**1. INTRODUCTION**

Process of converting simple text into encrypted text is known as Cryptography. Cryptography simply means encryption and decryption of the text, file or other resources without any additional procedures such as integrity checks, authentication etc. But sometimes it can be associated with the mathematical theory and computer science practices. It also cares about the confidentiality, integrity, Authentication, Non-repudiation of the data. Cryptography can be in many of the sectors like banking transaction cards, computer passwords, and e-commerce transactions. The process of cryptography follows the various techniques i.e. ciphers. There are mainly three types of cryptography techniques – Symmetric key cryptography, public key cryptography, Hash Functions. Symmetric key cryptography may be defining as single key cryptography. It is technique when both sender and receiver uses same single key. Sender uses a key to encrypt the text and sends that encrypted message to receiver. At the other end, receiver decrypts that text with the help of that key and gets the original message. Public key cryptography is the technique when two keys (public key and private key) are needed to perform this cryptography process. Public key can be distributed to all the users but private key will remain secret with each user, it can’t be distributed. For encryption of text public is used but for the decryption of the text private key is mainly used. Hash Functions is the technique in which no keys are mainly used to generate the message digest of the text. It is an irreversible process. It generates a fix length message digest from the variable length original text. In this process, we can’t get original text from the message digest. So, it can be more secure. Cryptography is basically in a great demand in today’s world.

We design a software based on cryptography. Our tool name is Crypto media. The main theme of project is that it consists of multiple hashing techniques and algorithms. A single platform for all the needs of cyber on the basis of encryption and decryption. Also the software works on every platform, we build a package installation model for the software, platform like windows, Linux, android.

Moreover, we added one module known as Integrity Checker. As we all know, now-a-days data integrity is one of the most important and crucial part of our life. We always become unable to judge which data is original data and which data is modified one by any individual or intruder in the network. Therefore, it is a process of comparing or verifying the current state of data with previous state of data to check the modifications.

**2.**  **PROFILE OF THE PROBLEM**

As we all know, now-a-days so many Capture The Flag (CTF) challenges are going on. Many cyber security individuals keep on participating in these CTFs. In these CTFs, there are many challenges related to cryptography which basically belongs to the encryption and decryption of the hidden messages. At this stage, to find the plain message by trying different techniques on different websites consumes more time. So, for this major problem we created a software which contains multiple hashing techniques as well as various algorithms to get the plain messages at that point itself by trying different techniques rather than to move on various websites. It consumes less time and we all know time is major parameter to win this capture the flag challenge. We are providing the execution time also to each cyber security individual who are using this software, so that they can compare or judge which software is better which we created or to finding through the various websites. Our software is very much user friendly in nature as compared to others calculators that are given in various websites. We provided this software on various platforms like Windows, Linux, Android. If user doing his/her CTF challenge on Windows, Linux, Android platform can do their cryptography process on the same platform without moving on different platform.

Everyday there are large number of cyber-attacks, new security techniques, other cyber news are coming in market, but user is unaware for that and getting no interest to check various news in day-to-day life. So, we provided an additional functionality in our software which is providing latest news to the individuals so that they can be more aware about the cyber news in their day to day life.

Our software is not just for the CTF users. It can be used by any other individual who belongs to computer world and who wants to generate some unintelligible content from their original content to secure their data from the users who want to access their data unintentionally as well as in un-securable form.

As in today’s world, so many attacks are keeps on occurring just to compromise the data of users, which includes data modifications too. So, that’s why for this set of problem we designed a module known as Integrity Checker Module. This will clearly give the result whether data is original one or modified one.

So, our software providing the solution for these problems making cyber individuals’ challenges and life better and easy.

**3. EXISTING SYSTEM**

**3.1 Introduction**

Our project **CRYPTOPEDIA** idea is to design a software that serves the need of encrypting and decrypting texts, codes, etc. A single platform for all the needs of cyber on the basis of encryption and decryption. Also the software works on every platform, we build a package installation model for the software, platform like windows, Linux, android. Everything in one place, perishing the need of searching the web for decrypting a text or encrypting one. As we students of cyber security facing problems with these not having them in one place and wasting our time in searching on internet, with these the further students see it easier and useful. The software or the API works on all platforms, reliable and feasible for a system, able to encrypt or decrypt any text based on any algorithm, or hashing technique that is available in the database it has been given.

**3.2 Existing Software**

**3.3 DFD for Present System**

**3.3.1 For Window Platform**



**3.4 What’s new in the system to be developed**

As our project is for different platforms, we have created by using different software’s. For Android, we developed an android app for the cryptography. The new thing we have done with this android app is – it is collection of multiple algorithms and hashing techniques. So, many apps were developed earlier for cryptography but this was having very much basic ciphers like Ceaser ciphers which is most least usable cipher now-a-days. We are providing advance ciphers to encrypt as well as decrypt the text. Moreover, we are giving the execution time of each and every algorithm by which we can easily compare the algorithms. We are providing the latest cyber news functionality to enhance the knowledge about the cyber for the cyber people as well as layman users. So, our app can be easily usable by the layman users. The main thing that we develop is checking the integrity of the files. We develop to compare or verify the current state of data with previous state of data to check the modifications. It will show whether the files are identical or integrity of these files is broken by someone else.

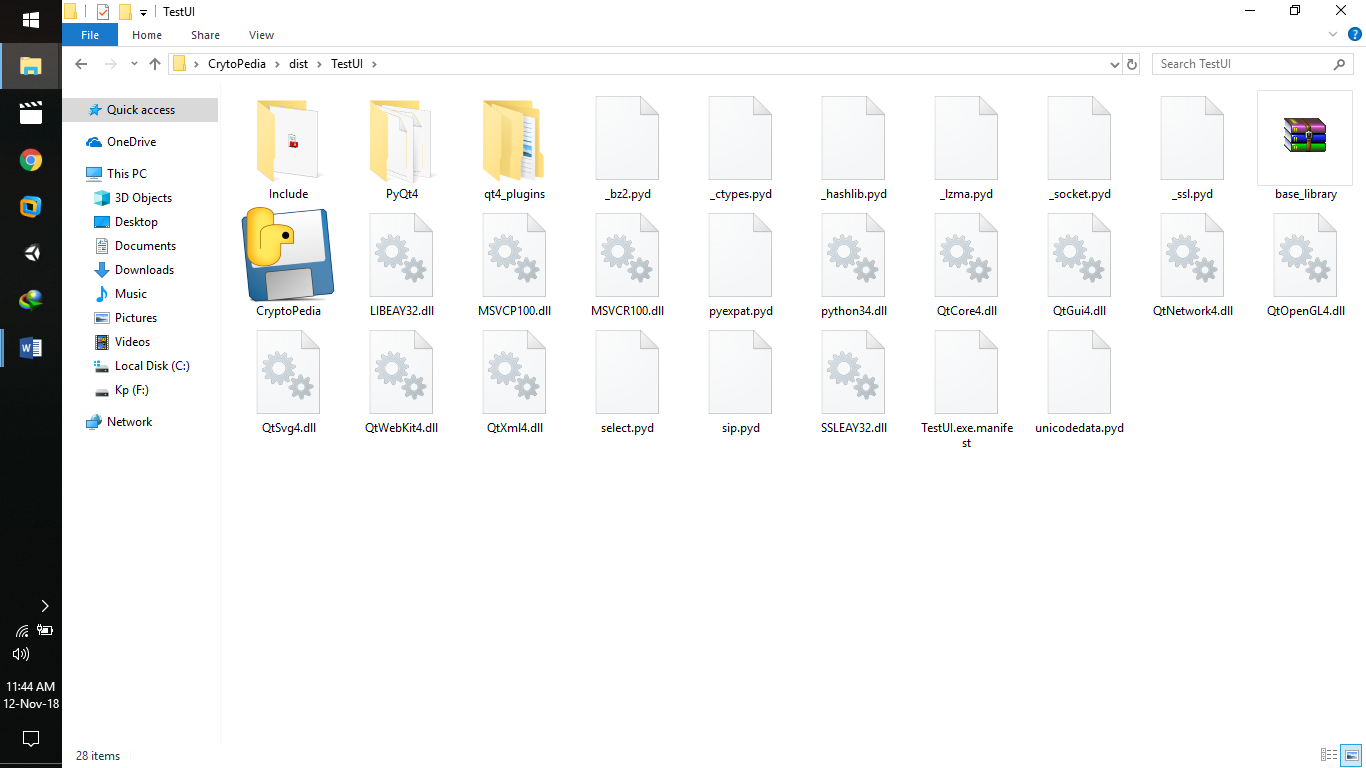
**4. PROBLEM ANALYSIS**

**4.1 Product definition**

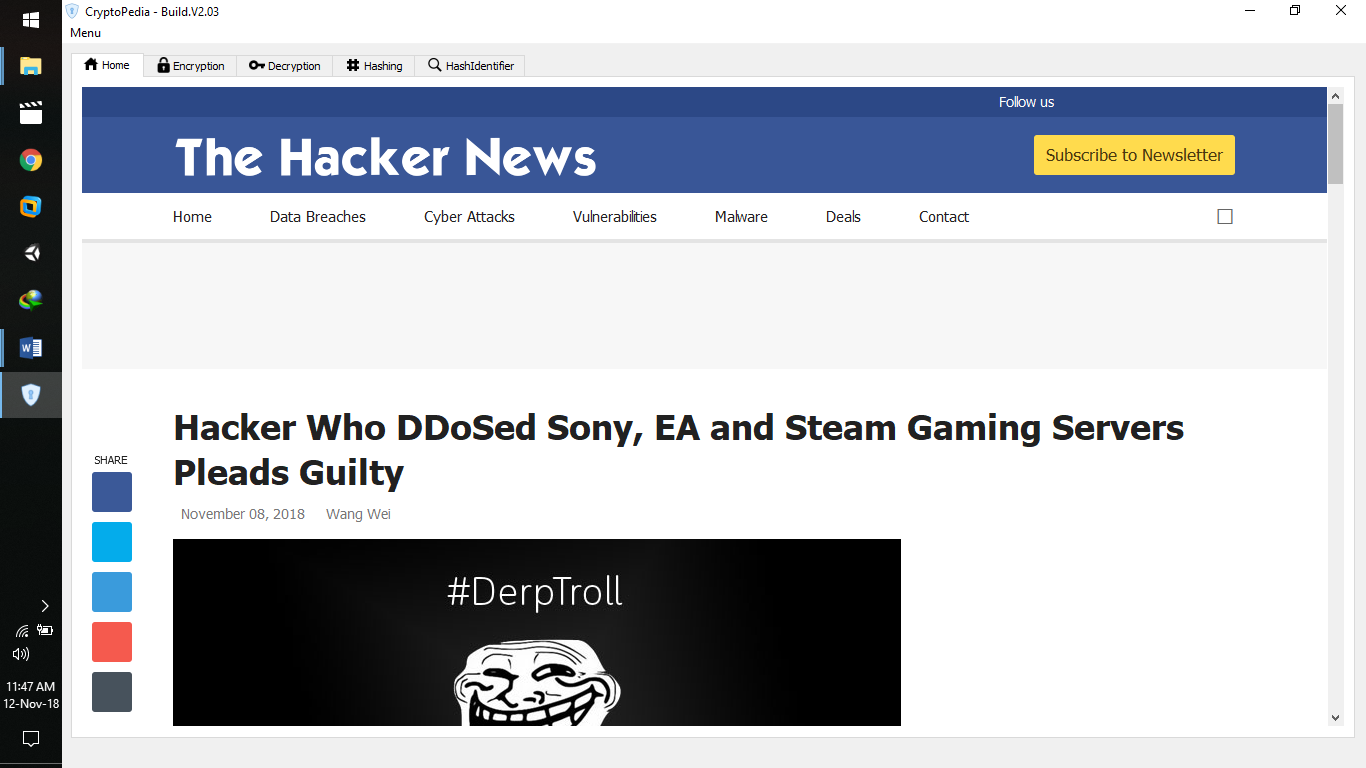
The project is a software based product which works on the concept of cryptography. Designed for the platforms windows, Linux, and android.

For Windows:

The windows application is designed as simple as it can be made, making it easy for the user to use it has been built like a package application which consists of every dependency and library required to run the application the user just need some files in which (.exe) file is defined that runs whole application.



As you can see in the above picture it consists of an (.exe) file named cryptopedia. On running that file the user gets full access of the application created as follows:



**4.2 Feasibility Analysis**

To produce a system that adequately serves its users there is the need to seek their present need as well as the recommended solution to tackle such needs and some society works as well. This project can be implemented using affordable software technology making it economically, technically and operationally feasible. Since, maximum of the project doesn’t take any data dynamically from the web and can run on any system or platform making it available for windows, Linux, and android makes it even more feasible.

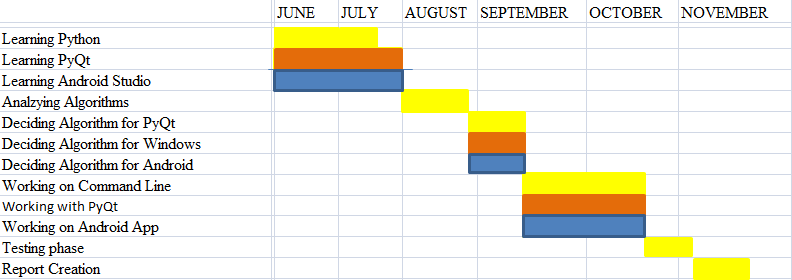
**4.2.1 Economically Feasibility:** This project is windows (64-bits), Linux (Kali Linux, Ubuntu etc.), Android phone (API levels-26,27,28) based which is easily available, affordable. This projects uses visual studio, android studio, Python3 which is economically feasible. Not even little cost is there.

**4.2.2 Technical Feasibility:** We did complete study of the project in terms of input, processes, output, fields, programs, and procedures. All the requirements of the project are easily achievable. When we were developing the project we got each and every functionality as same as we planned earlier. Therefore, development of this software is completely feasible.

**4.2.3 Operationally Feasibility:** This software is very much easy to use, user friendly interface for all the platforms will be offered to the users. The one who knows how to use android phone, Linux, Windows can operate our software very easily.

**4.3 Project Plan:**

Time-Span details taken by us to implement the project to its fullest.

****

We want to do some more additions in future:

To make our project globally available. We want to access the cloud services as well as we want to use the servers so that we can apply encryption/decryption process on client/server paradigm for layman users also to enhance the security in their day-to –day life.

**5. SOFTWARE REQUIREMENT ANALYSIS**

**5.1 Introduction**

Software requirement analysis describes the overall requirements and features of this proposed model of the software. When we come on software development we all need some sort of analysis. It includes requirement knowledge, Identification or stakeholders, collection of requirements, analysis of all collected requirements, all specifications of the software, management of software requirements. There is mainly a huge consideration of analysis of what is being introduced in the society, how it is going to work and how will market respond to it. The analysis is really very important if we want success in software development task and due to that, this development task will definitely lead to the height of perfection when the software will be introduced.

**5.2 General description**

**5.2.1 Product perspective**

An interface (graphical user interface, command-line, android app) that provides multiple functionalities to the user based on the cryptography. Main idea of the product to provide various encryption/decryption techniques to the user and to provide the response time for each of the encryption/decryption technique. A model where user can easily make their data in unintelligible form for provide more confidentiality and from unintelligible to the original message so that he/she can easily get their original data easily. A user can easily get latest cyber news on daily basis which will help them to enhance their knowledge of cyber world. A model for comparing or verifying the current state of data with previous state of data to check the modifications. As we know, now-a-days data integrity is one of the most important and crucial part of our life. So we provided this as integrity checker module.

Therefore, this product is not just for cyber security individuals but layman users can also use this, as we added integrity checking as well as cyber news module.

**5.2.2 Design and implementation constraints**

Since screen sizes, Unix systems and OS version are different from different mobile manufacturers, application performance might not likely be same for all users. The larger the screen size, the better performance, the newer the OS the better the GUI. Internet connection might also be a constraint for the end users for getting the latest news of cyber world.

**5.2.3 Assumptions and dependencies**

The assumption taking into consideration in the development of this project is that the gui/Unix interface/app will be used on system/device easily on any operating system but recommendation is for latest operating system/versions like: for windows- windows 10, windows 8.1, windows 7, for Unix system- kali Linux, for android phone- marshmallow, naugat, Oreo**.** These are the major assumptions/dependencies that can be followed at the time of the usage of this software. Module of integrity checker in the android mainly, has one very interesting functionality. Suppose any of the android phone is not having file manager application in it. It will pop up the message to install file manager application in the phone from any of the secure store, at that time itself

**5.3 Specific requirements**

**5.3.1 Functional Requirements:** This requirement outlines the functional capabilities that the system can be able to meet the user’s demands. The data encryption/decryption system has the following functional requirements.

* Functional Requirements 1:

Description: Entering the data

Input: Enter the data you want to encrypt

Output: Data is in supported format

* Functional Requirements 2:

Description: Applying Algorithms for encryption

Input: Different Algorithms for encryption

Output: Data Encrypted

* Functional Requirements 3:

Description: Decryption

Input: Applying appropriate algorithm

Output: Data Decrypted

* Functional Requirements 4:

Description: Execution Time

Input: Encrypt/Decrypt the data

Output: Time will be displayed

* Functional Requirements 5:

Description: Cyber News

Input: Choose the appropriate browser

Output: News will display

* Functional Requirements 6:

Description: Integrity Checker

Input: Choose the different files

Output: Integrity results will display

**5.3.2 External Requirements**

**5.3.3 Performance Requirements**

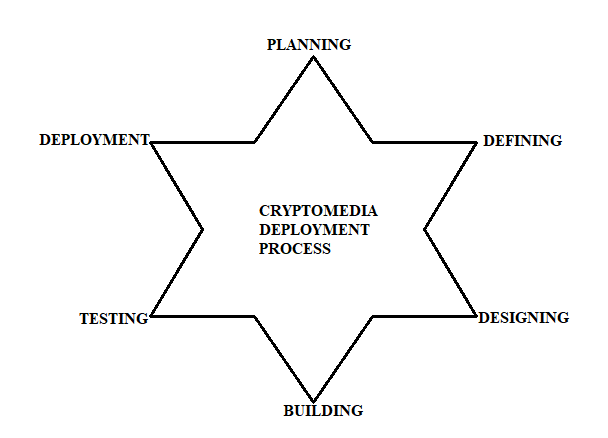
* **OS:** Device can be running any of the OS Version windows, Android, Linux etc. but  
  latest versions or above (major recommendations).
* **STORGAE:** There are no any specific requirements of storage space in the system
* **MEMORY:** Device should have a least 1GB of RAM.
* **TIME:** Very minimum amount of time is consumed to perform a process.

**5.3.4 Non Functional Requirements:**

* **Security**: The tool should be secure i.e. data of one user should not be visible to other  
  user**.** No one can see which user is encrypted or decrypted which type of data
* **Portability**: The tool should be able to work on every platform like Windows, Linux,  
  Android.
* **Capacity**: The tool should be able to have good capacity so that it can take maximum  
  amount of data in each time for encryption as well as decryption.
* **Availability:** The availability of cryptopedia will be 24x7, it is not facing any network error because usage of internet is very less only in the case of cyber news.
* **Maintainability:** cryptopedia can be easily extended. The code should be written in a way that it favors implementation of new functions, in order for functions to be easily added to the software in the future.

**5.4 Software Development Life cycle**

This process consists of a overall plan explaining how to develop, maintain, replace and alter the software. It is process that we used at the time of the deployment of our project. It helps us for enhancing the quality of the software and the overall development process. The process which follows at the time of development is given below:



**6. DESIGN**

**6.1 System Design**

As our tool for Windows, Linux, Android, so system differs but the working is same for all of them.

For Windows:

The main motive of windows platform is that, making the work of cryptography available for every window user as a Graphical User Interface(GUI). So, that the end user can easily grab the knowledge of what’s happening and would be very helpful when solving a CTF which is the main problem we want to solve in this project.

Therefore, the design is created as simple as it can be. The application consists of five tabs named: Home Tab, Encryption Tab, Decryption Tab, Hashing Tab, Hash Identifier Tab. Each tab is provided with some tools which the end user can use.

The Home tab provides latest cyber new time to time using the web services and the only module that requires internet to run, giving the end user knowledge about happenings around, whenever he/she uses the application.

Likewise, the Encryption tab, Decryption tab is provided with some algorithms that can be used to encrypt or decrypt any data. The Hashing tab provided with some algorithm’s so that hashes of several data can be made. The Hash Identifier tab consists of the tool in sequel with the tool provided for kali-Linux i.e., Hash-Identifier.

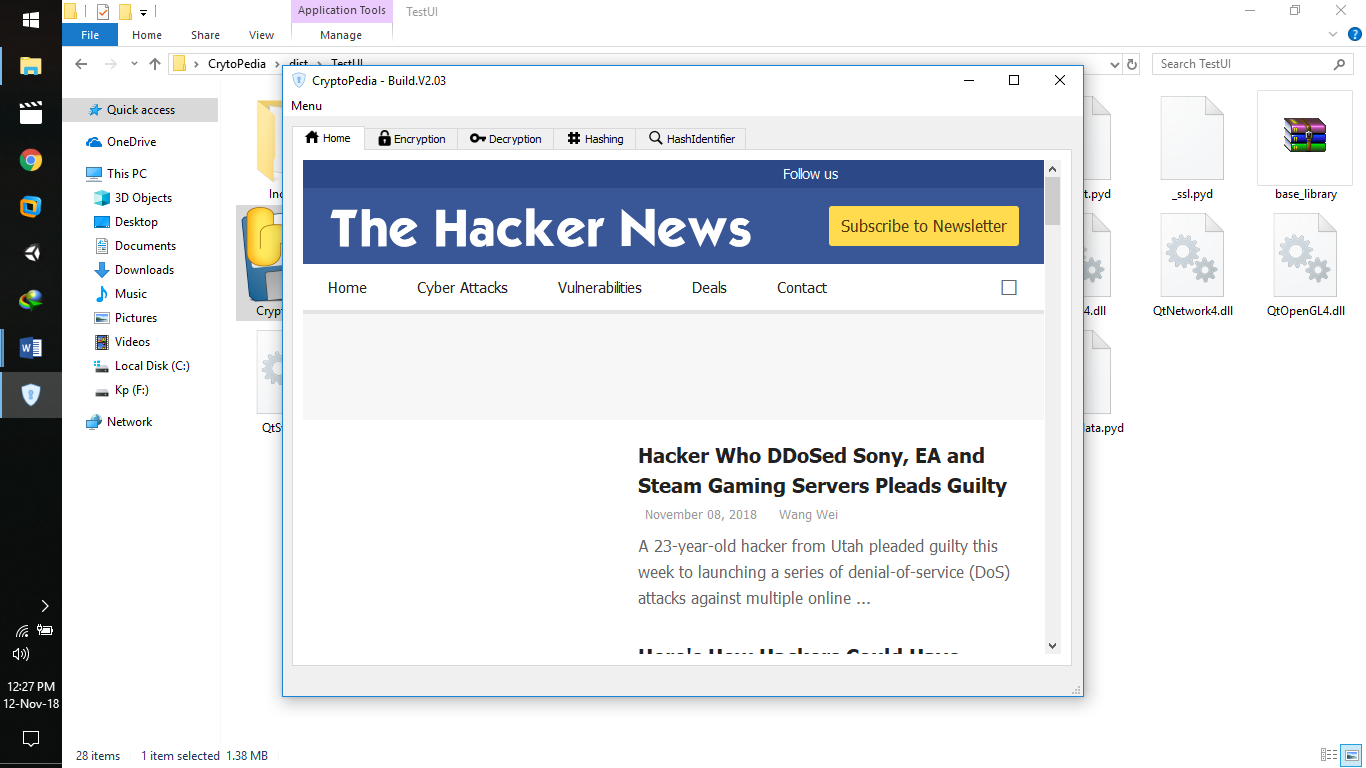
The Application can be further developed by adding more algorithms to it and making it more efficient to use for the end user.

**6.2 DETAILED DESIGN:**

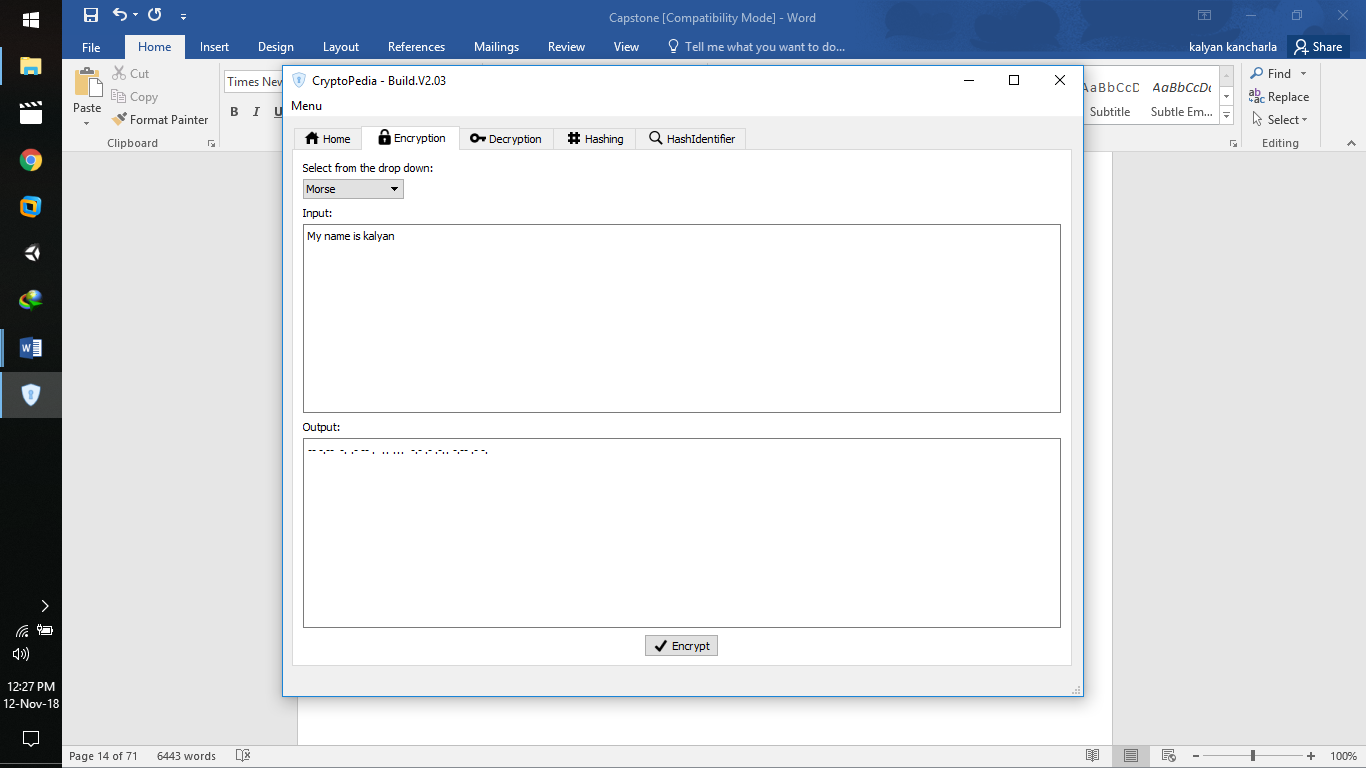
For Windows:

The Application consists of five tabs as follows:

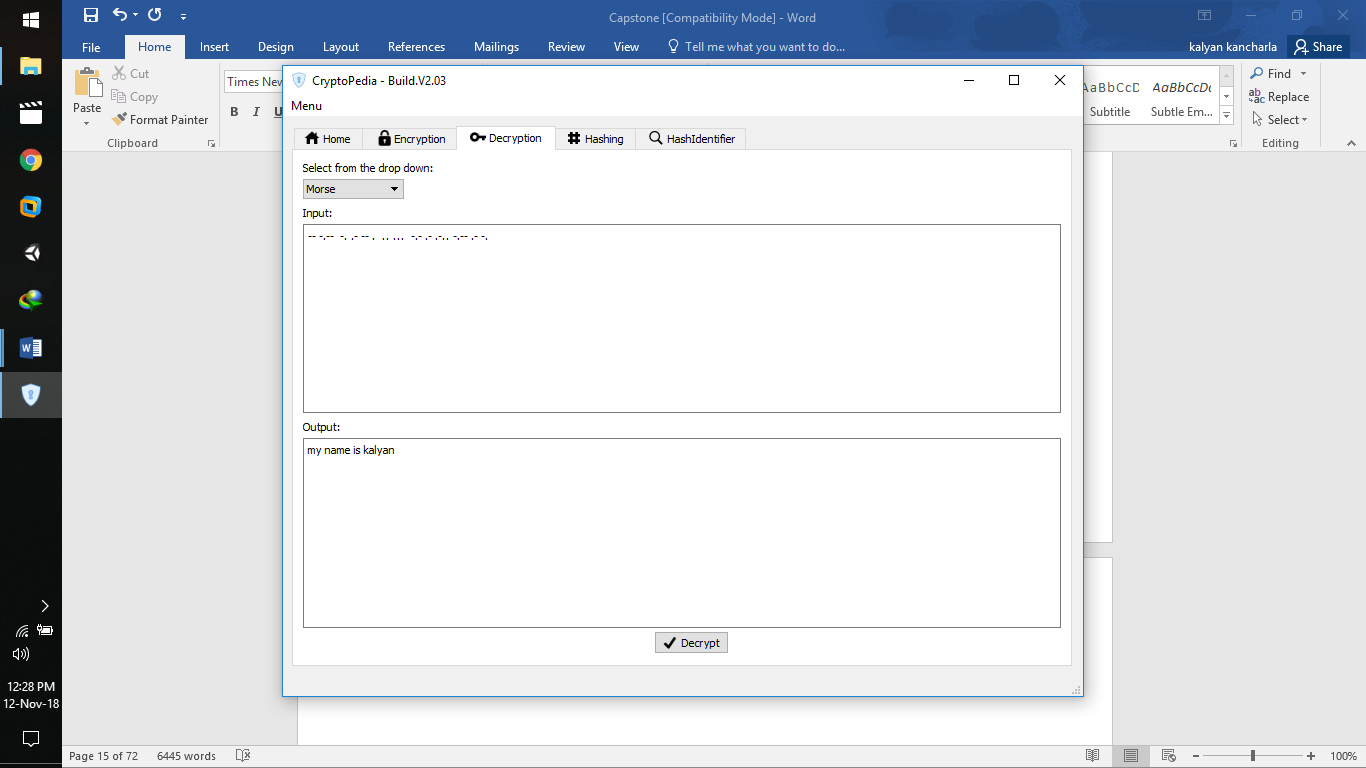
Home tab:



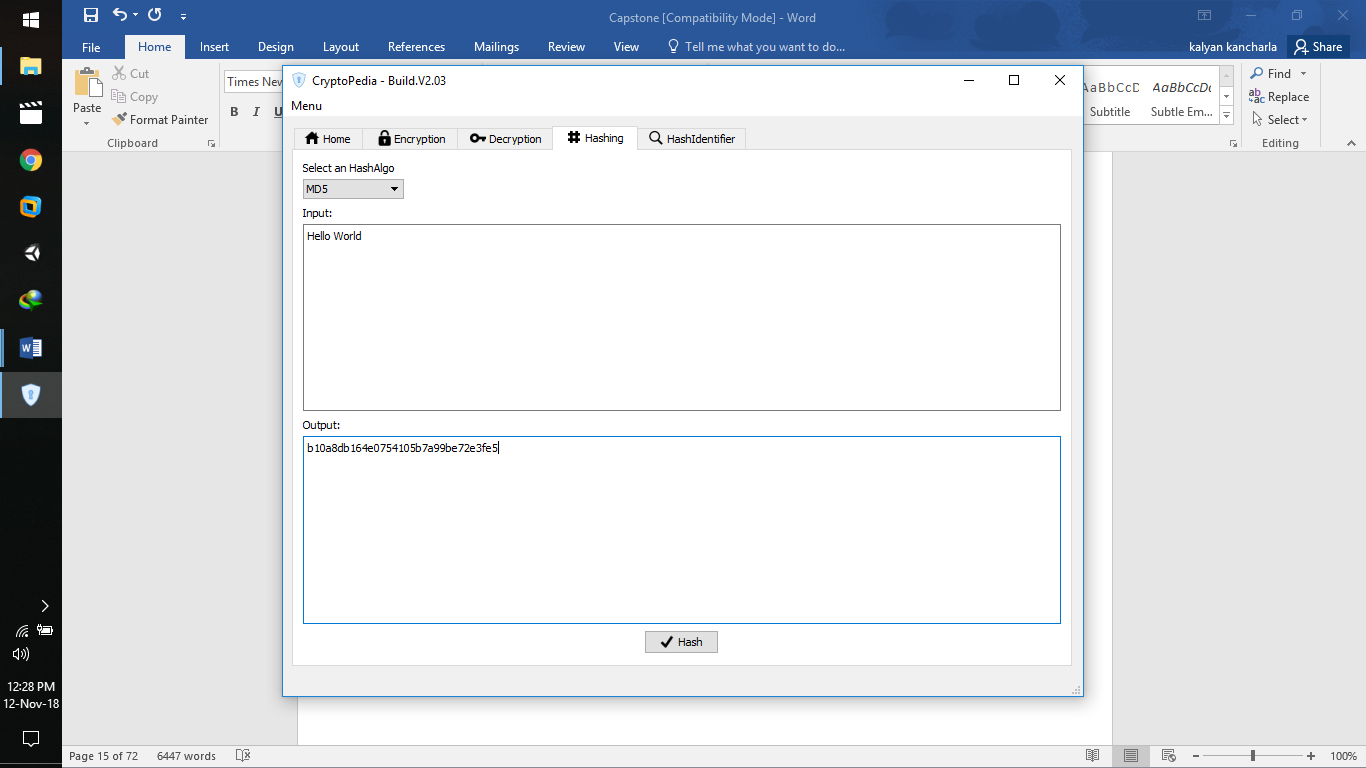
Encryption Tab:



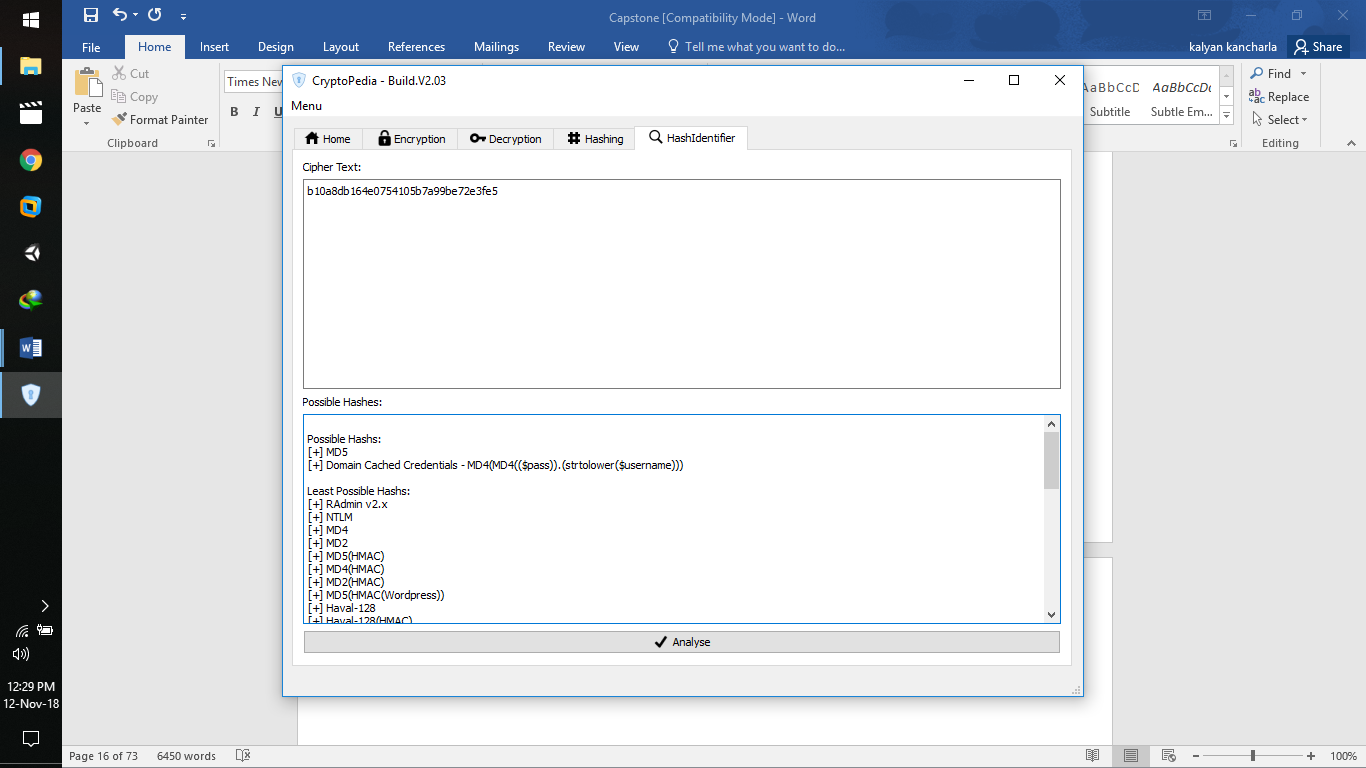
Decryption Tab:



Hashing Tab:



Hash Identifier Tab:



**6.3 FLOW CHARTS:**



Complete Flow Chart of Window’s GUI Application

**6.4 PSEUDO CODE:**

For Windows:

The main code of the project to the application is:

import sys

from PyQt4 import QtGui

from main import Ui\_MainWindow

import Base64, Ceaser, MorseCode, MD5, SHA1, SHA224, SHA256, SHA384, SHA512, ROT13, HashId

class MyQtApp(QtGui.QMainWindow):

def \_\_init\_\_(self):

QtGui.QMainWindow.\_\_init\_\_(self)

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.setWindowTitle("CryptoPedia - Build.V2.03")

self.ui.Enc\_Button.clicked.connect(self.encrypt)

self.ui.Dec\_Button.clicked.connect(self.decrypt)

self.ui.Hash\_Button.clicked.connect(self.hashing)

self.ui.Analyse\_Button.clicked.connect(self.hashidentifier)

def encrypt(self):

c1\_selection = self.ui.comboBox.currentText()

if c1\_selection == "Base64":

Base64.Base64.enc(self)

elif c1\_selection == "Ceaser":

Ceaser.Ceaser.enc(self)

elif c1\_selection == "Morse":

MorseCode.MorseCode.encrypt(self)

elif c1\_selection == "ROT13":

ROT13.ROT13.encrypt(self)

else:

print("Please Select an Algorithm")

def decrypt(self):

c2\_selection = self.ui.comboBox\_2.currentText()

if c2\_selection == "Base64":

Base64.Base64.dec(self)

elif c2\_selection == "Ceaser":

Ceaser.Ceaser.dec(self)

elif c2\_selection == "Morse":

MorseCode.MorseCode.decrypt(self)

elif c2\_selection == "ROT13":

ROT13.ROT13.decrypt(self)

else:

print("Please select an Algorithm")

def hashing(self):

c3\_selection = self.ui.comboBox\_3.currentText()

if c3\_selection == "MD5":

MD5.MD5.hashing(self)

elif c3\_selection == "SHA1":

SHA1.SHA1.hashing(self)

elif c3\_selection == "SHA224":

SHA224.SHA224.hashing(self)

elif c3\_selection == "SHA256":

SHA256.SHA256.hashing(self)

elif c3\_selection == "SHA384":

SHA384.SHA384.hashing(self)

elif c3\_selection == "SHA512":

SHA512.SHA512.hashing(self)

def hashidentifier(self):

HashId.HashIdentifier.hashing(self)

if \_\_name\_\_ == '\_\_main\_\_':

app = QtGui.QApplication(sys.argv)

qt\_app = MyQtApp()

qt\_app.show()

sys.exit(app.exec\_())

**7.PROJECT LEGACY:**

**7.1 Current status of the project:**

Currently the applications are working with full functionality and are not showing any error’s. Making it available for the platforms such as windows, Linux, and android has made the project more feasible as it can be used by many end users which requires very low system requirements as the application is now deployed only locally.

**7.2 Technical and managerial lessons learnt:**

**Technical**: -

For Windows:

The technical aspect of this application design and developments plays a major role when it comes to its design in GUI to make it more easy for the user to interact and code to further develop the features and tools provided in the application. For this application the GUI has been developed using PyQt which is called the GUI module in python and the rest of the coding is done using python. Designed the application to its fullest to make it easy for the end user.

Managerial:

The managerial learning came out of this project was that we learnt how to work in a group and to manage people who are working along with us and how to solve problems and conflicts faced by each of us. Then the problems aroused in implementing the project lead us to exploration of news things that could be useful for our future.

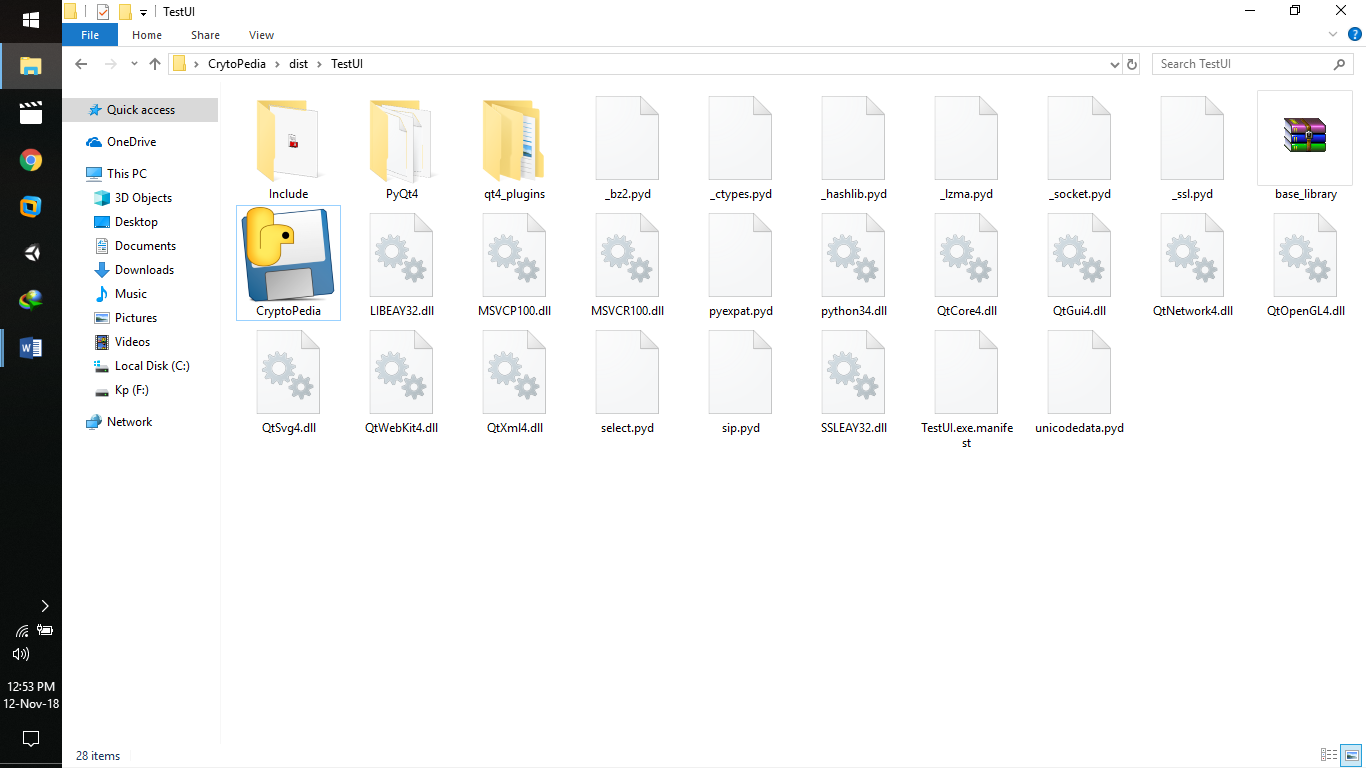
**APPENDIX-1**

**USER MANUAL:**

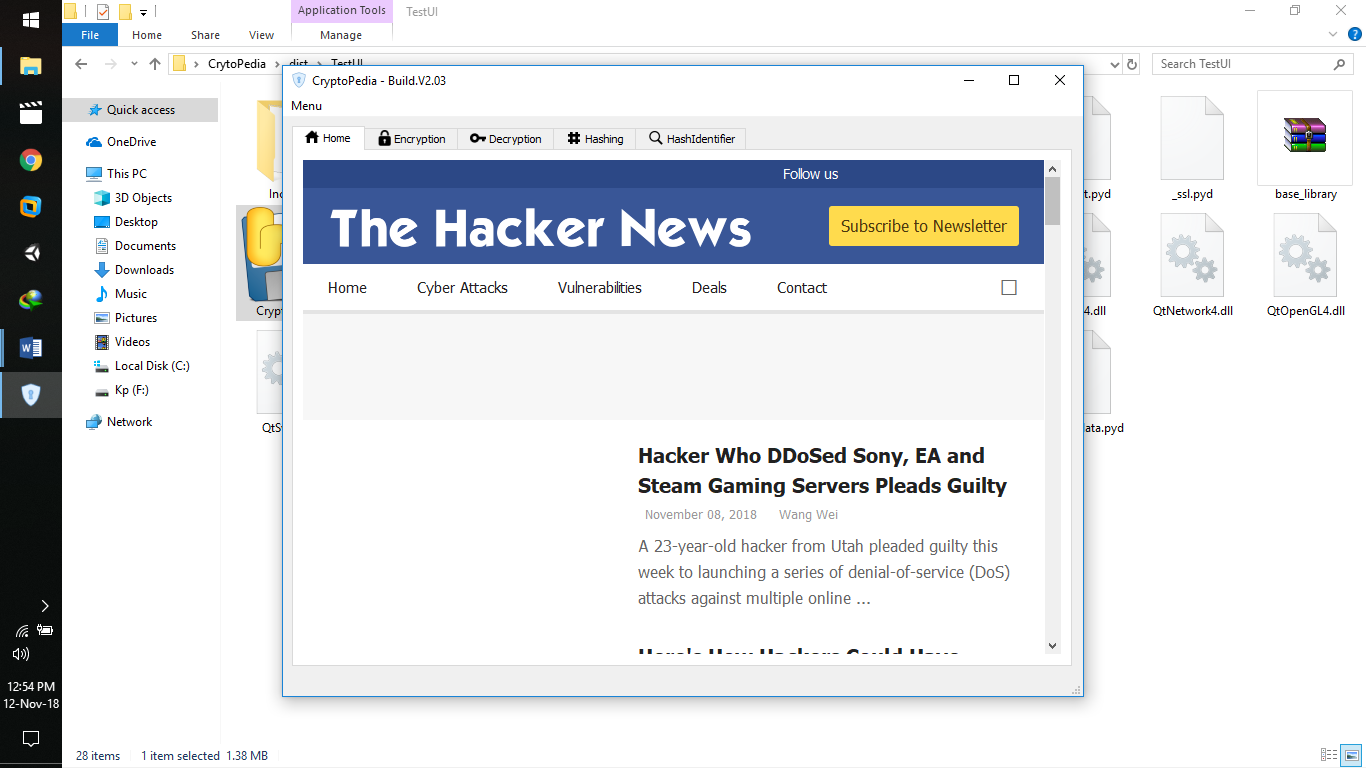
User Manual for windows:

A user guide or user's guide, also commonly known as a manual, is a technical communication document intended to give assistance to people using a particular system. User manual for our project is written below: As for the windows application goes as follows:

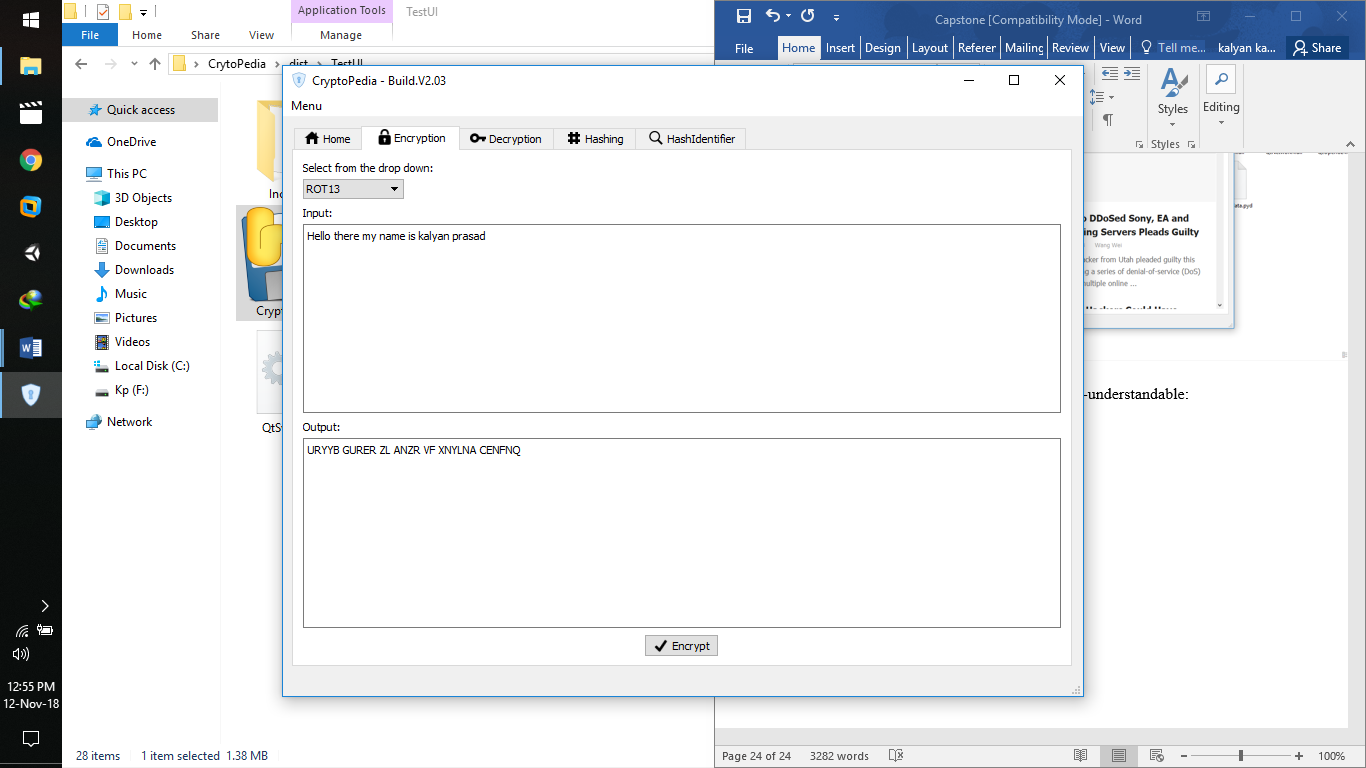
The user will be given a folder that contains every library and dependencies giving the freedom of not installing any other software we used as developing.



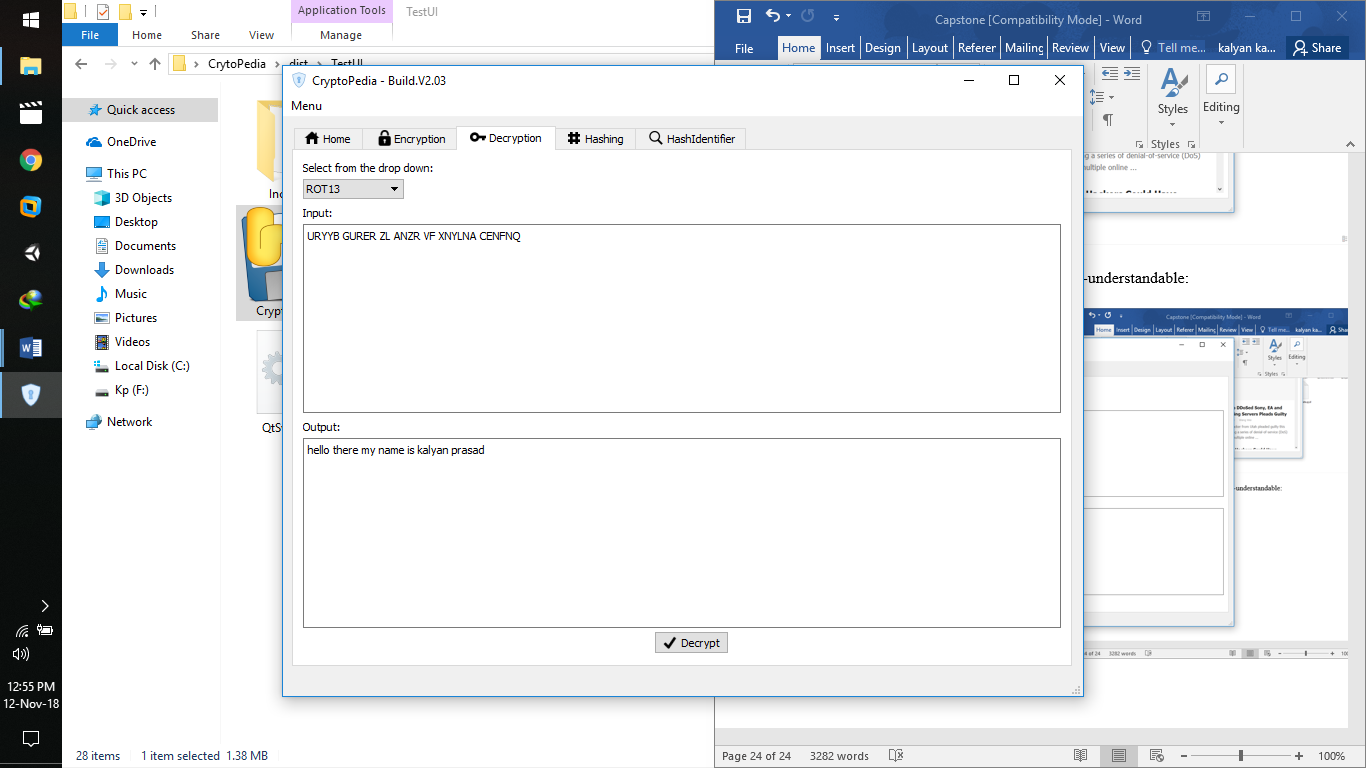
Once the application opens there opens a home tab that consists of cyber news on daily basis, which is the only module in the project that requires web connectivity.



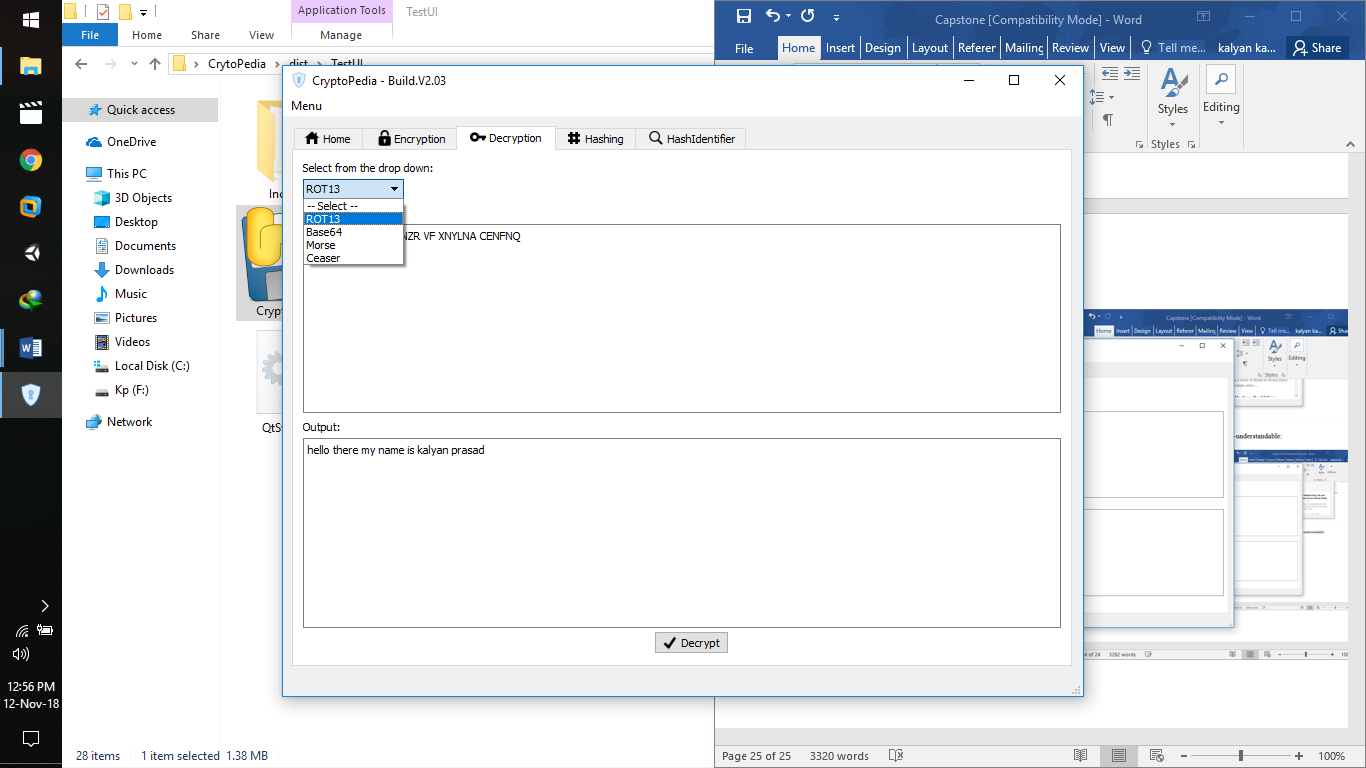
When the user wants to encrypt any data to keep it un-understandable:



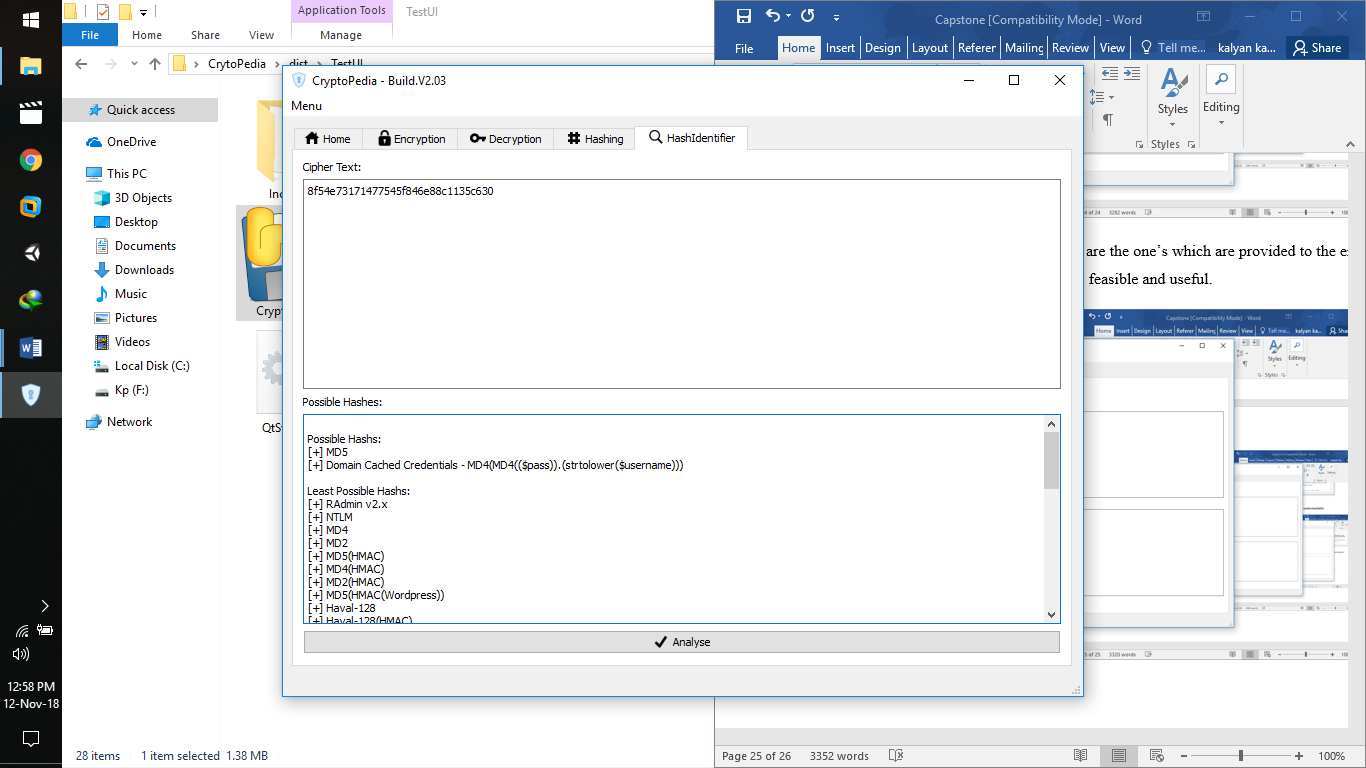
The user can use the Decryption tab in the time CTF’s which is main problem we want to solve, as the CTF comes with the questions related to cryptography encrypting every data given to us as a clue.



The algorithms mentioned in the drop down as below are the one’s which are provided to the end user and further to be developed to make it even more feasible and useful.



The hash identifier tab works in sequel with the tool in Kali-Linux. The user can use this tab to identify the hash with which the data is encrypted.



**APPENDIX-2**

**SOURCE CODE:**

For Windows:

The Code for the GUI:

from PyQt4 import QtCore, QtGui

try:

\_fromUtf8 = QtCore.QString.fromUtf8

except AttributeError:

def \_fromUtf8(s):

return s

try:

\_encoding = QtGui.QApplication.UnicodeUTF8

def \_translate(context, text, disambig):

return QtGui.QApplication.translate(context, text, disambig, \_encoding)

except AttributeError:

def \_translate(context, text, disambig):

return QtGui.QApplication.translate(context, text, disambig)

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName(\_fromUtf8("MainWindow"))

MainWindow.resize(800, 600)

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-security-lock-40.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

MainWindow.setWindowIcon(icon)

self.centralwidget = QtGui.QWidget(MainWindow)

self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))

self.verticalLayout = QtGui.QVBoxLayout(self.centralwidget)

self.verticalLayout.setObjectName(\_fromUtf8("verticalLayout"))

self.tabWidget = QtGui.QTabWidget(self.centralwidget)

self.tabWidget.setObjectName(\_fromUtf8("tabWidget"))

self.Home\_tab = QtGui.QWidget()

self.Home\_tab.setObjectName(\_fromUtf8("Home\_tab"))

self.gridLayout\_2 = QtGui.QGridLayout(self.Home\_tab)

self.gridLayout\_2.setObjectName(\_fromUtf8("gridLayout\_2"))

self.webView = QtWebKit.QWebView(self.Home\_tab)

self.webView.setUrl(QtCore.QUrl(\_fromUtf8("https://thehackernews.com/")))

self.webView.setObjectName(\_fromUtf8("webView"))

self.gridLayout\_2.addWidget(self.webView, 0, 0, 1, 1)

icon1 = QtGui.QIcon()

icon1.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-home-page-24.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.tabWidget.addTab(self.Home\_tab, icon1, \_fromUtf8(""))

self.Enc\_tab = QtGui.QWidget()

self.Enc\_tab.setObjectName(\_fromUtf8("Enc\_tab"))

self.Enc\_Input = QtGui.QTextEdit(self.Enc\_tab)

self.Enc\_Input.setGeometry(QtCore.QRect(9, 73, 758, 189))

self.Enc\_Input.setObjectName(\_fromUtf8("Enc\_Input"))

self.Enc\_Output = QtGui.QTextEdit(self.Enc\_tab)

self.Enc\_Output.setGeometry(QtCore.QRect(9, 287, 758, 190))

self.Enc\_Output.setObjectName(\_fromUtf8("Enc\_Output"))

self.Enc\_Button = QtGui.QPushButton(self.Enc\_tab)

self.Enc\_Button.setGeometry(QtCore.QRect(350, 483, 75, 23))

icon2 = QtGui.QIcon()

icon2.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-checkmark-26.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.Enc\_Button.setIcon(icon2)

self.Enc\_Button.setObjectName(\_fromUtf8("Enc\_Button"))

self.label = QtGui.QLabel(self.Enc\_tab)

self.label.setGeometry(QtCore.QRect(9, 54, 30, 16))

self.label.setObjectName(\_fromUtf8("label"))

self.label\_2 = QtGui.QLabel(self.Enc\_tab)

self.label\_2.setGeometry(QtCore.QRect(9, 268, 38, 16))

self.label\_2.setObjectName(\_fromUtf8("label\_2"))

self.label\_3 = QtGui.QLabel(self.Enc\_tab)

self.label\_3.setGeometry(QtCore.QRect(9, 9, 141, 16))

self.label\_3.setObjectName(\_fromUtf8("label\_3"))

self.comboBox = QtGui.QComboBox(self.Enc\_tab)

self.comboBox.setGeometry(QtCore.QRect(9, 28, 101, 20))

self.comboBox.setObjectName(\_fromUtf8("comboBox"))

self.comboBox.addItem(\_fromUtf8(""))

self.comboBox.addItem(\_fromUtf8(""))

self.comboBox.addItem(\_fromUtf8(""))

self.comboBox.addItem(\_fromUtf8(""))

self.comboBox.addItem(\_fromUtf8(""))

icon3 = QtGui.QIcon()

icon3.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-lock-26.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.tabWidget.addTab(self.Enc\_tab, icon3, \_fromUtf8(""))

self.Dec\_tab = QtGui.QWidget()

self.Dec\_tab.setObjectName(\_fromUtf8("Dec\_tab"))

self.comboBox\_2 = QtGui.QComboBox(self.Dec\_tab)

self.comboBox\_2.setGeometry(QtCore.QRect(9, 28, 101, 20))

self.comboBox\_2.setObjectName(\_fromUtf8("comboBox\_2"))

self.comboBox\_2.addItem(\_fromUtf8(""))

self.comboBox\_2.addItem(\_fromUtf8(""))

self.comboBox\_2.addItem(\_fromUtf8(""))

self.comboBox\_2.addItem(\_fromUtf8(""))

self.comboBox\_2.addItem(\_fromUtf8(""))

self.label\_5 = QtGui.QLabel(self.Dec\_tab)

self.label\_5.setGeometry(QtCore.QRect(9, 54, 30, 16))

self.label\_5.setObjectName(\_fromUtf8("label\_5"))

self.Dec\_Input = QtGui.QTextEdit(self.Dec\_tab)

self.Dec\_Input.setGeometry(QtCore.QRect(9, 73, 758, 189))

self.Dec\_Input.setObjectName(\_fromUtf8("Dec\_Input"))

self.label\_6 = QtGui.QLabel(self.Dec\_tab)

self.label\_6.setGeometry(QtCore.QRect(9, 268, 38, 16))

self.label\_6.setObjectName(\_fromUtf8("label\_6"))

self.Dec\_Output = QtGui.QTextEdit(self.Dec\_tab)

self.Dec\_Output.setGeometry(QtCore.QRect(9, 287, 758, 190))

self.Dec\_Output.setObjectName(\_fromUtf8("Dec\_Output"))

self.Dec\_Button = QtGui.QPushButton(self.Dec\_tab)

self.Dec\_Button.setGeometry(QtCore.QRect(360, 480, 75, 23))

self.Dec\_Button.setIcon(icon2)

self.Dec\_Button.setObjectName(\_fromUtf8("Dec\_Button"))

self.label\_4 = QtGui.QLabel(self.Dec\_tab)

self.label\_4.setGeometry(QtCore.QRect(9, 9, 131, 16))

self.label\_4.setObjectName(\_fromUtf8("label\_4"))

icon4 = QtGui.QIcon()

icon4.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-key-24.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.tabWidget.addTab(self.Dec\_tab, icon4, \_fromUtf8(""))

self.Hashing\_tab = QtGui.QWidget()

self.Hashing\_tab.setObjectName(\_fromUtf8("Hashing\_tab"))

self.Hash\_Input = QtGui.QTextEdit(self.Hashing\_tab)

self.Hash\_Input.setGeometry(QtCore.QRect(9, 73, 758, 187))

self.Hash\_Input.setObjectName(\_fromUtf8("Hash\_Input"))

self.Hash\_Output = QtGui.QTextEdit(self.Hashing\_tab)

self.Hash\_Output.setGeometry(QtCore.QRect(9, 285, 758, 188))

self.Hash\_Output.setObjectName(\_fromUtf8("Hash\_Output"))

self.Hash\_Button = QtGui.QPushButton(self.Hashing\_tab)

self.Hash\_Button.setGeometry(QtCore.QRect(350, 479, 75, 24))

self.Hash\_Button.setIcon(icon2)

self.Hash\_Button.setObjectName(\_fromUtf8("Hash\_Button"))

self.label\_9 = QtGui.QLabel(self.Hashing\_tab)

self.label\_9.setGeometry(QtCore.QRect(9, 266, 38, 16))

self.label\_9.setObjectName(\_fromUtf8("label\_9"))

self.label\_10 = QtGui.QLabel(self.Hashing\_tab)

self.label\_10.setGeometry(QtCore.QRect(9, 54, 30, 16))

self.label\_10.setObjectName(\_fromUtf8("label\_10"))

self.label\_11 = QtGui.QLabel(self.Hashing\_tab)

self.label\_11.setGeometry(QtCore.QRect(9, 9, 92, 16))

self.label\_11.setObjectName(\_fromUtf8("label\_11"))

self.comboBox\_3 = QtGui.QComboBox(self.Hashing\_tab)

self.comboBox\_3.setGeometry(QtCore.QRect(9, 28, 101, 20))

self.comboBox\_3.setObjectName(\_fromUtf8("comboBox\_3"))

self.comboBox\_3.addItem(\_fromUtf8(""))

self.comboBox\_3.addItem(\_fromUtf8(""))

self.comboBox\_3.addItem(\_fromUtf8(""))

self.comboBox\_3.addItem(\_fromUtf8(""))

self.comboBox\_3.addItem(\_fromUtf8(""))

self.comboBox\_3.addItem(\_fromUtf8(""))

self.comboBox\_3.addItem(\_fromUtf8(""))

icon5 = QtGui.QIcon()

icon5.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-hashtag-large-24.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.tabWidget.addTab(self.Hashing\_tab, icon5, \_fromUtf8(""))

self.HashIdentifier\_tab = QtGui.QWidget()

self.HashIdentifier\_tab.setObjectName(\_fromUtf8("HashIdentifier\_tab"))

self.verticalLayout\_2 = QtGui.QVBoxLayout(self.HashIdentifier\_tab)

self.verticalLayout\_2.setObjectName(\_fromUtf8("verticalLayout\_2"))

self.label\_8 = QtGui.QLabel(self.HashIdentifier\_tab)

self.label\_8.setObjectName(\_fromUtf8("label\_8"))

self.verticalLayout\_2.addWidget(self.label\_8)

self.HashId\_Input = QtGui.QTextEdit(self.HashIdentifier\_tab)

self.HashId\_Input.setObjectName(\_fromUtf8("HashId\_Input"))

self.verticalLayout\_2.addWidget(self.HashId\_Input)

self.label\_7 = QtGui.QLabel(self.HashIdentifier\_tab)

self.label\_7.setObjectName(\_fromUtf8("label\_7"))

self.verticalLayout\_2.addWidget(self.label\_7)

self.HashId\_Output = QtGui.QTextEdit(self.HashIdentifier\_tab)

self.HashId\_Output.setObjectName(\_fromUtf8("HashId\_Output"))

self.verticalLayout\_2.addWidget(self.HashId\_Output)

self.Analyse\_Button = QtGui.QPushButton(self.HashIdentifier\_tab)

self.Analyse\_Button.setIcon(icon2)

self.Analyse\_Button.setObjectName(\_fromUtf8("Analyse\_Button"))

self.verticalLayout\_2.addWidget(self.Analyse\_Button, QtCore.Qt.AlignHCenter)

icon6 = QtGui.QIcon()

icon6.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-search-filled-50.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.tabWidget.addTab(self.HashIdentifier\_tab, icon6, \_fromUtf8(""))

self.verticalLayout.addWidget(self.tabWidget)

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtGui.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 21))

self.menubar.setObjectName(\_fromUtf8("menubar"))

self.menuMenu = QtGui.QMenu(self.menubar)

self.menuMenu.setObjectName(\_fromUtf8("menuMenu"))

MainWindow.setMenuBar(self.menubar)

self.statusbar = QtGui.QStatusBar(MainWindow)

self.statusbar.setObjectName(\_fromUtf8("statusbar"))

MainWindow.setStatusBar(self.statusbar)

self.actionClose = QtGui.QAction(MainWindow)

icon7 = QtGui.QIcon()

icon7.addPixmap(QtGui.QPixmap(\_fromUtf8(":/images/images/icons8-delete-26.png")), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.actionClose.setIcon(icon7)

self.actionClose.setObjectName(\_fromUtf8("actionClose"))

self.menuMenu.addAction(self.actionClose)

self.menubar.addAction(self.menuMenu.menuAction())

self.retranslateUi(MainWindow)

self.tabWidget.setCurrentIndex(0)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow", None))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.Home\_tab), \_translate("MainWindow", "Home", None))

self.Enc\_Button.setText(\_translate("MainWindow", "Encrypt", None))

self.label.setText(\_translate("MainWindow", "Input:", None))

self.label\_2.setText(\_translate("MainWindow", "Output:", None))

self.label\_3.setText(\_translate("MainWindow", "Select from the drop down:", None))

self.comboBox.setItemText(0, \_translate("MainWindow", "-- Select --", None))

self.comboBox.setItemText(1, \_translate("MainWindow", "ROT13", None))

self.comboBox.setItemText(2, \_translate("MainWindow", "Base64", None))

self.comboBox.setItemText(3, \_translate("MainWindow", "Morse", None))

self.comboBox.setItemText(4, \_translate("MainWindow", "Ceaser", None))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.Enc\_tab), \_translate("MainWindow", "Encryption", None))

self.comboBox\_2.setItemText(0, \_translate("MainWindow", "-- Select --", None))

self.comboBox\_2.setItemText(1, \_translate("MainWindow", "ROT13", None))

self.comboBox\_2.setItemText(2, \_translate("MainWindow", "Base64", None))

self.comboBox\_2.setItemText(3, \_translate("MainWindow", "Morse", None))

self.comboBox\_2.setItemText(4, \_translate("MainWindow", "Ceaser", None))

self.label\_5.setText(\_translate("MainWindow", "Input:", None))

self.label\_6.setText(\_translate("MainWindow", "Output:", None))

self.Dec\_Button.setText(\_translate("MainWindow", "Decrypt", None))

self.label\_4.setText(\_translate("MainWindow", "Select from the drop down:", None))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.Dec\_tab), \_translate("MainWindow", "Decryption", None))

self.Hash\_Button.setText(\_translate("MainWindow", "Hash", None))

self.label\_9.setText(\_translate("MainWindow", "Output:", None))

self.label\_10.setText(\_translate("MainWindow", "Input:", None))

self.label\_11.setText(\_translate("MainWindow", "Select an HashAlgo:", None))

self.comboBox\_3.setItemText(0, \_translate("MainWindow", "-- Select --", None))

self.comboBox\_3.setItemText(1, \_translate("MainWindow", "MD5", None))

self.comboBox\_3.setItemText(2, \_translate("MainWindow", "SHA1", None))

self.comboBox\_3.setItemText(3, \_translate("MainWindow", "SHA224", None))

self.comboBox\_3.setItemText(4, \_translate("MainWindow", "SHA256", None))

self.comboBox\_3.setItemText(5, \_translate("MainWindow", "SHA384", None))

self.comboBox\_3.setItemText(6, \_translate("MainWindow", "SHA512", None))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.Hashing\_tab), \_translate("MainWindow", "Hashing", None))

self.label\_8.setText(\_translate("MainWindow", "Cipher Text:", None))

self.label\_7.setText(\_translate("MainWindow", "Possible Hashes:", None))

self.Analyse\_Button.setText(\_translate("MainWindow", "Analyse", None))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.HashIdentifier\_tab), \_translate("MainWindow", "HashIdentifier", None))

self.menuMenu.setTitle(\_translate("MainWindow", "Menu", None))

self.actionClose.setText(\_translate("MainWindow", "Close", None))

from PyQt4 import QtWebKit

import icons\_rc

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtGui.QApplication(sys.argv)

MainWindow = QtGui.QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

MainWindow.show()

sys.exit(app.exec\_())

##Code for Hash Identificaton:

import sys

from PyQt4 import QtGui

from main import Ui\_MainWindow

from builtins import input

from sys import argv, exit

class HashIdentifier(QtGui.QMainWindow):

def \_\_init\_\_(self):

super(HashIdentifier, self).\_\_init\_\_()

self.ui.setupUi(self)

def hashing(self):

algorithms={"102020":"ADLER-32", "102040":"CRC-32", "102060":"CRC-32B", "101020":"CRC-16", "101040":"CRC-16-CCITT", "104020":"DES(Unix)", "101060":"FCS-16", "103040":"GHash-32-3", "103020":"GHash-32-5", "115060":"GOST R 34.11-94", "109100":"Haval-160", "109200":"Haval-160(HMAC)", "110040":"Haval-192", "110080":"Haval-192(HMAC)", "114040":"Haval-224", "114080":"Haval-224(HMAC)", "115040":"Haval-256", "115140":"Haval-256(HMAC)", "107080":"Lineage II C4", "106025":"Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))", "102080":"XOR-32", "105060":"MD5(Half)", "105040":"MD5(Middle)", "105020":"MySQL", "107040":"MD5(phpBB3)", "107060":"MD5(Unix)", "107020":"MD5(Wordpress)", "108020":"MD5(APR)", "106160":"Haval-128", "106165":"Haval-128(HMAC)", "106060":"MD2", "106120":"MD2(HMAC)", "106040":"MD4", "106100":"MD4(HMAC)", "106020":"MD5", "106080":"MD5(HMAC)", "106140":"MD5(HMAC(Wordpress))", "106029":"NTLM", "106027":"RAdmin v2.x", "106180":"RipeMD-128", "106185":"RipeMD-128(HMAC)", "106200":"SNEFRU-128", "106205":"SNEFRU-128(HMAC)", "106220":"Tiger-128", "106225":"Tiger-128(HMAC)", "106240":"md5($pass.$salt)", "106260":"md5($salt.'-'.md5($pass))", "106280":"md5($salt.$pass)", "106300":"md5($salt.$pass.$salt)", "106320":"md5($salt.$pass.$username)", "106340":"md5($salt.md5($pass))", "106360":"md5($salt.md5($pass).$salt)", "106380":"md5($salt.md5($pass.$salt))", "106400":"md5($salt.md5($salt.$pass))", "106420":"md5($salt.md5(md5($pass).$salt))", "106440":"md5($username.0.$pass)", "106460":"md5($username.LF.$pass)", "106480":"md5($username.md5($pass).$salt)", "106500":"md5(md5($pass))", "106520":"md5(md5($pass).$salt)", "106540":"md5(md5($pass).md5($salt))", "106560":"md5(md5($salt).$pass)", "106580":"md5(md5($salt).md5($pass))", "106600":"md5(md5($username.$pass).$salt)", "106620":"md5(md5(md5($pass)))", "106640":"md5(md5(md5(md5($pass))))", "106660":"md5(md5(md5(md5(md5($pass)))))", "106680":"md5(sha1($pass))", "106700":"md5(sha1(md5($pass)))", "106720":"md5(sha1(md5(sha1($pass))))", "106740":"md5(strtoupper(md5($pass)))", "109040":"MySQL5 - SHA-1(SHA-1($pass))", "109060":"MySQL 160bit - SHA-1(SHA-1($pass))", "109180":"RipeMD-160(HMAC)", "109120":"RipeMD-160", "109020":"SHA-1", "109140":"SHA-1(HMAC)", "109220":"SHA-1(MaNGOS)", "109240":"SHA-1(MaNGOS2)", "109080":"Tiger-160", "109160":"Tiger-160(HMAC)", "109260":"sha1($pass.$salt)", "109280":"sha1($salt.$pass)", "109300":"sha1($salt.md5($pass))", "109320":"sha1($salt.md5($pass).$salt)", "109340":"sha1($salt.sha1($pass))", "109360":"sha1($salt.sha1($salt.sha1($pass)))", "109380":"sha1($username.$pass)", "109400":"sha1($username.$pass.$salt)", "1094202":"sha1(md5($pass))", "109440":"sha1(md5($pass).$salt)", "109460":"sha1(md5(sha1($pass)))", "109480":"sha1(sha1($pass))", "109500":"sha1(sha1($pass).$salt)", "109520":"sha1(sha1($pass).substr($pass,0,3))", "109540":"sha1(sha1($salt.$pass))", "109560":"sha1(sha1(sha1($pass)))", "109580":"sha1(strtolower($username).$pass)", "110020":"Tiger-192", "110060":"Tiger-192(HMAC)", "112020":"md5($pass.$salt) - Joomla", "113020":"SHA-1(Django)", "114020":"SHA-224", "114060":"SHA-224(HMAC)", "115080":"RipeMD-256", "115160":"RipeMD-256(HMAC)", "115100":"SNEFRU-256", "115180":"SNEFRU-256(HMAC)", "115200":"SHA-256(md5($pass))", "115220":"SHA-256(sha1($pass))", "115020":"SHA-256", "115120":"SHA-256(HMAC)", "116020":"md5($pass.$salt) - Joomla", "116040":"SAM - (LM\_hash:NT\_hash)", "117020":"SHA-256(Django)", "118020":"RipeMD-320", "118040":"RipeMD-320(HMAC)", "119020":"SHA-384", "119040":"SHA-384(HMAC)", "120020":"SHA-256", "121020":"SHA-384(Django)", "122020":"SHA-512", "122060":"SHA-512(HMAC)", "122040":"Whirlpool", "122080":"Whirlpool(HMAC)"}

def CRC16(hash):

hs='4607'

if len(hash)==len(hs) and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("101020")

def CRC16CCITT(hash):

hs='3d08'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("101040")

def FCS16(hash):

hs='0e5b'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("101060")

def CRC32(hash):

hs='b33fd057'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("102040")

def ADLER32(hash):

hs='0607cb42'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("102020")

def CRC32B(hash):

hs='b764a0d9'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("102060")

def XOR32(hash):

hs='0000003f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("102080")

def GHash323(hash):

hs='80000000'

if len(hash)==len(hs) and hash.isdigit()==True and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("103040")

def GHash325(hash):

hs='85318985'

if len(hash)==len(hs) and hash.isdigit()==True and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("103020")

def DESUnix(hash):

hs='ZiY8YtDKXJwYQ'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False:

jerar.append("104020")

def MD5Half(hash):

hs='ae11fd697ec92c7c'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("105060")

def MD5Middle(hash):

hs='7ec92c7c98de3fac'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("105040")

def MySQL(hash):

hs='63cea4673fd25f46'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("105020")

def DomainCachedCredentials(hash):

hs='f42005ec1afe77967cbc83dce1b4d714'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106025")

def Haval128(hash):

hs='d6e3ec49aa0f138a619f27609022df10'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106160")

def Haval128HMAC(hash):

hs='3ce8b0ffd75bc240fc7d967729cd6637'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106165")

def MD2(hash):

hs='08bbef4754d98806c373f2cd7d9a43c4'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106060")

def MD2HMAC(hash):

hs='4b61b72ead2b0eb0fa3b8a56556a6dca'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106120")

def MD4(hash):

hs='a2acde400e61410e79dacbdfc3413151'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106040")

def MD4HMAC(hash):

hs='6be20b66f2211fe937294c1c95d1cd4f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106100")

def MD5(hash):

hs='ae11fd697ec92c7c98de3fac23aba525'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106020")

def MD5HMAC(hash):

hs='d57e43d2c7e397bf788f66541d6fdef9'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106080")

def MD5HMACWordpress(hash):

hs='3f47886719268dfa83468630948228f6'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106140")

def NTLM(hash):

hs='cc348bace876ea440a28ddaeb9fd3550'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106029")

def RAdminv2x(hash):

hs='baea31c728cbf0cd548476aa687add4b'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106027")

def RipeMD128(hash):

hs='4985351cd74aff0abc5a75a0c8a54115'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106180")

def RipeMD128HMAC(hash):

hs='ae1995b931cf4cbcf1ac6fbf1a83d1d3'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106185")

def SNEFRU128(hash):

hs='4fb58702b617ac4f7ca87ec77b93da8a'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106200")

def SNEFRU128HMAC(hash):

hs='59b2b9dcc7a9a7d089cecf1b83520350'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106205")

def Tiger128(hash):

hs='c086184486ec6388ff81ec9f23528727'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106220")

def Tiger128HMAC(hash):

hs='c87032009e7c4b2ea27eb6f99723454b'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106225")

def md5passsalt(hash):

hs='5634cc3b922578434d6e9342ff5913f7'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106240")

def md5saltmd5pass(hash):

hs='245c5763b95ba42d4b02d44bbcd916f1'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106260")

def md5saltpass(hash):

hs='22cc5ce1a1ef747cd3fa06106c148dfa'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106280")

def md5saltpasssalt(hash):

hs='469e9cdcaff745460595a7a386c4db0c'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106300")

def md5saltpassusername(hash):

hs='9ae20f88189f6e3a62711608ddb6f5fd'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106320")

def md5saltmd5pass(hash):

hs='aca2a052962b2564027ee62933d2382f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106340")

def md5saltmd5passsalt(hash):

hs='de0237dc03a8efdf6552fbe7788b2fdd'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106360")

def md5saltmd5passsalt(hash):

hs='5b8b12ca69d3e7b2a3e2308e7bef3e6f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106380")

def md5saltmd5saltpass(hash):

hs='d8f3b3f004d387086aae24326b575b23'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106400")

def md5saltmd5md5passsalt(hash):

hs='81f181454e23319779b03d74d062b1a2'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106420")

def md5username0pass(hash):

hs='e44a60f8f2106492ae16581c91edb3ba'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106440")

def md5usernameLFpass(hash):

hs='654741780db415732eaee12b1b909119'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106460")

def md5usernamemd5passsalt(hash):

hs='954ac5505fd1843bbb97d1b2cda0b98f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106480")

def md5md5pass(hash):

hs='a96103d267d024583d5565436e52dfb3'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106500")

def md5md5passsalt(hash):

hs='5848c73c2482d3c2c7b6af134ed8dd89'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106520")

def md5md5passmd5salt(hash):

hs='8dc71ef37197b2edba02d48c30217b32'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106540")

def md5md5saltpass(hash):

hs='9032fabd905e273b9ceb1e124631bd67'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106560")

def md5md5saltmd5pass(hash):

hs='8966f37dbb4aca377a71a9d3d09cd1ac'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106580")

def md5md5usernamepasssalt(hash):

hs='4319a3befce729b34c3105dbc29d0c40'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106600")

def md5md5md5pass(hash):

hs='ea086739755920e732d0f4d8c1b6ad8d'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106620")

def md5md5md5md5pass(hash):

hs='02528c1f2ed8ac7d83fe76f3cf1c133f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106640")

def md5md5md5md5md5pass(hash):

hs='4548d2c062933dff53928fd4ae427fc0'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106660")

def md5sha1pass(hash):

hs='cb4ebaaedfd536d965c452d9569a6b1e'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106680")

def md5sha1md5pass(hash):

hs='099b8a59795e07c334a696a10c0ebce0'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106700")

def md5sha1md5sha1pass(hash):

hs='06e4af76833da7cc138d90602ef80070'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106720")

def md5strtouppermd5pass(hash):

hs='519de146f1a658ab5e5e2aa9b7d2eec8'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("106740")

def LineageIIC4(hash):

hs='0x49a57f66bd3d5ba6abda5579c264a0e4'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True and hash[0:2].find('0x')==0:

jerar.append("107080")

def MD5phpBB3(hash):

hs='$H$9kyOtE8CDqMJ44yfn9PFz2E.L2oVzL1'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:3].find('$H$')==0:

jerar.append("107040")

def MD5Unix(hash):

hs='$1$cTuJH0Ju$1J8rI.mJReeMvpKUZbSlY/'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:3].find('$1$')==0:

jerar.append("107060")

def MD5Wordpress(hash):

hs='$P$BiTOhOj3ukMgCci2juN0HRbCdDRqeh.'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:3].find('$P$')==0:

jerar.append("107020")

def MD5APR(hash):

hs='$apr1$qAUKoKlG$3LuCncByN76eLxZAh/Ldr1'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash[0:4].find('$apr')==0:

jerar.append("108020")

def Haval160(hash):

hs='a106e921284dd69dad06192a4411ec32fce83dbb'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109100")

def Haval160HMAC(hash):

hs='29206f83edc1d6c3f680ff11276ec20642881243'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109200")

def MySQL5(hash):

hs='9bb2fb57063821c762cc009f7584ddae9da431ff'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109040")

def MySQL160bit(hash):

hs='\*2470c0c06dee42fd1618bb99005adca2ec9d1e19'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:1].find('\*')==0:

jerar.append("109060")

def RipeMD160(hash):

hs='dc65552812c66997ea7320ddfb51f5625d74721b'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109120")

def RipeMD160HMAC(hash):

hs='ca28af47653b4f21e96c1235984cb50229331359'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109180")

def SHA1(hash):

hs='4a1d4dbc1e193ec3ab2e9213876ceb8f4db72333'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109020")

def SHA1HMAC(hash):

hs='6f5daac3fee96ba1382a09b1ba326ca73dccf9e7'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109140")

def SHA1MaNGOS(hash):

hs='a2c0cdb6d1ebd1b9f85c6e25e0f8732e88f02f96'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109220")

def SHA1MaNGOS2(hash):

hs='644a29679136e09d0bd99dfd9e8c5be84108b5fd'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109240")

def Tiger160(hash):

hs='c086184486ec6388ff81ec9f235287270429b225'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109080")

def Tiger160HMAC(hash):

hs='6603161719da5e56e1866e4f61f79496334e6a10'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109160")

def sha1passsalt(hash):

hs='f006a1863663c21c541c8d600355abfeeaadb5e4'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109260")

def sha1saltpass(hash):

hs='299c3d65a0dcab1fc38421783d64d0ecf4113448'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109280")

def sha1saltmd5pass(hash):

hs='860465ede0625deebb4fbbedcb0db9dc65faec30'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109300")

def sha1saltmd5passsalt(hash):

hs='6716d047c98c25a9c2cc54ee6134c73e6315a0ff'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109320")

def sha1saltsha1pass(hash):

hs='58714327f9407097c64032a2fd5bff3a260cb85f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109340")

def sha1saltsha1saltsha1pass(hash):

hs='cc600a2903130c945aa178396910135cc7f93c63'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109360")

def sha1usernamepass(hash):

hs='3de3d8093bf04b8eb5f595bc2da3f37358522c9f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109380")

def sha1usernamepasssalt(hash):

hs='00025111b3c4d0ac1635558ce2393f77e94770c5'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109400")

def sha1md5pass(hash):

hs='fa960056c0dea57de94776d3759fb555a15cae87'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("1094202")

def sha1md5passsalt(hash):

hs='1dad2b71432d83312e61d25aeb627593295bcc9a'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109440")

def sha1md5sha1pass(hash):

hs='8bceaeed74c17571c15cdb9494e992db3c263695'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109460")

def sha1sha1pass(hash):

hs='3109b810188fcde0900f9907d2ebcaa10277d10e'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109480")

def sha1sha1passsalt(hash):

hs='780d43fa11693b61875321b6b54905ee488d7760'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109500")

def sha1sha1passsubstrpass03(hash):

hs='5ed6bc680b59c580db4a38df307bd4621759324e'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109520")

def sha1sha1saltpass(hash):

hs='70506bac605485b4143ca114cbd4a3580d76a413'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109540")

def sha1sha1sha1pass(hash):

hs='3328ee2a3b4bf41805bd6aab8e894a992fa91549'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109560")

def sha1strtolowerusernamepass(hash):

hs='79f575543061e158c2da3799f999eb7c95261f07'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("109580")

def Haval192(hash):

hs='cd3a90a3bebd3fa6b6797eba5dab8441f16a7dfa96c6e641'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("110040")

def Haval192HMAC(hash):

hs='39b4d8ecf70534e2fd86bb04a877d01dbf9387e640366029'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("110080")

def Tiger192(hash):

hs='c086184486ec6388ff81ec9f235287270429b2253b248a70'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("110020")

def Tiger192HMAC(hash):

hs='8e914bb64353d4d29ab680e693272d0bd38023afa3943a41'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("110060")

def MD5passsaltjoomla1(hash):

hs='35d1c0d69a2df62be2df13b087343dc9:BeKMviAfcXeTPTlX'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[32:33].find(':')==0:

jerar.append("112020")

def SHA1Django(hash):

hs='sha1$Zion3R$299c3d65a0dcab1fc38421783d64d0ecf4113448'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:5].find('sha1$')==0:

jerar.append("113020")

def Haval224(hash):

hs='f65d3c0ef6c56f4c74ea884815414c24dbf0195635b550f47eac651a'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("114040")

def Haval224HMAC(hash):

hs='f10de2518a9f7aed5cf09b455112114d18487f0c894e349c3c76a681'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("114080")

def SHA224(hash):

hs='e301f414993d5ec2bd1d780688d37fe41512f8b57f6923d054ef8e59'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("114020")

def SHA224HMAC(hash):

hs='c15ff86a859892b5e95cdfd50af17d05268824a6c9caaa54e4bf1514'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("114060")

def SHA256(hash):

hs='2c740d20dab7f14ec30510a11f8fd78b82bc3a711abe8a993acdb323e78e6d5e'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115020")

def SHA256HMAC(hash):

hs='d3dd251b7668b8b6c12e639c681e88f2c9b81105ef41caccb25fcde7673a1132'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115120")

def Haval256(hash):

hs='7169ecae19a5cd729f6e9574228b8b3c91699175324e6222dec569d4281d4a4a'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115040")

def Haval256HMAC(hash):

hs='6aa856a2cfd349fb4ee781749d2d92a1ba2d38866e337a4a1db907654d4d4d7a'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115140")

def GOSTR341194(hash):

hs='ab709d384cce5fda0793becd3da0cb6a926c86a8f3460efb471adddee1c63793'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115060")

def RipeMD256(hash):

hs='5fcbe06df20ce8ee16e92542e591bdea706fbdc2442aecbf42c223f4461a12af'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115080")

def RipeMD256HMAC(hash):

hs='43227322be1b8d743e004c628e0042184f1288f27c13155412f08beeee0e54bf'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115160")

def SNEFRU256(hash):

hs='3a654de48e8d6b669258b2d33fe6fb179356083eed6ff67e27c5ebfa4d9732bb'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115100")

def SNEFRU256HMAC(hash):

hs='4e9418436e301a488f675c9508a2d518d8f8f99e966136f2dd7e308b194d74f9'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115180")

def SHA256md5pass(hash):

hs='b419557099cfa18a86d1d693e2b3b3e979e7a5aba361d9c4ec585a1a70c7bde4'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115200")

def SHA256sha1pass(hash):

hs='afbed6e0c79338dbfe0000efe6b8e74e3b7121fe73c383ae22f5b505cb39c886'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("115220")

def MD5passsaltjoomla2(hash):

hs='fb33e01e4f8787dc8beb93dac4107209:fxJUXVjYRafVauT77Cze8XwFrWaeAYB2'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[32:33].find(':')==0:

jerar.append("116020")

def SAM(hash):

hs='4318B176C3D8E3DEAAD3B435B51404EE:B7C899154197E8A2A33121D76A240AB5'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash.islower()==False and hash[32:33].find(':')==0:

jerar.append("116040")

def SHA256Django(hash):

hs='sha256$Zion3R$9e1a08aa28a22dfff722fad7517bae68a55444bb5e2f909d340767cec9acf2c3'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:6].find('sha256')==0:

jerar.append("117020")

def RipeMD320(hash):

hs='b4f7c8993a389eac4f421b9b3b2bfb3a241d05949324a8dab1286069a18de69aaf5ecc3c2009d8ef'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("118020")

def RipeMD320HMAC(hash):

hs='244516688f8ad7dd625836c0d0bfc3a888854f7c0161f01de81351f61e98807dcd55b39ffe5d7a78'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("118040")

def SHA384(hash):

hs='3b21c44f8d830fa55ee9328a7713c6aad548fe6d7a4a438723a0da67c48c485220081a2fbc3e8c17fd9bd65f8d4b4e6b'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("119020")

def SHA384HMAC(hash):

hs='bef0dd791e814d28b4115eb6924a10beb53da47d463171fe8e63f68207521a4171219bb91d0580bca37b0f96fddeeb8b'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("119040")

def SHA256s(hash):

hs='$6$g4TpUQzk$OmsZBJFwvy6MwZckPvVYfDnwsgktm2CckOlNJGy9HNwHSuHFvywGIuwkJ6Bjn3kKbB6zoyEjIYNMpHWBNxJ6g.'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:3].find('$6$')==0:

jerar.append("120020")

def SHA384Django(hash):

hs='sha384$Zion3R$88cfd5bc332a4af9f09aa33a1593f24eddc01de00b84395765193c3887f4deac46dc723ac14ddeb4d3a9b958816b7bba'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==False and hash[0:6].find('sha384')==0:

jerar.append("121020")

def SHA512(hash):

hs='ea8e6f0935b34e2e6573b89c0856c81b831ef2cadfdee9f44eb9aa0955155ba5e8dd97f85c73f030666846773c91404fb0e12fb38936c56f8cf38a33ac89a24e'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("122020")

def SHA512HMAC(hash):

hs='dd0ada8693250b31d9f44f3ec2d4a106003a6ce67eaa92e384b356d1b4ef6d66a818d47c1f3a2c6e8a9a9b9bdbd28d485e06161ccd0f528c8bbb5541c3fef36f'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("122060")

def Whirlpool(hash):

hs='76df96157e632410998ad7f823d82930f79a96578acc8ac5ce1bfc34346cf64b4610aefa8a549da3f0c1da36dad314927cebf8ca6f3fcd0649d363c5a370dddb'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("122040")

def WhirlpoolHMAC(hash):

hs='77996016cf6111e97d6ad31484bab1bf7de7b7ee64aebbc243e650a75a2f9256cef104e504d3cf29405888fca5a231fcac85d36cd614b1d52fce850b53ddf7f9'

if len(hash)==len(hs) and hash.isdigit()==False and hash.isalpha()==False and hash.isalnum()==True:

jerar.append("122080")

jerar=[]

x = []

try:

first = str(argv[1])

except:

first = None

if first:

h = first

else:

h = self.ui.HashId\_Input.toPlainText()

ADLER32(h); CRC16(h); CRC16CCITT(h); CRC32(h); CRC32B(h); DESUnix(h); DomainCachedCredentials(h); FCS16(h); GHash323(h); GHash325(h); GOSTR341194(h); Haval128(h); Haval128HMAC(h); Haval160(h); Haval160HMAC(h); Haval192(h); Haval192HMAC(h); Haval224(h); Haval224HMAC(h); Haval256(h); Haval256HMAC(h); LineageIIC4(h); MD2(h); MD2HMAC(h); MD4(h); MD4HMAC(h); MD5(h); MD5APR(h); MD5HMAC(h); MD5HMACWordpress(h); MD5phpBB3(h); MD5Unix(h); MD5Wordpress(h); MD5Half(h); MD5Middle(h); MD5passsaltjoomla1(h); MD5passsaltjoomla2(h); MySQL(h); MySQL5(h); MySQL160bit(h); NTLM(h); RAdminv2x(h); RipeMD128(h); RipeMD128HMAC(h); RipeMD160(h); RipeMD160HMAC(h); RipeMD256(h); RipeMD256HMAC(h); RipeMD320(h); RipeMD320HMAC(h); SAM(h); SHA1(h); SHA1Django(h); SHA1HMAC(h); SHA1MaNGOS(h); SHA1MaNGOS2(h); SHA224(h); SHA224HMAC(h); SHA256(h); SHA256s(h); SHA256Django(h); SHA256HMAC(h); SHA256md5pass(h); SHA256sha1pass(h); SHA384(h); SHA384Django(h); SHA384HMAC(h); SHA512(h); SHA512HMAC(h); SNEFRU128(h); SNEFRU128HMAC(h); SNEFRU256(h); SNEFRU256HMAC(h); Tiger128(h); Tiger128HMAC(h); Tiger160(h); Tiger160HMAC(h); Tiger192(h); Tiger192HMAC(h); Whirlpool(h); WhirlpoolHMAC(h); XOR32(h); md5passsalt(h); md5saltmd5pass(h); md5saltpass(h); md5saltpasssalt(h); md5saltpassusername(h); md5saltmd5pass(h); md5saltmd5passsalt(h); md5saltmd5passsalt(h); md5saltmd5saltpass(h); md5saltmd5md5passsalt(h); md5username0pass(h); md5usernameLFpass(h); md5usernamemd5passsalt(h); md5md5pass(h); md5md5passsalt(h); md5md5passmd5salt(h); md5md5saltpass(h); md5md5saltmd5pass(h); md5md5usernamepasssalt(h); md5md5md5pass(h); md5md5md5md5pass(h); md5md5md5md5md5pass(h); md5sha1pass(h); md5sha1md5pass(h); md5sha1md5sha1pass(h); md5strtouppermd5pass(h); sha1passsalt(h); sha1saltpass(h); sha1saltmd5pass(h); sha1saltmd5passsalt(h); sha1saltsha1pass(h); sha1saltsha1saltsha1pass(h); sha1usernamepass(h); sha1usernamepasssalt(h); sha1md5pass(h); sha1md5passsalt(h); sha1md5sha1pass(h); sha1sha1pass(h); sha1sha1passsalt(h); sha1sha1passsubstrpass03(h); sha1sha1saltpass(h); sha1sha1sha1pass(h); sha1strtolowerusernamepass(h)

if len(jerar)==0:

self.ui.HashId\_Output.setPlainText("\n Not Found.")

elif len(jerar)>2:

jerar.sort()

self.ui.HashId\_Output.append("\nPossible Hashs:")

self.ui.HashId\_Output.append("[+] "+str(algorithms[jerar[0]]))

self.ui.HashId\_Output.append("[+] "+str(algorithms[jerar[1]]))

self.ui.HashId\_Output.append("\nLeast Possible Hashs:")

for a in range(int(len(jerar))-2):

self.ui.HashId\_Output.append("[+] "+str(algorithms[jerar[a+2]]))

else:

jerar.sort()

self.ui.HashId\_Output.append("\nPossible Hashs:")

for a in range(len(jerar)):

self.ui.HashId\_Output.append("[+] "+str(algorithms[jerar[a]]))

first = None