A Secure Communicating MD5 based DICOM Annotation tool

Sweta Dey
Computer Science & Engineering
IIT Ropar
Ropar-140001, India
swetadey.iitcse@gmail.com

***Abstract*— DICOM proceeds through a volume of .dcm files and maintaining all standard as well but still it is not free from attacks. In this paper I present the RTStructure based GUI and the security model of DICOM communication. Here, I use massage digest with the metadata of each volume of DICOM. Metadata is nothing but data about data and it is required for constructing each volume of .dcm files. In my case, it is standardized and static with the help of python packages. After successful construction .dcm files, cryptographic hash function is applied. Hence, a secure RT-Structured based annotation model is ready for storing and accessing medical information among various medical devices by maintaining HIPPA and PACS standard.**

Keywords*— *Metadata, DICOM, PACS, HIPPA, IaaS, MD5, grid structure.

I. INTRODUCTION

DICOM is an acronym for **Digital Imaging and Communications in Medicine**. DICOM has two major parts namely, Header and DataSet.

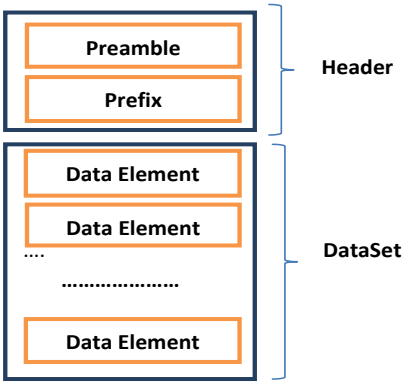


Fig. 1. Construction of DICOM

Where, Header is the collection of Preamble (i.e. constructed by 128 bytes), and 4 bytes of Prefix. DataSet is the collection of Data Elements; where each of the Data Element (DE) consists of Value Multiplicity(VM), Data Element Tag, Value Representation(VR), Value Length(VL) etc.VM of DE specifies the total no. of values that can be encoded in the value field of that DE. Tag is the collection of group number and element number. We have three types of tags, and from those type1 and type2 tags are mandatory; type3 tag is optional.

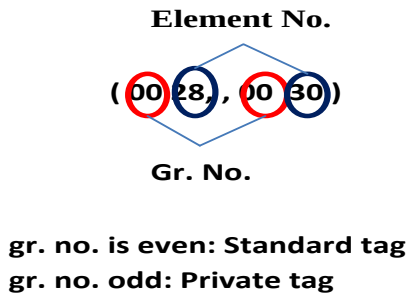


Fig. 2. Configuration of DICOM tags

DICOM proceeds by both of its public and private tags and simultaneously maintains all the standards though making a secure RT-Structured based annotation construction of DICOM is a big challenge. PACS maintains some of the security concerns but DICOM itself and medical devices are not free from attacks (e.g. bio attacks, cryptographic attacks, cyber-attacks etc.). To get rid from these issues my model use Little Endian Transfer Syntax UID, Contour Geometry, grid structure with the help of python "gdcm" package, cryptographic hash function, and the **Structure Set** of the **DICOM-RT** objects.

Grid structure is constructed in such a way, so that it can able to accommodate the objects of .dcm files and process the scientific applications effectively as well. Here, my main goal is to use grid structure is to grid my standardized metadata from a huge dataset. Hence, it can easily construct .dcm file and hence each volume of DICOM.

Cloud Computing [1], [2], [3], [4], [5] is also a well-known mathematical tool which deploys various kinds of service models (i.e. Software as a Service (SaaS), Plateform as a Service(PaaS), Infrastructure as a Service (IaaS)),and each of these model has their own services. Here, I use IaaS model for making the resources infinite in terms of virtual machines (VMs) i.e. it uses replication. With the help of these VMs, cloudlets, brokers (all of these are cloud entities which are used for constructing communication among medical devices);I can able to connect each volume of .dcm files with the environment and finally able to combine all these required entities in python environment by using " PiCloud" and "dispy" library packages.

After a successful construction of the required volume of DICOM; I can able to establish a communication among

various medical devices by using python packages as well as the anaconda environment [6]. But, this communication is not a safe because of various attacks like bio attacks as well as cryptographic attacks. So, to get remove from these issues I applied cryptographic hash function.

In general, Bio attacks [7] are caused by bioterrorism and this terrorism is the end result of the intentional usage of biological, chemical, nuclear, or radiological agents of the bio attackers and they can also use the metadata of DICOM for successful attack. They always start their Sampling from a very large data set of active neurons (because human brain contains billions of neurons) from metadata. Various kinds of Chemical and biological weapons can cause a wide range of nervous system damage and neurobehavioral effects; so human brains are not free from these attacks.

For establishing an efficient secure communication; the requirements are each volume .dcm file, RT-Structure set and medical devices but these devices as well as the volumes of .dcm files are not free from attacks like cyber-attack[8], [9]. Cyber-attack [8], [9] is a kind of attack by which attacker can attack on the medical devices and forge the functionality of these crucial devices such as implants, exposing the medical records of patients, and also potentially granting the access of the prescription infrastructure by modifying the RT-Structure set of many repute organizations for some illicit activities. This kind of attack can also have the potential to make the operating system of the devices more vulnerable. So that it can be embedded on that device only. It can also be able to hide itself from various sensitive sandboxes as well as other defensive tools in a safe environment.

So, to establish a secure communication among various medical devices; I use cryptographic one-way hash function [10] inside each volume of DICOM. I used MD5 (Message Digest 5) [11] as a cryptographic hash function.

The road map of this paper is organized as follows: **Section 2** describes the related works. **Section 3** describes the problem formulation. **Section 4** describes proposed model, in **Section 5**, I conclude my paper, and **Section** 6 present future works.

II. RELATED WORKS

DICOM is a standard for .dcm files and medical images as well. It is actually used for all the kind of modification of medical images and data as well. It supports various modules of DICOM; such as, CT, MR, X-ray, PET, radiotherapy (RT) etc. In this paper, I mainly concentrated on RTStructureSet module since I am interested on CT images. In DICOM standard, we have lot many RT Structure based modules such as: RT Image, RT Dose, RTSTRUCT, RT Plan and RT Treatment Record etc. These modules are required in .dcm files in form of volume because DICOM is a globally accepted standard. DICOM users are familiar with RTSTRUCT module. So whenever they want to identify some therapy details of the patient they use to write the contour data with the help of contour geometry. The usage of RTSTRUCT module is to address the requirements for transfer of patient structures.

A. RTSTRUCT

DICOM standard [12], [13] provides all the available modules of .dcm file. My work is mainly based on the RTSTRUCT module. Here, I present some of the mandatory objects of RTSTRUCT module in Table 1which is used for the construction of my model.

TABLE I. CONTAINS MANDATORY OBJECTS OF RTSTRUCT MODULE.

Patient's Information	**Mandatory Objects**
Patient	(i) Patient
Study	(ii) General Study
Series	(iii) RT Series
Equipment	(iv) General Equipment
Structure Set	(v) Structure Set (vi) ROI Contour (vii) RT ROI Observations (viii) SOP Common

Table 1: Mandatory objects of RTSTRUCT module

My model is also follows the contour geometry for contour construction [14]. These are described as follows:

POINT type geometry: When it deals with only a single point

OPEN_PLANAR: When it deals with coplanar points

OPEN_NONPLANAR: When it deals with non-coplanar points

CLOSED_PLANAR: When it deals with closed coplanar points

For making the model secure, I used hashing function namely MD5 [11] inside every medical devices. So that, patient's details are safe form the attacker. Hence, these devices are protected and my model can able to construct a successful efficient communication.

B. MD5[11]

Message digest functions are also known as hash functions and these functions are mostly used for reproducing the digital summaries of information called message digests. These functions are mostly 128 to 160 bits long in length and with the help of this length they are able to provide a digital identifier. These hash functions are a kind of special mathematical functions which are able to process some computational dynamic information to produce a different kind of more secure one way hash functions for each unique computational resource. Same kinds of resources have the same hash functions though any one bit of the messages has been changed. It is done just because of the message digest functions. These functions are much shorter as compared to the original computational resources which are actually generated as digests and then it has a length of finite measurement. Similarly the copied message digests are known as collisions. However, a good message digest function can be used as a one-way hash function, which is mathematically and computationally proves that reverse of the message digest process and discovering the original resources both are infeasible to in reality. However, it is infeasible for attacker also.

III. PROBLEM FORMULATION

Table2 contains notation of my proposed model.

TABLE II. NOTATION OF THE MODEL

Symbol	Meaning
px	x-coordinate of the voxel (i, j) in the frame's image plane in mm
py	y-coordinate of the voxel (i, j) in the frame's image plane in mm
pz	z-coordinate of the voxel (i, j) in the frame's image plane in mm
Xx, Xy, Xz	Values from the row (X) direction cosine of ImageOrientationPatient.
Yx, Yy, Yz	Values from the column (Y) direction cosine of ImageOrientationPatient
i	Column index.
j	Row index.
Δi	Column pixel resolution of PixelSpacing in mm.
Δj	Row pixel resolution of PixelSpacing in mm.
im	Hash value of DICOM
h	MD5 one way hashing algorithm

For constructing the annotation tool; first I create the header of .dcm file with the help of the metadata. Then after, RTStructure Set has been constructed. For annotation, contour vector geometry is used:

$$\begin{bmatrix} px \\ py \\ pz \\ 1 \end{bmatrix} = \begin{bmatrix} Xx\Delta i & Yx\Delta j & 0 & Sx \\ Xy\Delta i & Yy\Delta j & 0 & Sy \\ Xz\Delta i & Yy\Delta j & 0 & Sz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} i \\ j \\ 0 \\ 1 \end{bmatrix}$$

After successful construction of RTStructure Set; I made a secure communication among various medical devices.

IV. PROPOSED MODEL

In this section, my proposed methodology is delineated in a detail.

To achieve the goal, my proposed model has three phases. Phase1 is responsible for the construction of .dcm files. Phase2 is responsible for the construction of contours, Phase3 is responsible for the construction of RTStructure set, and Phase4 is responsible for the construction of a secure communication among various medical devices.

Phase1:

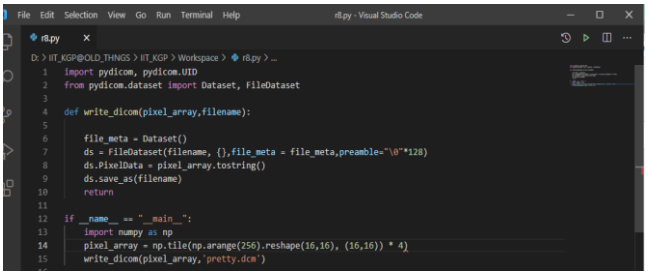
Construction of .dcm files:

Since, DICOM is totally a different format among the others. Here, I used CT images for my experiment. DICOM proceeds by gathering the information into a dataset, and then after, the header of DICOM, and image datasets are then packed into a single file in form of a volume of a .dcm file. Then finally, these .dcm files are shared via internet. The information within the header is organized as a constant and standardized series of tags. So, Construction of .dcm files can further has two sub phases:

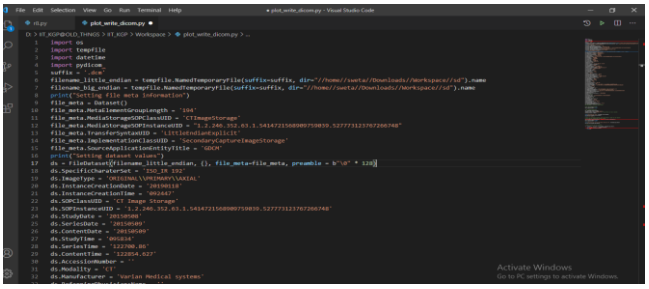
Sub phase1:

Construction of header:

Step1: Create 128 preamble bytes and 4 prefix bytes ('D', 'I', 'C', 'M').



Step2: Add type 1 and type 2 tags within the dataset.



Sub phase2:

Construction of dataset:

Step 1: Create the dataset with the help of type1, type2, and type3 tags.

Step2: Pack header and dataset into a single volume of .dcm file.

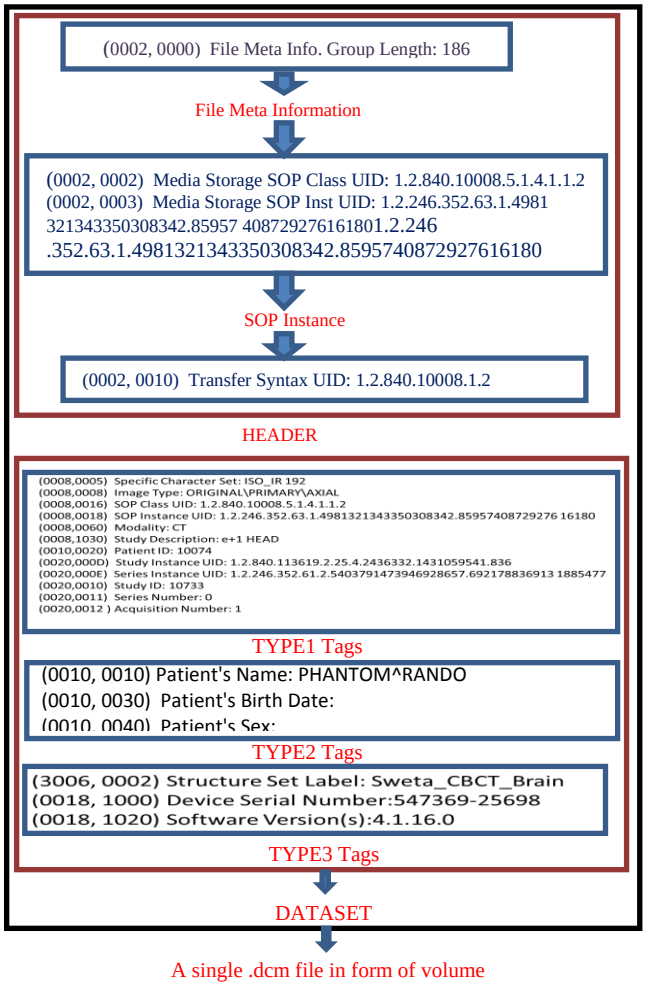


Fig. 3. Block diagram of phase1

```
ds.save_as(filename_little_endian)
print("File saved.")

# Write as a different transfer syntax XXX shouldn't need this but pydicom
# 0.9.5 bug not recognizing transfer syntax
ds.file_meta.TransferSyntaxUID = pydicom.uid.ExplicitVRBigEndian
ds.is_little_endian = False
ds.is_implicit_VR = False

print("Writing test file as Big Endian Explicit VR", filename_big_endian)
ds.save_as(filename_big_endian)

# reopen the data just for checking

for filename in (filename_little_endian, filename_big_endian):
    print('Load file {} ...'.format(filename))
    ds = pydicom.dcmread(filename)
    print(ds)
```

Phase2:

Construction of Contours:

Contours can be constructed by using the following formulae:

Step1: Convert pixel value to voxel value using the following formula:

$$\begin{bmatrix} px \\ py \\ pz \\ 1 \end{bmatrix} = \begin{bmatrix} Xx\Delta i & Yx\Delta j & 0 & Sx \\ Xy\Delta i & Yy\Delta j & 0 & Sy \\ Xz\Delta i & Yy\Delta j & 0 & Sz \\ 0 & 0 & 0 & 1 \end{bmatrix} = M \begin{bmatrix} i \\ j \\ 0 \\ 1 \end{bmatrix}$$

Step2: Join those voxel points using contour vector geometry. After successful joining of those voxel points; phase2 has been completed.

```
############## ROI CONTOUR 1
de = gdcm.DataElement(gdcm.Tag(0x3006, 0x002a))
ROIDisplayColor = "255\\0\\0"
de.SetByteValue(ROIDisplayColor, gdcm.VL(len(ROIDisplayColor)))
de.SetVR(gdcm.VR(gdcm.VR.IS))

de1 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0084))
ReferencedROINumber = "1"
de1.SetByteValue(ReferencedROINumber, gdcm.VL(len(ReferencedROINumber)))
de1.SetVR(gdcm.VR(gdcm.VR.IS))

it=gdcm.Item()
it.SetVLToUndefined()
nds=it.GetNestedDataSet()
nds.Insert(de)
nds.Insert(de1)
#######################
sde = gdcm.DataElement(gdcm.Tag(0x3006, 0x0042))
ContourGeometricType = "POINT"
sde.SetByteValue(ContourGeometricType, gdcm.VL(len(ContourGeometricType)))
sde.SetVR(gdcm.VR(gdcm.VR.CS))

try:
    sde1 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0046))
    NumberOfContourPoints = open("/home/sweta/Downloads/Workspace/count34.txt", "r").read()
    sde1.SetByteValue(NumberOfContourPoints, gdcm.VL(len(NumberOfContourPoints)))
    sde1.SetVR(gdcm.VR(gdcm.VR.IS))
```

Phase3:

Construction of RTStructure Set:

RT Structure is created by following two step mechanisms:

Step1: Creation of empty .dcm dataset (for reference):

Structure Set is one of the **DICOM-RT** objects and it includes **Structure Set**, **ROI Contour** and **RT ROI Observations** modules. All these modules first create the Data Elements (using type of tags) for a single volume of .dcm file.

```
def OnButtonClickedsave(self, evt):

    r = gdcm.Reader()
    filename = os.path.join( '/home/sweta/Downloads/Workspace/MyHead/blankdataset.dcm' )
    r.SetFileName( filename )
    r.Read()
    f = r.GetFile()
    ds = f.GetDataSet()
    uid = "1.2.840.10008.5.1.4.1.1.481.3"
    de = gdcm.DataElement( gdcm.Tag(0x0008,0x0016) )
    de.SetByteValue( uid, gdcm.VL(len(uid)) )
    vr = gdcm.VR( gdcm.VR.UI )
    de.SetVR( vr )
    ds.Replace( de )

    ano = gdcm.Anonymizer()
    ano.SetFile( r.GetFile() )

    gen = gdcm.UIDGenerator()

    uid = gen.Generate()
    de.SetTag( gdcm.Tag(0x0008,0x0018) )
    de.SetByteValue( uid, gdcm.VL(len(uid)) )
    ds.Insert( de )

    swde = gdcm.DataElement(gdcm.Tag(0x0008,0x0060))
    Modality = "RTSTRUCT"
    swde.SetByteValue(Modality, gdcm.VL(len(Modality)))
    swde.SetVR(gdcm.VR(gdcm.VR.CS))
    ds.Insert(swde)

    swde1 = gdcm.DataElement(gdcm.Tag(0x0008,0x1010))
```

Step2: Add all required data element to the empty dataset. After construction of data elements, these data elements are added to the dataset. Then I construct the sequence because all objects of **Structure Set** are in the form of sequence; sequences are created using "**dycompyler**" [15], "**gdcm"** package and the steps are as follows:

1. Create Data Element
2. Create an item
3. Create a Sequence
4. Insert the Sequence into the DataSet

After successful completion of phase3, I add some annotation features inside my annotation tool such as: Zoomin-Zoomout, Movement of Pan, checking modality, add levels etc. Hence, produced an efficient user-friendly GUI; namely "DICOM RTStructured based Annotation Tool".

```
del1 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0082))
ObservationNumber = "0"
del1.SetByteValue(ObservationNumber, gdcm.VL(len(ObservationNumber)))
del1.SetVR(gdcm.VR(gdcm.VR.IS))

del2 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0084))
ReferencedROINumber = "4"
del2.SetByteValue(ReferencedROINumber, gdcm.VL(len(ReferencedROINumber)))
del2.SetVR(gdcm.VR(gdcm.VR.IS))

del3 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0085))
ROIObservationlabel = "BrainStem"
del3.SetByteValue(ROIObservationlabel, gdcm.VL(len(ROIObservationlabel)))
del3.SetVR(gdcm.VR(gdcm.VR.SH))

del4 = gdcm.DataElement(gdcm.Tag(0x3006, 0x00a4))
RTROIInterpretedType = "ORGAN"
del4.SetByteValue(RTROIInterpretedType, gdcm.VL(len(RTROIInterpretedType)))
del4.SetVR(gdcm.VR(gdcm.VR.CS))

del5 = gdcm.DataElement(gdcm.Tag(0x3006, 0x00a6))
ROIInterpreter = ""
del5.SetByteValue(ROIInterpreter, gdcm.VL(len(ROIInterpreter)))
del5.SetVR(gdcm.VR(gdcm.VR.PN))

itm10=gdcm.Item()
itm10.SetVLToUndefined()
ndds10=itm10.GetNestedDataSet()
ndds10.Insert(del1)
ndds10.Insert(del2)
ndds10.Insert(del3)
ndds10.Insert(del4)
NORMAL
```

Phase4:

Construction of a secure communication:

For constructing a communication among various medical devices; I used cloud IaaS layer and integrate cloud IaaS layer with python by using “PiCloud” and “dispy” library packages. With the help of these two packages I can able to create a server and from this server I can able to transfer the information to the other medical devices by maintaining PACs and HIPPA and vice-versa. Hence, achieve a successful communication. For security, I applied MD5, one way hashing function to each volume of .dcm file.

```
################################### end of nested seq 43 ####################
sde = gdcm.DataElement(gdcm.Tag(0x3006, 0x0042))
ContourGeometricType = "CLOSED_PLANAR"
sde.SetByteValue(ContourGeometricType, gdcm.VL(len(ContourGeometricType)))
sde.SetVR(gdcm.VR(gdcm.VR.CS))

try:
    sde1 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0046))
    NumberOfContourPoints = open("/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/count8.txt", "r").read()
    sde1.SetByteValue(NumberOfContourPoints, gdcm.VL(len(NumberOfContourPoints)))
    sde1.SetVR(gdcm.VR(gdcm.VR.IS))

    sde2 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0050))
    ContourData = open("/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/anno_img8_CS.txt", "r").read()
    ContourData = ContourData[:-1]
    sde2.SetByteValue(ContourData, gdcm.VL(len(ContourData)))
    sde2.SetVR(gdcm.VR(gdcm.VR.SH))
except:
    pass

it44=gdcm.Item()
it44.SetVLToUndefined()
nds1=it44.GetNestedDataSet()
nds1.Insert(sde)
nds1.Insert(sde1)
nds1.Insert(sde2)

sqde1 = gdcm.DataElement(gdcm.Tag(0x0008, 0x1150))
```

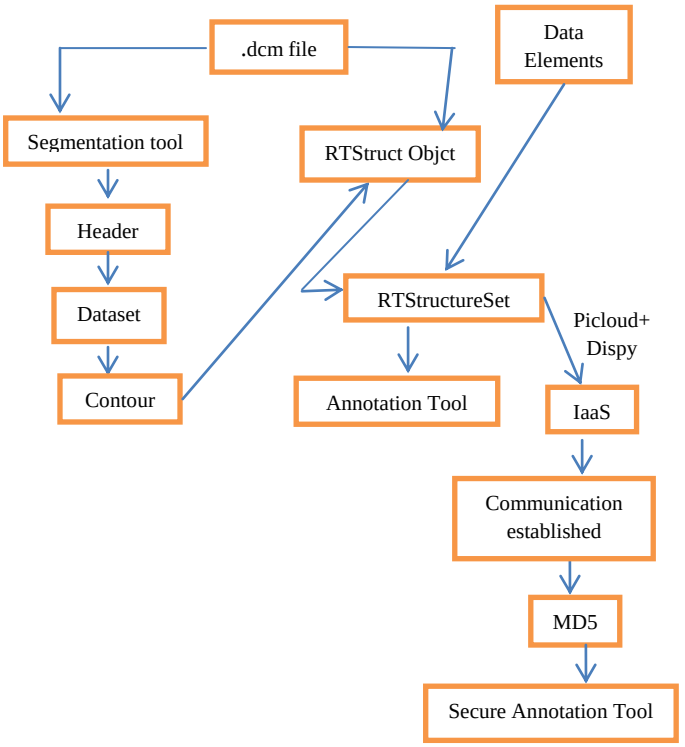
Hashing (im): calculate the one way hash value for all of the voxel points for each volume of .dcm files by using the following formulae:

$$im = h\left(\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}\right)$$

Since, MD5 is a one way hash function; so after applying MD5 in each volume of DICOM, each of 128 rounds is responsible for securing the information of patients. Hence, achieve a secure .dcm files as well as the medical devices. Because of the modular, and factorization properties of one-way hash function [12], and each round of MD5, the voxel values are changed in each 128 rounds and after the completion of these entire 128 rounds client devices are getting the correct information from the server and vice-versa. So, for the intruder it is impossible to break the system, and modify the information. Hence, achieve an efficient secure communication among various medical devices.

```
class plugin2DView(wx.Panel):
    """Plugin to display DICOM image, RT Structure, RT Dose in 2D."""

    def __init__(self):
        wx.Panel.__init__(self)

    def Init(self, res):
        """Method called after the panel has been initialized."""

        # Bind ui events to the proper methods
        self.Bind(wx.EVT_PAINT, self.OnPaint)
        self.Bind(wx.EVT_SIZE, self.OnSize)
        self.Bind(wx.EVT_WINDOW_DESTROY, self.OnDestroy)
        ######################################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

        self.Bind(wx.EVT_BUTTON, self.OnButtonClickedPan)
        self.Bind(wx.EVT_BUTTON, self.OnButtonClickedAnno)
        self.Bind(wx.EVT_BUTTON, self.OnButtonClickedsave)

        ###############################################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

        # Initialize variables

        self.images = []
        self.structures = {}
        self.window = 0
        self.level = 0
        self.zoom = 1
        self.pan = [0, 0]
        self.bwidth = 0
        self.bheight = 0
```

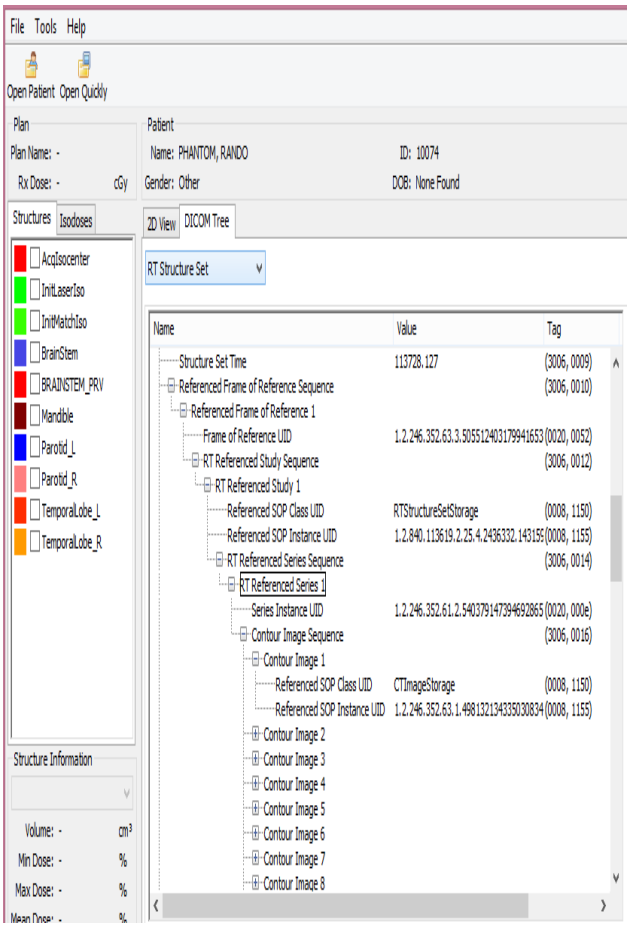
```
# Setup toolbar controls
if guiutil.IsGtk():
    drawingstyles = ['Solid', 'Transparent', 'Dot']
else:
    drawingstyles = ['Solid', 'Transparent', 'Dot', 'Dash', 'Dot Dash']

zoominbmp = wx.Bitmap(util.GetResourcePath('magnifier_zoom_in.png'))
zoomoutbmp = wx.Bitmap(util.GetResourcePath('magnifier_zoom_out.png'))
toolsbmp = wx.Bitmap(util.GetResourcePath('cog.png'))
#################################################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
panon = wx.Bitmap(util.GetResourcePath('on.jpg'))
anno = wx.Bitmap(util.GetResourcePath('annotation.jpg'))
save = wx.Bitmap(util.GetResourcePath('table_multiple.png'))

#################################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
self.tools = []
self.tools.append({'label':"Zoom In", 'bmp':zoominbmp, 'shortHelp':"Zoom In", 'eventhandler':self.OnZoomIn})
self.tools.append({'label':"Zoom Out", 'bmp':zoomoutbmp, 'shortHelp':"Zoom Out", 'eventhandler':self.OnZoomOut})
self.tools.append({'label':"Tools", 'bmp':toolsbmp, 'shortHelp':"Tools", 'eventhandler':self.OnToolsMenu})
#######################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

self.tools.append({'label':"pan", 'bmp':panon, 'shortHelp':"pan", 'eventhandler':self.OnButtonClickedPan})
self.tools.append({'label':"annotation", 'bmp':anno, 'shortHelp':"annotation", 'eventhandler':self.OnButtonClickedAnno})
self.tools.append({'label':"Save", 'bmp':save, 'shortHelp':"save", 'eventhandler':self.OnButtonClickedsave})

############################################################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


# Set up preferences
self.preferences = [
    {'Drawing Settings':
```




```
des=gdcm.DataElement(gdcm.Tag(0x3006,0x0016))
des.SetVR(gdcm.VR.SQ))
des.SetValue(sq._ref_())
des.SetVLToUndefined()
nds1.Insert(des)
############################## end of nested seq 39 ################
sde = gdcm.DataElement(gdcm.Tag(0x3006, 0x0042))
ContourGeometricType = "CLOSED_PLANAR"
sde.SetByteValue(ContourGeometricType, gdcm.VL(len(ContourGeometricType)))
sde.SetVR(gdcm.VR(gdcm.VR.CS))

try:
    sde1 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0046))
    NumberOfContourPoints = open("/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/count4.txt", "r").read()
    sde1.SetByteValue(NumberOfContourPoints, gdcm.VL(len(NumberOfContourPoints)))
    sde1.SetVR(gdcm.VR(gdcm.VR.IS))

    sde2 = gdcm.DataElement(gdcm.Tag(0x3006, 0x0050))
    ContourData = open("/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/anno_img4_CS.txt", "r").read()
    ContourData = ContourData[:-1]
    sde2.SetByteValue(ContourData, gdcm.VL(len(ContourData)))
    sde2.SetVR(gdcm.VR(gdcm.VR.SH))
except:
    pass
```

Fig. 4. Block diagram of proposed Annotation tool

```
from wx.lib.pubsub import pub
#from matplotlib import _cntr as cntr
from matplotlib import __version__ as mplversion
import numpy as np
try:
    from dicompyler import guiutil, util
except:
    pass
import gdcm
import sys,os

globalFlag = False
panEnable = False
annoEnable = False

def setFlagFalse():
    global panEnable,annoEnable
    panEnable = globalFlag
    annoEnable = globalFlag


def pluginProperties():
    """Properties of the plugin."""

    props = {}
    props['name'] = '2D View'
    props['description'] = "Display image, structure and dose data in 2D"
    props['author'] = 'Aditya Panchal'
    props['version'] = "0.5.0"
    props['plugin_type'] = 'main'
    props['plugin_version'] = 1
    props['min_dicom'] = ['images']
```



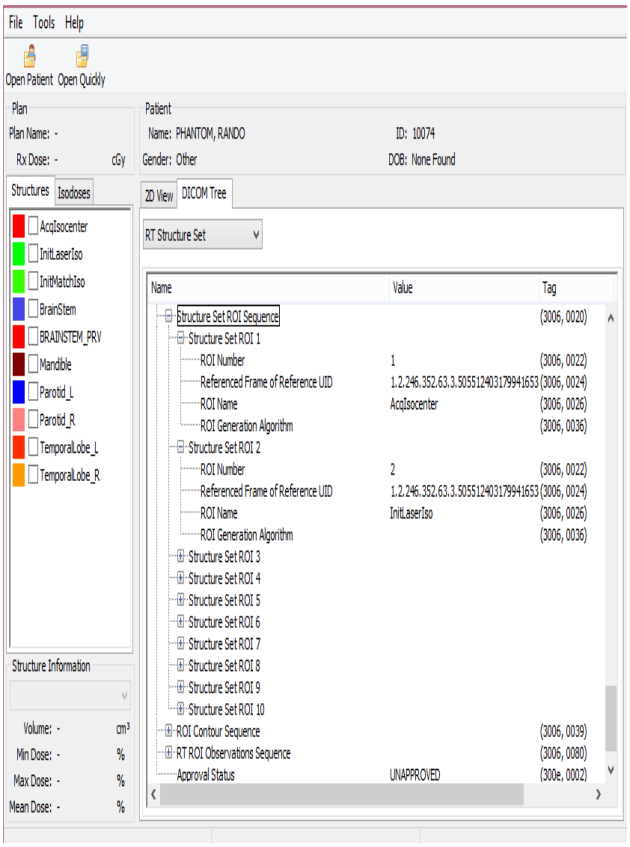
A. RESULT ANALYSIS

In this section, I present how my developed tool works.

Referenced Frame of Reference Sequence:

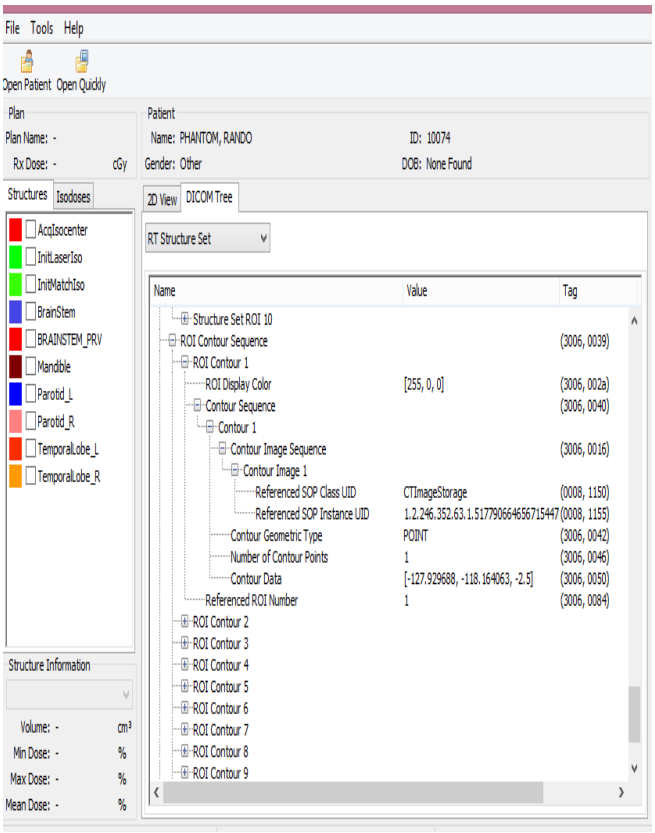
Structure Set ROI Sequence:

```
it40=gdcm.Item()
it40.SetVLToUndefined()
nds1=it40.GetNestedDataSet()
nds1.Insert(sde)
nds1.Insert(sde1)
nds1.Insert(sde2)

sqde1 = gdcm.DataElement(gdcm.Tag(0x0008, 0x1150))
ReferencedSOPClassUID = "CTImageStorage"
sqde1.SetByteValue(ReferencedSOPClassUID, gdcm.VL(len(ReferencedSOPClassUID)))
sqde1.SetVR(gdcm.VR(gdcm.VR.UI))

sqde2 = gdcm.DataElement(gdcm.Tag(0x0008, 0x1155))
ReferencedSOPInstanceUID = "1.2.246.352.63.1.5369381223347546636.4813187339420828830"
sqde2.SetByteValue(ReferencedSOPInstanceUID, gdcm.VL(len(ReferencedSOPInstanceUID)))
sqde2.SetVR(gdcm.VR(gdcm.VR.UI))

it2=gdcm.Item()
it2.SetVLToUndefined()
nds2=it2.GetNestedDataSet()
nds2.Insert(sqde1)
nds2.Insert(sqde2)

sq=gdcm.SequenceOfItems().New()
sq.SetLengthToUndefined()
sq.AddItem(it2)

des=gdcm.DataElement(gdcm.Tag(0x3006,0x0016))
des.SetVR(gdcm.VR(gdcm.VR.SQ))
des.SetValue(sq.__ref__())
```



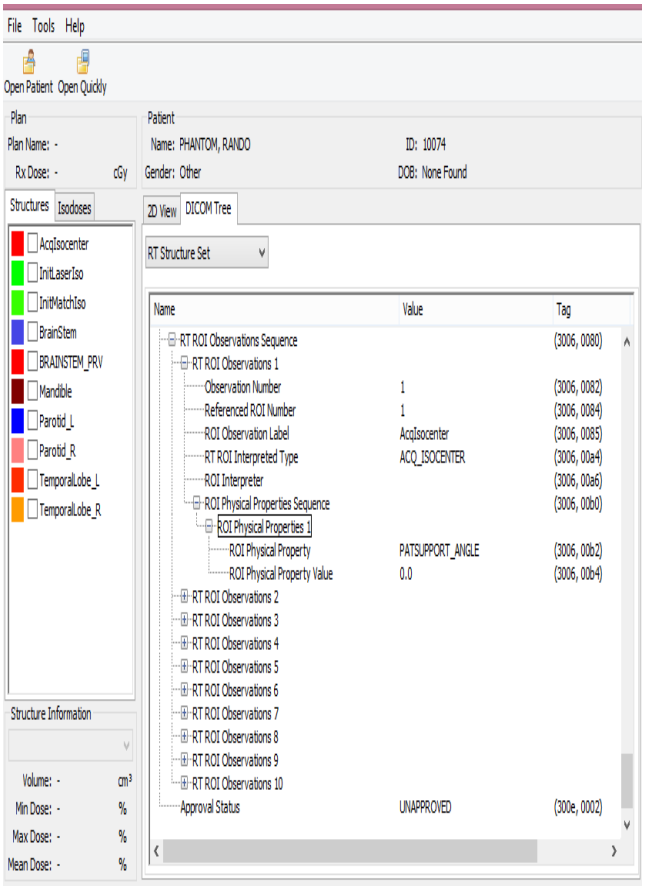
ROI contour Sequence:

```
z_val = 85.0
if(z_val != o1):

    with open("/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/anno.txt", "a+") as myfileAnno:

        m = float(x  * float(xx[0]) + float(yy[0]))
        n = float(y  * float(xx[1]) + float(yy[1]))
        o = yy[2]
        np.savetxt(myfileAnno, np.c_[m, n, o], fmt='%f', delimiter='\t')
        #exit()

        #np.savetxt(myfileAnno, np.c_[a11], fmt='%s', delimiter='\t')

else:
    with open("/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/anno_img1.txt", "a+") as myfileAnno:

        m = float(x * float(xx[0]) + float(yy[0]))
        n = float(y * float(xx[1]) + float(yy[1]))
        o = yy[2]
        np.savetxt(myfileAnno, np.c_[m, n, o], fmt='%f', delimiter='\t')

    with open("/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/anno_img1_CS.txt", "a+") as myfileAnno1:

        m = float((x - 135.0) * float(xx[0]) + float(yy[0]))
        n = float((y - 12.0) * float(xx[1]) + float(yy[1]))
        o = yy[2]
        np.savetxt(myfileAnno1, np.c_[m, n, o],fmt='%f', delimiter='\\', newline='\\')

    def out():
        count=0
        with open ('/home/sweta/Downloads/dicompyler-master/dicompyler/baseplugins/anno_img1.txt','r') as f:
            for line in f:
```

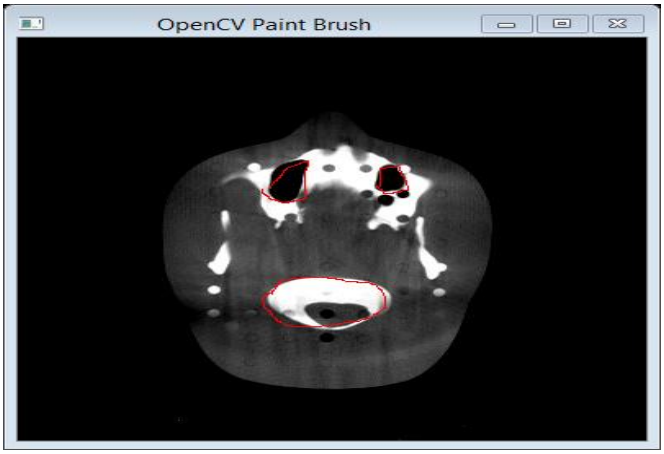


RT ROI Observations Sequence:

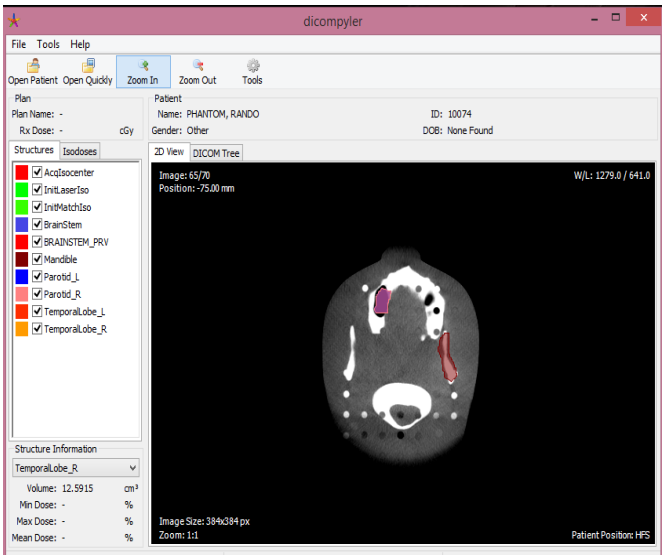
```
it43=gdcm.Item()
it43.SetVLToUndefined()
nds1=it43.GetNestedDataSet()
nds1.Insert(sde)
nds1.Insert(sde1)
nds1.Insert(sde2)

sqde1 = gdcm.DataElement(gdcm.Tag(0x0008, 0x1150))
ReferencedSOPClassUID = "CTImageStorage"
sqde1.SetByteValue(ReferencedSOPClassUID, gdcm.VL(len(ReferencedSOPClassUID)))
sqde1.SetVR(gdcm.VR(gdcm.VR.UI))

sqde2 = gdcm.DataElement(gdcm.Tag(0x0008, 0x1155))
ReferencedSOPInstanceUID = "1.2.246.352.63.1.5369381223347546636.4813187339420828830"
sqde2.SetByteValue(ReferencedSOPInstanceUID, gdcm.VL(len(ReferencedSOPInstanceUID)))
sqde2.SetVR(gdcm.VR(gdcm.VR.UI))

it2=gdcm.Item()
it2.SetVLToUndefined()
nds2=it2.GetNestedDataSet()
nds2.Insert(sqde1)
nds2.Insert(sqde2)

sq=gdcm.SequenceOfItems().New()
sq.SetLengthToUndefined()
sq.AddItem(it2)

des=gdcm.DataElement(gdcm.Tag(0x3006,0x0016))
des.SetVR(gdcm.VR(gdcm.VR.SQ))
```

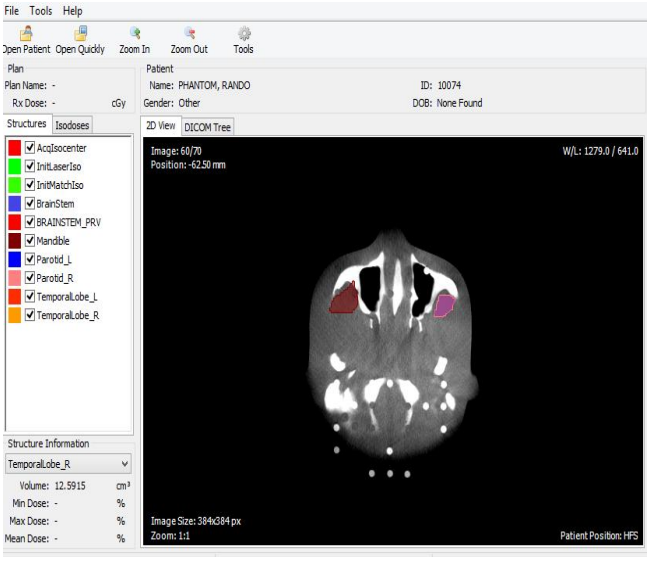


Screenshot of contour images:

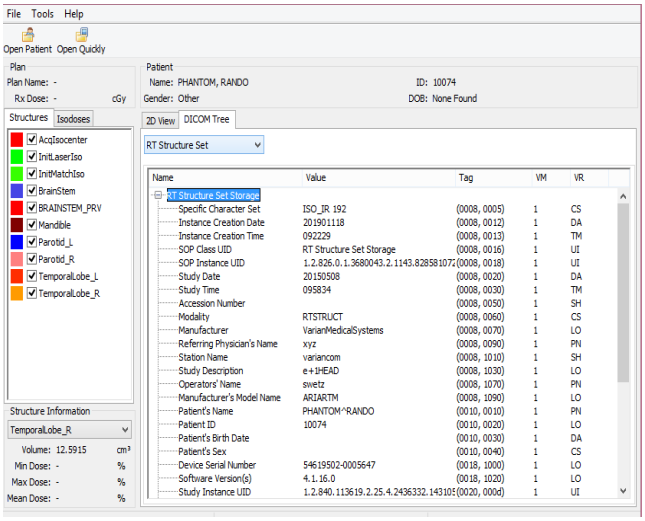


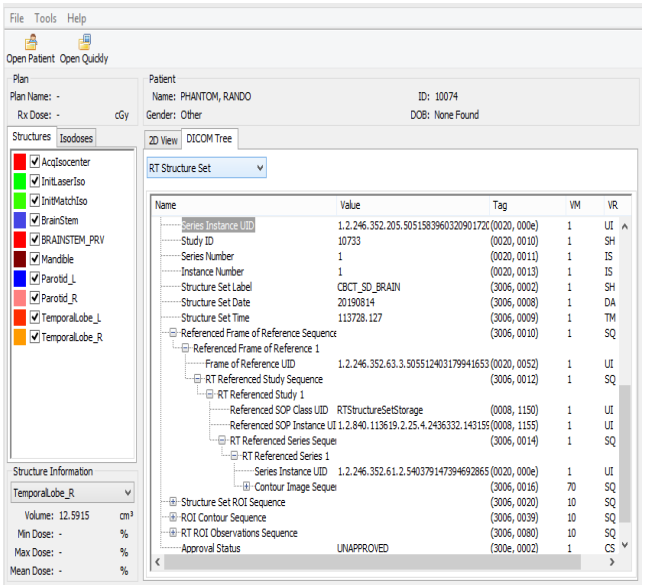
Screenshot of RTStructureSet DICOM:





Screenshot of RT Structure set:





V. CONCLUSION

In this paper, I have presented a Secure Annotation Tool of DICOM which makes DICOM easy to understand, and communicate among various medical devices securely. It is able to annotate all .dcm files efficiently in form of volume and able to records the therapy details of patients in form of RT Structure. Hence, my tool is able to successfully address the requirements for the transfer of Patients structures and related data defined on medical devices, and also communicate with each other securely.

VI. FUTURE WORK

My future plan is to add different views of DICOM as well as to add some authentication technique of the medical devices. For adding different views on my developed GUI; my plan is to use some of the python packages like "vtk" etc., and for authentication; my plan is to add digital signature inside each volume of .dcm file.

REFERENCES

[1] Dey, S. (2018, November). SBPSO-LB: SB-PSO based Secure Load Balancing Approach for Cloud Data Center with RSA. In 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON) (pp. 1-6). IEEE.

[2] Ramesh, D., & Dey, S. (2018, March). SCLBA-CC: Slot Based Carton Load Balancing Approach for Cloud Environment. In 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT) (pp. 1-5). IEEE.

[3] Dey, S. (2018, October). A Secure Fair Queuing and SLA based Slotted Round Robin Load Balancing Approach for Cloud Data Centers. In 2018 3rd International Conference on Communication and Electronics Systems (ICCES) (pp. 382-387). IEEE.

[4] Dey, S., & Kaur, U. (2019, January). RSA and SFQ based Secure Heuristic Load Balancing Approach for Cloud Data Centers. In 2019 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-7). IEEE.

[5] Dey, S. (2019, April). SSSD-LB: SRNN based Secure Slotted Dynamic Load Balancing Scheme. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 1212-1217). IEEE.

[6] https://docs.anaconda.com/anaconda/

[7] Pavlidis, I., & Faltesek, T. (2002, September). A video-based surveillance solution for protecting the air-intakes of buildings from chem-bio attacks. In Proceedings. International Conference on Image Processing (Vol. 1, pp. I-I). IEEE.

[8] Miciolino, E. E., Bernieri, G., Pascucci, F., & Setola, R. (2015, November). Communications network analysis in a SCADA system testbed under cyber-attacks. In 2015 23rd Telecommunications Forum Telfor (TELFOR) (pp. 341-344). IEEE.

[9] https://en.wikipedia.org/wiki/Cyberattack

[10] Rivest, R. (1992). The MD5 Message-Digest Algorithm. Document of MIT Laboratory for computer Science and RSA Data Security, Inc.

[11] Zhou, X. Q., Huang, H. K., & Lou, S. L. (2001). Authenticity and integrity of digital mammography images. IEEE transactions on medical imaging, 20(8), 784-791.

[12] https://www.dicomstandard.org/

[13] Gorthi, S., Bach, C. M., & Thiran, J. P. (2009). Exporting contours to DICOM-RT structure set. Insight Journal, (1), 1-18.

[14] https://dicom.innolitics.com/ciods

[15] https://www.dicompyler.com/