**Addition to knowledge**

This thesis research will bring significant contribution to the field of computer security, cryptography and medical record privacy. Various advance encryption model is available with different approach or method of encrypting and securing data. The sturdy reveal why Counter CTR encryption mode should be used in combination with RSA encryption algorithm. This advantage of fast encryption and decryption process provided by CTR encryption mode cryptography and RSA algorithm utilization for key distributions. The proposed approach will greatly improve the preservation of confidential medical data across the globe.

**Appendix**

**Aes\_ctrmode.py file**

import base64

from ctypes import sizeof

from Crypto.Cipher import AES

from numpy import byte

import time

data = b'ade-12-yes-1995'

# https://pycryptodome.readthedocs.io/en/latest/src/cipher/classic.html#ctr-

# https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

import json

from base64 import b64encode

from Crypto.Cipher import AES

from Crypto.Util import Counter

from Crypto.Random import get\_random\_bytes

from pathlib import Path

import json

from base64 import b64decode

from Crypto.Cipher import AES

import csv as cv

import rsa\_encryption as rsa

from binascii import hexlify, unhexlify

class AESCTRXEncryption:

def \_\_init\_\_(self) -> None:

self.key = None

# generate and RSA key for each encryption

self.rsa = rsa.RSAEncryption()

self.private , self.public = ('', '')

def encrypt(self, plain\_text, id , passcode):

# GENERATING RSA KEY

self.rsa.generateKeys()

self.private , self.public = self.rsa.loadKeys() # retrurn a turple of private, public key

# data = b"secret"

data = bytes(plain\_text, 'utf-8')

# self.key = get\_random\_bytes(16)

# counter = Counter.new(nbits=16, prefix= unhexlify('f0f1f2f3f4f5f6f7f8f9fafbfcfd'), initial\_value=0xfeff)

self.key = bytes(passcode, 'utf-8')

# print("KEY SIZE" , len(self.key))

cipher = AES.new(self.key, AES.MODE\_CTR)

start\_time = time.time()

ct\_bytes = cipher.encrypt(data)

nonce = b64encode(cipher.nonce).decode('utf-8')

# ENCRYPT the PASSCODE before placed in the BINARY FILE

# USING THE RSA ALGORITHM......

passcode = self.rsa.encrypt(passcode , self.public)

end\_time = time.time()

# print('Encryption Time ', (end\_time-start\_time)\*1000)

# print('decrypted ', passcode)

pk\_file = open(str(id), 'wb')

pk\_file.write(passcode)

pk\_file.close()

# passcode = base64.b64encode(passcode)

# print(passcode)

# self.cerfertext = self.encrypt(self.key, self.public)

ct = b64encode(ct\_bytes).decode('utf-8')

result = json.dumps({'nonce':nonce, 'ciphertext':ct})

Path(str(id)+".json").write\_text(result)

# print(result)

return (end\_time-start\_time)\*1000

def decrypt(self, id\_file , passcode):

# We assume that the key was securely shared beforehand

data = Path(str(id\_file)+'.json').read\_text()

try:

b64 = json.loads(data)

nonce = b64decode(b64['nonce'])

ct = b64decode(b64['ciphertext'])

# key\_text = b64['Code']

# decrypting the AES KEY with RSA encryption

# key\_text = key\_text.decode("ascii")

# rsa\_key\_decrypt = self.r.decrypt(self.key , self.private)

self.private , self.public = self.rsa.loadKeys()

# retrurn a turple of private, public key

# DECRYPT the CTR KEY.. LOAD BINARY FILE..

# USING THE RSA ALGORITHM......

pk\_read = open(str(id\_file), 'rb')

pass\_key = pk\_read.read()

pk\_read.close()

# print('AES KEY SIZE ' , len(pass\_key))

# print(f'RSA key SIZE private:{sizeof(self.private)} public:{sizeof(self.public)}')

start\_time2 = time.time()

pass\_key = self.rsa.decrypt(pass\_key, self.private)

# print('decrypted ', pass\_key)

# temp key movement

self.key = bytes(pass\_key, 'utf-8')

# key = bytes(rsa\_key\_decrypt, 'utf-8')

cipher = AES.new(self.key, AES.MODE\_CTR, nonce=nonce)

pt = cipher.decrypt(ct)

end\_time2 = time.time()

# print("The message was: ", pt)

return pt , (end\_time2-start\_time2)\*1000

except (ValueError, KeyError):

print("Incorrect decryption")

**database.py file**

import sqlite3 as database

import csv as cv

import patient as pt

class Database:

def load\_csv\_data(self, filename)-> list:

with open(filename) as file:

dataset = cv.reader(file)

header = [next(dataset)]

datalist = []

for row in list(dataset):

print('row==>:' , row)

datalist.append(row)

return datalist

def \_\_init\_\_(self) -> None:

self.connection = None

with database.connect('medical\_record.db') as db :

self.connection = db

print('connection established')

def create\_medical\_record\_table(self):

self.connection.execute("""CREATE TABLE IF NOT EXISTS medical\_table (

id INT NOT NULL,

name TEXT NOT NULL,

age INT NOT NULL,

sex TEXT NOT NULL,

antivirals TEXT NOT NULL,

liverbig TEXT NOT NULL,

bilirubin TEXT NOT NULL,

Albumin TEXT NOT NULL,

PRIMARY KEY (id)

)

""")

self.connection.commit()

print('medical record table created.........')

def add\_record(self, patient):

id = patient.id

name = patient.name

age = patient.age

sex = patient.sex

antivirals = patient.antiviral

liverbig = patient.liverbig

bilirubin = patient.bilirubin

albumin = patient.albumin

# print('ID .....====> ', id)

self.connection.execute(

'INSERT INTO medical\_table VALUES(?,?,?,?,?,?,?,?)', (id, name, age, sex,antivirals, liverbig,bilirubin,albumin)

)

self.connection.commit()

print(patient.id , ' .... record inserted.....')

def fetch\_all\_record(self):

record\_list = []

cr = self.connection.cursor().execute('SELECT \* FROM medical\_table')

data = cr.fetchall()

for patient in data:

# id , name , age, sex, antivirals, liverbig, bilirubin, albumin

us = pt.Patient(patient[0], patient[1], patient[2], patient[3], patient[4], patient[5], patient[6], patient[7])

record\_list.append(us)

return record\_list

def csv\_to\_database(self, f):

csv\_record = self.load\_csv\_data(f)

patient\_list = []

for record in csv\_record:

id, age, sex, antivirals, liverbig, bilirubin, albumin , name = record

patient = pt.Patient(id, name, age, sex, antivirals, liverbig, bilirubin, albumin)

print(f"{patient}")

self.add\_record(patient)

else:

print('all record inserted completely')

def database\_to\_csv(self):

patient\_records = self.fetch\_all\_record()

# write to csv

with open("datas.csv", "w") as file:

writer = cv.writer(file)

writer.writerow(['ID', 'NAME', 'AGE', 'SEX', 'ANTIVIRALS' , 'LIVERBIG', 'BILIRUBIN', 'ALBUMIN'])

for patient in patient\_records:

id, name, age, sex, antivirals, liverbig, bilirubin, albumin = patient.id, patient.name, patient.age, patient.sex, patient.antiviral, patient.liverbig, patient.bilirubin, patient.albumin

writer.writerow([id, name, age, sex, antivirals, liverbig, bilirubin, albumin])

# print(f"{id} {name} {age}")

print('program completed....')

# db = Database()

# # db.create\_medical\_record\_table()

# patient = db.fetch\_all\_record()

# # db.load\_csv\_data('final\_medical\_record.csv')

# # db.csv\_to\_database('final\_medical\_record.csv')

# # for p in patient:

# # print('record : ' , p)

# # print([str(p) for p in patient])

# db.database\_to\_csv()

**Rsa\_encryption.py file**

import rsa

class RSAEncryption():

def generateKeys(self):

(publicKey, privateKey) = rsa.newkeys(1024)

with open('keys/publicKey.pem', 'wb') as p:

p.write(publicKey.save\_pkcs1('PEM'))

with open('keys/privateKey.pem', 'wb') as p:

p.write(privateKey.save\_pkcs1('PEM'))

def loadKeys(self):

with open('keys/publicKey.pem', 'rb') as p:

publicKey = rsa.PublicKey.load\_pkcs1(p.read())

with open('keys/privateKey.pem', 'rb') as p:

privateKey = rsa.PrivateKey.load\_pkcs1(p.read())

return privateKey, publicKey

def encrypt(self, message, key):

return rsa.encrypt(message.encode('ascii'), key)

def decrypt(self, ciphertext, key):

try:

return rsa.decrypt(ciphertext, key).decode('ascii')

except:

return False

def decrypt(self, ciphertext, key):

try:

return rsa.decrypt(ciphertext, key).decode('ascii')

except:

return False

# rs = RSAEncryption()

# # rs.generateKeys()

# private, public = rs.loadKeys()

# print(f'private {private} \n public {public}')

# cipher\_text = rs.encrypt('1995' , public)

# print('message encrypted succesfully')

# message = rs.decrypt(cipher\_text, private)

# print(f'original\_message {message}')

**layout.kivy file**

ScreenController:

LoginScreen:

MainScreen:

MedicalDetailScreen:

<LoginScreen>

name:'login\_screen'

MDScreen:

MDCard:

size\_hint: .80, .8

pos\_hint:{'center\_x':0.5 , 'center\_y':0.5}

orientation: 'vertical'

elevation:10

padding:30

spacing:8

MDLabel:

text: "CRYPTOMEDIC ENCRYPTION SYSTEM"

font\_size:25

# text\_color:app.theme\_cls.accent\_color

theme\_text\_color:"Primary"

halign:'center'

size\_hint\_y: None

height:self.texture\_size[1]

MDLabel:

text: "(RSA --- CTR PARALLEL ENCRYPTION MODE)"

font\_size:18

text\_color:app.theme\_cls.accent\_color

# theme\_text\_color:"Primary"

halign:'center'

size\_hint\_y: None

height:self.texture\_size[1]

Widget:

size\_hint\_y:None

height:40

MDTextField:

hint\_text:'Enter ID Code'

icon\_left:"account"

padding:15

normal\_color:app.theme\_cls.accent\_color

line\_color\_normal:app.theme\_cls.accent\_color

color\_mode:"custom"

size\_hint\_x:.70

pos\_hint:{'center\_x':.50}

MDTextField:

hint\_text:'Enter Access Code'

icon\_left:"key-variant"

hint\_color:app.theme\_cls.accent\_color

password:True

padding:15

normal\_color:app.theme\_cls.accent\_color

line\_color\_normal:app.theme\_cls.accent\_color

size\_hint\_x:.70

pos\_hint:{'center\_x':.50}

MDCard:

orientation:'horizontal'

spacing:10

padding:10

pos\_hint:{'center\_x':.50}

size\_hint:None, None

height:80

width: 180

MDRectangleFlatButton:

text: 'Doctor'

# text\_color:app.theme\_cls.accent\_color

# theme\_text\_color:'Primary'

# md\_bg\_color: app.theme\_cls.primary\_light

pos\_hint: {'center\_x': 0.5,'center\_y': 0.5}

on\_release:app.root.current='main\_screen'

MDRectangleFlatButton:

text: 'Patient'

# text\_color:app.theme\_cls.accent\_color

# theme\_text\_color:'Primary'

# md\_bg\_color: app.theme\_cls.primary\_light

pos\_hint: {'center\_x': 0.5,'center\_y': 0.5}

on\_release:app.root.current='detail\_screen'

# MDRectangleFlatButton:

# text: 'Cancel'

# # text\_color:app.theme\_cls.accent\_color

# # theme\_text\_color:'Primary'

# # md\_bg\_color: app.theme\_cls.primary\_light

# pos\_hint: {'center\_x': 0.5,'center\_y': 0.5}

<MainScreen>

name:"main\_screen"

MDCard:

id:layout

orientation:'vertical'

pos\_hint:{"center\_x":.5 , "center\_y":.5}

size\_hint:1,1

padding:10

# MDDataTable:

# id:table

MDCard:

orientation:'vertical'

pos\_hint:{"center\_x":.5}

# md\_bg\_color:app.theme\_cls.primary\_light

size\_hint:.7,1

spacing:50

padding:10

MDRectangleFlatButton:

text: 'CTR-RSA SECURE RECORD'

# text\_color:app.theme\_cls.accent\_color

# theme\_text\_color:'Primary'

# md\_bg\_color: app.theme\_cls.primary\_light

pos\_hint: {'center\_x': 0.5}

on\_release:app.encryption\_box()

MDCard:

orientation:'horizontal'

pos\_hint:{"center\_x":.5}

# md\_bg\_color:app.theme\_cls.primary\_light

size\_hint:.7,1

spacing:20

padding:10

MDTextField:

hint\_text:'ENTER CTR KEY'

icon\_right:"key-variant"

# mode:"rectangle"

id:text\_encryption\_key

hint\_color:app.theme\_cls.accent\_color

password:False

padding:15

max\_text\_length:16

normal\_color:app.theme\_cls.accent\_color

line\_color\_normal:app.theme\_cls.accent\_color

size\_hint\_x:.20

pos\_hint:{'center\_x':.60}

MDTextField:

hint\_text:'Search Record'

icon\_right:"magnifier"

mode:'rectangle'

hint\_color:app.theme\_cls.accent\_color

password:False

padding:15

normal\_color:app.theme\_cls.accent\_color

line\_color\_normal:app.theme\_cls.accent\_color

size\_hint\_x:.80

pos\_hint:{'center\_x':.50}

<MedicalDetailScreen>

name:'detail\_screen'

MDCard:

orientation:'vertical'

pos\_hint:{"center\_x":.5 , "center\_y":.5}

size\_hint:.8,.8

padding:8

MDLabel:

text:"RECORD PRESAVATION SYSTEM"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

font\_size:20

text\_align:'center'

# ROW 1 MEDICAL ID DATA

MDCard:

id:layout

orientation:'horizontal'

padding:10

spacing:10

# THE DETATIL VERTICAL SPLITTING............

MDCard:

id:layout

orientation:'vertical'

padding:2

spacing:1

# LEFT SIDE WITH ID, NAME, AGE AND SEX

# =============================

FloatLayout:

canvas:

Color:

rgb: 1, 1, 1

Ellipse:

pos: 180, 400

size: 100 , 100

source: 'userIcon.png'

angle\_start: 0

angle\_end: 360

MDCard:

id:layout

orientation:'horizontal'

padding:2

spacing:2

MDLabel:

text:"MEDICAL ID"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

id:label\_id

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

# ROW 2 MEDICAL NAME DATA

MDCard:

id:layout

orientation:'horizontal'

padding:2

spacing:2

MDLabel:

text:"NAME"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

id:label\_name

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

# ROW 3 MEDICAL ID DATA

MDCard:

id:layout

orientation:'horizontal'

padding:2

spacing:2

MDLabel:

text:"SEX"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

size\_hint\_y: None

id:label\_sex

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

# ROW 4 MEDICAL ID DATA

MDCard:

id:layout

orientation:'horizontal'

padding:2

spacing:2

MDLabel:

text:"AGE"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

id:label\_age

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDCard:

orientation:'horizontal'

spacing:10

padding:10

pos\_hint:{'center\_x':.50}

size\_hint:None, None

height:80

width: 180

MDRectangleFlatButton:

text: 'Logout'

# text\_color:app.theme\_cls.accent\_color

# theme\_text\_color:'Primary'

# md\_bg\_color: app.theme\_cls.primary\_light

pos\_hint: {'center\_x': 0.5,'center\_y': 0.5}

size\_hint:.5, .6

on\_release:app.root.current='login\_screen'

MDRectangleFlatButton:

text: 'Decrypt'

# text\_color:app.theme\_cls.accent\_color

# theme\_text\_color:'Primary'

# md\_bg\_color: app.theme\_cls.primary\_light

pos\_hint: {'center\_x': 0.5,'center\_y': 0.5}

size\_hint:.5, .6

# on\_release:app.root.current='detail\_screen'

on\_release: app.decryption\_box()

MDTextField:

hint\_text:'Enter ID for Decryption File'

icon\_right:"magnifier"

id:id\_txt\_id\_file

mode:'rectangle'

hint\_color:app.theme\_cls.accent\_color

password:False

padding:5

normal\_color:app.theme\_cls.accent\_color

line\_color\_normal:app.theme\_cls.accent\_color

size\_hint\_x:.80

pos\_hint:{'center\_x':.50}

Widget:

size\_hint\_y:None

height:0

# RIGHT SIDE WITH ID, NAME, AGE AND SEX

# =============================

MDCard:

id:layout

orientation:'vertical'

padding:10

spacing:2

MDCard:

id:layout

orientation:'horizontal'

padding:5

spacing:2

MDLabel:

text:"ANTIVIRAL"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

id:label\_antiviral

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

# ROW 6 MEDICAL ID DATA

MDCard:

id:layout

orientation:'horizontal'

padding:5

spacing:2

MDLabel:

text:"LIVERBIG"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

id:label\_liverbig

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

# ROW 7 MEDICAL ID DATA

MDCard:

id:layout

orientation:'horizontal'

padding:5

spacing:2

MDLabel:

text:"BILLIRUBIN"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

id:label\_bilirubin

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDCard:

id:layout

orientation:'horizontal'

padding:5

spacing:2

MDLabel:

text:"ALBUMIN"

size\_hint\_y: None

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'

MDLabel:

text:"XXXXXX"

size\_hint\_y: None

id:label\_albumin

height:self.texture\_size[1]

theme\_text\_color:"Primary"

halign:'center'

text\_align:'center'