

A
Project Report On
CyberSec – Blogs and Bombe
Submitted in partial fulfilment of the
requirements for the award of the degree of
BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING
BY

SVN Ramakanth

(1602-19-733-118)

Venkateshwarlu Guptha

(1602-19-733-119)

Under the guidance of

Smt. B. Syamala

ASSISTANT PROFESSOR

Sri. I. Navakanth

ASSISTANT PROFESSOR



Department of Computer Science & Engineering

Vasavi College of Engineering.

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad.

TABLE OF CONTENTS

CONTENT	PAGE NO.
1. Acknowledgement	03
2. Abstract	04-06
3. SRS	07
4. Project Database and Tables	08-09
5. User Policy	10
6. Implementation of Code	11-28
7. Output Screenshots	29-34
8. Technologies Used	35-36
9. Conclusion	37
10. References	37

1. ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to Navakanth Sir, our project guide, for his valuable guidance and constant support, along with her capable instruction and persistent encouragement.

We are grateful to our Head of Department, Dr. T. Adilakshmi, for her steady support and the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal, Dr. S. V. RAMANA, as well as the management of the institute, for having designed an excellent learning atmosphere.

2. ABSTRACT

2.1 OBJECTIVE:

This application provides the user with blogs regarding cyber security so that they can stay updated about upcoming technologies, awareness, security and tools. The project also features Bombe machine in which encoding and decoding of information takes place using combined cryptography and steganography.

2.2 WORKING:

This web application is made using Python(Django) Framework in back-end and HTML in the front-end. Registration is free for any user and they can read all the blogs even without having an account but having an account is essential for using the Bombe machine. It is an advanced Software that enables the readers to read blogs regarding cybersecurity and lets them encrypt and decrypt their data using two ciphers namely, Affine cipher and Caesar cipher. For steganographic encryption and decryption, Least Significant Bit (LSB) technique is used.

Caesar is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. Caesar cipher encryption formula is $E_n(x) = (x+n) \bmod 26$ i.e., Encryption Phase with shift n and The decryption formula is $D_n(x) = (x-n)$ decryption phase with shift n . The Affine cipher is a type of mono alphabetic substitution cipher, wherein each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. LSB steganography is used in the project and the idea behind LSB embedding is that if we change the last bit value of a pixel, there won't be much visible change in the color. For example, 0 is black. Changing the value to 1 won't make much of a difference since it is still black, just a lighter shade. Using this basic logic as working principle, encoding of data is done into pixels of the image given and retrieved in decoding phase

Features:

- **Homepage** – For homepage, one will be able to have all the basic access in the whole system such as home, sign up, login, user policy, encode, decode, my-encodes and contact sections if the user is logged in then model blogs and about us section will also be displayed wherein administrators github link will also be provided to know more about the admins.
- **Sign Up**– For sign up, one will fill the forms such as username, password, email, first name and last name.
- **Login** – For login, one must give his right username and password.

- **Policy** – One can read the user policy here.
- **Logout** – One can logout by using this button.
- **Encode** – The user can give the data to be encoded in form of text and the public key should also be given for encryption using ciphers. For stego encoding, an image has to be given as input. And the stego encoded image will be downloaded after data encryption into pixels of the image and private key will also be generated.
- **My encodes** – The history of all the encoding activities of the user will be displayed in the field.
- **Decode** – Using this the user can decode his/her encoded image by giving the stego encoded image and the private key. Upon successful validation of image and the private key, the intended text will be decoded and displayed to the user.
- **Contact** – User can post any sort of complains, security concerns or suggestions using the contact session. If the user is interested in writing his/her own blog their blog can be sent to the admin using this feature. If the blog is not fallacious and misleading and gets approved by the administrator of the site, it will be posted in the homepage and credit will be given to the user who wrote the blog. Blogs that are written by the admin himself will be highlighted by red colored label named “admin”.

2.3 SCOPE:

This application can be easily implemented under various situations. We can add new features as and when we require. Reusability is possible as and when require in this application. There is flexibility in all the modules.

•Extensibility:

This software is extendable in ways that its original developers may not expect. The following principles enhances extensibility like hide data structure, avoid traversing multiple links or methods, avoid case statements on object type and distinguish public and private operations.

•Reusability:

Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability: Sharing of newly written code within a project and reuse of previously written code on new projects.

•Understandability:

A method is understandable if someone other than the creator of the method can understand the code (as well as the creator after a time lapse). We use the method, whichsmall and coherent helps to accomplish this.

• Cost- effectiveness:

Its cost is under the budget and make within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement. Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

3. SRS

A Software Requirements Specification (SRS) is a complete set of information about the system on which the developed project will be running. It includes all the hardware as well as the recommended system requirement for running the software is also mentioned in detail separately. The aim of this document is to gather and analyze and give in-depth insight of the complete software requirement of the software.

a) Hardware Required: • Processor: Intel i3 or Above

- RAM: 2GB or above
- Hard Disk: 1 GB or above
- Input Devices: Keyboard, Mouse
- Output Devices: Monitor

b) Software Required: • Operating System: Linux, Ubuntu, Mac, Windows 10

- Frontend: HTML, CSS, Bootstrap
- Backend: Python
- Framework: Django
- IDE: PyCharm, VS Code

4. Project Database and Tables

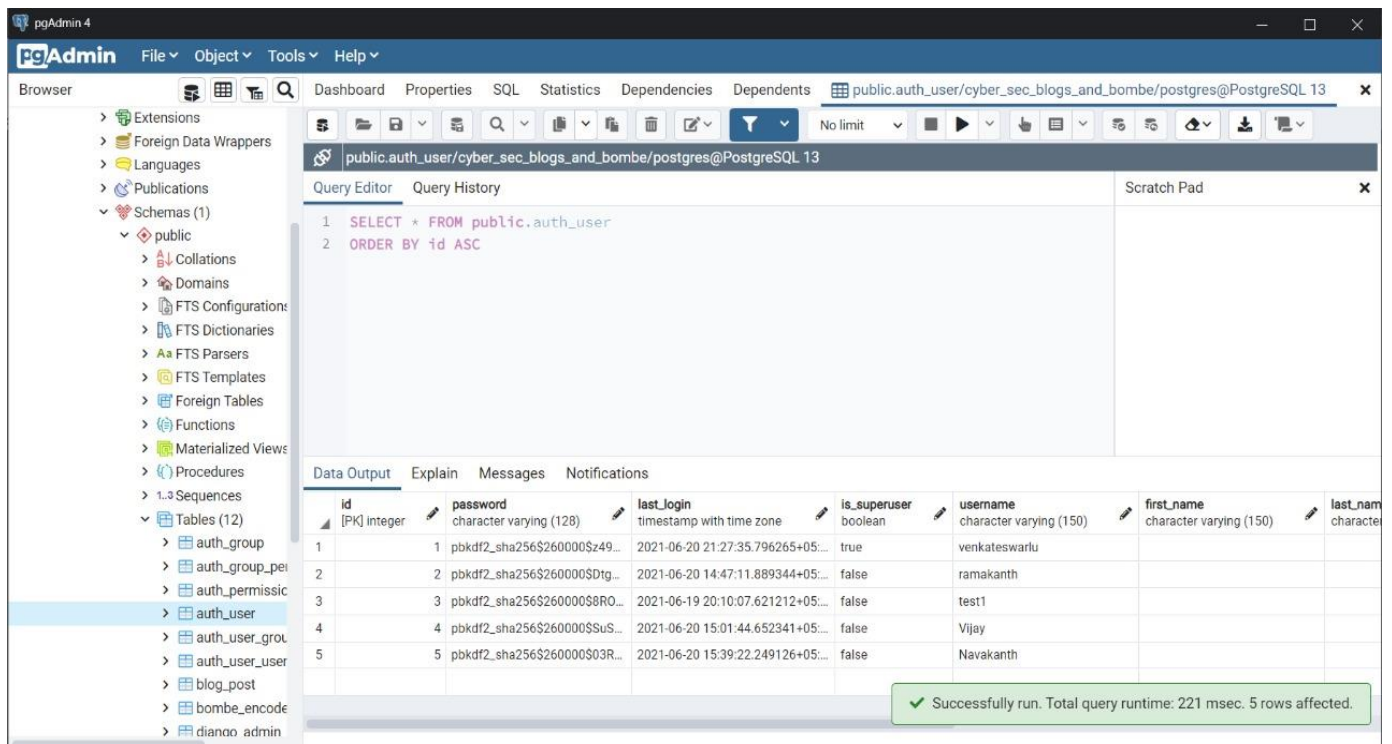
Database is critical for all businesses. A good database does not allow any form of anomalies and stores only relevant information in an ordered manner. If a database has anomalies, it is affecting the efficiency and data integrity. For example, delete anomaly arise upon the deletion of a row which also forces other useful data to be lost. As such, the tables need to be normalized. This fulfils the last objective of ensuring data are accurate and retrieved correctly.

Database files are the key source of information into the system. It is the process of designing database files, which are the key source of information to the system. The files should be properly designed and planned for collection, accumulation, editing and retrieving the required information.

The organization of data in database aims to achieve three major objectives:

- Data integration
- Data integrity
- Data independence

User Authorization Database :



The screenshot displays the pgAdmin 4 web interface. On the left, the 'Browser' pane shows the database structure, with 'public' expanded to show 'Tables (12)', including 'auth_user'. The main pane shows the 'Query Editor' with the following SQL query:

```
1 SELECT * FROM public.auth_user
2 ORDER BY id ASC
```

The 'Data Output' tab is active, showing the results of the query. The table has 5 rows and 7 columns: id, password, last_login, is_superuser, username, first_name, and last_name. A green status bar at the bottom indicates: 'Successfully run. Total query runtime: 221 msec. 5 rows affected.'

id	password	last_login	is_superuser	username	first_name	last_name
1	pbkdf2_sha256\$260000\$z49...	2021-06-20 21:27:35.796265+05:...	true	venkateswarlu		
2	pbkdf2_sha256\$260000\$Dtg...	2021-06-20 14:47:11.889344+05:...	false	ramakanth		
3	pbkdf2_sha256\$260000\$8RO...	2021-06-19 20:10:07.621212+05:...	false	test1		
4	pbkdf2_sha256\$260000\$SuS...	2021-06-20 15:01:44.652341+05:...	false	Vijay		
5	pbkdf2_sha256\$260000\$03R...	2021-06-20 15:39:22.249126+05:...	false	Navakanth		

Blogs Database:

The screenshot shows the pgAdmin 4 interface with the 'public.blog_post/cyber_sec_blogs_and_bombe' database selected. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT * FROM public.blog_post
2 ORDER BY id ASC
```

The 'Data Output' tab shows the results of the query, which are 3 rows. The columns are: id (PK) bigint, title character varying (200), slug character varying (200), updated_on timestamp with time zone, content text, category text, and created_on timestamp with time zone.

id	title	slug	updated_on	content	category	created_on
1	What is Django?	what-is-django	2021-06-19 07:38:16.816807+05:30	Python fra...	Django	2021-06-19 07:38:16.816807+05:30
2	The evolving threat of ransom...	the-evolving-threat-of-ransom...	2021-06-20 07:17:46.901538+05:30	The advanc...	Attacks	2021-06-20 07:16:18.628211+05:30
3	How to Boost Cyber Security ...	how-to-boost-cyber-security-a...	2021-06-20 07:21:02.80737+05:30	Cyber secur...	Security	2021-06-20 07:21:02.80737+05:30

A green message box at the bottom right indicates: 'Successfully run. Total query runtime: 84 msec. 3 rows affected.'

Bombe Database:

The screenshot shows the pgAdmin 4 interface with the 'public.bombe_encodeddetails/cyber_sec_blogs_and_bombe' database selected. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT * FROM public.bombe_encodeddetails
2 ORDER BY id ASC
```

The 'Data Output' tab shows the results of the query, which are 4 rows. The columns are: id (PK) bigint, title character varying (32), private_key character varying (256), img character varying (100), and user_id integer.

id	title	private_key	img	user_id
1	Virat Kohli	18181	/media/virat.png	2
2	Venkateswarlu	1231231231231	/media/virat_6uelatV.png	1
3	5 Numbers	[9, '8', '7', '6', '5']	/media/d.png	1
4	Navakanth Sir Demo	123451234512	/media/d_p4qrrLU.png	5

A green message box at the bottom right indicates: 'Successfully run. Total query runtime: 94 msec. 4 rows affected.'

5.User policy

In order to protect the viewers from getting misleading and malicious information, we undertake the following user policy. All users are requested to accede with the same below.

Terms and conditions:

- All blogs are posted subject to the admin of the page.
- The admin is responsible for the content posted in the blog.
- If any viewer is interested in writing their own blog, the blog can be sent as a message in CONTACT section of the website.
- If the blog is acceptable and doesn't require any user discretion it will be posted.
- In case a blog is posted after being reviewed, then the user details will be posted.
- Bombe machine in our website is used for demonstration of Encryption and decryption using combined steganography and cryptography.
- Use of Bombe machine has to be restricted to educational purposes only.
- Admins of this site do not hold any responsibility for any sort of misuse of the Bombe machine.
- The user can complain against any blog by giving the blog details in CONTACT section and action will be taken if found fallacious.
- Any suggestions or issues with aspect to technical development or interface structure improvement can be brought to admin's notice.

6.IMPLEMENTATION OF CODE

Blog Folder:

- **Admin.py**

```
from
django.contrib
import admin

from .models import Post

class PostAdmin(admin.ModelAdmin):
    list_display = ('title', 'slug', 'status', 'created_on',)
    list_filter = ('status',)
    search_fields = ['title', 'content']
    prepopulated_fields = {'slug': ('title',)}

admin.site.register(Post, PostAdmin)
```

- **Forms.py**

```
from
django
import
forms

from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

def should_be_empty(value):
    if value:
        raise forms.ValidationError('Field is not empty')

class ContactForm(forms.Form):
    name = forms.CharField(max_length=80)
    message = forms.CharField(widget=forms.Textarea)
    email = forms.EmailField()

# Override the UserCreationForm
class UserCreateForm(UserCreationForm):
    email = forms.EmailField(required=True, label='Email', error_messages={'exists': 'This already exists!'})
    pi = 3.14
    class Meta:
        model = User
        fields = ('username', 'email', 'password1', 'password2')
```

```

def save(self, commit=True):
    user = super(UserCreateForm, self).save(commit=False)
    user.email = self.cleaned_data['email']
    if commit:
        user.save()
    return user

def clean_email(self):
    if User.objects.filter(email=self.cleaned_data['email']).exists():
        raise forms.ValidationError(self.fields['email'].error_messages['exists'])
    return self.cleaned_data['email']

```

- **Apps.py**

```

from
django.apps
import
AppConfig

```

```

class BlogConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'blog'

```

Bombe Folder:

- **Admin.py**

```

from
django.contrib
import admin

```

```

from .models import EncodedDetails
admin.site.register(EncodedDetails)

```

- **Apps.py**

```

from
django.apps
import
AppConfig

```

```

class BombeConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'bombe'

```

- **Logic.py**

#

Implementation

of Affine

Cipher in

Python

```
from PIL import Image
```

```
# Extended Euclidean Algorithm for finding modular inverse
```

```
# eg: modinv(7, 26) = 15
```

```
def egcd(a, b):
```

```
    x,y, u,v = 0,1, 1,0
```

```
    while a != 0:
```

```
        q, r = b//a, b%a
```

```
        m, n = x-u*q, y-v*q
```

```
        b,a, x,y, u,v = a,r, u,v, m,n
```

```
    gcd = b
```

```
    return gcd, x, y
```

```
def modinv(a, m):
```

```
    gcd, x, y = egcd(a, m)
```

```
    if gcd != 1:
```

```
        return None # modular inverse does not exist
```

```
    else:
```

```
        return x % m
```

```
# affine cipher encrytion function
```

```
# returns the cipher text
```

```
def affine_encrypt(text, key):
```

```
    '''
```

```
    C = (a*P + b) % 26
```

```
    '''
```

```
    return ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26)
```

```
                      + ord('A')) for t in text.upper().replace(' ', '') ])
```

```
# affine cipher decryption function
```

```
# returns original text
```

```
def affine_decrypt(cipher, key):
```

```
    '''
```

```
    P = (a^-1 * (C - b)) % 26
```

```
    '''
```

```
    return ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') - key[1]))
```

```
                        % 26) + ord('A')) for c in cipher ])
```

```

def generateKey(string, key):
    key = list(key)
    if len(string) == len(key):
        return(key)
    elif len(string) > len(key):
        for i in range(len(string) - len(key)):
            key.append(key[i % len(key)])
        return("".join(key))
    else:
        key = key[:len(string)]
        return(key)
print(key)

# This function returns the
# encrypted text generated
# with the help of the key
def cipherText(string, key):
    cipher_text = []
    for i in range(len(string)):
        x = (ord(string[i]) + ord(key[i])) % 26
        x += ord('A')
        cipher_text.append(chr(x))
    return("".join(cipher_text))

# This function decrypts the
# encrypted text and returns
# the original text
def originalText(cipher_text, key):
    orig_text = []
    for i in range(len(cipher_text)):
        x = (ord(cipher_text[i]) - ord(key[i]) + 26) % 26
        x += ord('A')
        orig_text.append(chr(x))
    return("".join(orig_text))

# Python program implementing Image Steganography

# PIL module is used to extract
# pixels of image and modify it

# Convert encoding data into 8-bit binary
# form using ASCII value of characters
def genData(data):

```

```

# list of binary codes
# of given data
newd = []

for i in data:
    newd.append(format(ord(i), '08b'))
return newd

# Pixels are modified according to the
# 8-bit binary data and finally returned
def modPix(pix, data):

    datalist = genData(data)
    lendata = len(datalist)
    imdata = iter(pix)

    for i in range(lendata):

        # Extracting 3 pixels at a time
        pix = [value for value in imdata.__next__():3] +
                imdata.__next__():3] +
                imdata.__next__():3]]

        # Pixel value should be made
        # odd for 1 and even for 0
        for j in range(0, 8):
            if (datalist[i][j] == '0' and pix[j]% 2 != 0):
                pix[j] -= 1

            elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
                if(pix[j] != 0):
                    pix[j] -= 1
                else:
                    pix[j] += 1
                # pix[j] -= 1

        # Eighth pixel of every set tells
        # whether to stop ot read further.
        # 0 means keep reading; 1 means thec
        # message is over.
        if (i == lendata - 1):
            if (pix[-1] % 2 == 0):
                if(pix[-1] != 0):

```

```

        pix[-1] -= 1
    else:
        pix[-1] += 1

    else:
        if (pix[-1] % 2 != 0):
            pix[-1] -= 1

    pix = tuple(pix)
    yield pix[0:3]
    yield pix[3:6]
    yield pix[6:9]

def encode_enc(newimg, data):
    w = newimg.size[0]
    (x, y) = (0, 0)

    for pixel in modPix(newimg.getdata(), data):

        # Putting modified pixels in the new image
        newimg.putpixel((x, y), pixel)
        if (x == w - 1):
            x = 0
            y += 1
        else:
            x += 1

    # Encode data into image
    def encode(cipher_text, img):

        image = Image.open(img, 'r')
        print('Image opened')
        data = cipher_text
        if (len(data) == 0):
            print('Data length is zero')
            raise ValueError('Data is empty')

        newimg = image.copy()
        print(newimg)
        encode_enc(newimg, data)

        newimg.save(img, str(img.split(".")[1].upper()))
        return newimg

```



```

# Decode the data in the image
def decode(img):

    image = Image.open(img, 'r')

    data = ''
    imgdata = iter(image.getdata())

    while (True):
        pixels = [value for value in imgdata.__next__()[ :3] +
                  imgdata.__next__()[ :3] +
                  imgdata.__next__()[ :3]]

        # string of binary data
        binstr = ''

        for i in pixels[:8]:
            if (i % 2 == 0):
                binstr += '0'
            else:
                binstr += '1'

        data += chr(int(binstr, 2))
        if (pixels[-1] % 2 != 0):
            return data

```

Core Folder:

- **Asgi.py**

-
-

```

"""

```

ASGI config for core project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

<https://docs.djangoproject.com/en/3.2/howto/deployment/asgi/>

```

"""

```

```

import os

```

```

from django.core.asgi import get_asgi_application

```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')
```

```
application = get_asgi_application()
```

- **Settings.py**

```
"""
Django settings for core project.
Generated by 'django-admin startproject' using Django 3.2.4.
For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-6*$@&6mm$0ny%wiwvsqd!ueg_8z))y3)!$-e#kxn_(09#)_p74'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
```

```

'django.contrib.messages',
'django.contrib.staticfiles',
'blog.apps.BlogConfig',
'bombe.apps.BombeConfig',
]

MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'core.urls'

TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]

WSGI_APPLICATION = 'core.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
'default': {
'ENGINE': 'django.db.backends.postgresql_psycopg2',
'NAME': 'cyber_sec_blogs_and_bombe',
'USER': 'heyuser',

```

```

        'PASSWORD': 'vasavi',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True

USE_L10N = True

USE_TZ = True

```

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_USE_TLS = True
EMAIL_PORT = 587
EMAIL_HOST_USER = 'hostdjango@gmail.com'
EMAIL_HOST_PASSWORD = 'hostpwd'

LOGIN_REDIRECT_URL = 'blog:home'
LOGOUT_REDIRECT_URL = 'blog:home'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

• **Urls.py**

"""core URL

Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
<https://docs.djangoproject.com/en/3.2/topics/http/urls/>

Examples:

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `path('', views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `path('', Home.as_view(), name='home')`

Including another URLconf

1. Import the `include()` function: `from django.urls import include, path`
2. Add a URL to `urlpatterns`: `path('blog/', include('blog.urls'))`

"""

`from django.contrib import admin`

`from django.urls import path, include`

`from django.conf import settings`

```

from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('bombe/', include('bombe.urls')),
    path('', include('blog.urls')),
    path('accounts/', include('django.contrib.auth.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

- **Wsgi.py**

```

"""
WSGI config for core project.
It exposes the WSGI callable as a module-level variable named ``application``.
For more information on this file, see
https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')

application = get_wsgi_application()

```

Bombe machine.py (Raw code):

```

#
Implementation
of Affine
Cipher in
Python

# Extended Euclidean Algorithm for finding modular inverse
# eg: modinv(7, 26) = 15
def egcd(a, b):
    x,y, u,v = 0,1, 1,0
    while a != 0:
        q, r = b//a, b%a
        m, n = x-u*q, y-v*q

```

```

        b,a, x,y, u,v = a,r, u,v, m,n
gcd = b
return gcd, x, y

def modinv(a, m):
    gcd, x, y = egcd(a, m)
    if gcd != 1:
        return None # modular inverse does not exist
    else:
        return x % m

# affine cipher encryption function
# returns the cipher text
def affine_encrypt(text, key):
    '''
    C = (a*P + b) % 26
    '''
    return ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26)
                        + ord('A')) for t in text.upper().replace(' ', '') ])

# affine cipher decryption function
# returns original text
def affine_decrypt(cipher, key):
    '''
    P = (a^-1 * (C - b)) % 26
    '''
    return ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') - key[1]))
                        % 26) + ord('A')) for c in cipher ])

def generateKey(string, key):
    key = list(key)
    if len(string) == len(key):
        return(key)
    elif len(string) > len(key):
        for i in range(len(string) - len(key)):
            key.append(key[i % len(key)])
        return("".join(key))
    else:
        key = key[:len(string)]
        return(key)
print(key)

```

```

# This function returns the
# encrypted text generated
# with the help of the key
def cipherText(string, key):
    cipher_text = []
    for i in range(len(string)):
        x = (ord(string[i]) + ord(key[i])) % 26
        x += ord('A')
        cipher_text.append(chr(x))
    return("".join(cipher_text))

# This function decrypts the
# encrypted text and returns
# the original text
def originalText(cipher_text, key):
    orig_text = []
    for i in range(len(cipher_text)):
        x = (ord(cipher_text[i]) - ord(key[i]) + 26) % 26
        x += ord('A')
        orig_text.append(chr(x))
    return("".join(orig_text))

# Python program implementing Image Steganography

# PIL module is used to extract
# pixels of image and modify it
from PIL import Image

# Convert encoding data into 8-bit binary
# form using ASCII value of characters
def genData(data):

    # list of binary codes
    # of given data
    newd = []

    for i in data:
        newd.append(format(ord(i), '08b'))

    return newd

# Pixels are modified according to the
# 8-bit binary data and finally returned
def modPix(pix, data):

```



```

datalist = genData(data)
lendata = len(datalist)
imdata = iter(pix)

for i in range(lendata):

    # Extracting 3 pixels at a time
    pix = [value for value in imdata.__next__():3] +
           imdata.__next__():3] +
           imdata.__next__():3]]

    # Pixel value should be made
    # odd for 1 and even for 0
    for j in range(0, 8):
        if (datalist[i][j] == '0' and pix[j]% 2 != 0):
            pix[j] -= 1

        elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
            if(pix[j] != 0):
                pix[j] -= 1
            else:
                pix[j] += 1
            # pix[j] -= 1

    # Eighth pixel of every set tells
    # whether to stop or read further.
    # 0 means keep reading; 1 means the
    # message is over.
    if (i == lendata - 1):
        if (pix[-1] % 2 == 0):
            if(pix[-1] != 0):
                pix[-1] -= 1
            else:
                pix[-1] += 1

    else:
        if (pix[-1] % 2 != 0):
            pix[-1] -= 1

    pix = tuple(pix)
    yield pix[0:3]
    yield pix[3:6]
    yield pix[6:9]

```

```

def encode_enc(newimg, data):
    w = newimg.size[0]
    (x, y) = (0, 0)

    for pixel in modPix(newimg.getdata(), data):

        # Putting modified pixels in the new image
        newimg.putpixel((x, y), pixel)
        if (x == w - 1):
            x = 0
            y += 1
        else:
            x += 1

# Encode data into image
def encode(cipher_text):
    img = input("Enter image name(with extension) for stego-encoding: ")
    image = Image.open(img, 'r')

    data = cipher_text
    if (len(data) == 0):
        raise ValueError('Data is empty')

    newimg = image.copy()
    encode_enc(newimg, data)

    new_img_name = input("Enter the name of new image(with extension) : ")
    newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))

# Decode the data in the image
def decode():
    img = input("Enter image name(with extension) for stego-decoding : ")
    image = Image.open(img, 'r')

    data = ''
    imgdata = iter(image.getdata())

    while (True):
        pixels = [value for value in imgdata.__next__()[ :3] +
                  imgdata.__next__()[ :3] +
                  imgdata.__next__()[ :3]]

```

```

        # string of binary data
        binstr = ''

        for i in pixels[:8]:
            if (i % 2 == 0):
                binstr += '0'
            else:
                binstr += '1'

        data += chr(int(binstr, 2))
        if (pixels[-1] % 2 != 0):
            return data

def main():

    a = int(input(":: Welcome to Steganography ::\n1. Encode\n2. Decode\n3. exit\n"))
    if( a== 1):
        text = input('Enter plain text:\n')
        key = [17,20]

        # calling affine encryption function
        affine_encrypted_text = affine_encrypt(text,key)
        print('affine encrypted Text: {}'.format( affine_encrypted_text ))

        # calling ceaser encryption function
        string = affine_encrypted_text
        keyword = input("enter the public key\n")
        key2 = generateKey(string, keyword)
        ceaser_encrypted_text = cipherText(string,key2)
        print('private key is :')
        print(key2)
        print("ceaser encrypted text : ",ceaser_encrypted_text)

        # calling steganographic functions
        encode(ceaser_encrypted_text)
        main()

    elif( a== 2):
        key = [17,20]
        key2 = input('Enter private key\n')
        stegoDecrypt = decode()
        print("\nDecoded Word : " + stegoDecrypt)
        # calling ceaser decryption function
        ceaserDecrypt = originalText(stegoDecrypt, key2)

```

```

        print("Ceaser Decrypted Text : ", ceaserDecrypt)

    # calling affine decryption function
    print('Original/ Affine decrypted Text: {}'.format( affine_decrypt(ceaserDecrypt,
key) ))
    main()

else:
    exit(0)
if __name__ == '__main__':
    main()

```

Manage.py:

```

#!/usr/bin/env
python

        """Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did "
            you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()

```

7. Output Screenshots

USER MODULE

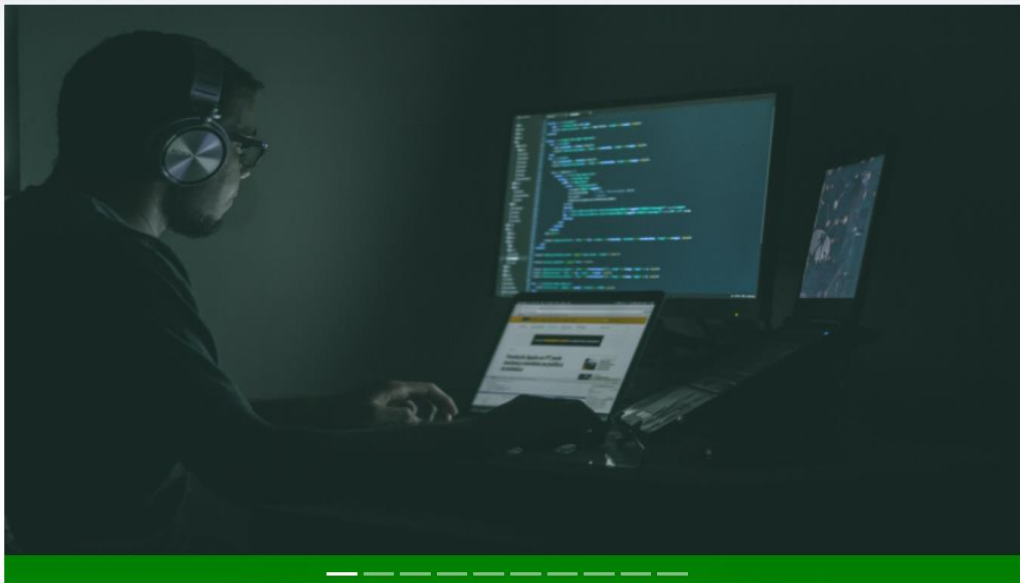
i. Home Page:

This is the view of the website's homepage where all the options are visible to the user

CyberSec Blogs & Bombe

[Home](#) [Policy](#) [Logout](#) [Encode](#) [My Encodes](#) [Decode](#) [Contact](#)

Welcome to CyberSec Blogs ramakanth



ii. Login Page:

This is where the user can login

CyberSec Blogs & Bombe

[Home](#) [Policy](#) [Sign in](#) [Sign Up](#) [Contact](#)

Login

Username:

Password:

Login

Reset

[Not a Member...Sign up](#)

[Forgot Password](#)

iii. Sign Up Page:

New users can provide their details and create their new account here

← → ↻ ⓘ 127.0.0.1:8000/signup/

🔖 📄 🔍 | 🌟 🗑️ 👤 ⋮

CyberSec Blogs & Bombe

[Home](#) [Policy](#) [Sign in](#) [Sign Up](#) [Contact](#)

Sign Up

Username:

Email:

Password:

Password confirmation:

Sign up

Reset

[Already a Member](#)

iv. Password Reset:

This is where the user can reset his/her password. A link will be sent to the users registered mail ID where the following form will be displayed to reset the password

Django administration

Home » Password reset

Password reset

Forgotten your password? Enter your email address below, and we'll email instructions for setting a new one.

Email address:

Reset my password

v. About & Blogs:

Here the model blogs and About us will be displayed

CyberSec Blogs & Bombe

Home Policy Logout Encode My Encodes Decode Contact

How to Boost Cyber Security As An Ethical Hacker

ramakanth | June 20, 2021, 7:21 a.m.

Cyber security is protecting the Information systems from damage or theft of the software, hardware and to the sensitive information on them. It involves the controlling of the access to the hardware

Read More →

The evolving threat of ransomware and how to guard against attacks

venkateswarlu | June 20, 2021, 7:16 a.m.

The advanced ransomware attacks of today take seconds to compromise endpoints and have the potential to cause untold damage to systems and infrastructure, making it critical to ensure organizations ar

Read More →

What is Django?

venkateswarlu | June 19, 2021, 7:38 a.m.

Python framework

About Us

Build with ❤️ by Ramakanth and Venkatesh

This project is build in order to stay up to date about Cybersecurity, its inventions and cutting edge technologies using Blogs. Cybersecurity is one such that domain that keeps updating and hence one requires to attain knowledge constantly about the field. In this busy world we hardly find any time to browse the internet for updates. This approach evidently provides us with information regarding CS domain as blogs. Blogs can relate to any concept such as Cybersecurity awareness, upcoming technologies, preferred tools, security breaches, Ethical Hacking, etc.

HAPPY BLOGGING!!!!

Know more about Venkatesh!

Know more about Ramakanth!

vi. Blogs:

This is the view of a sample blog with admin tag

CyberSec Blogs & Bombe Home Policy Logout Encode My Encodes Decode Contact

The evolving threat of ransomware and how to guard against attacks

venkateswarlu | June 20, 2021, 7:16 a.m. | admin

The advanced ransomware attacks of today take seconds to compromise endpoints and have the potential to cause untold damage to systems and infrastructure, making it critical to ensure organizations are prepared. As attacks grow in sophistication, the impact they can have goes far beyond the financial losses and decreased productivity often associated with systems going down. With digital transformation taking a hold of organizations globally, the convergence of IT and OT systems has led ransomware attacks to target new data and technology types. Devices in the field including the Industrial Internet of Things (IIoT) have become new targets, resulting in malicious actors shifting their focus from corporate networks to the OT edge. At the OT edge, devices carry far more value than sensitive information and are responsible for people's physical safety, demonstrating the severity of attacks on these networks. As a result, power grids, transportation management infrastructures, medical systems, and other critical resources are being threatened more than ever before. The importance of engaging all internal and

About Us

Build with ❤ by Ramakanth and Venkatesh

This project is build in order to stay up to date about Cybersecurity, its inventions and cutting edge technologies using Blogs. Cybersecurity is one such that domain that keeps updating and hence one requires to attain knowledge constantly about the field. In this busy world we hardly find any time to browse the internet for updates. This approach evidently provides us with information regarding CS domain as blogs. Blogs can relate to any concept such as Cybersecurity awareness, upcoming technologies, preferred tools, security breaches, Ethical Hacking, etc.

HAPPY BLOGGING!!!!

[Know more about Venkatesh!](#)

vii. User policy:

These are the terms and conditions for using our website

CyberSec Blogs & Bombe Home Policy Logout Encode My Encodes Decode Contact

Our Policy

- All blogs posted are subject to the admin of the page.
- The admin is responsible for the content posted in the blog.
- If any viewer is interested in writing their own blog, the blog can be sent as a message in CONTACT section of the website.
- If the blog is acceptable and doesn't require any user discretion, it will be posted.
- In case a blog is posted after being reviewed, then the user details will be posted.
- Bombe machine in our website is used for demonstration of Encryption and decryption using combined steganography and cryptography.
- Use of Bombe machine has to be restricted to educational purposes only.
- Admins of this site do not hold any responsibility for any sort of misuse of the Bombe machine.
- The user can complain against any blog by giving the blog details in CONTACT section and action will be taken if found fallacious.
- Any suggestions or security concerns with aspect to technical development or interface structure improvement can be brought to admins notice.

viii. Encoding in bombe:

This is the encoding section where the user will give the following details and get their information encoded into the given image

CyberSec Blogs & Bombe

Home Policy Logout Encode My Encodes Decode Contact

Title:

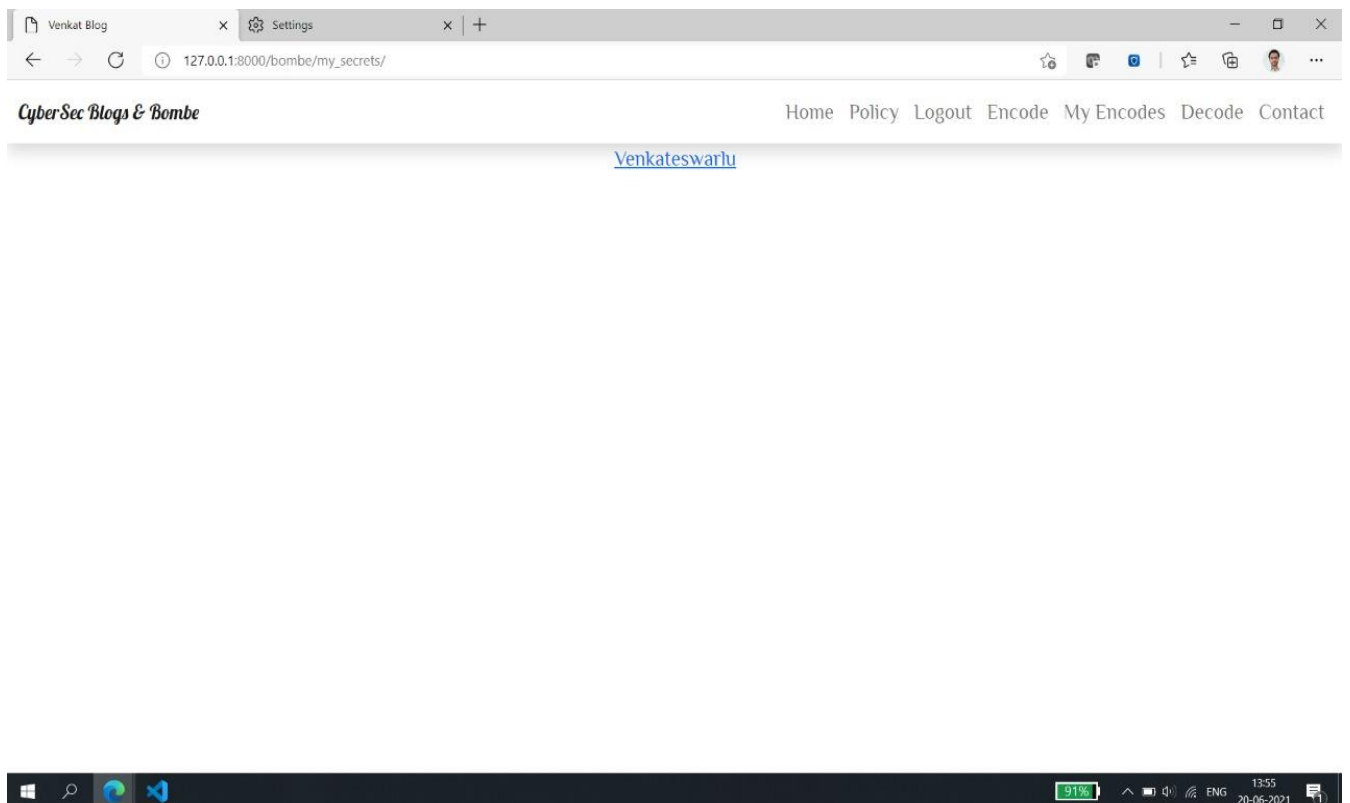
Plain Text:

Public Key:

No file chosen

ix. My encodes:

This is the history of encodes



x. Decoding in bombe:

This is the decoding section where the user will give the following details and get their image decoded and obtain the intended information.

Private Key:

Image Name:

Enter

8. TECHNOLOGIES USED

Tools

- HTML
- CSS
- Bootstrap
- Django
- SQLite

a. HTML

Every webpage you look at is written in a language called HTML. You can think of HTML as the skeleton that gives every webpage structure. In this course, we will use HTML to add paragraphs, headings, images, and links to a webpage. In the editor to the right, there is a tab called test.html. This is the file we will type our HTML into. Like any language, it has its own special syntax. A browser's job is to transform the code in test.html into a recognizable webpage! It knows how to lay out the page by following the HTML syntax.

b. CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications. CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .CSS file, and reduce complexity and repetition in the structural content.

c. Bootstrap

Bootstrap is a potent front-end framework used to create modern websites and web apps. It's open-source and free to use yet features numerous HTML and CSS templates for UI interface elements such as forms typography, forms, buttons, navigation, and other interface components. Bootstrap also supports JavaScript extensions. Responsive design makes it possible for a web page or app to detect the visitor's screen size and orientation and automatically adapt the display; accordingly, the mobile first approach assumes that smartphones, tablets and task-specific Mobile apps are employees' primary tools for getting work done and addresses the requirements of those technologies in design. Bootstrap includes user

interface components, layouts, and JS tools along with the framework for implementation. The software is available precompiled or as source code.

d. Django

Django (named after the Django Reinhardt) is a high-level python-based free and open-source web framework that follows the model-view-template (MVT) architectural pattern. It is slightly different from the MVC pattern as it maintains its own conventions, so, the controller is handled by the framework itself. The template is a presentation layer. It is an HTML file mixed with Django Template Language (DTL). The developer provides the model, the view, and the template then maps it to a URL, and finally, Django serves it to the user.

Django can be broken into many components:

Models.py file: This file defines your data model by extending your single line of code into full database tables and add a pre-built administration section to manage content.

Urls.py file: It uses a regular expression to capture URL patterns for processing.

Views.py file: It is the main part of Django. The actual processing happens in view.

When a visitor lands on Django page, first Django checks the URLs pattern you have created and used the information to retrieve the view. After that view processes the request, querying your database if necessary, and passes the requested information to a template.

Magane.py file: It is user for local host website deployment for testing and demonstration purposes.

e. PostgreSQL

PostgreSQL also known as Postgres, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley. In 1996, the project was renamed to PostgreSQL to reflect its support for SQL. After a review in 2007, the development team decided to keep the name PostgreSQL and the alias Postgres.

PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users. It is the default database for macOS Server and is also available for Windows, Linux, FreeBSD, and OpenBSD

9. CONCLUSION

Now-a-days peeps are pretty busy and hardly find any time to browse regarding particular topics in cybersecurity. Cybersecurity is one such domain which keeps constantly updating and one requires to stay up to date regarding cutting edge technologies, security updates, upcoming softwares, security breaches, tools, etc. Cybersec provides the user brief information regarding the updates in the form of blogs in brief and concise manner. In order to transfer any sort of confidential information, CIA (Confidentiality, Integrity and Accessibility) triad has to be preserved. Keeping this in view, highly advanced Bombe machine is designed wherein the data can be safely encoded in the pixels of an image and the generated stego-image can be decoded to get the intended information. The system is built fully in Django Framework in back-end and HTML, CSS in front-end. It is free to register for any user and they can get all the latest updates regarding cybersecurity including security breaches and security improvements across the world.

GITHUB LINK: [Ramakanth001/CyberSec-Blogs-Bombe](https://github.com/Ramakanth001/CyberSec-Blogs-Bombe):

10. REFERENCES

- <https://docs.djangoproject.com> - Django Documentation
- International Journal of Research Studies in Computer Science and Engineering (IJRSCSE) Volume 2, Issue 5, May 2015, PP 45-49 – PHD on “**A Study on Combined Cryptography and Steganography**” by Vishnu S Babu and Prof. Helen K J
- <https://getbootstrap.com> - Bootstrap
- <https://developer.mozilla.org/en-US/docs/Web/HTML> - HTML Documentation
- <https://www.postgresql.org/docs/> - postgresql Documentation
- <https://tutorial.djangogirls.org/en/> - Tutorials of Django