
-> Input: 2.6 <-
-> State: 2.6 <-
  agA.state = send3alpha
  Iactive = 1
-> Input: 2.7 <-
-> State: 2.7 <-
  agA.count = 1
  agB.state = receive3alpha
  agI.state = receive3beta
-> Input: 2.8 <-
-> State: 2.8 <-
  agB.state = send4alpha
  agB.count = 1
  agI.state = send3beta
  agI.count = 1
-> Input: 2.9 <-
-> State: 2.9 <-
  agA.state = receive4alpha
  agB.state = receive3beta
  agB.authenticated = 1
-> Input: 2.10 <-
-> State: 2.10 <-
  agA.state = wait
  agA.authenticated = 1
  agB.state = send4beta
```

```
  agB.count = 2
-> Input: 2.11 <-
-> State: 2.11 <-
  agA.state = send1
  agI.state = receive4beta
```

B.2 Replay Attack overcome using Freshness Concept

```
-- specification AG !(agA.count < agB.count)  is true
-- specification AG !(agI.state = receive4beta)  is true
```

References

[1] M. Shell. (2007) IEEEtran webpage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/IEEEtran/

[2] Y. Okada, K. Dejima, and T. Ohishi, "Analysis and comparison of PM synchronous motor and induction motor type magnetic bearings," *IEEE Trans. on EE*, vol. 31, pp. 1047–1053, Sep./Oct. 1995.

[3] R. K. Gupta and S. D. Senturia, "Pull-in time dynamics as a measure of absolute pressure," in *Proc. IEEE International Workshop on Microelectromechanical Systems (MEMS'97)*, Nagoya, Japan, Jan. 1997, pp. 290–294.

[4] B. D. Cullity, *Introduction to Magnetic Materials*. Reading, MA: Addison-Wesley, 1972.

[5] *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[6] R. Jain, K. K. Ramakrishnan, and D. M. Chiu, "Congestion avoidance in computer networks with a connectionless network layer," Digital Equipment Corporation, MA, Tech. Rep. DEC-TR-506, Aug 1987.

[7] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, jan 1999.

[8] Q. Li, "Delay characterization and performance control of wide-area networks," Ph.D. dissertation, Univ. of Delaware, Newark, May 2000. [Online]. Available: http://www.ece.udel.edu/~qli

[9] L. Roberts, "Enhanced proportional rate control algorithm PRCA," ATM Forum Contribution 94-0735R1, Aug. 1994.

Table B.1: Project Detail

Student Details

Student Name	Your Name		
Registration Number	170911xxx	Section/Roll No.	A/01
Email Address	xyz@gmail.com	Phone No.(M)	9891000000
Student Name	Your Name		
Registration Number	170911xxx	Section/Roll No.	A/01
Email Address	yourname@yahoo.com	Phone No.(M)	9891000000

Project Details

Project Title	Title of your project		
Project Duration	4-6 Months	Date of Reporting	dd-mm-2021

Organization Details

Organization Name	Name of your organization
Full Postal Address	Whitefield, B'lore
Website Address	www.abc.com

Supervisor Details

Supervisor Full Name	Name		
Designation	Project Leader or Manager		
Full Contact Address with PIN Code	#1,Whitefield, B'lore		
Email Address	xyz@abc.in	Phone No.(M)	9767541234

Internal Guide Details

Faculty Name	Name
Full Contact Address with PIN Code	Department of Information and Communication Technology, Manipal Institute of Technology, Manipal-576104
Email Address	abc@manipal.edu

INSPECTOR

ORIGINALITY REPORT

8%	7%	7%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	hectordelgado.me Internet Source	2%
2	hal.inria.fr Internet Source	1%
3	www.aclweb.org Internet Source	1%
4	Shankar Bhausabheb Nikam. "Wavelet energy signature and GLCM features-based fingerprint anti-spoofing", 2008 International Conference on Wavelet Analysis and Pattern Recognition, 08/2008 Publication	1%
5	Jinesh Mehta, Eshaan Ramnani, Sanjay Singh. "Face Detection and Tagging Using Deep Learning", 2018 International Conference on Computer, Communication, and Signal Processing (ICCCSP), 2018 Publication	1%
6	Internet Research, Volume 23, Issue 2 (2013-05-27)	1%

SAMSUNG

Samsung Semiconductor India Research (SSIR),
Division of Samsung R&D Institute India – Bangalore Pvt. Ltd.
South Block, Bagmane Goldstone Building,
Outer Ring Road, Mahadevpura,
Bangalore-560048, Karnataka

Date: December 10, 2019

Dear [redacted]

Congratulations! We are pleased to offer you the position of **"Student Trainee"** at Samsung R&D Institute, Bangalore, from **6 January 2020** to **19 June 2020**.

Stipend and Benefits:

1. You will be entitled to a stipend of **INR.** [redacted] per month.
2. You are eligible to use the company transport (to and from the company) and cafeteria free of cost.
3. You will be covered under the **Group Personal Accident Insurance Policy** (as per the company policy).

Please bring a copy of all your testimonials along with the originals for verification on the date of joining.