MAJOR PROJECT REPORT

ON

Application for Issuing Digitally Signed Academic Certificates

Submitted by

Mani Kumar Reddy K (13B81A0579)

Under Supervision of

Prof. M. Badrinarayana

Professor of Department of Computer Science

CVR College of Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CVR COLLEGE OF ENGINEERING

An Autonomous Institution

(Affiliated to JNTU University, Hyderabad)

Mangalpally, Ibrahimpatnam-501 510

2016-2017

CVR COLLEGE OF ENGINEERING

(An Autonomous Institution)
ACCREDITED BY NATIONAL BOARD OF ACCREDITATION, AICTE
(Approved by A.I.C.T.E. & Govt. of Andhra Pradesh and Affiliated to JNT University)
Vastunagar, Mangalpalli(V), Ibrahimpatnam(M), R.R. District, PIN – 501 510
Web: http://cvr.ac.in, email: info@cvr.ac.in
Phones: Code within A.P.: 958414; Code from Outside: 08414
General: 252222, 252369 Office Telefax: 252396, Principal: 252396 (O)

CERTIFICATE

This is to certify that the Project Report entitled **"Application for Issuing Digitally Signed Academic Certificates"** is a bonafide record of work carried out by **Mani Kumar Reddy K (13B81A0579)** under my guidance and Supervision in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University, Hyderabad during the academic year 2016-2017.

Prof. M. Badrinarayana
Project Guide
Department of CSE
CVR college of Engineering

Prof .L.C. Siva Reddy
Head of the Department
Department of CSE
CVR college of Engineering

External Examiner

DECLARATION

I hereby declare that the project report entitled “Application for Issuing Digitally Signed Academic Certificates" submitted by us to CVR College of Engineering, in partial fulfillment of the requirement for the award of the degree of B.Tech, in Computer Science Engineering is a record of bonafide project work carried out by me under the guidance of **Prof. M. Badrinarayana**. I further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Signature of the Student

Mani Kumar Reddy K

ACKNOWLEDGEMENT

Apart from my efforts, the success of any project depends largely on the encouragement and guidelines of many other. I take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

I am also thankful to the Head of the Department **Prof. L.C. Siva Reddy** for providing excellent infrastructure and a nice atmosphere for completing this project successfully.

I take this opportunity to express our profound gratitude and deep regards one more time to our guide **Prof. M. Badrinarayana** of CSE department for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him, time to time shall carry us a long way in the journey of life on which we are about to embark.

The guidance and support received from all the members of **CVR College of Engineering** who contributed to this project, was vital for the success of the project. I am grateful for their constant support and help.

Mani Kumar Reddy K (13B81A0579)

ABSTRACT

APPLICATION FOR ISSUING DIGITALLY SIGNED ACADEMIC CERTIFICATES

With increased emphasis on Digital Technology by Govt., Non-Govt. agencies at various levels, a need is felt to use all the available technologies towards going Digital which has many advantages such as Efficiency, Traceability, Confidentiality, Integrity of documents and Origin, Non-repudiation etc. It also has the benefit of being environment friendly.

The System acts as a tool for the Controller Of Examinations (COE) to generate bulk digitally signed academic documents by providing student academic data input and a digital certificate and key store(a PKI physical token). The system is developed using Python and Java and their supported text/image/crypto libraries.

Through its user-friendly Graphical User Interface (GUI), the system collects details of the student academic performance as .xls file, the scanned signature , the photo etc. and outputs the Digitally Signed certificates with all the fields properly laid out.

TABLE OF CONTENTS

1. INTRODUCTION

With the rise of Digital India movement, it is crucial to try and move many of the existing system to a paper-less digital environment, and while adopting digital technology to sensitive and critical entities such as academic certificates, it is very important to ensure it's security and authenticity.

Cryptography is one best technology that has made giant effect in protecting data and information in recent years. It is the science of securing your information by means of a code. Cryptography provides an encryption for the data and information that passes via single/ multiple channels. This is done to keep the data from any external or third party influence.
Digital signature is one of the kinds of encrypting your signature that is specific to you and saves it from forgery of any kind.

A digital signature or e-signature for short is an electronic signature that can be utilized to authenticate the identity of the sender of a message or the signer of a document. A digital certificate contains the digital signature of the certificate-issuing authority so that anyone can verify that the certificate is real. This indeed is so commonly observed now in internet transactions .

A digital signature can be used with any kind of message, transactions and the like, whether it is encrypted or not, simply so that the receiver can be sure of the sender's identity and that the message arrived intact.

Hash value of a message when encrypted with the private key of a person is his digital signature on that e-Document. Digital Signature of a person therefore varies from document to document thus ensuring authenticity of each word of that document. As the public key of the signer is known, anybody can verify the message and the digital signature.

Paper Signatures vs Digital Signatures

These are the various major differences:

1. **Verification Method**
For a conventional signature, when the recipient receives a document, she compares the signature on the document with the signature on file. For a digital signature, the recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

2. **Relationship**
For a conventional signature, there is normally a one-to-many relationship between a signature and documents. For a digital signature, there is a one-to-one relationship between a signature and a message.

3. **Duplicity**
In conventional signature, a copy of the signed document can be distinguished from the original one on file. In digital signature, there is no such distinction unless there is a factor of time on the document.

Parameter	Paper	Electronic
Authenticity	May be forged	Cannot be copied
Integrity	Signature independent of the document	Signature depends on the contents of the document
Non-repudiation	a. Handwriting expert needed b. Error prone	a) Any computer user b) Error free

Table: Differences between Parameter and Electronic signatures.

How digital signature works?

A message encrypted with one element of the pair can be decrypted ONLY by the other element of the same pair. Two members of a pair are called keys, the Public Key & the Private Key. User himself generates his own key pair on his computer using Public Key Infrastructure(PKI).

Any message irrespective of its length can be compressed or abridged uniquely into a smaller length message called the Digest or the Hash. Smallest change in the message will change the Hash value.

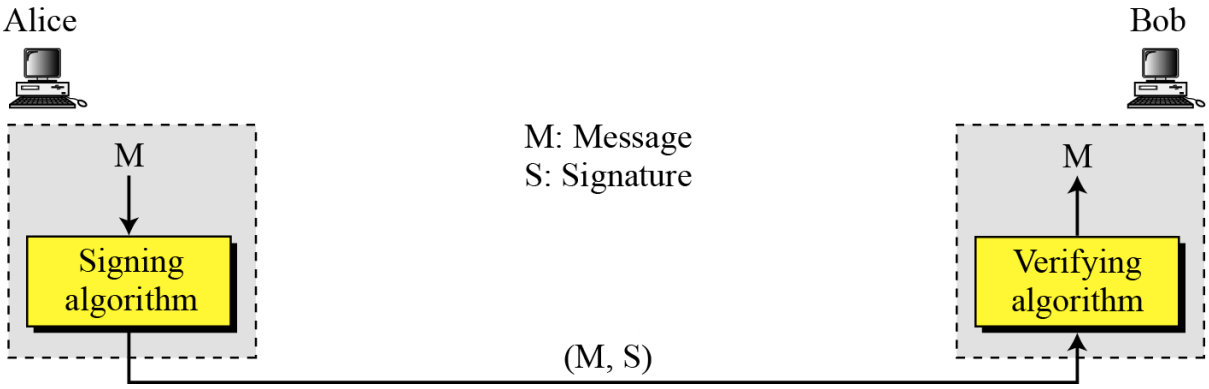


Fig: How digital Signatures work

- Each individual generates his own key pair
- [Public key known to everyone & Private Key only to the owner]
- Private Key – Used for making digital signature
- Public Key – Used to verify the digital signature

The signing and the verifying ceremony are done with the help of suitable algorithms.

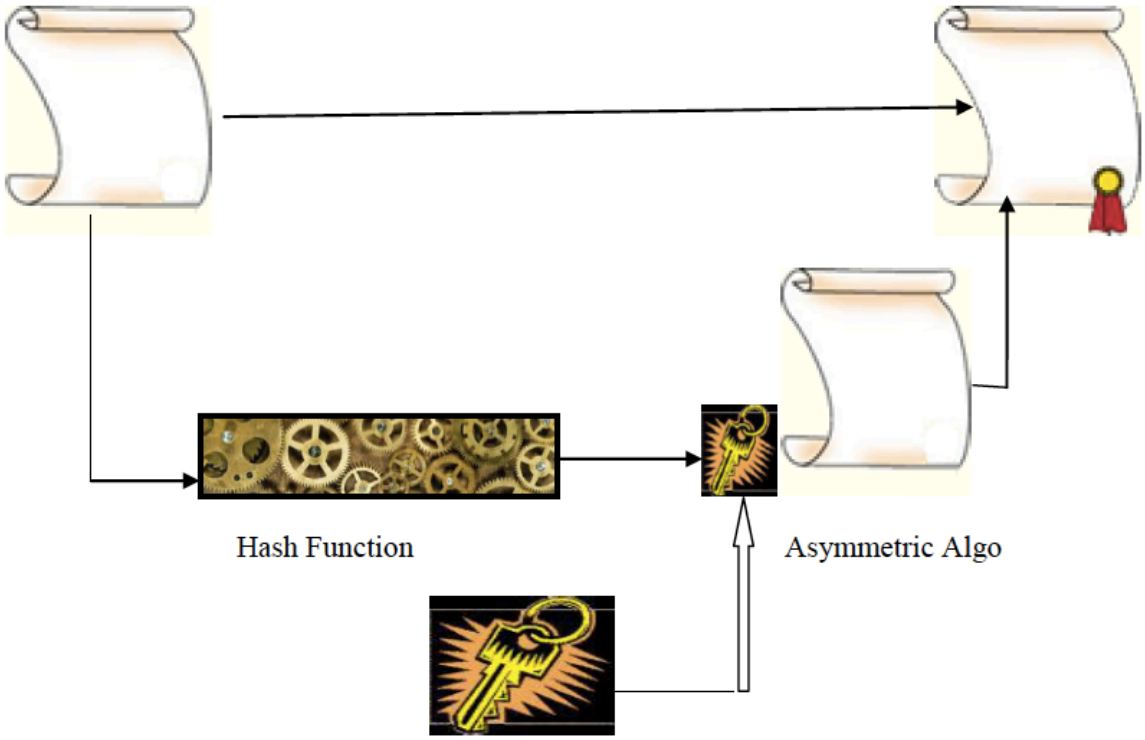


Fig: Digital Signature Mechanism

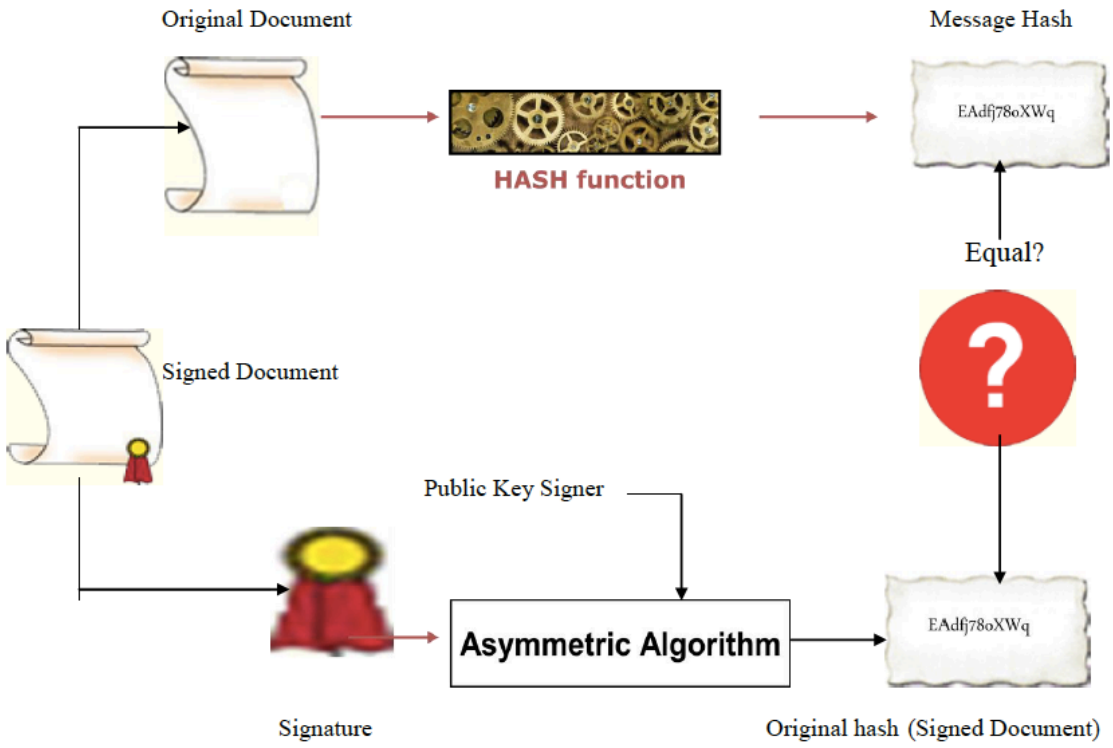


Fig: Digital Signing Verification

Services provided by Digital Signatures

a) To provide Authenticity, Integrity and Non-repudiation to electronic documents

b) To use the Internet as the safe and secure medium for e-Commerce and e-Governance

c) Providing accountability

d) Providing document integrity

e) Providing non-repudiation

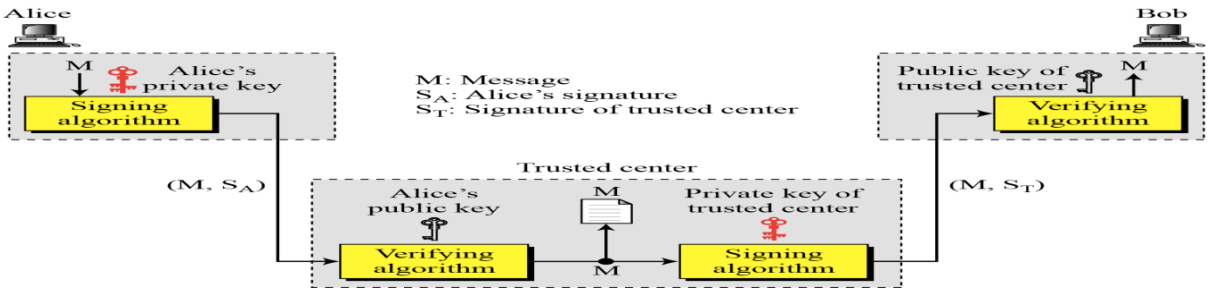
f) Providing satisfactory evidence of: Authorship, Approval, Review, and Authentication

g) Infrastructural pattern to be further profiled by domain specific groups (e-Prescribing, e-Referral).

Why use Digital Signatures in Academic Records

a) **Authentication and Integrity**: By placing a digital signature on any kind of document, especially one that requires it, shows the one who signed it is real and accepts responsibility for the signed document as being real and legal. A digital signature will also authenticate the document as being real and valid. It will prove to the executioner that the information contained on the document is valid, non-modified and can be put in action.

b) **Non-repudiation**: The digital signature also ensures that the sender cannot deny that he/ she is the source of the document and needs to accept full responsibility for it. Nonrepudiation can be provided using a trusted party (Certification Authority).



Digital Signatures and PKI

The signing algorithms and standards and a subset of Public Key Infrastructure(PKI) and the key generation, management, publishing and revocation is done using PKI.

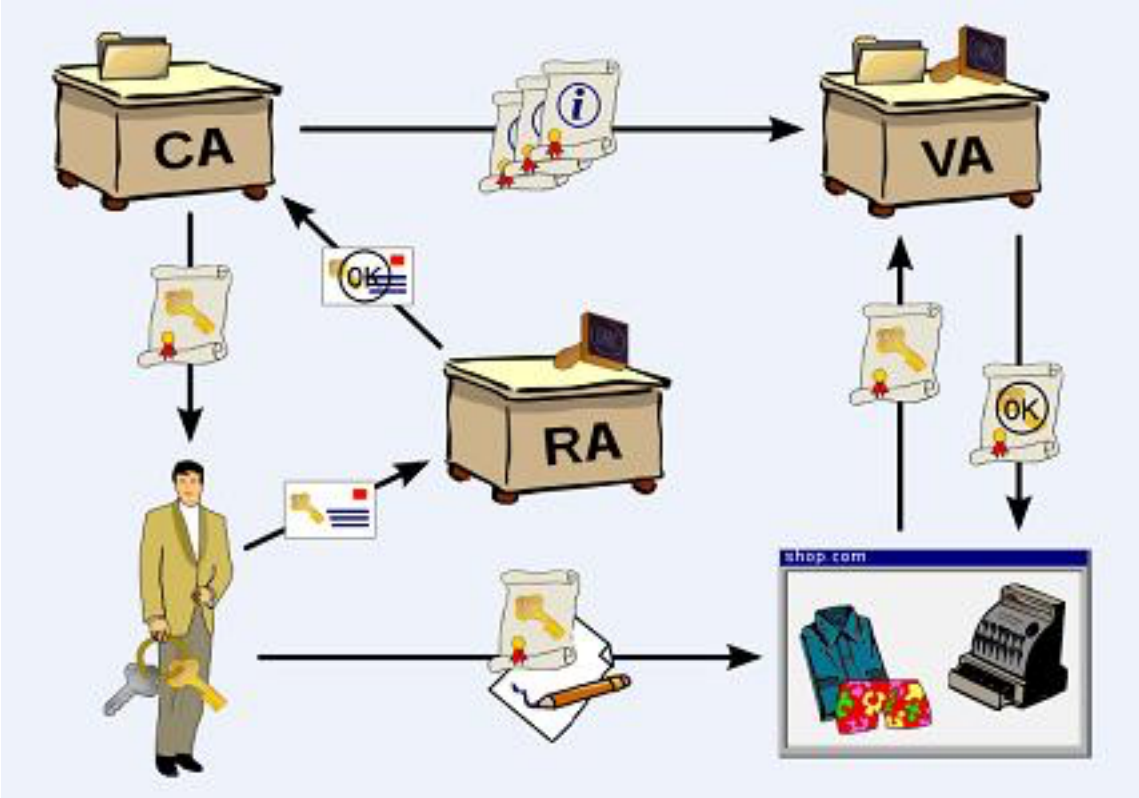


Fig. Certification Authority

A user applies for a certificate with his public key at a registration authority (RA). The latter confirms the user's identity to the certification authority (CA) which in turn issues the digital certificate(the authorized proof of his public key's validity). The user can then digitally sign a document using his new certificate. His identity is then checked by the verification party with a validation authority (VA) which again receives information about issued certificates by the certification authority.

ACT FOR DIGITAL SIGNATURE IN INDIA

The Information Technology Act, 2000 provides for the use of Digital Signatures on the documents submitted in electronic form. Under the provision of IT Act, 2000, the office of Controller of Certifying Authorities (CCA) appoints the Certifying Authorities (CA) by issuing Certificates for the same. These CA will issue the Digital Signature to the End Users Directly or through the Registration Authorities (RA) /Local Registration Authorities (LRA).

It must be obtained from an ISO 17090 compliant Certificate Authority.

Including the role extension for the signer's role in the healthcare profession.

For purposes of signature verification, the signer's certificate (public key portion) must be available.

How Digital Signature will be issued to the End-users:

Information Technology Act, 2000

Controller of Certifying Authorities (CCA)

Certifying Authorities (CA)

Registration Authorities (RA)

Local Registration Authorities (LRA)

End Users

The Certification Agencies available in India are:

- Tata Consultancy Services Ltd.
- National Informatics Centre
- Institute for Development & Research in Banking Technology (IDRBT)
- MTNL
- Customs & Central Excise (CBEC)
- Code Solutions Ltd., (A division of Gujarat Narmada Valley Fertilizers Company Ltd.)
- SafeScrypt from Sify Communications
- E Mudhra

2. SOFTWARE REQUIREMENT SPECIFICATIONS

2.1 System Environment

Presentation Layer	:	Tcl/Tk widgets
Languages	:	Python, Java
Operating Systems	:	Windows 7, Windows 8, Linux, Mac
Drivers	:	Aladdin/SafeNet eToken drivers
Cryptographic Tokens	:	SafeNet 5100 USB Token Authenticator
Libraries	:	PCKS#11, Python reportlab, iTextPDF and BouncyCastle Cryptographic Providers

2.2 FUNCTIONAL REQUIREMENTS

- User should be able to generate academic certificates using structured student data such as an Excel sheet(.xls).
- User should be able to generate digitally signed documents using legally valid digital certificates from a cryptographic token.
- User should have the capability to select appropriate certificate from the cryptographic token to sign the academic certificates.
- A user-friendly GUI(Graphical User Interface) to operate

2.3 NON FUNCTIONAL REQUIREMENTS

Reliability: The system should operate continuously with no or minimal failures.

Security: Private keys shall not be exportable and token password shall never be stored.

Human Factors: All menus must have a consistent format.

Ease of Installation: A user can install and operate the program with minimal assistance.

Speed: The system shall not take more than two seconds to generate a signed academic document.

2.4 MODULES

- User Interface module
- Provisional Certificate generation module
- Memorandum of Marks Generation module
- Consolidated Marks Memorandum module
- Certificate Signature module
- List Certificates module

User Interface module:

User interface module serves as a bridge between operator and other modules. It provides an user-friendly GUI to invoke the functionalities of other modules.

This module consists of following requirements:

- An introduction window
    - It contains a welcome message greeting the operator and also contains the organisational details.
    - It contains multiple buttons that serve as an entry point to GUI for each of the modules:
        - Provisional Certificates UI.
        - Consolidated Marks Memorandum UI.
        - Memorandum of Marks UI.
- Provisional Certificate Window
    - It acts as a form asking for input parameters to generate provisional certificates.
    - It contains the following fields:
        - Input data file field
        - Signing Private key alias
        - Cryptographic token password
        - Reason for digital signature
        - Location of signature
    - Input data field - This field asks for the path of input file that contains the information about students and their information for generating the certificates. It also has a “choose file” button that opens a File Dialog window to select the file from desired location so that path is autofilled.

        - Private Key alias - This fields asks for the unique identifier of a Private Key / Digital Certificate called alias, also called as container name(CN). This helps to identify the certificate that is used to generate the digital signature. It includes a “copy key” button that can be used to list all the aliases right now connected to the computer. This button acts as an invoker to call List Certificates module to display all available certificates. A new dialog shall open showing the available aliases from which you can copy-paste the required.
        - Cryptographic token password - The password needed to login to the USB token. The typed characters are hidden to enhance security.
        - Signing reason - An option field to enter the reason we are signing the documents.
        - Signing location - An optional field to specify the location the document is signed.
        - It also includes a button to display a sample input sheet so as to form a template for input file.
        - It includes a generate certificate button to create certificates.
- Memorandum of Marks Window
    - It acts as a form asking for input parameters to generate memorandums of marks.
    - It contains the following fields:
        - Input data file field
        - Signing Private key alias
        - Cryptographic token password
        - Reason for digital signature
        - Location of signature
    - Input data field - This field asks for the path of input file that contains the information about students and their information for generating the certificates. It also has a “choose file” button that opens a File Dialog window to select the file from desired location so that path is autofilled.
    - Private Key alias - This fields asks for the unique identifier of a Private Key / Digital Certificate called alias, also called as container name(CN). This helps to identify the certificate that is used to generate the digital signature. It includes a “copy key” button that can be used to list all the aliases right now connected to the computer. This button acts as an invoker to call List Certificates module to display all available certificates. A new dialog shall open showing the available aliases from which you can copy-paste the required.

    - Cryptographic token password - The password needed to login to the USB token. The typed characters are hidden to enhance security.
    - Signing reason - An option field to enter the reason we are signing the documents.
    - Signing location - An optional field to specify the location the document is signed.
    - It also includes a button to display a sample input sheet so as to form a template for input file.
    - It includes a generate certificate button to create certificates.
- Consolidated Marks Memorandum Window
    - It acts as a form asking for input parameters to generate consolidated marks memorandums.
    - It contains the following fields:
        - Input data file field
        - Signing Private key alias
        - Cryptographic token password
        - Reason for digital signature
        - Location of signature
    - Input data field - This field asks for the path of input file that contains the information about students and their information for generating the certificates. It also has a “choose file” button that opens a File Dialog window to select the file from desired location so that path is autofilled.
    - Private Key alias - This fields asks for the unique identifier of a Private Key / Digital Certificate called alias, also called as container name(CN). This helps to identify the certificate that is used to generate the digital signature. It includes a “copy key” button that can be used to list all the aliases right now connected to the computer. This button acts as an invoker to call List Certificates module to display all available certificates. A new dialog shall open showing the available aliases from which you can copy-paste the required.
    - Cryptographic token password - The password needed to login to the USB token. The typed characters are hidden to enhance security.
    - Signing reason - An option field to enter the reason we are signing the documents.
    - Signing location - An optional field to specify the location the document is signed.
    - It also includes a button to display a template for input file.
    - It includes a generate certificate button to create certificates.

Provisional Certificates module

This module is responsible for parsing the input data and generate provisional certificates. It parses the following information:

- Name of the student.
- Hall ticket number.
- Father's name and Mother's name.
- Gender.
- Percentage of marks obtained.
- Consolidated Marks Memorandum(CMM) number.
- Provisional Certificate(PC) number.
- Month and Year of Examination.

This module reads the above information and uses it determine the Grade, Class and Branch of the candidate to create Provisional Certificate.

It later utilizes the certificate signing module to sign each provisional certificate generated to digitally sign the document.

Memorandum of Marks module

This module is responsible for parsing the input data and generate provisional certificates. It parses the following information:

- Name of the student.
- Hall ticket number.
- Father's name and Mother's name.
- Gender.
- Subjects opted by the student.
- Marks obtained by student in each subject.
- Credits for each subject.
- Month and Year of Examination.
- Name of the Examination

This module reads the above information and uses it determine the Grade, Class, Percentage obtained, Credits received and Final status of the candidate.

It later utilizes the certificate signing module to sign each provisional certificate generated to digitally sign the document.

Consolidated Marks Memorandum module

This module is responsible for parsing the input data and generate provisional certificates. It parses the following information:

- Name of the student.
- Hall ticket number.
- Father's name and Mother's name.
- Gender.
- Subjects opted by the student.
- Marks obtained by student in each subject.
- Credits for each subject.

This module reads the above information and uses it determine the Grade, Class, Percentage obtained, Credits received and Final status of the candidate.

It later utilizes the certificate signing module to sign each provisional certificate generated to digitally sign the document.

Certificate Signing Module

This module is responsible for generating legally valid digital signatures using certificates securely stored in USB-base cryptographic eTokens. It shall support industry standard PKCS#11 keystores and support enterprise tokens such as SafeNet.

It solely handles the process of generating the digest, encrypting it with private key, adding the certificate chain, integrating the signature into PDF document, handling signature appearance and signature validation.

List Certificates Module

This is a small module intended to help user find aliases / container names that are used to uniquely identify the entities of a Token(Certificates, Keys, etc.).

3. DESIGN

3.1. UML Design

The unified modeling language is a standard language for specifying, Visualizing, Constructing and documenting the software system and its components. It is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed.

It is used to understand, design, configure, maintain and control Information about the systems.

3.1.1. Aims of Modeling

Models help us to visualize a system as it is or as we want to be

- Models permit us to specify the structure of system.
- Model gives us template that guides us in constructing system.
- Models can document the decisions we have made.
- Models can contain relationships among them.

3.1.2 Relationships

- Dependencies:

  Specifying a change in the specification of one thing may effect another thing, but not necessarily the reverse.

- Generalizations:

  A generalization is a relationship between a general thing and a more specific kind of that thing.

- Associations

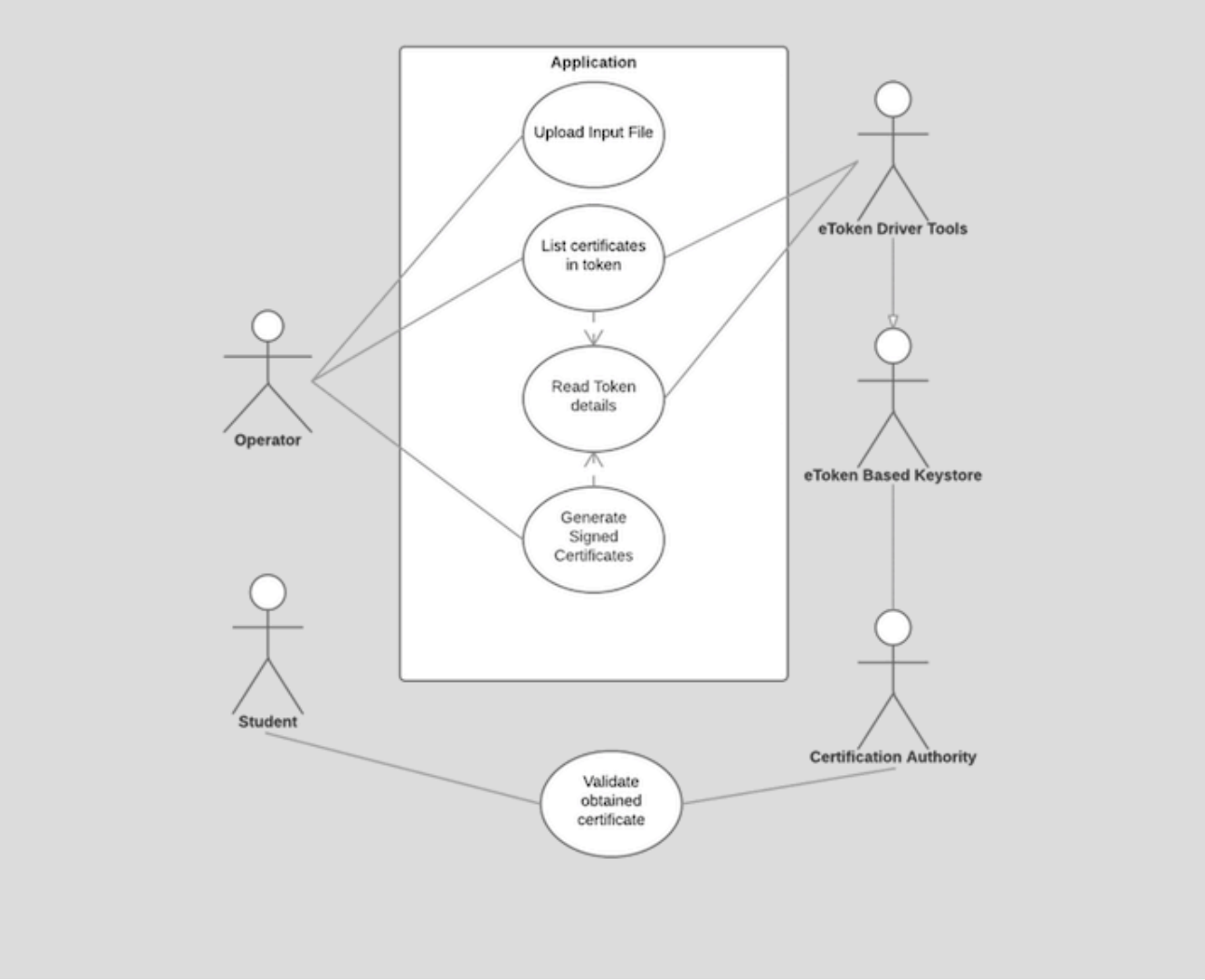
  An association is a structural relationship, specifying that objects of one thing are connected to object of another.

- Realizations

  A realization is a semantic relationship between classifiers in which one classifier specifies a contract that another classifier guarantees to carry out.

3.2 Use Case Diagram

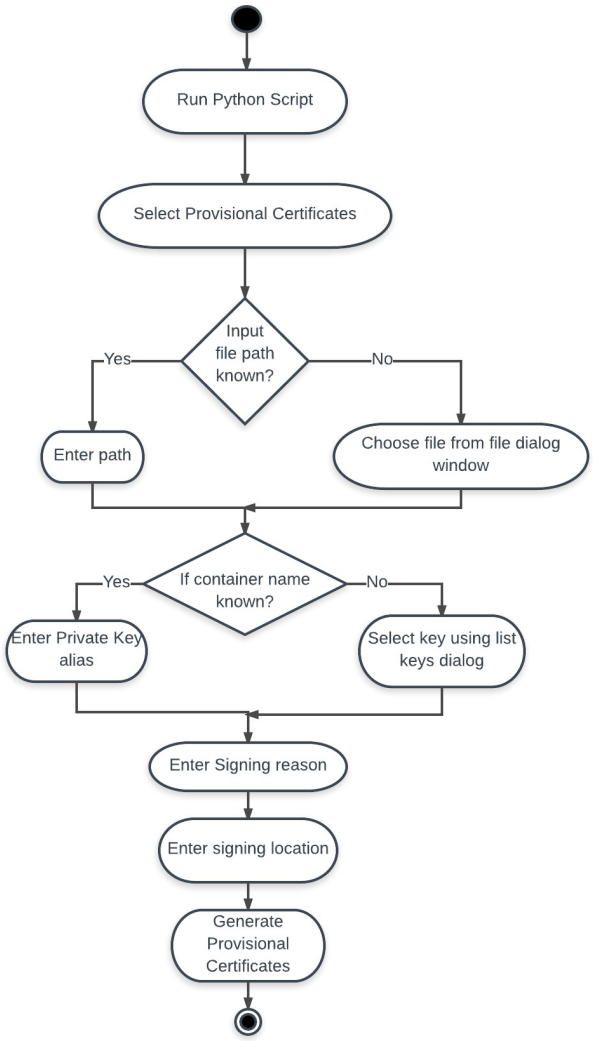
A use case diagram acts as a focus for the description of user requirements. It describes the relationships between requirements, users, and the major components. It does not describe the requirements in detail; these can be described in separate diagrams or in documents that can be linked to each use case.



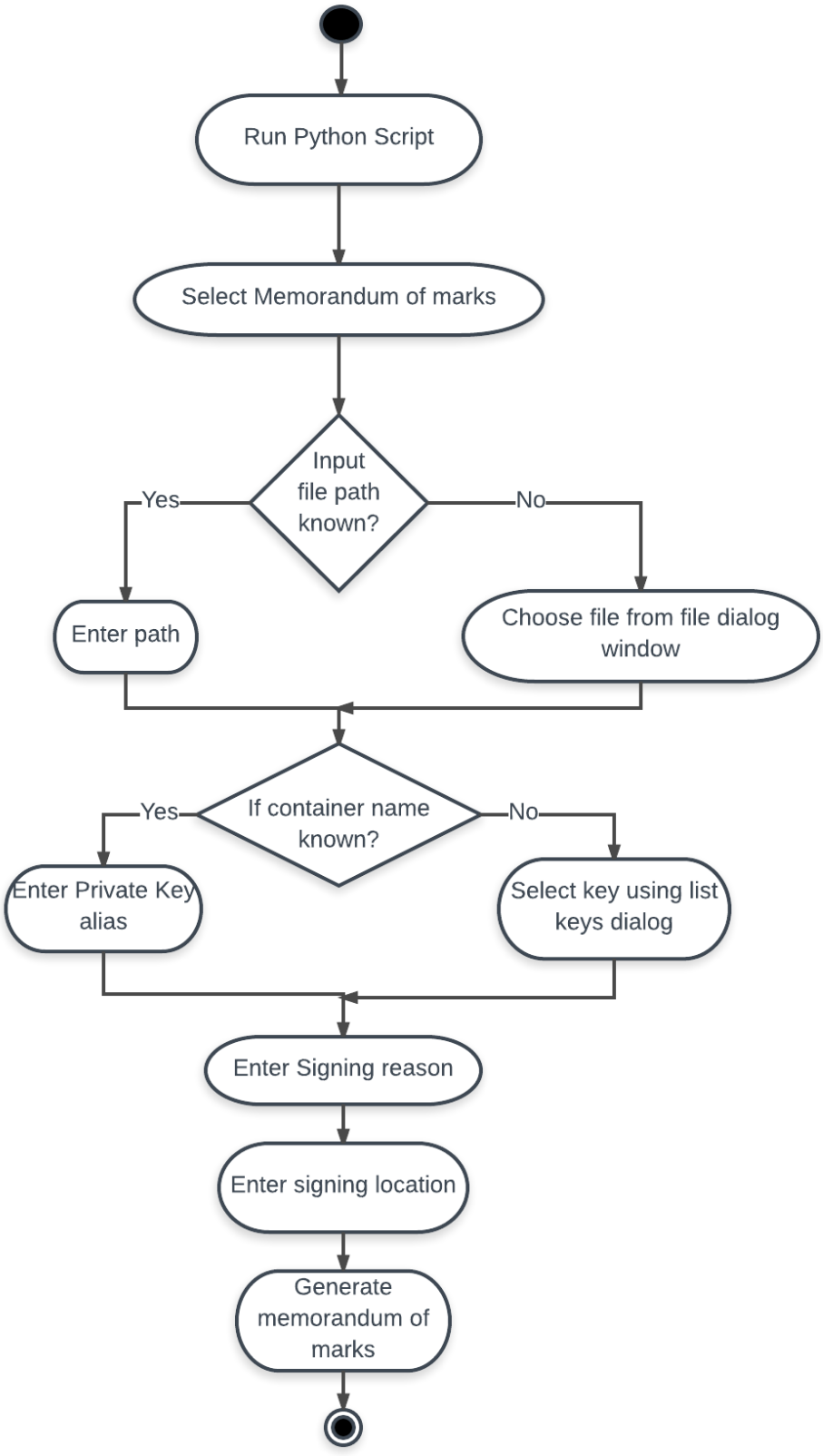
3.3 Activity Diagrams

Activity Diagrams shows the flow of control from activity to activity. It specifies dynamic aspects of a system. It is ongoing non atomic execution unit in a state machine .Activity diagrams contain action states, activity states, transitions, objects, notes and constraints. Action state is an atomic execution unit which may represent calling an operation on the object, send a signal to an object, computing an expression, creating and destroying an object.

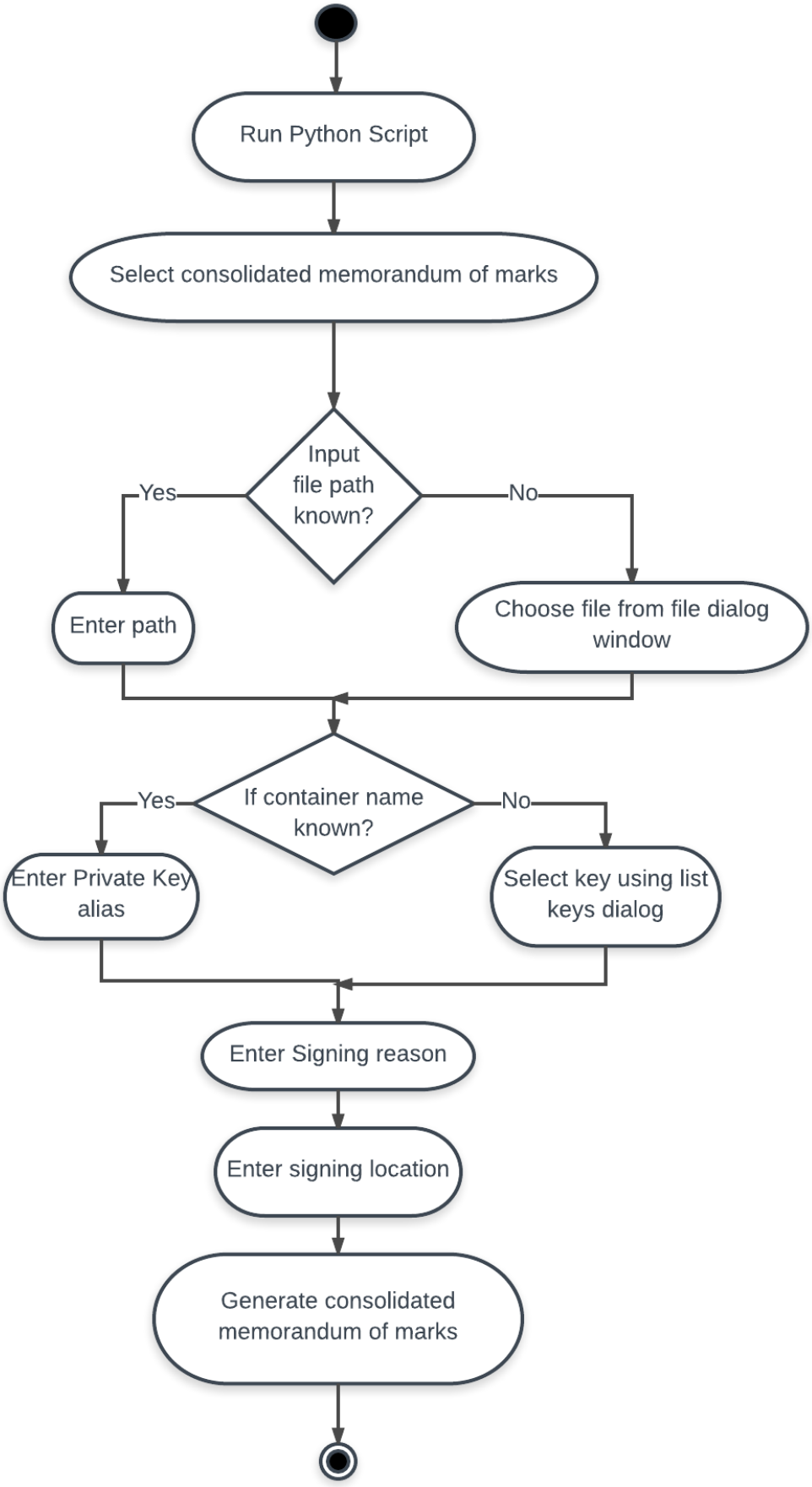
Activity Diagram for creating Provisional Certificates:



Activity Diagram for creating Memorandum of Marks:



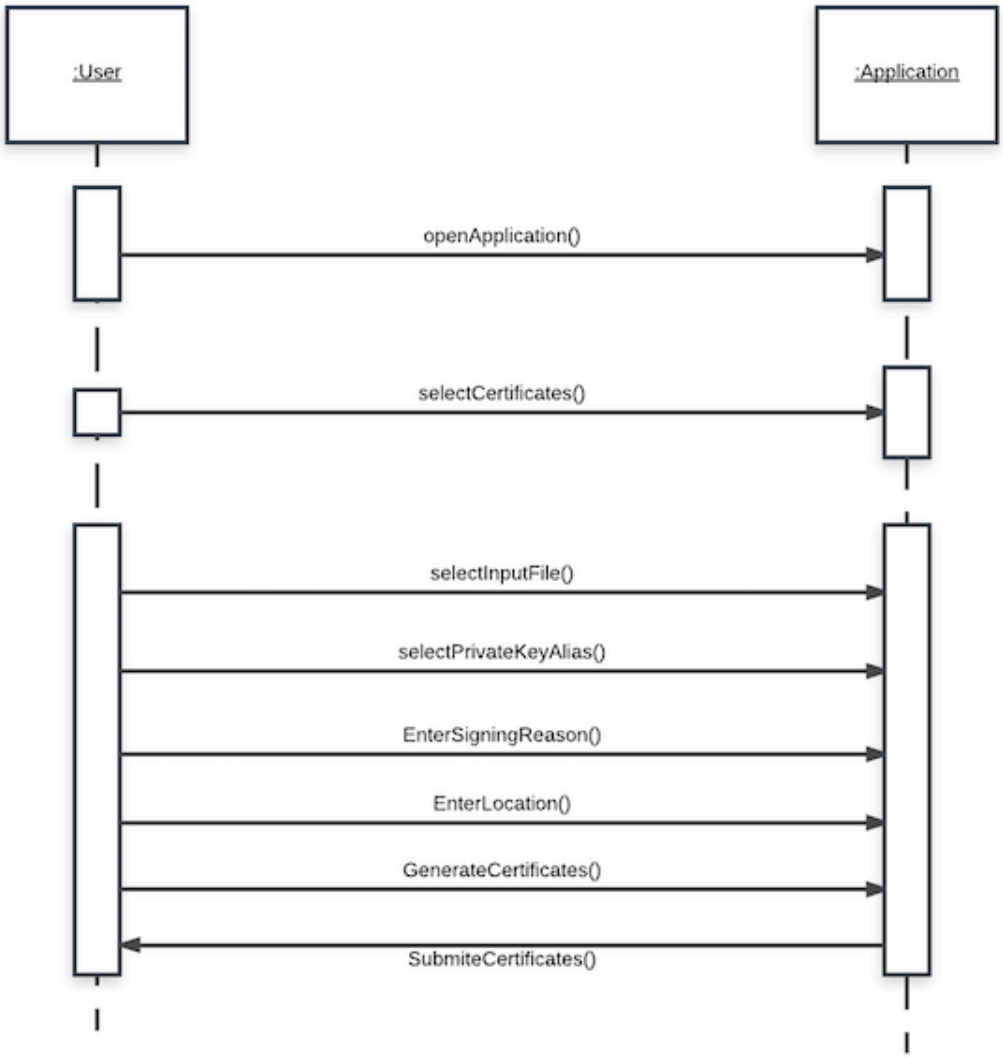
Activity Diagram for creating Memorandum of Marks:



3.4 Sequence Diagram

The purpose of the sequence diagram is to show the flow of the functionality through a use case. In other words, we call it mapping process in terms of data transfers from the actor through corresponding objects. The diagrams are read left to right and descending. A Sequence diagram is a model that describes how group of objects collaborate in some behaviour over time and capturing behaviour of a single use case.

Sequence diagram for operator:



4. IMPLEMENTATION

Gui.py

This program provides an user interface using Tcl/Tk widgets to interact with the system. This acts as the User Interface module of the system.

```
import Tkinter, Tkconstants, tkFileDialog, tkMessageBox

import provisional_certificate

from sys import platform


class MainWindow(Tkinter.Frame):


  def __init__(self, root):


    Tkinter.Frame.__init__(self, root)


    # options for buttons

    button_opt = {'fill': Tkconstants.BOTH, 'padx': 5, 'pady': 5}

    self.file_opt = {}

    self.file_opt['defaultextension'] = '.xls'

    # self.file_opt['filetypes'] = [('all files', '.*'), ('text files', '.txt'), ('Excel files', '.xls*')]

    self.file_opt['initialdir'] = '/Users/' if platform == "darwin" else 'C:\\Users\\'

    self.file_opt['initialfile'] = 'input.xls'

    self.file_opt['parent'] = root
```

```
    # define widgets


    Tkinter.Label(self, text="Input file path:").pack()

    self.filenameinput = Tkinter.Entry(self)

    self.filenameinput.pack()

    Tkinter.Button(self, text='Select Input file',
command=self.choose_input_file).pack(**button_opt)

    Tkinter.Label(self, text="Enter private key alias").pack()

    self.aliasinput = Tkinter.Entry(self)

    self.aliasinput.insert(Tkinter.END, 'le-9f8bfd35-9ccd-41d2-b396-b25def9e02b3')

    self.aliasinput.pack()

    Tkinter.Label(self, text="Enter token password").pack()

    self.passwordinput = Tkinter.Entry(self, show="*")

    self.passwordinput.pack()

    Tkinter.Label(self, text="Enter signing reason").pack()

    self.reasoninput = Tkinter.Entry(self)

    self.reasoninput.pack()

    Tkinter.Label(self, text="Enter signing location").pack()

    self.locationinput = Tkinter.Entry(self)

    self.locationinput.pack()
```

```
    Tkinter.Button(self, text='Create Priovisional Certificates',
command=self.create_provisional_certificates).pack(**button_opt)

    # define options for opening or saving a file


  def choose_input_file(self):


    """Returns an opened file in read mode.

    This time the dialog just returns a filename and the file is opened by your own code.

    """

    filename = tkFileDialog.askopenfilename(**self.file_opt)

    self.filenameinput.delete(0, Tkinter.END)

    self.filenameinput.insert(0,filename)

    return


  def create_provisional_certificates(self):

    inputfilename = self.filenameinput.get()

    alias = self.aliasinput.get()

    reason = self.reasoninput.get()

    location = self.locationinput.get()

    password = self.passwordinput.get()

    res = provisional_certificate.main(['create_sign_script', inputfilename, password, alias,
reason, location])
```

```
    if res == 0:
        tkMessageBox.showinfo("Success!", "Successfully created signed provisional
certificates.")
    else:
        tkMessageBox.showinfo("Certificate creation failed. Stack trace: " + res)
    return




if __name__=='__main__':
  root = Tkinter.Tk()
  MainWindow(root).pack()
  root.mainloop()
```

Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with the standard Microsoft Windows and Mac OS X install of Python.

The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5.[4][5] This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter).

There are several popular GUI library alternatives available, such as wxPython, PyQt (PySide), Pygame, Pyglet, and PyGTK.

Tkinter is free software released under a Python license.

provisional_certificates.py

This is used for implementing the provisional certificate module.
It is responsible for parsing the input data and generate provisional certificates.

It uses Reportlab package to generate PDF objects and generate a complete PDF as a certificate.

It collects Name of the student, Hall ticket number, Father's name and Mother's name, Gender, Percentage of marks obtained, Consolidated Marks Memorandum(CMM) number, Provisional Certificate(PC) number, Month and Year of Examination and calculates criteria such as Class, Grade, Status, etc. to be presented in the Provisional Certificate.

It then calls the Certificate Signing module to generate digitally signed provisional certificates.

```
from xlrd import open_workbook
from reportlab.pdfgen import canvas
from reportlab.platypus import Paragraph
from reportlab.lib.styles import ParagraphStyle
from reportlab.lib.units import inch
from reportlab.lib.enums import TA_JUSTIFY
from reportlab.lib.colors import black
from reportlab.lib.pagesizes import letter, A4
import os
import sys
from datetime import date

def main(argv):
        if len(argv) > 1:
                filename = argv[1]
        else:
```

```
            filename = "input_formats/provisional_certificate.xls"
        wb = open_workbook(filename=filename, on_demand=True)
        sheet = wb.sheet_by_index(0)
        for row_number in xrange(1,sheet.nrows):
            htno = sheet.cell(row_number, 1).value
            name = sheet.cell(row_number, 2).value
            father_name = sheet.cell(row_number, 3).value
            mother_name = sheet.cell(row_number, 4).value
            month_of_passing = sheet.cell(row_number, 5).value
            year_of_passing = sheet.cell(row_number, 6).value
            percentage = sheet.cell(row_number, 7).value
            branch_code = sheet.cell(row_number, 10).value
            pc_track_no = sheet.cell(row_number, 11).value
            gender = sheet.cell(row_number, 13).value
            cmm_track_no = sheet.cell(row_number, 15).value
            class_string = "PASS CLASS"
            if percentage >= 70:
                class_string = "FIRST CLASS WITH DISTINCTION"
            elif percentage >= 60:
                class_string = "FIRST CLASS"
            elif percentage >= 50:
                class_string = "SECOND CLASS"
            cnv = canvas.Canvas("output/" + name + "_pc.pdf", pagesize=A4)
            cnv.drawImage('templates/provisional_certificate.jpg', 0, 0, height = A4[1],
width = A4[0])
            cnv.drawString(1.2 * inch, 10.25 * inch, pc_track_no)
            cnv.drawString(5.6 * inch, 9.85 * inch, htno)
            cnv.drawString(5.6 * inch, 9.4 * inch, cmm_track_no)
            cnv.drawString(1.25 * inch, 0.45 * inch, date.today().strftime('%d, %b %Y'))
            style = ParagraphStyle(
        'Body Text',
        fontName='Helvetica',
        fontSize=20,
```

```
        leading=24,
        leftIndent=0,
        rightIndent=0,
        firstLineIndent=0,
        alignment=TA_JUSTIFY,
        spaceBefore=0,
        spaceAfter=0,
        bulletFontName='Helvetica',
        bulletFontSize=10,
        bulletIndent=0,
        textColor= black,
        backColor=None,
        wordWrap=None,
        borderWidth= 0,
        borderPadding= 0,
        borderColor= None,
        borderRadius= None,
        allowWidows= 1,
        allowOrphans= 0,
        textTransform=None,
        endDots=None,
        splitLongWords=1,
    )
            p = Paragraph("This is to certify that <b>" + ("Mr. " if gender == "M" else
"Ms. ") + name + "</b><br/>son of <b>Mr. " + father_name + "</b> and <b>Mrs. " +
mother_name + "</b><br/>passed <b>B.TECH. Computer Science and Engineering</b>
examination of the JNTUH, Hyderabad, held in " + month_of_passing + ", " +
year_of_passing + "<br/>placed in <b>" + class_string + "</b>.<br/>Student has satisfied all
the requirements for the award of the degree.",style)
            p.wrap(7.4 * inch, 4.5 * inch)
            p.drawOn(cnv, 0.5 * inch, 5 * inch)
            cnv.showPage()
            cnv.save()
```

```
        if len(argv) == 5:
            sign_password = argv[2]
            sign_location = argv[3]
            sign_reason = argv[4]
            alias = argv[5]
        else:
            sign_password = "rupee@123"
            sign_location = "CVR College"
            sign_reason = "Authentication and Non-repudation"
            alias = "le-9f8bfd35-9ccd-41d2-b396-b25def9e02b3"
        if sys.platform == "darwin":
            res = os.system("java -jar java/SignWithUSB.jar \"output/" + name +
"_pc.pdf\" \"output/" + name + "_pc_signed.pdf\" /usr/local/lib/libeTPkcs11.dylib
\""+sign_password+"\" \""+alias+"\" \"" + sign_reason + "\" \"" + sign_location + "\"")
        else:
            res = os.system("java -jar \"java\\SignWithUSB.jar\" \"output\\" +
name + "_pc.pdf\" \"output\\" + name + "_pc_signed.pdf\" \"C:\\Windows\\System\
\eTPkcs11.dll\" \""+sign_password+"\" \""+alias+"\" \"" + sign_reason + "\" \"" +
sign_location + "\"")
    return res


if __name__ == "__main__":
    main(sys.argv)
```

Reportlab

ReportLab is the time-proven, ultra-robust open-source engine for creating complex, data-driven PDF documents and custom vector graphics. It's free, open-source , and written in Python. The package sees 50,000+ downloads per month, is part of standard Linux distributions, is embedded in many products, and was selected to power the print/export feature for Wikipedia.

The ReportLab Toolkit has evolved over the years in direct response to the real-world reporting needs of large institutions. The library implements three main layers:

- A graphics canvas API that 'draws' PDF pages
- A charts and widgets library for creating reusable data graphics.
- A page layout engine - PLATYPUS ("Page Layout and TYPography Using Scripts") - which builds documents from elements such as headlines, paragraphs, fonts, tables and vector graphics.

SignWithPKCS11USB.java

This is the Certificate signing module that uses Java's Cryptographic libraries, Sun's Security Libraries, Bouncy Castle's Cryptographic Provider's and iTextPDF's PDF modifying libraries.

```
import java.io.ByteArrayInputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.security.GeneralSecurityException;

import java.security.KeyStore;

import java.security.PrivateKey;

import java.security.Provider;

import java.security.Security;

import java.security.cert.Certificate;

import java.security.cert.X509Certificate;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import java.util.Enumeration;

import java.util.Properties;


import org.bouncycastle.jce.provider.BouncyCastleProvider;
```

```
import sun.security.pkcs11.SunPKCS11;

import sun.security.pkcs11.wrapper.CK_C_INITIALIZE_ARGS;

import sun.security.pkcs11.wrapper.CK_TOKEN_INFO;

import sun.security.pkcs11.wrapper.PKCS11;

import sun.security.pkcs11.wrapper.PKCS11Exception;



import com.itextpdf.text.DocumentException;

import com.itextpdf.text.Rectangle;

import com.itextpdf.text.log.LoggerFactory;

import com.itextpdf.text.log.SysoLogger;

import com.itextpdf.text.pdf.PdfReader;

import com.itextpdf.text.pdf.PdfSignatureAppearance;

import com.itextpdf.text.pdf.PdfStamper;

import com.itextpdf.text.pdf.security.BouncyCastleDigest;

import com.itextpdf.text.pdf.security.CertificateUtil;

import com.itextpdf.text.pdf.security.CrlClient;

import com.itextpdf.text.pdf.security.CrlClientOnline;

import com.itextpdf.text.pdf.security.DigestAlgorithms;

import com.itextpdf.text.pdf.security.ExternalDigest;

import com.itextpdf.text.pdf.security.ExternalSignature;
```

```
import com.itextpdf.text.pdf.security.MakeSignature;

import com.itextpdf.text.pdf.security.OcspClient;

import com.itextpdf.text.pdf.security.OcspClientBouncyCastle;

import com.itextpdf.text.pdf.security.PrivateKeySignature;

import com.itextpdf.text.pdf.security.TSAClient;

import com.itextpdf.text.pdf.security.TSAClientBouncyCastle;

import com.itextpdf.text.pdf.security.MakeSignature.CryptoStandard;


public class SignWithPKCS11USB {


    public static void main(String[] args) throws IOException, GeneralSecurityException,
DocumentException {

        LoggerFactory.getInstance().setLogger(new SysoLogger());

        String src = args[0];

        String dest = args[1];

        String DLL = args[2];

        char[] pass = args[3].toCharArray();

        String alias = args[4];

        String reason = args[5];

        String location = args[6];

        long slotsWithTokens[] = new long[10];

        slotsWithTokens = getSlotsWithTokens(DLL);
```

```
String config = "name=safenet5100\n" +
"library=" + DLL + "\n" +
"slotListIndex = " + slotsWithTokens[0];
ByteArrayInputStream bais = new ByteArrayInputStream(config.getBytes());
Provider providerPKCS11 = new SunPKCS11(bais);
Security.addProvider(providerPKCS11);
System.out.println(providerPKCS11.getName());
BouncyCastleProvider providerBC = new BouncyCastleProvider();
Security.addProvider(providerBC);
KeyStore ks = KeyStore.getInstance("PKCS11");
ks.load(null, pass);
PrivateKey pk = (PrivateKey)ks.getKey(alias, pass);
Certificate[] chain = ks.getCertificateChain(alias);
// Creating the reader and the stamper
PdfReader reader = new PdfReader(src);
FileOutputStream os = new FileOutputStream(dest);
PdfStamper stamper = PdfStamper.createSignature(reader, os, '\0');
// Creating the appearance
PdfSignatureAppearance appearance = stamper.getSignatureAppearance();
appearance.setReason(reason);
appearance.setLocation(location);
appearance.setVisibleSignature(new Rectangle(36, 748, 144, 780), 1, "signature");
```

```
        // Creating the signature
        ExternalSignature pks = new PrivateKeySignature(pk, DigestAlgorithms.SHA256,
providerPKCS11.getName());
        ExternalDigest digest = new BouncyCastleDigest();
        MakeSignature.signDetached(appearance, digest, pks, chain, null, null, null, 0,
CryptoStandard.CMS);
    }


    public static long[] getSlotsWithTokens(String libraryPath) throws IOException {
        CK_C_INITIALIZE_ARGS initArgs = new CK_C_INITIALIZE_ARGS();
        String functionList = "C_GetFunctionList";

        initArgs.flags = 0;
        PKCS11 tmpPKCS11 = null;
        long[] slotList = null;
        try {
            try {
                tmpPKCS11 = PKCS11.getInstance(libraryPath, functionList, initArgs, false);
            } catch (IOException ex) {
                ex.printStackTrace();
                throw ex;
```

```
        }
    } catch (PKCS11Exception e) {
        try {
            initArgs = null;
            tmpPKCS11 = PKCS11.getInstance(libraryPath, functionList, initArgs, true);
        } catch (IOException ex) {
         ex.printStackTrace();
     } catch (PKCS11Exception ex) {
         ex.printStackTrace();
     }
}

try {
    slotList = tmpPKCS11.C_GetSlotList(true);

    for (long slot : slotList){
        CK_TOKEN_INFO tokenInfo = tmpPKCS11.C_GetTokenInfo(slot);
        System.out.println("slot: "+slot+"\nmanufacturerID: "
            + String.valueOf(tokenInfo.manufacturerID) + "\nmodel: "
            + String.valueOf(tokenInfo.model));
    }
} catch (PKCS11Exception ex) {
```

```
            ex.printStackTrace();
        } catch (Throwable t) {
            t.printStackTrace();
        }

        return slotList;

        }
}
```

ListAliases.java

It is the implementation for List Private key module used to show the available certificates in the token.

```
import java.io.ByteArrayInputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.security.GeneralSecurityException;

import java.security.KeyStore;

import java.security.PrivateKey;

import java.security.Provider;

import java.security.Security;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import java.util.Enumeration;

import java.util.Properties;


import org.bouncycastle.jce.provider.BouncyCastleProvider;


import sun.security.pkcs11.SunPKCS11;

import sun.security.pkcs11.wrapper.CK_C_INITIALIZE_ARGS;
```

```
import sun.security.pkcs11.wrapper.CK_TOKEN_INFO;

import sun.security.pkcs11.wrapper.PKCS11;

import sun.security.pkcs11.wrapper.PKCS11Exception;


public class SignWithPKCS11USB {


    public static void main(String[] args) throws IOException, GeneralSecurityException,
DocumentException {

        LoggerFactory.getInstance().setLogger(new SysoLogger());

        String src = args[0];

        String dest = args[1];

        String DLL = args[2];

        char[] pass = args[3].toCharArray();

        String alias = args[4];

        String reason = args[5];

        String location = args[6];

        long slotsWithTokens[] = new long[10];

        slotsWithTokens = getSlotsWithTokens(DLL);

        String config = "name=safenet5100\n" +

        "library=" + DLL + "\n" +

        "slotListIndex = " + slotsWithTokens[0];

        ByteArrayInputStream bais = new ByteArrayInputStream(config.getBytes());
```

```
    Provider providerPKCS11 = new SunPKCS11(bais);
    Security.addProvider(providerPKCS11);
    System.out.println(providerPKCS11.getName());
    BouncyCastleProvider providerBC = new BouncyCastleProvider();
    Security.addProvider(providerBC);
    KeyStore ks = KeyStore.getInstance("PKCS11");
    ks.load(null, pass);
    Enumeration<String> aliases = ks.aliases();
    while (aliases.hasMoreElements()) {
        System.out.println((String)aliases.nextElement() + "\n");
    }
    String alias = (String)ks.aliases().nextElement();
}



public static long[] getSlotsWithTokens(String libraryPath) throws IOException {
    CK_C_INITIALIZE_ARGS initArgs = new CK_C_INITIALIZE_ARGS();
    String functionList = "C_GetFunctionList";

    initArgs.flags = 0;
    PKCS11 tmpPKCS11 = null;
    long[] slotList = null;
```

```
  try {
    try {
      tmpPKCS11 = PKCS11.getInstance(libraryPath, functionList, initArgs, false);
    } catch (IOException ex) {
      ex.printStackTrace();
      throw ex;
    }
  } catch (PKCS11Exception e) {
    try {
      initArgs = null;
      tmpPKCS11 = PKCS11.getInstance(libraryPath, functionList, initArgs, true);
    } catch (IOException ex) {
     ex.printStackTrace();
   } catch (PKCS11Exception ex) {
     ex.printStackTrace();
   }
}

try {
  slotList = tmpPKCS11.C_GetSlotList(true);

  for (long slot : slotList){
```

```
            CK_TOKEN_INFO tokenInfo = tmpPKCS11.C_GetTokenInfo(slot);
            System.out.println("slot: "+slot+"\nmanufacturerID: "
                + String.valueOf(tokenInfo.manufacturerID) + "\nmodel: "
                + String.valueOf(tokenInfo.model));
        }
    } catch (PKCS11Exception ex) {
        ex.printStackTrace();
    } catch (Throwable t) {
        t.printStackTrace();
    }

    return slotList;

    }
}
```

iTextPDF

iText is an open source library for creating and manipulating PDF files in Java.

iText was written by Bruno Lowagie, Paulo Soares and others. The source code was initially distributed under the Mozilla Public License or the LGPL open source licenses. However, as of version 5.0.0 (released Dec 7, 2009) it is distributed under the Affero General Public License version 3. A fork of the LGPL/MPL licensed version of iText is currently actively maintained as the OpenPDF library on GitHub.[2] iText is also available through a proprietary license, distributed by iText Software Corp.

iText provides support for most advanced PDF features such as PKI-based signatures, 40-bit and 128-bit encryption, color correction, Tagged PDF, PDF forms (AcroForms), PDF/X, color management via ICC profiles and barcodes, and is used by several products and services, including Eclipse BIRT, Jasper Reports, Red Hat JBoss Seam, Windward Reports, and pdftk

iText adheres to most modern day PDF standards, including:

- ISO 32000-1 (PDF 1.7)
- ISO 19005 (PDF/A)
- ISO 14289 (PDF/UA)

Watch PDF and Standards, a talk by Adobe's PDF architect Leonard Rosenthol at the iText Summit in 2012.

MakeSignature.java

```
package com.itextpdf.text.pdf.security;


import com.itextpdf.text.DocumentException;
import com.itextpdf.text.io.RASInputStream;
import com.itextpdf.text.io.RandomAccessSource;
import com.itextpdf.text.io.RandomAccessSourceFactory;
import com.itextpdf.text.io.StreamUtil;
import com.itextpdf.text.log.Logger;
import com.itextpdf.text.log.LoggerFactory;
import com.itextpdf.text.pdf.AcroFields;
import com.itextpdf.text.pdf.ByteBuffer;
import com.itextpdf.text.pdf.PdfArray;
import com.itextpdf.text.pdf.PdfDate;
import com.itextpdf.text.pdf.PdfDeveloperExtension;
import com.itextpdf.text.pdf.PdfDictionary;
import com.itextpdf.text.pdf.PdfName;
import com.itextpdf.text.pdf.PdfReader;
import com.itextpdf.text.pdf.PdfSignature;
import com.itextpdf.text.pdf.PdfSignatureAppearance;
import com.itextpdf.text.pdf.PdfString;
import org.bouncycastle.asn1.esf.SignaturePolicyIdentifier;


import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.GeneralSecurityException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.Certificate;
import java.security.cert.X509Certificate;
import java.util.*;


/**
 * Class that signs your PDF.
 * @author Paulo Soares
 */
public class MakeSignature {


        /** The Logger instance. */
    private static final Logger LOGGER = LoggerFactory.getLogger(MakeSignature.class);
```

```
    public enum CryptoStandard {
            CMS, CADES
    }


    public static void signDetached(PdfSignatureAppearance sap, ExternalDigest externalDigest,
ExternalSignature externalSignature, Certificate[] chain, Collection<CrlClient> crlList, OcspClient ocspClient,
                                TSAClient tsaClient, int estimatedSize, CryptoStandard sigtype) throws IOException,
DocumentException, GeneralSecurityException {
        signDetached(sap, externalDigest, externalSignature, chain, crlList, ocspClient, tsaClient, estimatedSize,
sigtype, (SignaturePolicyIdentifier) null);
    }


    public static void signDetached(PdfSignatureAppearance sap, ExternalDigest externalDigest,
ExternalSignature externalSignature, Certificate[] chain, Collection<CrlClient> crlList, OcspClient ocspClient,
                                TSAClient tsaClient, int estimatedSize, CryptoStandard sigtype, SignaturePolicyInfo
signaturePolicy) throws IOException, DocumentException, GeneralSecurityException {
        signDetached(sap, externalDigest, externalSignature, chain, crlList, ocspClient, tsaClient, estimatedSize,
sigtype, signaturePolicy.toSignaturePolicyIdentifier());
    }


     */
    public static void signDetached(PdfSignatureAppearance sap, ExternalDigest externalDigest,
ExternalSignature externalSignature, Certificate[] chain, Collection<CrlClient> crlList, OcspClient ocspClient,
                                TSAClient tsaClient, int estimatedSize, CryptoStandard sigtype, SignaturePolicyIdentifier
signaturePolicy) throws IOException, DocumentException, GeneralSecurityException {
        Collection<byte[]> crlBytes = null;
        int i = 0;
        while (crlBytes == null && i < chain.length)
            crlBytes = processCrl(chain[i++], crlList);
            if (estimatedSize == 0) {
            estimatedSize = 8192;
            if (crlBytes != null) {
                for (byte[] element : crlBytes) {
                    estimatedSize += element.length + 10;
                }
            }
            if (ocspClient != null)
                estimatedSize += 4192;
            if (tsaClient != null)
                estimatedSize += 4192;
        }
        sap.setCertificate(chain[0]);
        if (sigtype == CryptoStandard.CADES) {
            sap.addDeveloperExtension(PdfDeveloperExtension.ESIC_1_7_EXTENSIONLEVEL2);
        }
```

```
        PdfSignature dic = new PdfSignature(PdfName.ADOBE_PPKLITE, sigtype == CryptoStandard.CADES ?
PdfName.ETSI_CADES_DETACHED : PdfName.ADBE_PKCS7_DETACHED);
        dic.setReason(sap.getReason());
        dic.setLocation(sap.getLocation());
        dic.setSignatureCreator(sap.getSignatureCreator());
        dic.setContact(sap.getContact());
        dic.setDate(new PdfDate(sap.getSignDate())); // time-stamp will over-rule this
        sap.setCryptoDictionary(dic);


        HashMap<PdfName, Integer> exc = new HashMap<PdfName, Integer>();
        exc.put(PdfName.CONTENTS, new Integer(estimatedSize * 2 + 2));
        sap.preClose(exc);


        String hashAlgorithm = externalSignature.getHashAlgorithm();
        PdfPKCS7 sgn = new PdfPKCS7(null, chain, hashAlgorithm, null, externalDigest, false);
        if (signaturePolicy != null) {
            sgn.setSignaturePolicy(signaturePolicy);
        }
        InputStream data = sap.getRangeStream();
        byte hash[] = DigestAlgorithms.digest(data, externalDigest.getMessageDigest(hashAlgorithm));
        byte[] ocsp = null;
        if (chain.length >= 2 && ocspClient != null) {
            ocsp = ocspClient.getEncoded((X509Certificate) chain[0], (X509Certificate) chain[1], null);
        }
        byte[] sh = sgn.getAuthenticatedAttributeBytes(hash, ocsp, crlBytes, sigtype);
        byte[] extSignature = externalSignature.sign(sh);
        sgn.setExternalDigest(extSignature, null, externalSignature.getEncryptionAlgorithm());


        byte[] encodedSig = sgn.getEncodedPKCS7(hash, tsaClient, ocsp, crlBytes, sigtype);


        if (estimatedSize < encodedSig.length)
            throw new IOException("Not enough space");


        byte[] paddedSig = new byte[estimatedSize];
        System.arraycopy(encodedSig, 0, paddedSig, 0, encodedSig.length);


        PdfDictionary dic2 = new PdfDictionary();
        dic2.put(PdfName.CONTENTS, new PdfString(paddedSig).setHexWriting(true));
        sap.close(dic2);
    }
```

```
public static Collection<byte[]> processCrl(Certificate cert, Collection<CrlClient> crlList) {
    if (crlList == null)
        return null;
    ArrayList<byte[]> crlBytes = new ArrayList<byte[]>();
    for (CrlClient cc : crlList) {
        if (cc == null)
            continue;
        LOGGER.info("Processing " + cc.getClass().getName());
        Collection<byte[]> b = cc.getEncoded((X509Certificate)cert, null);
        if (b == null)
            continue;
        crlBytes.addAll(b);
    }
    if (crlBytes.isEmpty())
        return null;
    else
        return crlBytes;
}
```

```
public static void signExternalContainer(PdfSignatureAppearance sap, ExternalSignatureContainer externalSignatureContainer, int estimatedSize) throws GeneralSecurityException, IOException, DocumentException {
    PdfSignature dic = new PdfSignature(null, null);
    dic.setReason(sap.getReason());
    dic.setLocation(sap.getLocation());
    dic.setSignatureCreator(sap.getSignatureCreator());
    dic.setContact(sap.getContact());
    dic.setDate(new PdfDate(sap.getSignDate())); // time-stamp will over-rule this
    externalSignatureContainer.modifySigningDictionary(dic);
    sap.setCryptoDictionary(dic);

    HashMap<PdfName, Integer> exc = new HashMap<PdfName, Integer>();
    exc.put(PdfName.CONTENTS, new Integer(estimatedSize * 2 + 2));
    sap.preClose(exc);

    InputStream data = sap.getRangeStream();
    byte[] encodedSig = externalSignatureContainer.sign(data);

    if (estimatedSize < encodedSig.length)
        throw new IOException("Not enough space");

    byte[] paddedSig = new byte[estimatedSize];
```

```
        System.arraycopy(encodedSig, 0, paddedSig, 0, encodedSig.length);


        PdfDictionary dic2 = new PdfDictionary();
        dic2.put(PdfName.CONTENTS, new PdfString(paddedSig).setHexWriting(true));
        sap.close(dic2);
    }

    public static void signDeferred(PdfReader reader, String fieldName, OutputStream outs,
ExternalSignatureContainer externalSignatureContainer) throws DocumentException, IOException,
GeneralSecurityException {
        AcroFields af = reader.getAcroFields();
        PdfDictionary v = af.getSignatureDictionary(fieldName);
        if (v == null)
            throw new DocumentException("No field");
        if (!af.signatureCoversWholeDocument(fieldName))
            throw new DocumentException("Not the last signature");
        PdfArray b = v.getAsArray(PdfName.BYTERANGE);
        long[] gaps = b.asLongArray();
        if (b.size() != 4 || gaps[0] != 0)
            throw new DocumentException("Single exclusion space supported");
        RandomAccessSource readerSource = reader.getSafeFile().createSourceView();
        InputStream rg = new RASInputStream(new RandomAccessSourceFactory().createRanged(readerSource,
gaps));
        byte[] signedContent = externalSignatureContainer.sign(rg);
        int spaceAvailable = (int)(gaps[2] - gaps[1]) - 2;
        if ((spaceAvailable & 1) != 0)
            throw new DocumentException("Gap is not a multiple of 2");
        spaceAvailable /= 2;
        if (spaceAvailable < signedContent.length)
            throw new DocumentException("Not enough space");
        StreamUtil.CopyBytes(readerSource, 0, gaps[1] + 1, outs);
        ByteBuffer bb = new ByteBuffer(spaceAvailable * 2);
        for (byte bi : signedContent) {
            bb.appendHex(bi);
        }
        int remain = (spaceAvailable - signedContent.length) * 2;
        for (int k = 0; k < remain; ++k) {
            bb.append((byte)48);
        }
        bb.writeTo(outs);
        StreamUtil.CopyBytes(readerSource, gaps[2] - 1, gaps[3] + 1, outs);
    }
}
```

Bouncy Castle Cryptographic Provider

Bouncy Castle is a collection of APIs used in cryptography. It includes APIs for both the Java and the C# programming languages. The APIs are supported by a registered Australian charitable organization: Legion of the Bouncy Castle Inc.

Bouncy Castle is Australian in origin and therefore American restrictions on the export of cryptographic software do not apply to it.

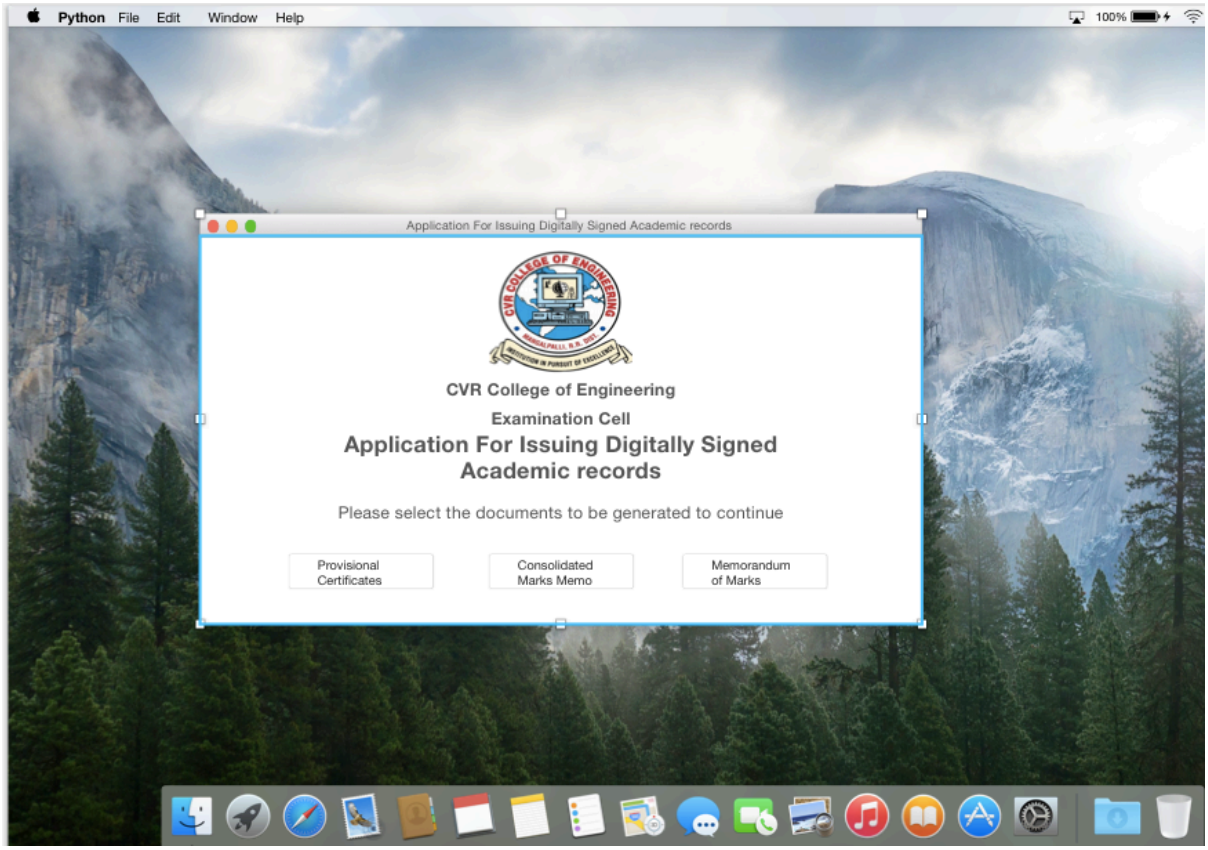
The Bouncy Castle architecture consists of two main components that support the base cryptographic capabilities. These are known as the 'light-weight' API, and the Java Cryptography Extension (JCE) provider. Further components built upon the JCE provider support additional functionality, such as PGP support, S/MIME, etc.

The low-level, or 'light-weight', API is a set of APIs that implement all the underlying cryptographic algorithms. The APIs were designed to be simple enough to use if needed, but provided the basic building blocks for the JCE provider. The intent is to use the low-level API in memory constrained devices (JavaME) or when easy access to the JCE libraries is not possible (such as distribution in an applet). As the light-weight API is just Java code, the Java virtual machine (JVM) does not impose any restrictions on the operation of the code, and at early times of the Bouncy Castle history it was the only way to develop strong cryptography that was not crippled by the Jurisdiction Policy files that prevented JCE providers from performing "strong" encryption.

The JCE-compatible provider is built upon the low-level APIs. As such, the source code for the JCE provider is an example of how to implement many of the "common" crypto problems using the low-level API. Many projects have been built using the JCE provider, including an Open Source Certificate Authority EJBCA.

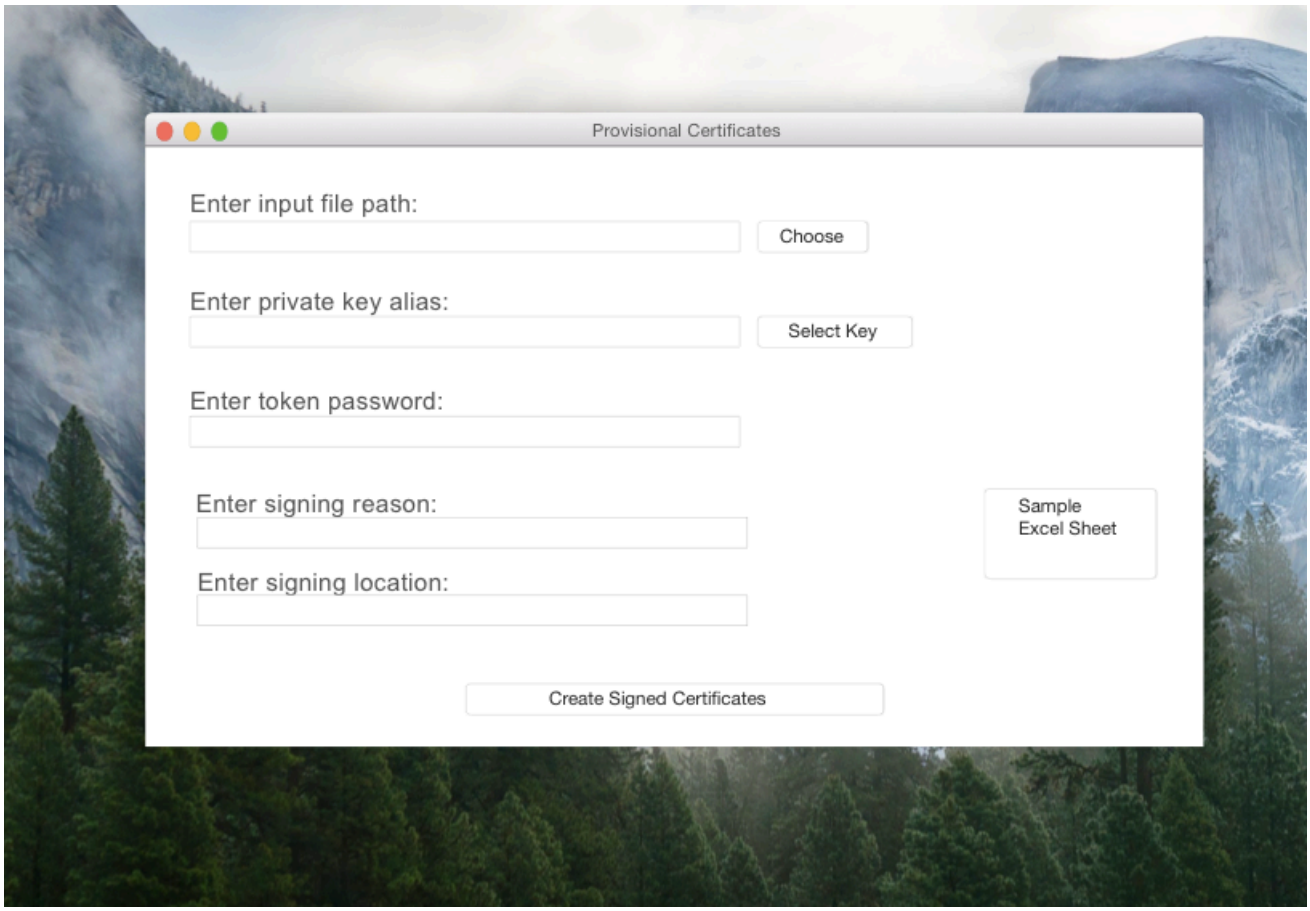
Sample Screenshots

Home Screen



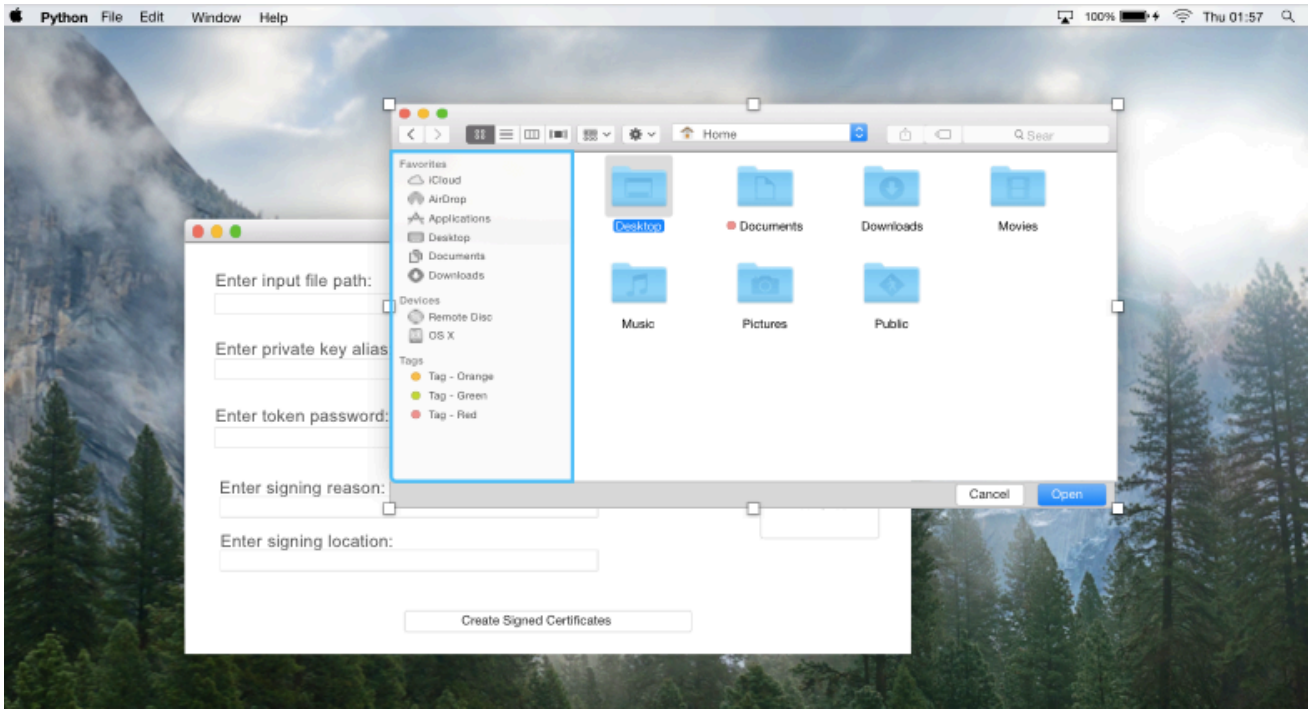
This is the home screen of the application where user can select one of the certificate generation applications. It marks the welcome screen of the application.

Provisional Certificate Window



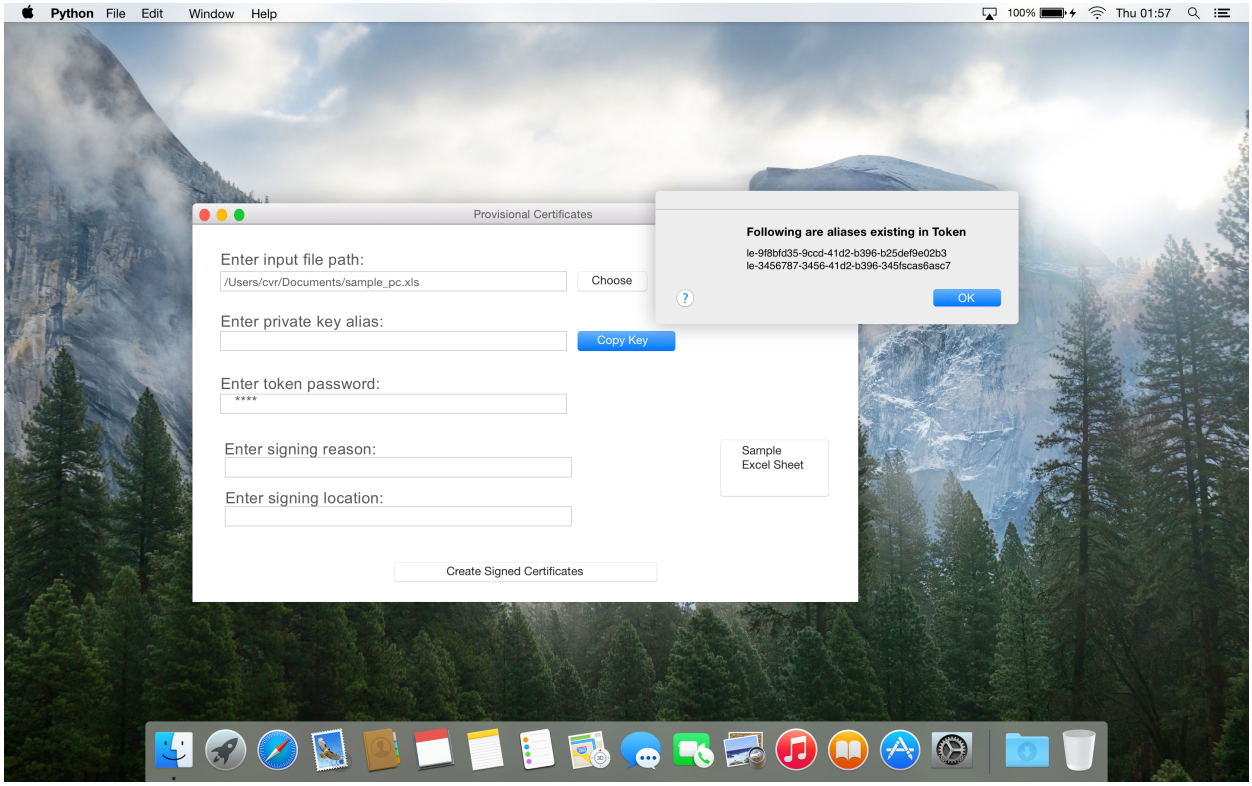
This provides a form to enter details required for generating and signing the provisional certificate

File Dialog Window



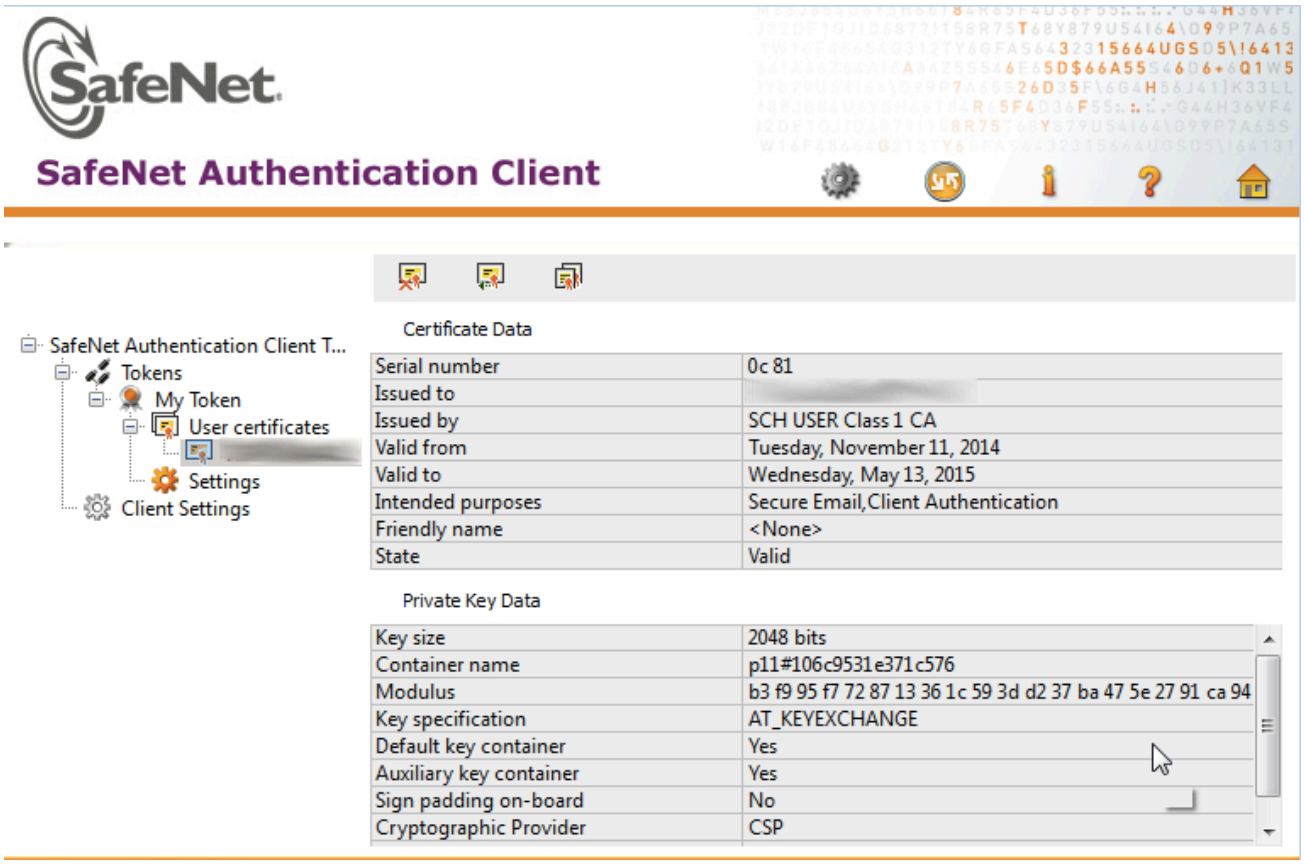
It is used to select the file path using the File Explorer.

Select Alias Window



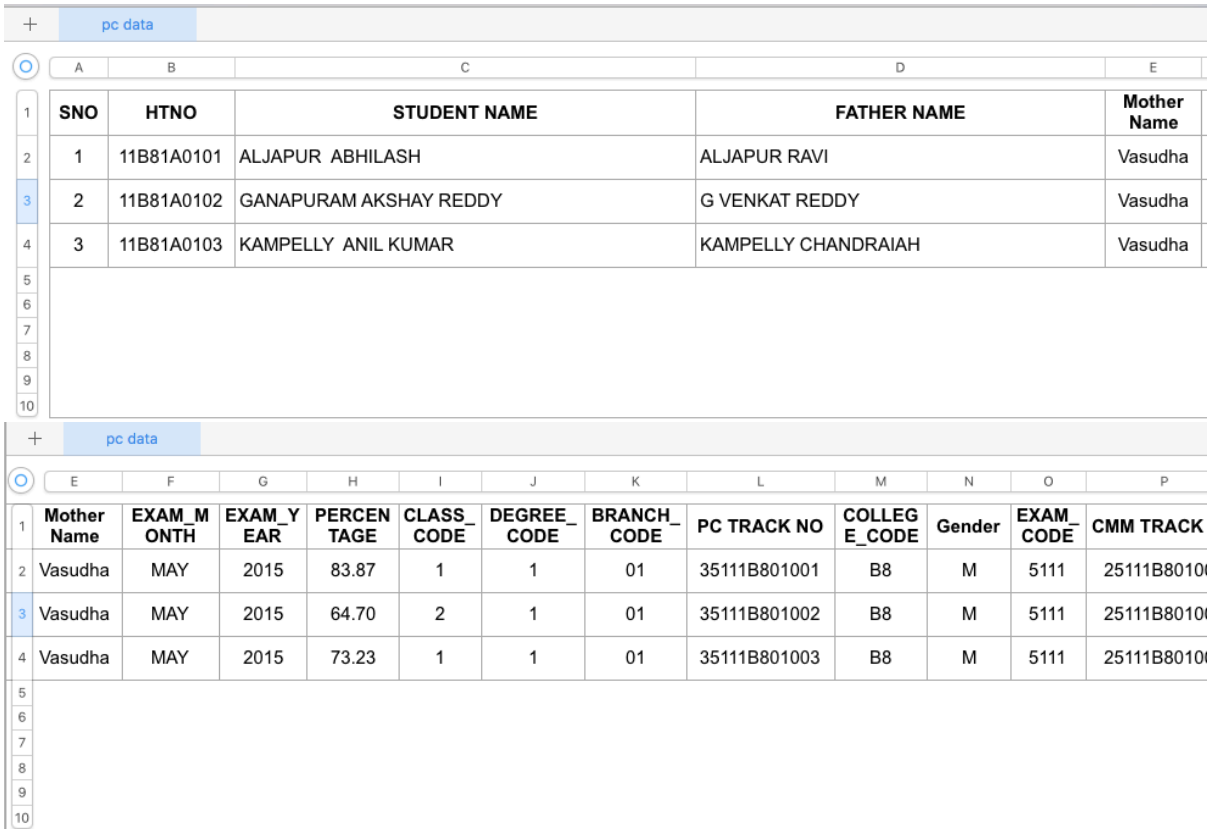
This lists the available aliases / container names(CNs) in the token so that user can copy the required alias.

SafeNet Client Tools(Driver Software)



This allows users to view all the available cryptographic entities and their attributes. User can observe a Certificate's alias listed as Container Name here.

Sample Provisional Certificate Data



pc data

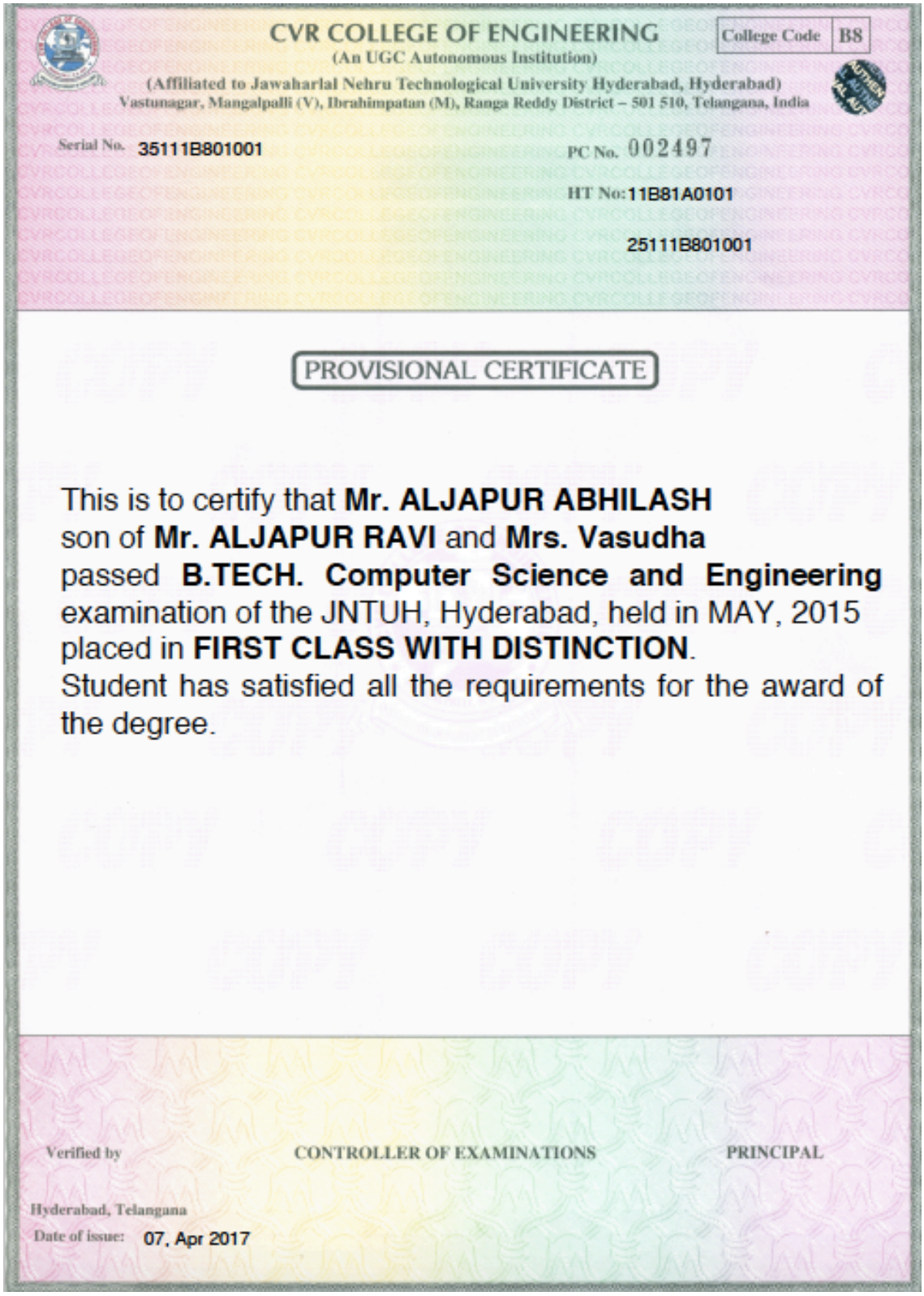
SNO	HTNO	STUDENT NAME	FATHER NAME	Mother Name
1	11B81A0101	ALJAPUR ABHILASH	ALJAPUR RAVI	Vasudha
2	11B81A0102	GANAPURAM AKSHAY REDDY	G VENKAT REDDY	Vasudha
3	11B81A0103	KAMPELLY ANIL KUMAR	KAMPELLY CHANDRAIAH	Vasudha

pc data

Mother Name	EXAM_MONTH	EXAM_YEAR	PERCENTAGE	CLASS_CODE	DEGREE_CODE	BRANCH_CODE	PC TRACK NO	COLLEGE_CODE	Gender	EXAM_CODE	CMM TRACK
Vasudha	MAY	2015	83.87	1	1	01	35111B801001	B8	M	5111	25111B8010
Vasudha	MAY	2015	64.70	2	1	01	35111B801002	B8	M	5111	25111B8010
Vasudha	MAY	2015	73.23	1	1	01	35111B801003	B8	M	5111	25111B8010

These show the records read by provisional certificate module to generate the certificates.

Digitally Signed PDF



CVR COLLEGE OF ENGINEERING

College Code B8

(An UGC Autonomous Institution)

(Affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad)

Vastunagar, Mangalpalli (V), Ibrahimpatan (M), Ranga Reddy District – 501 510, Telangana, India

Serial No. 35111B801001

PC No. 002497

HT No: 11B81A0101

25111B801001

PROVISIONAL CERTIFICATE

This is to certify that **Mr. ALJAPUR ABHILASH** son of **Mr. ALJAPUR RAVI** and **Mrs. Vasudha** passed **B.TECH. Computer Science and Engineering** examination of the JNTUH, Hyderabad, held in MAY, 2015 placed in **FIRST CLASS WITH DISTINCTION**.
Student has satisfied all the requirements for the award of the degree.

Verified by

CONTROLLER OF EXAMINATIONS

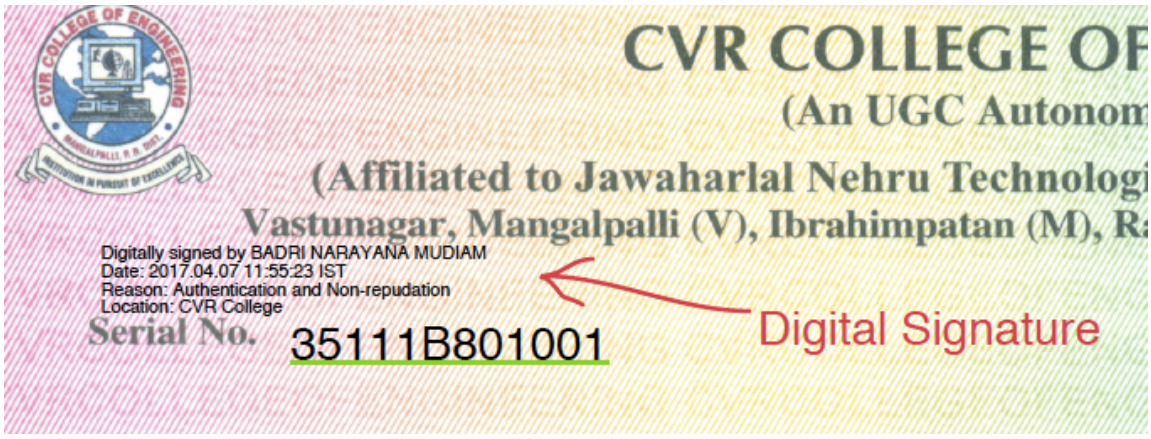
PRINCIPAL

Hyderabad, Telangana

Date of issue: 07, Apr 2017

A sample provisional certificate generated.

Digitally Signature



The above image shows a signature appearance showing the signer's name, timestamp, Reason and Location.

5. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceived fault or weakness in a work product.

Test Case: Testing if a valid token is present.

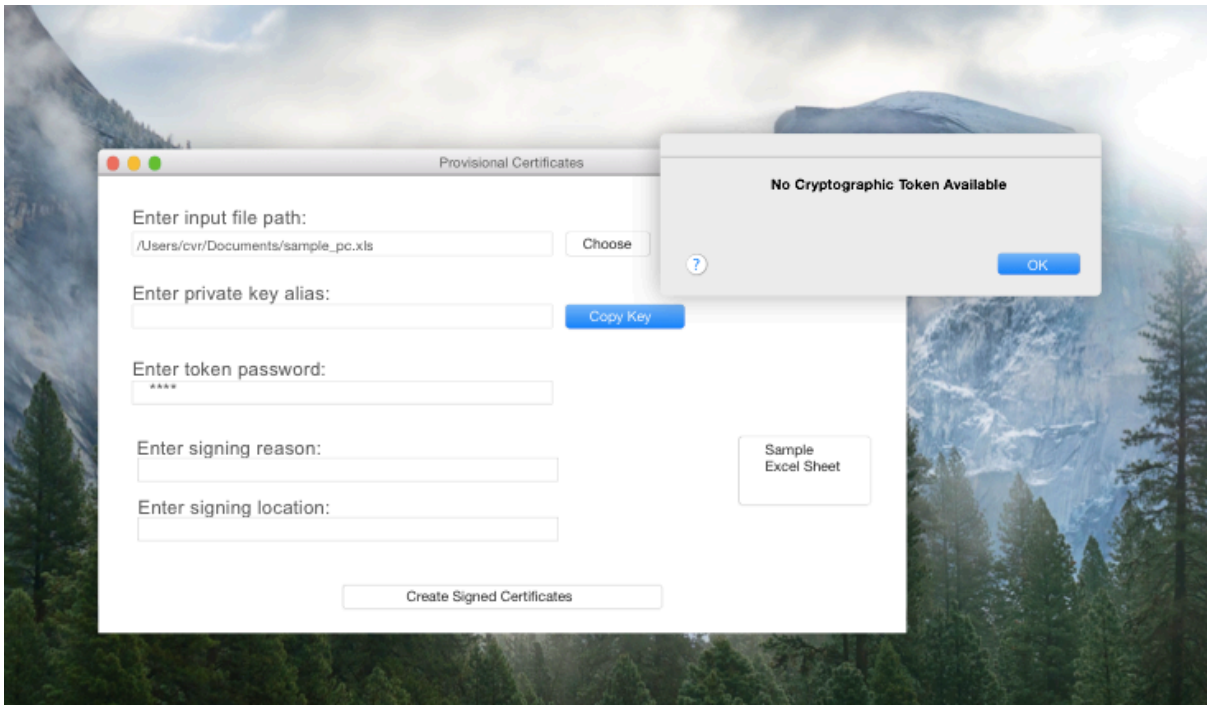
Description: Valid token must be present to sign a certificate.

Input: Invalid or No token presented by the user.

Expected Outcome: User must be given an error message informing the he must insert a valid token before generating certificates.

Actual Outcome: An error message is displayed when the user inserts invalid token.

The message displayed is “No Cryptographic Token Available.”



Test Case: Testing if password entered is right.

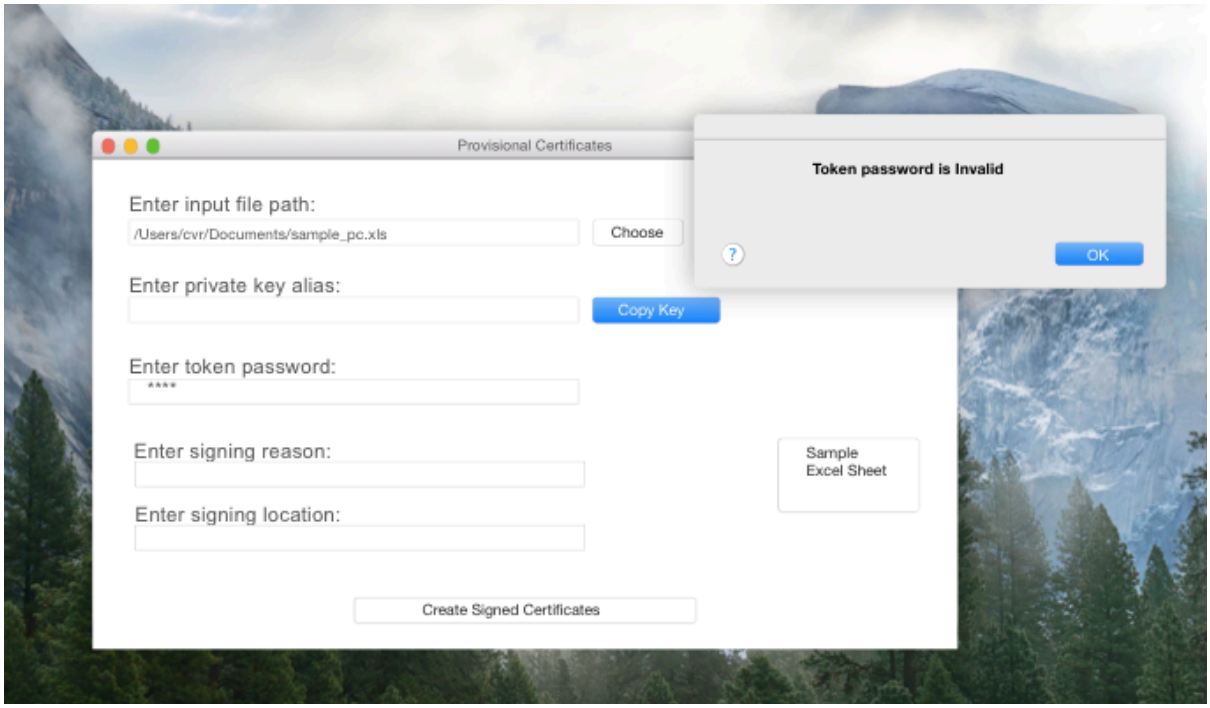
Description: Valid and correct password must be entered to sign a certificate.

Input: Wrong password entered by the user.

Expected Outcome: User must be given an error message informing the password entered is incorrect.

Actual Outcome: An error message is displayed when the user enters wrong password.

The message displayed is “Token Password is Invalid.”



6. CONCLUSION AND FUTURE SCOPE

The application can be used to generate legally valid academic certificates that can be distributed to the students. It reduces burden of generating multiple documents and the pain of signing multiple certificates multiple times.

A Digital Certificate consisting the public key used to sign the records will be published by the college so that other entities can use the certificate to validate the signature.

User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- Extend application to many other documents to provide a paper-less office.
- Provide acceptable records to Universities and other outside parties for documentation.
- The certificates can be delivered along with results in the website to provide validation to third-parties.
- Certificates can be mailed to the students as soon as generated and provide physical certificates only if necessary to reduce resources used by Examination branch.

7. REFERENCES

[1] Behrouz A. Forouzan, "Cryptography & Network Security": McGraw-Hill, 2015.

[2] "ReportLab PDF Library User Guide" Apr. 6, 2017. [Online]. Available: https://www.reportlab.com/docs/reportlab-userguide.pdf.

[3] Bruno Lowagie, "iText in Action Second Edition": Manning Publications Co, 2011.

[4] SafeNet, "Luna HSMs and Java PKCS#11 Providers Integration Guide ": SafeNet, Inc. , 2010.

[5] Jason R. Weiss, "Java Cryptography Extensions: Practical Guide for Programmers": Morgan Kaufmann, 2004.

[6] Brian Jones, "Python Cookbook": Shroff; Third edition, 2013

[7] "PKCS#11: Cryptographic Token Interface Standard" [Online]. Available: https://www.cryptsoft.com/pkcs11doc.

[8] https://stackoverflow.com

[9] https://pypi.python.org

[10] https://www.bouncycastle.org/wiki/

[11] https://docs.python.org