

**Hardware Security Module (LUNAEFT)**

Project report submitted in partial fulfillment of the requirement for the degree of  
Bachelor of Technology

In

**COMPUTER SCIENCE & ENGINEERING**

SUBMITTED BY

Name: Aditya Jadia

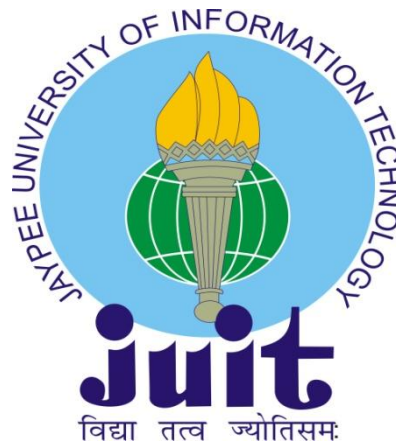
Roll no.: 141328

Under the Supervision of

**Prof. Dr Satya Prakash Gherera**

**Professor, Brig (Retd.) and Head, Dept. of CSE and IT**

TO



Department of Computer Science & Engineering and Information Technology

**Jaypee University of Information Technology Waknaghat,**

**Solan-173234, Himachal Pradesh**

## Candidate's Declaration

I hereby declare that the work presented in this report entitled "Hardware Security Module" in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the **Department of Computer Science & Engineering/Information Technology, Jaypee University of Information Technology, Wagnaghat** is an authentic record of my own work carried out over a period from 5<sup>th</sup> February 2018 to 23<sup>rd</sup> May 2018 under the supervision of **Prof. Dr. Satya Prakash Ghrera(Professor, Brig (Retd.) and Head, Dept. of CSE and IT)**.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Aditya Jadia

141328

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Prof. Dr. Satya Prakash Ghrera

Professor, Brig (Retd.) and Head, Dept. of CSE and IT

Department of Computer Science & Engineering/Information Technology

JUIT, Solan, HP

Dated:

## **CERTIFICATE**

This is to certify that Aditya Jadia, student of B.Tech CSE of Jaypee University of Information Technology has completed his industrial training dated from February 5<sup>th</sup> 2018 to 31<sup>st</sup> May 2018. He is working in Crypto Management Department on the product Hardware Security Module under the guidance of Mr. Suhail Shrivastava.

Signature of Manager

Mr. Suhail Shrivastava

Manager, Crypto Management

## **ACKNOWLEDGEMENT**

I would like to express my immense gratitude to Mr. Suhail Shrivastava (Manager, Crypto Management) for helping me with every aspect of learning and understanding the working of the organization. His guidance helped throughout the internship. I would like to thank Mr. Shivam Garg (Technical Lead), Mr. Vijendra Singh (Solutions & Services) and Mr. Rajat Kumar (Software Engineer) for the guidance they provided during my internship at Gemalto.

I would like to take this opportunity to thank Gemalto, Noida for providing me the golden opportunity to work on their flagship projects. My internship at Gemalto has been a wonderful experience, the internship helped me to explore the new aspects of Digital security and provided me a platform to work with some of the most advanced technologies in this domain.

I would like to express my special thanks for Prof. Dr. Mr. Satya Prakash Ghrera for all the support and guidance. He provided me solutions for all my problems and also was always ready with suggestions that helped me carry out the internship smoothly.

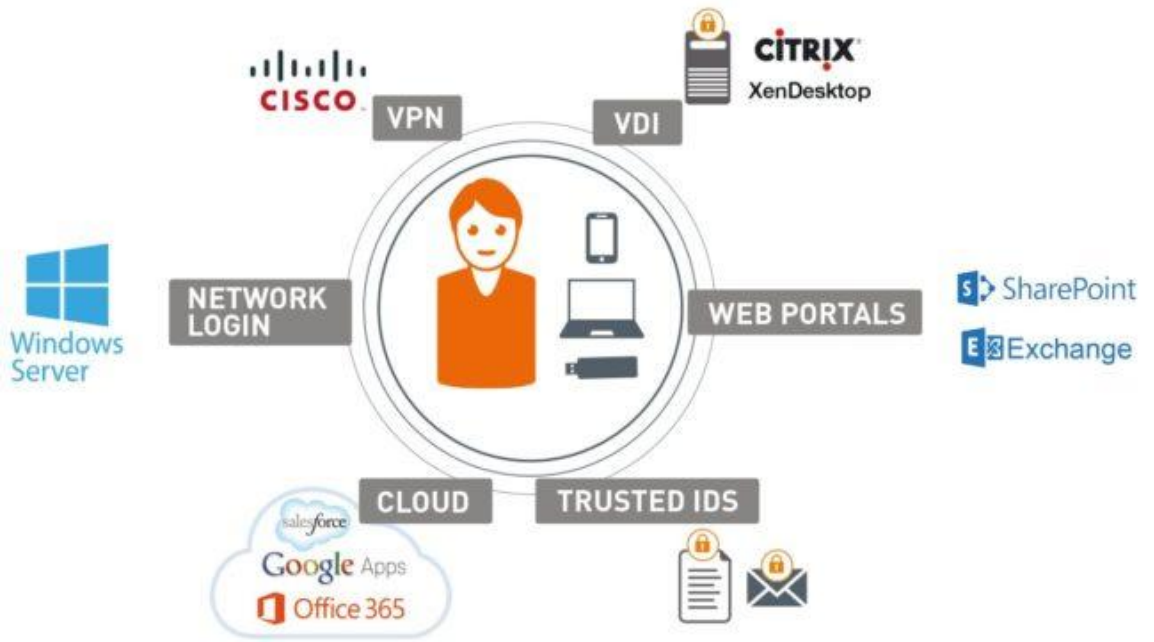
## **Gemalto: Digital Security Solutions & Services**

Gemalto is the World's leading provider of Digital security & Data Protection. Since 2006, Gemalto have helped Government agencies and Global corporations to secure their most precious assets and intellectual property. Our data-centric approach focuses on the protection of high value information throughout its lifecycle, from the data center to the cloud. Our security solutions covers a vast area of industries ranging from Banking and Payments, Biometrics to IOT and Software Monetization.

### **Protecting High Value Data**

Gemalto's security services & solutions have enabled our customers to adapt to the escalating internal and external threats to their high-value data, and rapidly evolve to address new business requirements and compliance mandates. Our products deliver persistent protection of sensitive data throughout the information lifecycle by:

- Providing protection for both User Ids and applications Ids.
- Protecting transactions of highly valuable data.
- Providing encryption operations for critical sets data that is being shared, moved , accessed or stored.
- Creating cloud-based infrastructure



# Content

<i>Declaration</i> -----	<i>i</i>
<i>Certificate</i> -----	<i>ii</i>
<i>Acknowledgement</i> -----	<i>iii</i>
<i>About the Company</i> -----	<i>iv</i>
<i>Table of Contents</i> -----	<i>vi</i>

## Unit 1: Introduction

### Chapter 1: Outline

1.1 Abstract	1
1.2 About H.S.M.	2
1.3 Product Overview	3
1.4 Project Overview	4

## Unit 2: Project

### Chapter 2: Requirement and Analysis

2.1 Software Requirements	5
2.1.1 Aim	5
2.1.2 Scope	5
2.1.3 Feasibility	5
2.1.3.1 Technical Feasibility	5
2.1.3.2 Cost Feasibility	6
2.1.3.3 Time Feasibility	6
2.1.4 Environment	7
2.1.4.1 Hardware & Software	7
2.1.4.2 Input / Output	8

2.1.4.3 Interaction	
2.1.4.4 Security	
2.2 Product Perspective	8
2.2.1 Interface	
2.2.2 H.S.M.'s Operations	
2.3 Functions	9
2.4 User Characteristics	9
2.5 Constraints	10
<b>Chapter 3: Design Specification</b>	
3.1 Introduction	11
3.2 Development Strategies	11
3.3 System Architecture	13
3.3.1 Purpose	13
3.3.2 Architecture	14
3.3.2.1 Lush	14
3.3.2.2 Function Module	15
3.3.2.3 Crypto Module	15
3.3.2.4 Console Module	15
3.3.2.5 Host Module	15
3.3.2.6 Base Module	15
3.4 Policies and Tactics	16
3.5 Coding Guidelines	16



## **Chapter 4: My Learning and Role**

4.1 Basics of Cryptography	18
4.2 Technology Studied/Used	21
4.2.1 Basics of Linux	21
4.2.2 C language	22
4.2.3 Coding Guidelines	23

## **Chapter 5: Development and Snapshots**

5.1 Dev tasks	25
5.2 Overall Flow of Task	25
5.3 List of Tasks	25
5.3.1 Understanding basic cryptographic key manipulation APIs	26
5.3.2 Automation framework hands on & Modification	27
5.3.3 Key Generation API Modification	27
5.3.4 Key View API Modification	28
5.3.5 Key Delete API Modification	28
5.3.6 Removed all C++ warnings from LUNAEFT code base	29
5.3.7 Coverity Fixes	29
5.4 Snapshots	31

## **Chapter 6: Conclusion**

6.1 Conclusion	37
----------------	----

<b>References</b>	<b>38</b>
-------------------	-----------

# Unit 1: Introduction

## Chapter 1: Outline

### 1.1 Abstract

Internship program provides students with an opportunity to experience and understand the work culture of an organization. It helps interns to form the foundation for their careers by providing them the exposure to industry and an understanding of industrial operations. Ability to understand problems and the ability to solve them enhances because professionals that work around you always motivates you and help you to do things with perfection.

### 1.2 About H.S.M.

I had a wonderful experience in the company. The internship helped me to expand my knowledge base by providing me with the opportunity to work with products for which Gemalto is famous for, as instance LUNAEFT 2.x . It is one of the many flagship projects of Gemalto. H.S.M.s are known for is reliable protection for applications, transactions and information assets by securing cryptographic keys.



Fig 1.1 LUNAEFT 2.x

## 1.2 Product Overview

Here are some H.S.M.s that are offered by Gemalto:

- General Purpose HSMs, Embedded



**Fig 1.2 General Purpose H.S.M.s**

Gemalto's embedded H.S.Ms provides tight integration b/w an application server and the H.S.M. Gemalto's embedded H.S.M provides a cost effective solutions of data protection without interrupting the business processes, making Gemalto's embedded H.S.M.s the best solution for the protection of cryptographic keys.

- General Purpose H.S.M.s, Network Attached



**Fig 1.3 General Purpose H.S.M., Network Attached**

In situations where H.S.M. is needed to be shared b/w multiple application servers, Network attached H.S.M.s protects cryptographic keys used to secure transactions and sensitive data.

- Payment H.S.M.s

Security solutions for electronic fund transfers are a growing concern in the market. Gemalto's Payment H.S.M.s, also known as LUNAEFT, is one of the highest performance solutions for the protection of online transactions, providing its user with a powerful end to end security for electronic fund transfers.

- Certifications and Validations



**Fig 1.5 FIPS Standard 140-2**

## **1.4 Project Overview**

My role in the project was to deal with modifying present APIs in payment H.S.M. to add the support for a new key known as AES BDK. APIs were to be modified in such a way, so that none of the existing functionality are impacted in a negative way. In order to accomplish my task I went through an extensive research about the basic concepts of cryptography and various algorithms dealing with Key manipulation in the H.S.M.

Apart from the major task of modifying the APIs, I was involved in removal of errors that were discovered during the static analysis of the source code for the H.S.M. and also was involved during the testing and debugging of smartcard functionalities and newly added features. Understanding of high level architecture and guidelines to manage the code was also a learning experience.

I was also given a task to modify the present automation framework used to test the smartcard functionality.

## **Unit 2: Project**

### **Chapter 2: Analysis and Requirements**

#### **2.1 Software Specifications**

##### **2.1.1 Aim**

The aim of this project was to change an existing API and adding new methods to address the new user requirements. The modification dealt with the change in key manipulation techniques used to handle cryptographic key stored in the H.S.M. Apart from the changes in the API, creation of new test case sets to harden APIs was also the aim of this project.

##### **2.1.2 Scope**

This project was a improvement task designed to satisfy new customer requirements. It was assigned to me as an assessment.

##### **2.1.3 Feasibility**

###### **2.1.3.1 Technical Feasibility**

After conducting the analysis of this project, we discovered that both the hardware and software components like the Operating System, Programming languages used for development of the APIs etc. which were essential for the development of the project were easily available in the market.

Since all resources are easily available and there was no such resource that was unavailable. Hence, we can conclude that the given project is technically feasible.

### **2.1.3.2 Cost Feasibility**

Basic COCOMO model was used to conduct the analysis of cost feasibility. According to the COCOMO model, the cost required for the completion of the task can be calculated using the following equations-

$$\text{Effort Applied} = a (\text{KLOC})^b$$

$$\text{Dev Time} = c (\text{Effort Applied})^d$$

$$\text{People Required} = \text{Effort Applied} / \text{Dev Time}$$

Where,

$$a = 2.4,$$

$$b = 1.05$$

$$c = 2.5$$

$$d = 0.38$$

& KLOC stands for estimated number of lines of source code delivered.

The calculated cost was much less than the revenue generated and also the required number of developers is 2.

### **2.1.3.3 Time Feasibility**

COCOMO model was used for this analysis and according to the results of the analysis, the time required for the development of this project was 1 month.

The resources required for the development of the project are available and there is no dependency of any sought which needs to be resolved, therefore requirement is not an issue.

However, It is my first experience with the product and therefore I took some time to understand the basic design and architecture of the product. Initially, I start with the study of Payment H.S.M. and gained knowledge of the existing API.

## **2.1.4 Environment**

### **2.1.4.1 Hardware and Software**

The specs are listed below:

**Platform used:** Windows & Linux OS

**Tools used:** VMware, Putty

**Hardware:** Any hardware can be used for the client

**Software:** Provides by Gemalto. No Additional software required.

### **2.1.4.2 Input / Output**

Input to the system are in the form of command or string , these are then used to call API which perform the task specified by the user. The function code and the data required by the API is supplied by the input.

The output of the system will provide response code for the call which will indicate the success of the failure of the command.

### **2.1.4.3 Interaction**

Interaction to the project being developed can only be made through a selected number of systems. These systems are specified by the admin using their IP addresses No System other than the one specified by the admin can interact with the API.

### **2.1.4.4 Security**

There was no requirement for the implementation of additional security measures for development, however any dependencies that results due to the changes made in the API must be resolve as soon as possible.



## **2.2 Product Perspective**

### **2.2.1 Interface**

The Payment H.S.M. (LUNAEFT) is not dependent on any other system for its input. Only test data or processing data are provided by external systems. The H.S.M. is Self Contained device and all operations for the H.S.M. are performed internally.

#### **User Interface (UI)**

In order to operate the H.S.M., Users are provided with the following screens-

1. Web Console
2. Luna Shell (Lush)

#### **Hardware Interface**

The hardware interface for the H.S.M. comes in the form of

1. LCD Screen
2. 3 USB Ports
3. Serial Port
4. Ethernet Ports

### **2.2.2 H.S.M's Operations**

The List of Operations that can be performed by the H.S.M. are provided by the user interface of the H.S.M. The UI can be used to call API and also to supply them with the appropriate data. I cannot disclose the list of operations as per the company policy however some generic operations include key entry, pin mailer etc.

### **2.3 Functions**

The project is responsible for a wide variety of tasks, however the API that I was developing is used to perform key manipulation operations like key generation, view and delete.

### **2.4 User Characteristics**

Here are some operations that can be performed by the admin user-

1. Create Partition
2. Change Passwords
3. Host Functions
4. Configure APIs
5. Key Entry
6. Network configurations
7. Enable Sensitive functions
8. View audit logs
9. View H.S.M. configurations
10. Delete Keys
11. View keys
12. Create Administrators
13. Enable web console

Operations that can be performed by non-admin users-

1. View audit logs
2. Configure APIs
3. Configure H.S.M

## **2.5 Constraints**

Certain constraints are imposed during the designing of the product, some of those constraints are listed below-

1. In order to support Client Server Architecture, a minimum of 64 MB of RAM must be giving to the Client.
2. Only Specific communication modes can be used to conduct the communication between the systems.

## Chapter 3: Design Specification

### 3.1 Introduction

The existing design used by the H.S.M. is not altered by the newly added functionality. It is simply an addition to the existing API. Hence, this section of the report would help us to understand the high-level design of the H.S.M.

All the operations which can be performed by the product are listed on the user interface. As per the company policies, I cannot disclose the list of operation in the report. But for your understanding I can mention some of the generic operation which includes key generation, Encryption pin mailer etc.

### 3.2 Development Strategies

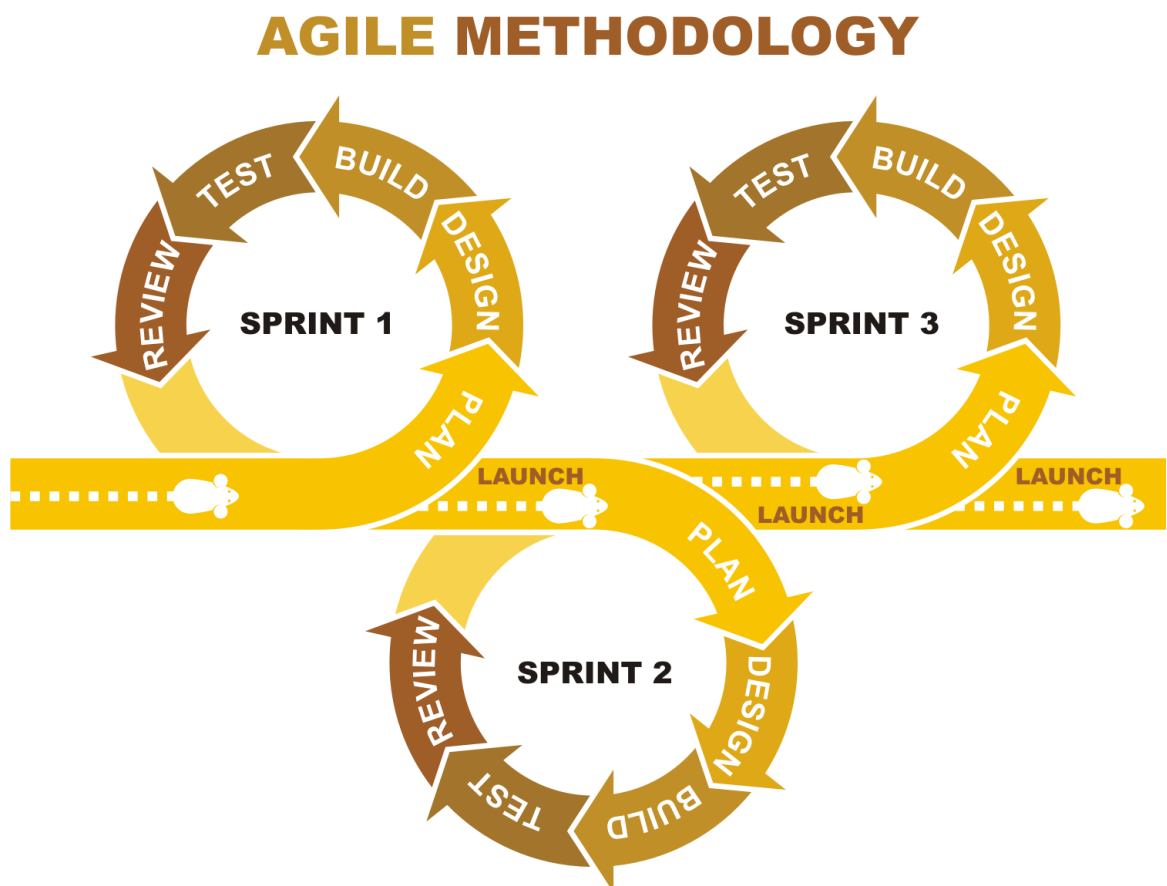


Fig 3.1- Agile Methodology

Gemalto follows agile method to divide tasks into modules and achieve smaller goals on a daily basis. Some of the basic terminologies used in agile developments are as follows -

- Story- A statement from the end user perspective that defines the software requirements.
- Sprint- Fixed period of time in which stories are converted into deliverables.
- Scrum- Scrum is one of the most popular Agile methods. It is an adaptive and iterative framework. It can be conducted in form of standup meetings that are held on a daily basis. Main focus areas of the meetings are-
  - What did you do yesterday that helped the team meet the sprint goal?
  - What will you do today to help the team meet the sprint goal?
  - Do you see any impediment that prevents you or the team from meeting the sprint goal?

### **Agile Training**

Gemalto officials conducted a 2 days session on Agile Development for helping new comers to understand the code of conduct of the organization. The Session was held in the company premises by my mentor Mr. Shivam Garg. The further understanding of the Scrum Framework and other Agile technique was provided by Mr. Manish Kumar, who is the Scrum Master for the LUNAEFT 2.x team.

### **Agile Process**

1. Initially , bigger tasks are broken into small stories by Project Lead Manager.
2. Specific measures are taken to assign priorities to the stories. These measures depend on the business conducted by the organization.
3. Team is assigned with some of the stories which are to be completed within the duration of the sprint.
4. A sprint can have a duration of 1 or 2 weeks depending on the team.

5. A functional product is delivered by the team after the completion of the sprint. Each delivered product have some market value.
6. The deliverables are then reviewed at the end of the sprint and new stories are decided for the next sprint.

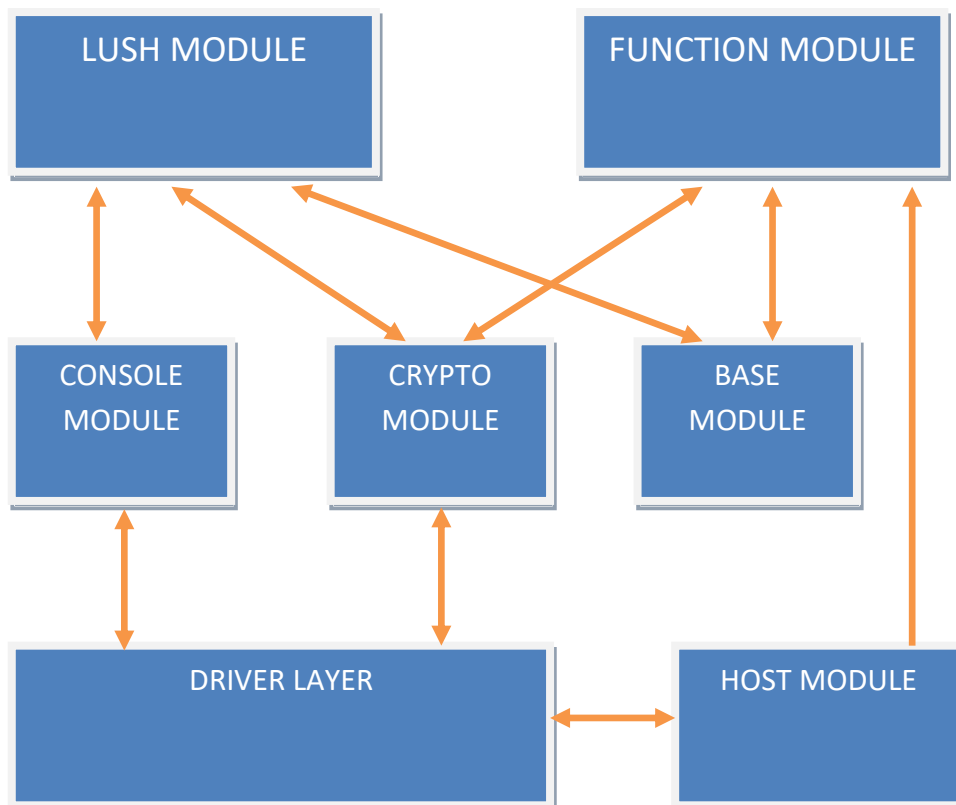
### 3.3 System Architecture

#### 3.3.1 Purpose

The sections aim at describing the high-level design of the General Purpose Hardware Security Module.

#### 3.3.2 Architecture

The diagram below depicts the high level view of the architecture of LUNAEFT

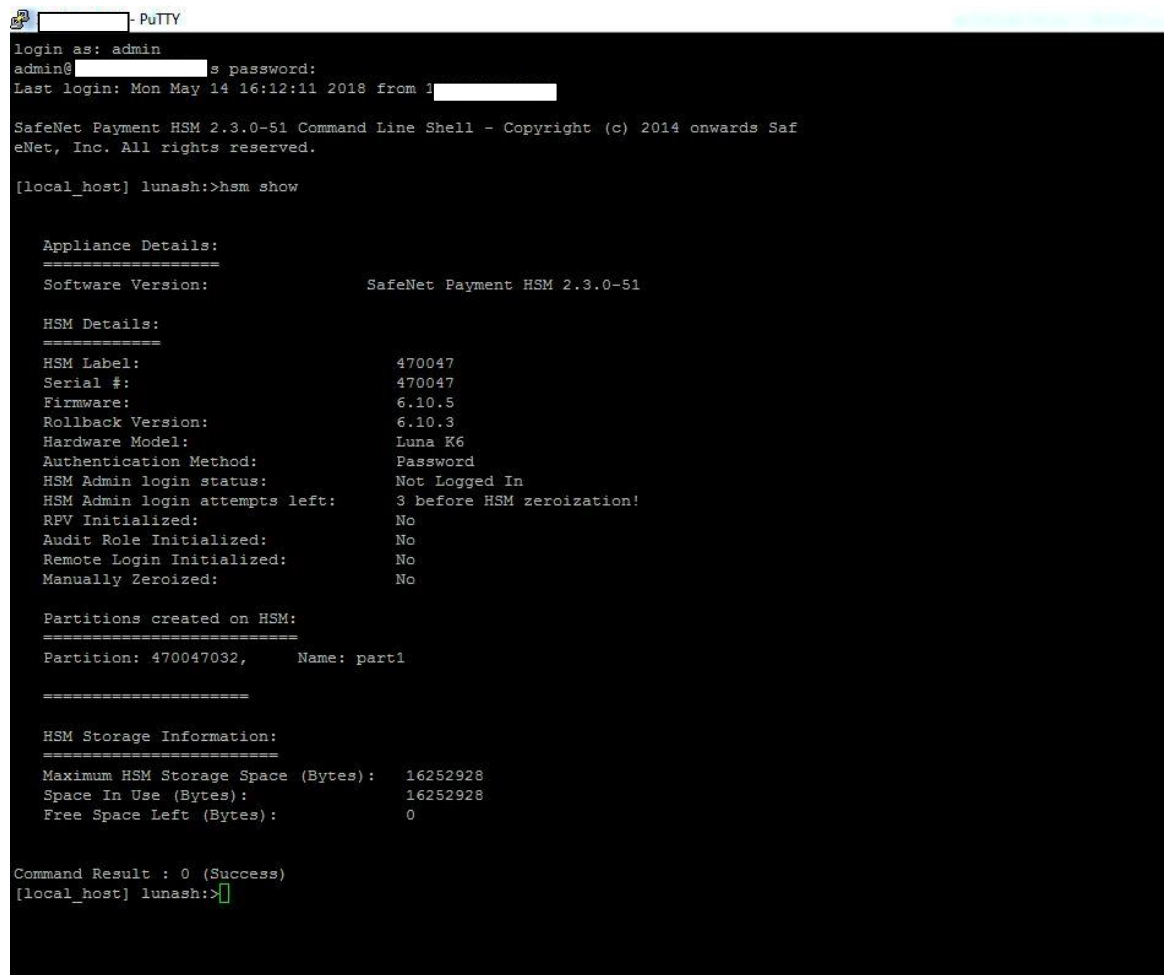


**Fig 3.2 Software Architecture**

### 3.3.2.1 Lush Console

“LUSH” is the short form for Luna Shell. It provides the user with a command line interface using which any operation can be performed on the H.S.M. Lush console is similar to the Bourne shell.

In order to use lush as an interface, user must connect to the Hardware Security Module machine via a serial cable or SSH. C & C++ are used to implement the Lush module and the implementation is such that new commands can be added in the future without changing much of the code.



```
login as: admin
admin@ [redacted] s password:
Last login: Mon May 14 16:12:11 2018 from 1 [redacted]

SafeNet Payment HSM 2.3.0-51 Command Line Shell - Copyright (c) 2014 onwards Saf
eNet, Inc. All rights reserved.

[local_host] lunash:>hsm show

Appliance Details:
=====
Software Version:                SafeNet Payment HSM 2.3.0-51

HSM Details:
=====
HSM Label:                        470047
Serial #:                          470047
Firmware:                          6.10.5
Rollback Version:                  6.10.3
Hardware Model:                    Luna K6
Authentication Method:             Password
HSM Admin login status:            Not Logged In
HSM Admin login attempts left:     3 before HSM zeroization!
RPV Initialized:                   No
Audit Role Initialized:            No
Remote Login Initialized:          No
Manually Zeroized:                 No

Partitions created on HSM:
=====
Partition: 470047032,              Name: part1
=====

HSM Storage Information:
=====
Maximum HSM Storage Space (Bytes): 16252928
Space In Use (Bytes):              16252928
Free Space Left (Bytes):            0

Command Result : 0 (Success)
[local_host] lunash:>
```

### **3.3.2.2 Function Module**

Methods that are responsible for the proper working of the hardware security module are placed in this module. It is also responsible for handling host requests and for redirecting these calls to appropriate APIs. Some of the major operations of the function module are creation of threads, to load and initialize modules, to initialize the communication between different components, entry and exit point setup etc. Disabling/enabling of functions is also handled by this module.

### **3.3.2.3 Crypto Module**

It includes implementation of symmetric and asymmetric cryptographic algorithms (both h/w & s/w). Key generation algorithms are also implemented here.

### **3.3.2.4 Console Module**

Console Module provides the H.S.M. administrator with the list of modules that are responsible for performing console operations. These operations may include initialization of admin credentials, key management etc.

### **3.3.2.5 Host Module**

H.S.M. users can use different modes of communication to make function requests. Now, these modes of communication are placed in the host module.

### **3.3.2.6 Base Module**

It consists of core libraries for the Hardware Security Module source code.



### 3.4 Policies & Tactics

Here are some of the policies and tactics that are followed during the making of this design spec document-

- **Product being used in Development**
  - C language
  - “gdb” compiler
  - Using MSEXCEL sheets as an input.
  
- **Guidelines & conventions**

Standard C language guidelines are used during the development of the API.
  
- **Testing of s/w**

Testing is done by both manually and through automation. New APIs are check for the expected outputs.
  
- **S/w maintenance**

All users can easily maintain the system as provided all the provided resources are easily available in the market.
  
- **Interfaces(UI, s/w, h/w, communications)-**

User are supposed to have a good knowledge of Linux & Windows OS and should also have in depth knowledge about H.S.M.s.

### 3.5 Coding guidelines

During our internship at Gemalto, we had to strictly follow the coding guidelines and conventions used by the development team. The guidelines came in the form of a

document, stating different approaches to make the code cleaner during compilation, much readable, and bug free.

Some of the best practices stated in the coding guideline are as follows-

- Do not use standard system header file name for any other header files.
- Use inline function in lieu of macros.
- Do not read uninitialized variables.
- Do not access a variable through a pointer of an incompatible type.
- Do not form or use out-of-bounds pointers or array subscripts.
- Do not subtract or compare two pointers that do not refer to the same array.
- Do not attempt to modify string literals (constants).
- Arguments to character-handling functions must be representable as an unsigned char.
- Always close files descriptors when they are no longer needed.
- Do not access a closed file.
- Do not access freed memory.

## Chapter 4: My Role and Learning

### 4.1 Basic of Cryptography

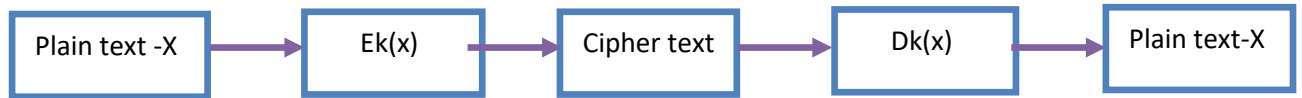


Fig 4.1 Cryptography Flow

#### 4.1.1 Cryptosystem

Cryptosystem as a whole consists of the following items-

- Required S/w
- Essential Protocols
- Encryption and Decryption Algorithms
- Cryptographic Keys

#### 4.1.2 Services of Cryptosystem

- **Confidentiality:** Confidentiality is an approach to avoid or restrict any unauthorized access to the sensitive data.
- **Authenticity:** Authenticity stated that the information being received during the communication between the sender and the receiver is coming from a valid sender and not from any imposter.
- **Integrity:** It means that the message being receiver will not be modified in any way during its transmission from the sender and the receiver. The modification can be accidental or intentional.
- **Non-Repudiation:** Sender cannot deny having sent a message.

### 4.1.3 Symmetric Cryptography

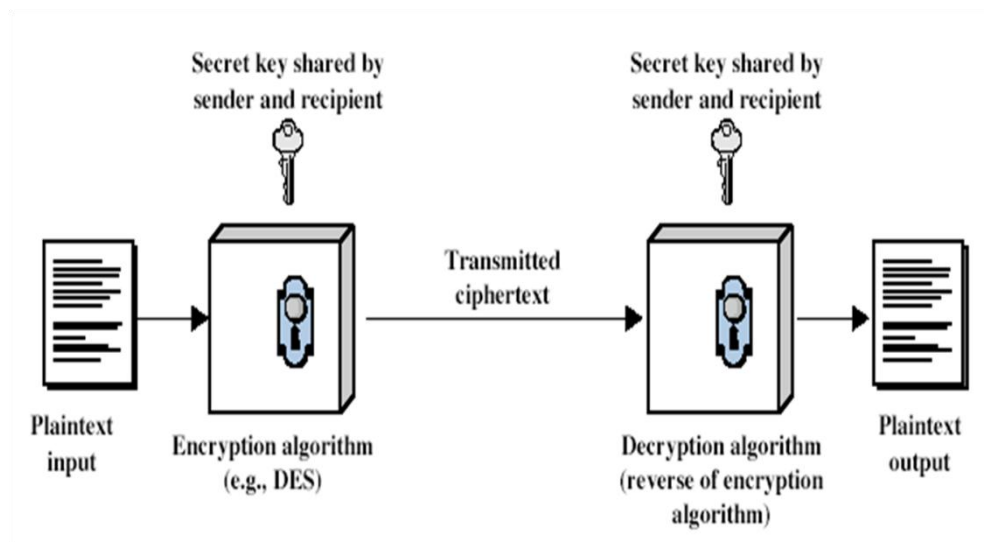
In symmetric cryptography, Same key is used for both encryption and decryption of data.

#### Strength:

- It is a fast method to encrypt and decrypt data.
- Cipher text is hard to break for large keys.

#### Weaknesses

- A secure mechanism is needed to share the key b/w the sender and the receiver of data.
- Key management becomes difficult with the increase in no of keys used by different individuals.
- Provides confidentiality but not authenticity or nonrepudiation.



**Fig 4.2 Symmetric Cryptography**

#### **Some of Symmetric Cryptography Algorithms**

- Data Encryption Standard (DES)
- Triple DES
- Advanced Encryption Standard (AES)
- International Data Encryption Algorithm

- RC4, RC5 and RC6
- Blowfish

#### 4.1.4 Asymmetric Cryptography

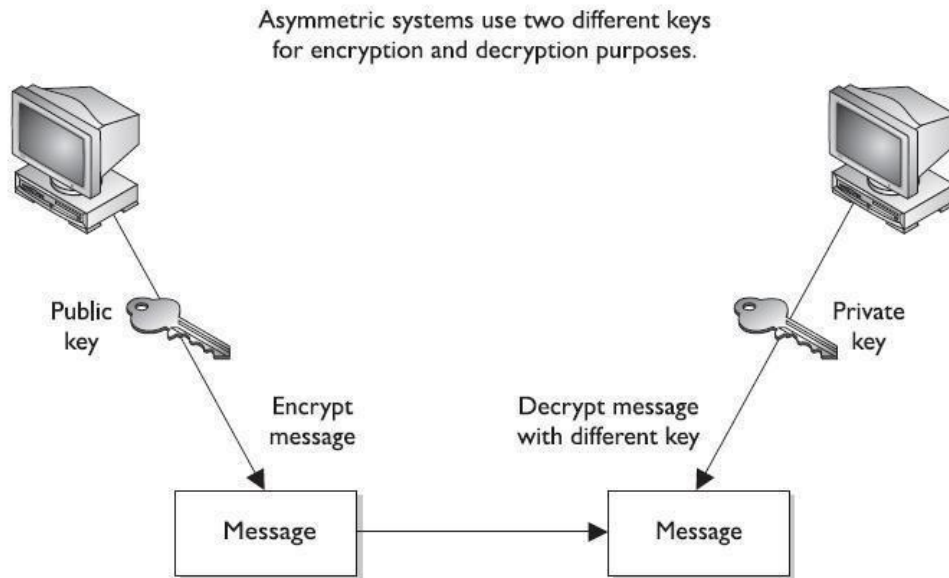


Fig 4.3 Asymmetric Cryptography

#### 4.1.5 Message Integrity

When exchanging sensitive data, the receiver must have the knowledge of whether the data being received the same and is not being altered in any way.

#### 4.1.6 Triple Data Encryption Standard

Double Data Encryption Standard has a key length of 112 but it is not very good for encrypting data efficiently as there are specific attacks against D-DES that reduces its work factor to about the same as DES. Even though it requires a greater amount of computation, D-DES is no more secure than DES.

T-DES on the other hand is highly resistant to differential cryptanalysis. This is because of its high number of computational round which makes it much more secured than DES and D-DES. But due to the extra computation that T-DES performs, there is a heavy performance penalty. Thus, for T-DES encryption and decryption operations can be 3 times slower when compared with encryption and decryption operations for DES.

#### **4.1.7 What are Hash Functions?**

Hash function is a mathematical function that can be used to detect changes in the data/message. A numerical value taken as an input is converted into a compressed numerical value. A hash function accepts a variable size input and produces a fixed sized output. Return value of the hash function is called a hash value or message digest.

Hash functions are one-way functions i.e. for a given input, output can be easily calculated, but calculating the input values from a given output values of a hash function is infeasible. Also a good hash function must have strong collision resistance.

#### **4.1.8 Some Popular Hash Functions**

- Message Digest (MD)
- Secure Hash Function

### **4.2 Technologies Studied/Used**

#### **4.2.1 Languages –C/C++**

C language was used to develop the APIs for H.S.M.. Some Advantages of the C Language are as follows-

1. Modularity-Functionality
2. Speed
3. Portability
4. Flexibility
5. Code Reusability

Also, C++ was used to program some testing modules which were used to perform unit testing on new APIs.

### 4.2.2 Basics of Linux

H.S.M. source code is deployed on virtual machines which run CENTOS kernel. Hence it was important that we have basic knowledge of Linux commands in so that we don't encounter any problems working with the virtual machines and the H.S.M. Here are some basic Linux commands that are generally used while working in the Linux environment-

- **cd:** In order to change the current working directory on linux or unix like system, this command is used.
- **mkdir:** This command is used to create new directories.
- **Pwd:** The pwd command displays the full pathname for the current directory.
- **grep:** The grep command is used for searching plain text data sets for a particular regular expression/sequence of characters.
- **cat:** The cat command have 3 major functions: Displaying text files, combining copies of text files and creating new text files.
- **mv:** The mv command is used to move, rename & remove files & directories.
- **cp:** The cp command is used to copy files and directories. Also the copies are independent of originals.
- **rm:** It is used to delete files and directories.
- **ps:** This command provides us with the lists of the currently running processes and there PIDs.

- **ls:** This command is used for listing the content of the working directory. The content listed can be a files or directories.
- **ping:** Ping is a shorthand for Packet Internet Groper. It can be used to test the connectivity b/w 2 nodes. It uses ICMP to communicate with other devices. In Linux, ping command keep executing until it is interrupted.
- **Ifconfig:** This command is used for network interface configurations.
- **chmod:** A File's access permissions can be modified using this command.
- **su:** This command allows the user to run a shell as the root user.
- **traceroute:** This commands shows the number of nodes taken by a packet to reach its destination.
- **route:** This command is used to manipulate and display IP routing table.
- **host:** This command is used to map IP addresses to names and vice versa
- **whoami:** It returns the name of the owner of the current login session.
- **file:** This command can classify file system objects that are provided to it as arguments.

### 4.2.3 GDB Debugger

GDB debugger is an open source debugger that can be used on unix-like operating systems. It allows the user to exercise control over the execution of the code and also give them the privilege to observe changes in the variables.

Using GDB a user can add breakpoint ,this allows the user to pause the execution at any point within the source code. User can also watch a step by step execution of the source code.

GDB is supported for the following languages-

- Ada
- Assembly
- C



- C++
- D
- Fortran
- Go
- Objective-C
- OpenCL
- Modula-2
- Pascal
- Rust

```
[root@local_host ~]#
[root@local_host ~]# gdb att 9823
GNU gdb Fedora (6.8-37.el5)
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i386-redhat-linux-gnu"...
att: No such file or directory.
Attaching to process 9823
Reading symbols from /usr/lunaeft/bin/partition_process...(no debugging symbols found)...done.

warning: .dynamic section for "/usr/lib/libz.so.1" is not at the expected address
warning: difference appears to be caused by prelink, adjusting expectations
warning: .dynamic section for "/lib/libm.so.6" is not at the expected address
warning: difference appears to be caused by prelink, adjusting expectations
warning: .dynamic section for "/lib/libgcc_s.so.1" is not at the expected address
warning: difference appears to be caused by prelink, adjusting expectations
Reading symbols from /lib/libpthread.so.0...(no debugging symbols found)...done.
[Thread debugging using libthread_db enabled]
[New Thread 0xb7fac6c0 (LWP 9823)]
[New Thread 0xb60d2b90 (LWP 10118)]
[New Thread 0xb6296b90 (LWP 10109)]
[New Thread 0xb6449b90 (LWP 10094)]
[New Thread 0xb644eb90 (LWP 10093)]
[New Thread 0xb6453b90 (LWP 10092)]
[New Thread 0xb66cbb90 (LWP 10091)]
[New Thread 0xb68cbb90 (LWP 10090)]
[New Thread 0xb690bb90 (LWP 10089)]
[New Thread 0xb6910b90 (LWP 10088)]
[New Thread 0xb6b10b90 (LWP 10087)]
[New Thread 0xb6b50b90 (LWP 10086)]
[New Thread 0xb6d50b90 (LWP 10085)]
[New Thread 0xb6eabb90 (LWP 10084)]
Loaded symbols for /lib/libpthread.so.0
Reading symbols from /lib/libdl.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/libdl.so.2
Reading symbols from /opt/eft-2.3.0-51/lib/libos2compat.so...(no debugging symbols found)...done.
Loaded symbols for /opt/eft-2.3.0-51/lib/libos2compat.so
Reading symbols from /opt/eft-2.3.0-51/lib/librt.so...(no debugging symbols found)...done.
```

**Fig 4.4 GDB Debugger**

## Chapter 5: Development

### 5.1 Dev Tasks

C language was used to accomplish the development tasks. All the coding was done using 'Eclipse IDE for C/C++'. It is a very powerful and efficient as it provides an excellent interface for coding and is also helpful for searching function, references and variable across the entire source code.

The source code cannot be shared as per the company policy.

### 5.2 Overall Flow of Task

Initially my goal was to gain the knowledge about LUNAEFT which includes-

1. Grasping the basic high-level architecture of the H.S.M.
2. Learning the procedure to deploy the code on a machine.
3. Setting up the virtual machines.
4. Understanding the procedure to compile and debug code.
5. Understanding the testing frameworks.
6. Understanding macros created to generate test cases in MS Excel.

#### Step by Step description

- **Learning about APIs:** Understanding the high level design and low level implementation of APIs.
- **Coding:** C Language and shell scripting were used to modify the source code of the H.S.M. Eclipse IDE was used for editing the code.
- **Compiling:** Linux virtual machine and 'gdb' compiler were used to compile the code.

- **Testing and Debugging:** ESM tester was used to test the modified APIs. If any bugs were captured after testing, 'gdb' debugger was also used for step by step debugging of the modified code.

## **5.3 List of Tasks**

### **5.3.1 Understanding basic cryptographic key manipulation APIs**

After understanding the architecture of H.S.M., basics of cryptography and development environment, the first task that was assigned to me comprised of understanding the basic key APIs used for key manipulation. Some of these APIs included the functionality for generating, viewing and deleting cryptographic keys.

High level manual test cases were executed to get a better understanding of the API's functionalities. The result for the manual test were observed and kept as a reference for testing the functionalities that would be included after modifying the APIs.

### **5.3.2 Automation framework hands on & Modification**

In order to test the functionality of the smartcard APIs, an automation framework was used for previous s/w releases. I was assigned with a duty to modify this automation framework so that it can be used for future s/w releases.

### **5.3.3 Key Generation API Modification**

The aim of this task was to add the support for a new key know as AES BDK.(Base Derivation Key). In order to accomplish this task, new functions must be added to the LUNAEFT code base and also some old functions must be modified to support the new key features. Lush command that triggers the generation of the AES BDK key was also modified in order to accept the change format of input and output of the functions. The changes were made using C language and shell script.

After the function was successfully, new test cases were written to test the functionality of the new functions .Also unit testing was done at my end.



```
[local_host] lunash:>
[local_host] lunash:>keyMgmt delete hsm [REDACTED]

Confirm deletion of the following key - Base Derivation Key (BDK)
[REDACTED]
[REDACTED]
[REDACTED] [Y/N]: y
RESULT : Operation successful

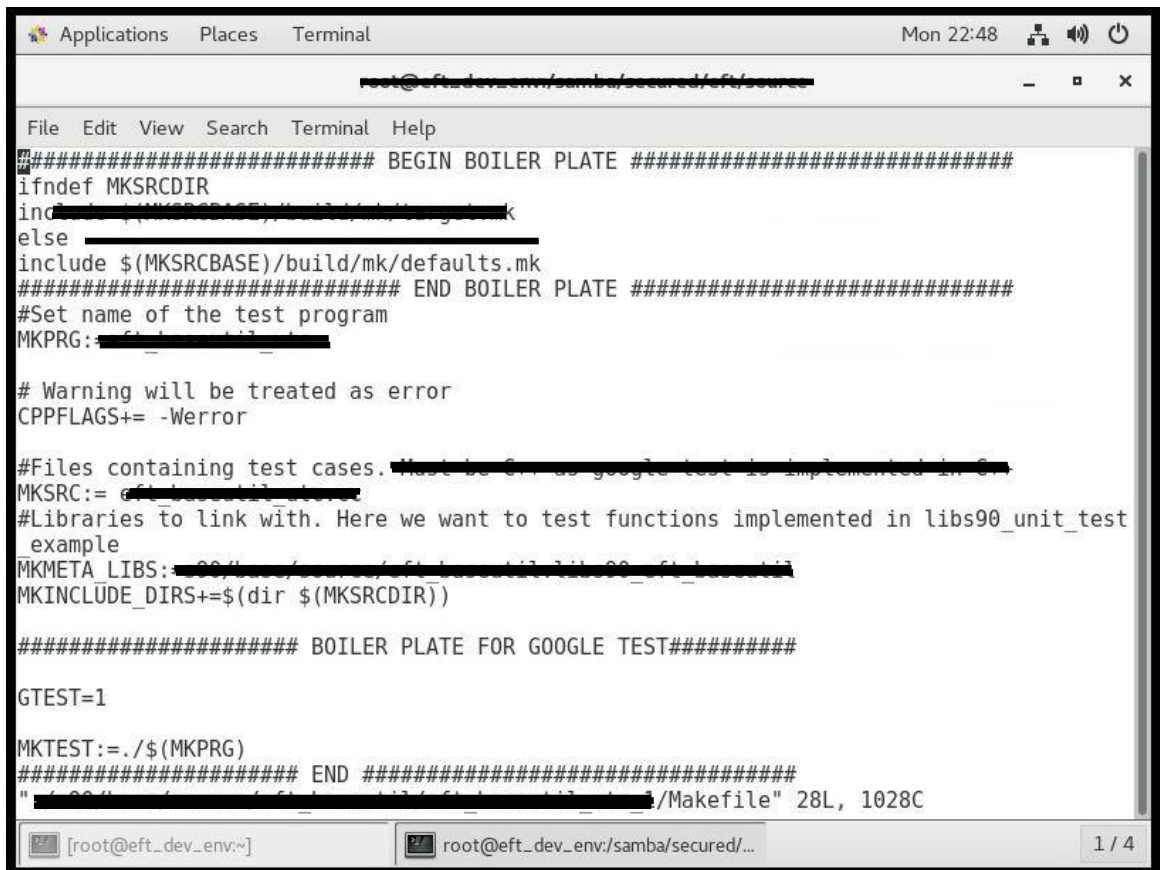
Command Result : 0 (Success)
[local_host] lunash:>
```

**Fig 5.3 Key delete lush command**

### **5.3.6 Removing all C++ warnings from LUNAEFT code base**

I was assigned a task to remove all C++ code warnings from LUNAEFT codebase to improve the quality of the code and make it look cleaner during compilations. Initially there were around 100 lines of warnings, but after completion of my task it was reduced to '0'.

After the removal of all the warnings, the root directory MakeFile was protected with –Werror Flag for C++ so that no further warnings can be introduced to the code. If any further warnings are introduced then they will be treated as errors and hence the code will not compile.



```
root@eft_dev_env/samba/secure/~/ft/source
File Edit View Search Terminal Help
##### BEGIN BOILER PLATE #####
ifndef MKSRCDIR
include $(MKSRCDIR)/build/mk/defaults.mk
else
include $(MKSRCDIR)/build/mk/defaults.mk
##### END BOILER PLATE #####
#Set name of the test program
MKPRG: ft-test

# Warning will be treated as error
CPPFLAGS+= -Werror

#Files containing test cases. Must be C++ as google test is implemented in C++
MKSRC:= ft-test.c
#Libraries to link with. Here we want to test functions implemented in libs90_unit_test
example
MKMETA_LIBS: -l90 -lunit_test
MKINCLUDE_DIRS+= $(dir $(MKSRCDIR))

##### BOILER PLATE FOR GOOGLE TEST#####
GTEST=1

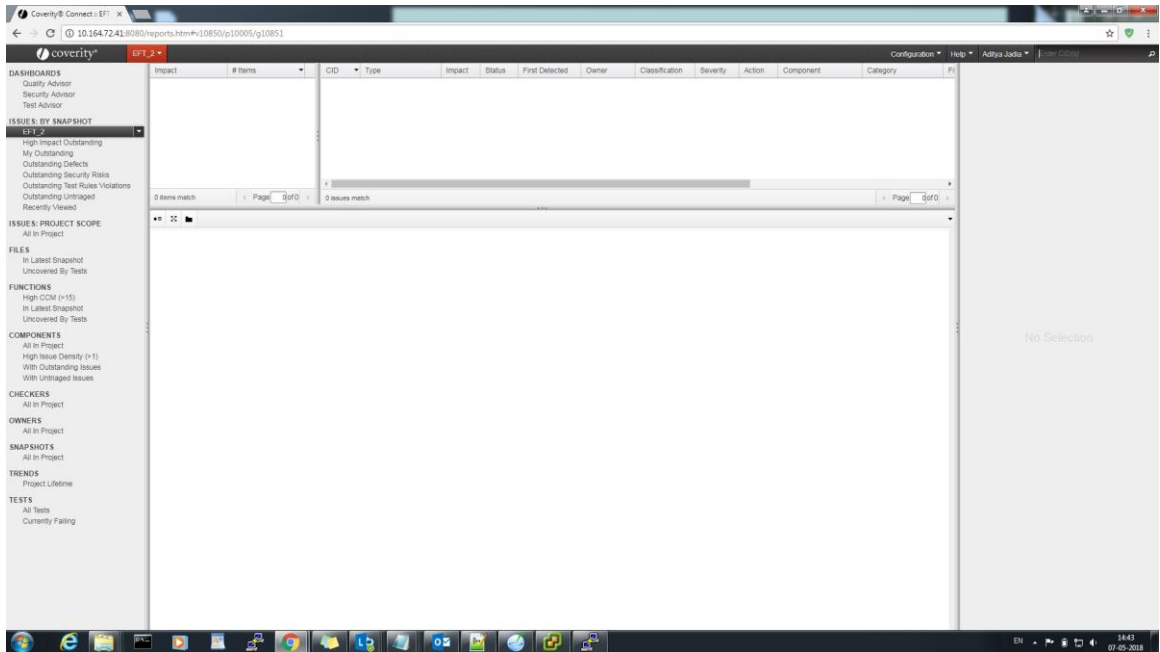
MKTEST:= ./$(MKPRG)
##### END #####
"~/ft/source/~/ft/Makefile" 28L, 1028C
```

**Fig 5.4 MakeFile (with new CPPFLAGS)**

### 5.3.7 Coverity Issues

Coverity is a powerful tool used to conduct both static and dynamic code analysis. Coverity is used to identify possible security bugs and vulnerability in the code. After every scan, coverity generates a report consisting of all the possible bugs in the code, each bug is provided with a coverity id and priority. Some examples of possible bugs are Stack overflow, Reference to a NULL pointer etc. After taking appropriate actions, you can mark the issue as fixed or false positive.

A team lead by my project lead Mr. Shivam Garg was given a task to remove all the coverity issues in our LUNAEFT codebase. I was a part of the team and was responsible for handling 40 issues. The task was completed without any complications.



**Fig 5.5 Coverity Dashboard for LUNAEFT**

## 5.4 Snapshots

Here are some snapshots of H.S.M interfaces, virtual machine etc.

- **LUNAEFT and available Accessories**



**Fig 5.6 Payment H.S.M.**



**Fig 5.7 Power Chord**



**Fig 5.8 Null-Modem Serial Cable**



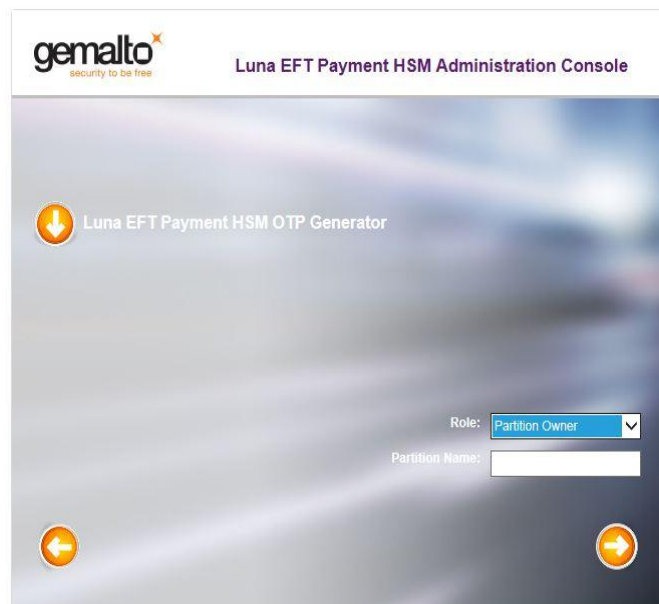


**Fig 5.9 Serial to console**

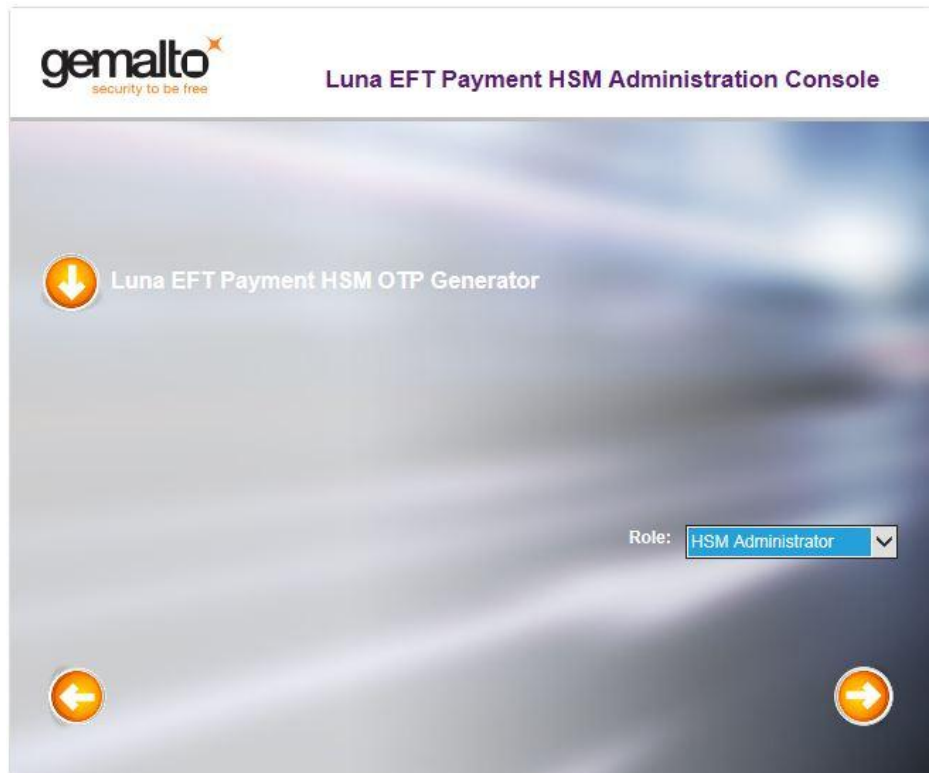


**Fig 5.10 Smartcard Reader**

- **Web console**

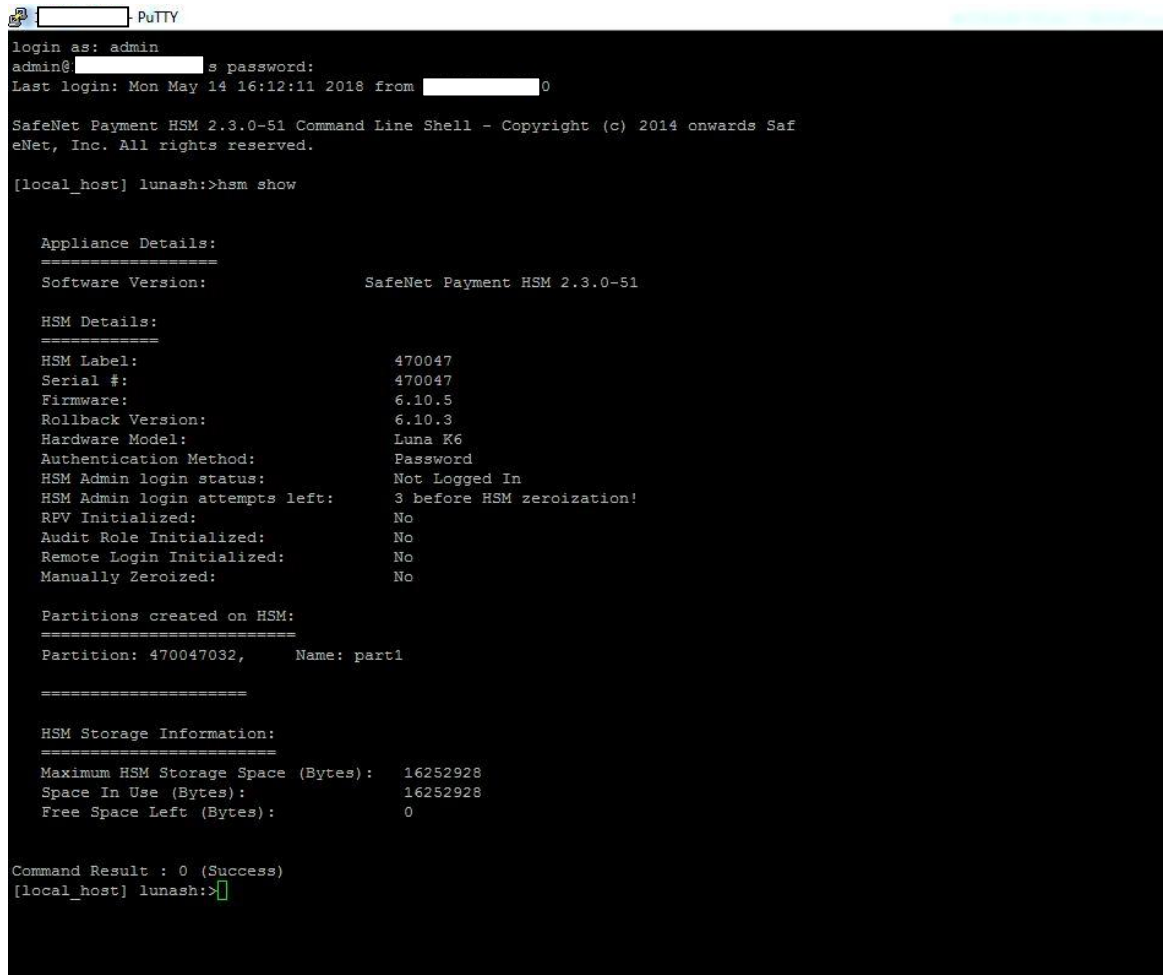


**Fig 5.11 Web Console Partition Owner**



**Fig 5.12 Web console Admin**

- **Lush**



```
login as: admin
admin@: s password:
Last login: Mon May 14 16:12:11 2018 from 0

SafeNet Payment HSM 2.3.0-51 Command Line Shell - Copyright (c) 2014 onwards Saf
eNet, Inc. All rights reserved.

[local_host] lunash:>hsm show

Appliance Details:
=====
Software Version:                SafeNet Payment HSM 2.3.0-51

HSM Details:
=====
HSM Label:                       470047
Serial #:                         470047
Firmware:                         6.10.5
Rollback Version:                 6.10.3
Hardware Model:                   Luna K6
Authentication Method:            Password
HSM Admin login status:           Not Logged In
HSM Admin login attempts left:    3 before HSM zeroization!
RPV Initialized:                  No
Audit Role Initialized:           No
Remote Login Initialized:         No
Manually Zeroized:                No

Partitions created on HSM:
=====
Partition: 470047032,            Name: part1

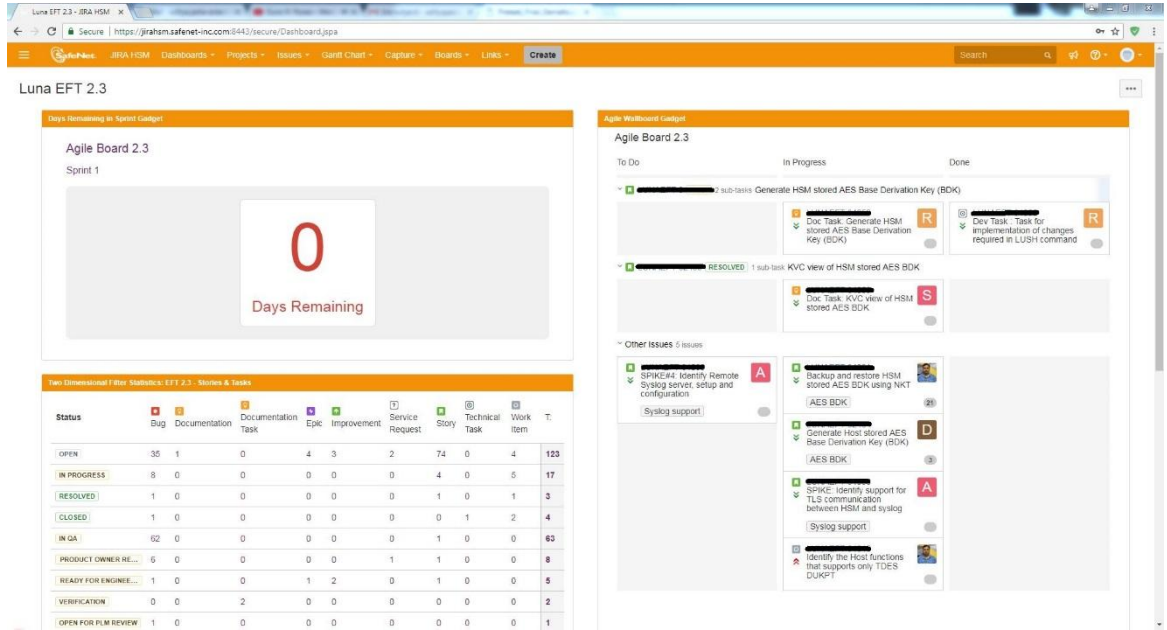
=====

HSM Storage Information:
=====
Maximum HSM Storage Space (Bytes): 16252928
Space In Use (Bytes):            16252928
Free Space Left (Bytes):         0

Command Result : 0 (Success)
[local_host] lunash:>
```

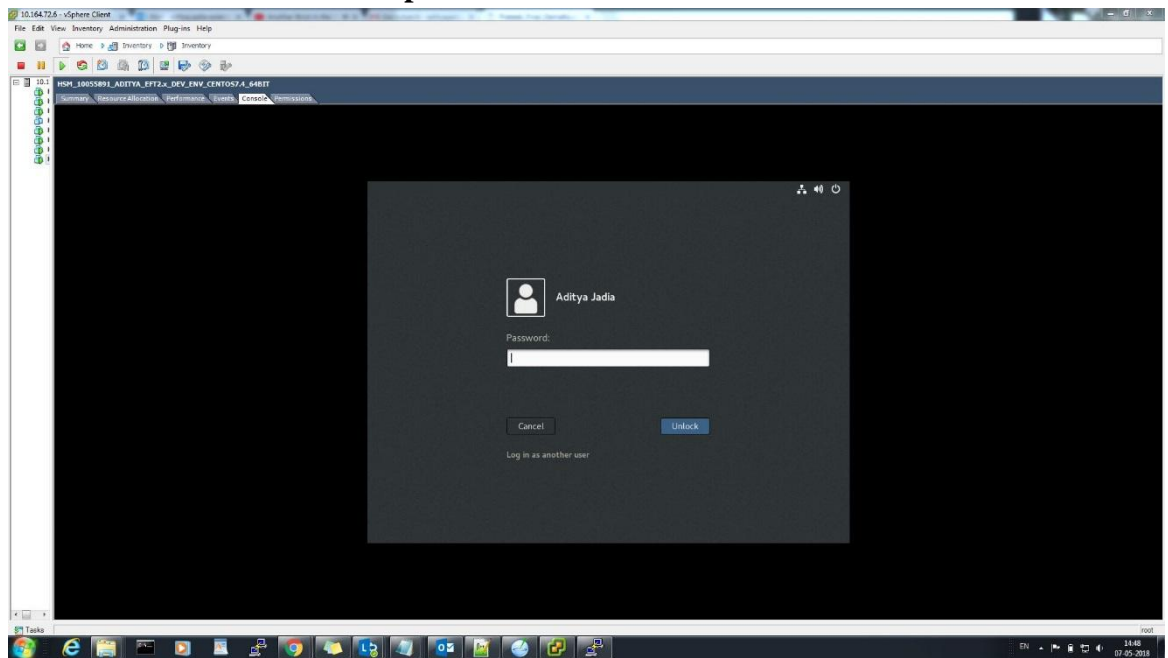
**Fig 5.12 Lush console**

■ **JIRA Dashboard**



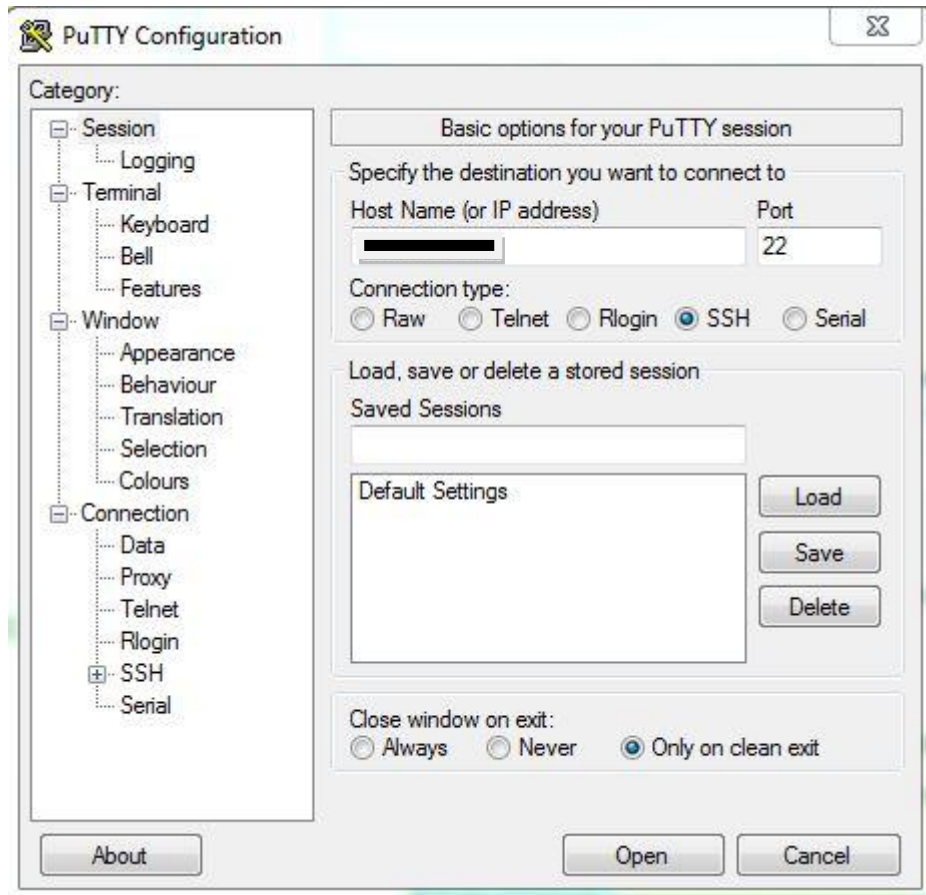
**Fig 5.13 JIRA**

- **Virtual Machine used to compile codes**



**Fig 5.14 Virtual Machine (CENTOS)**

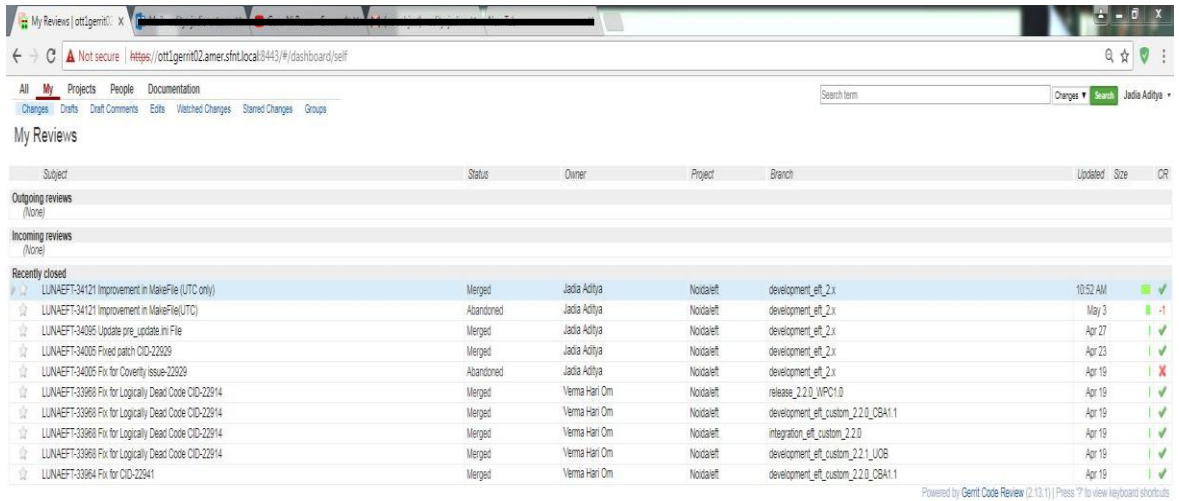
- **Putty**



**Fig 5.15 Putty**

Putty is an emulator that was used to connect to the H.S.M via SSH protocol or using a Serial cable. Using putty you can login into your H.S.M admin account and perform operations using built in H.S.M command.

- **Gerrit Dashboard**



**Fig 5.16 Gerrit Dashboard**

Gerrit portal was used to monitor the changes being done to the code by developers. The code that is being modified is first reviewed by the team leaders before merging it into the source code to ensure that the change does not impact the functionality in any way.



## **Chapter 6: Conclusion**

### **6.1 Conclusion**

During my 4 months of internship in Gemalto, I got a chance to work with some of the latest and cutting edge technologies of the IT industry. Gemalto's LUNAEFT is the most trusted Payment Hardware Security Module all around the globe. With high performance & temper resistant solutions, LUNAEFT provides protection for digital keys. Also LUNAEFT provide solutions for the protection of sensitive information like, User PINs and cardholder data with great efficiency. By achieving PCI compliance, Gemalto's LUNAEFT HSMs can not only assist them in meeting the basic compliance requirements but also leverage the technology to secure other key assets of their business. I got to understand the product and work on it.

In the end, I would like to conclude this report by saying that the entire project work assigned to me was completed without any major issue and was delivered on time. This project has helped me to understand the working of security industry and have added significant knowledge to my knowledge base.

# References

## Websites-

- <https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms/>
- <https://www.gerritcodereview.com/>

## Books/Guide

- HSM User Guide
- Network Security and Cryptography By William Stalling