



Vellore Institute
of Technology

Project Report

Network Security and Cryptography Fundamentals (CSE3045)

Project Title | Encryption and Decryption of Image,
Video and File using AES-CBC Cryptography

D r . R A B I N D R A K U M A R S I N G H

Lokesh kanna – 19MIA1014

Mogalapu Sri Sai Vivek – 19MIA1070

Annugraha.S- 19MIA1059

CERTIFICATE

This is to certify that the project work entitled “Encryption and Decryption of Image, Video and File using AES-CBC Cryptography” that is being submitted by the above-mentioned team members for Network Security and Cryptography Fundamentals (CSE3045) is a record of Bonafide work done under my supervision. The content of this project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place: Chennai

Date:02/12/2021

Signature of faculty

Dr. RABINDRA KUMAR SINGH

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Prof. Dr. Rabindra Kumar Singh who gave us the golden opportunity to do this wonderful project on the topic “Encryption and Decryption of Image, Video and File using AES-CBC Cryptography”, which also helped us in doing a lot of research and we came to know about so many new things we are really thankful to them. Secondly, we would also like to thank our friends and groupmates who helped us a lot in finalizing this project within the limited time frame.

Abstract

Using internet and network are increasing rapidly. Everyday a lot of files have been exchanging among users. Some of the files is sensitive that need to be protected from intruders. Encryption and decryption algorithms play vital roles to protect original data from unauthorized access. Various kind of algorithms already exist to encrypt or decrypt data. So this project is about implementation of Image, Video and File Encryption and Decryption using AES cryptography in CBC mode in a web browser.

Introduction

Advanced Encryption Standard (AES) algorithm is one on the most common and widely symmetric block cipher algorithm used in worldwide. This algorithm has an own particular structure to encrypt and decrypt sensitive data and is applied in hardware and software all over the world. It is extremely difficult to hackers to get the real data when encrypting by AES algorithm. Till date is not any evidence to crake this algorithm. AES has the ability to deal with three different key sizes such as AES 128, 192 and 256 bit and each of these ciphers has 128-bit block size. In this project we will be using AES in CBC mode with 256 bits.

CBC (Cipher Blocker Chaining) is an advanced form of block cipher encryption. With CBC mode encryption, each ciphertext block is dependent on all plaintext blocks processed up to that point. This adds an extra level of complexity to the encrypted data.

The main feature of this scheme of encryption is that identical blocks of the clear text belong to one message, are ciphered into various blocks of cipher text.

Here are the main characteristics of this scheme:

If one bit of the transferred message will be corrupted, it will damage the one more following block. Other blocks would be safe.

In case of loss or an insert at least one bit into cipher text, there will be a shift of bits and borders of blocks that will lead to a wrong decryption of all subsequent blocks of cipher text

The malefactor can add blocks by the end of the ciphered message, supplementing with that a clear text

Two identical messages have identical cipher texts if the same initialization vector (initialization vector (IV)) was used.

LITERATURE SURVEY:

1. This paper " A Study of Encryption Algorithms AES, DES and RSA for Security 2013," International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) has implemented experiments for three encryption techniques: AES, DES and RSA algorithms and compared their performance based on the analysis of their stimulated time at the time of encryption and decryption. Results of the experiments were used to analyze the effectiveness of each algorithm.
2. This paper "A Survey on Different Image Encryption and Decryption Techniques 2013," presented a survey of over 25 research papers dealing with image encryption techniques that scrambled the pixels of the image and decrease the correlation among the pixels, which lowers correlation among the pixel and produces the encrypted image. A survey of different existing image encryption and decryption techniques was given. Additionally, the paper focused on the functionality of Image encryption and decryption techniques.
3. This paper "Text and Image Encryption / Decryption Using Advanced Encryption Standard 2014", implemented text and image encryption and decryption using AES. Features of data depends on its types. Therefore, same encryption technique cannot be used for all types of data. If the Images have large data size and also have problems with real time constrain hence similar method cannot be used to protect images as well as text from unauthorized access. Few variations in method AES can be used to protect image as well as text.
4. In this paper "A Review on DES, AES and Blowfish for Image Encryption & Decryption 2015", the authors discussed and surveyed DES, AES and Blowfish for Image Encryption and Decryption. In today's world it is a crucial concern that while transferring image from one network to another over the internet, the proper encryption and decryption should be applied so that unauthorized access can be prevented. The authors also surveyed related research and did some problem identification and provided suggestions that can be useful for image encryption. This is to enhance the performance and encryption and decryption times of the image.

Implementation areas:

CBC is an example of effective challenge-response authentication. Because cipher block chaining relies on using previous ciphertext blocks to encrypt subsequent plaintext blocks, hackers and decryptors must have all ciphertext blocks available in order to successfully decrypt entire CBC outputs. This multistep encryption mechanism makes it difficult to deconstruct, thereby increasing the security of the messages it is trying to encrypt. A user or group that requires access to a certain set of documents must be able to present the necessary ciphertext blocks to successfully decrypt the entire message or text. CBC can also be found in most modern applications and devices that need encryption functionality such as WhatsApp, Facebook Messenger and Intel and AMD processor and Cisco devices like router, switch, etc.

Methodology and tools used:

Files will be encrypted using AES-CBC symmetric encryption. The encryption key is derived from the password which we provide and a random salt using PBKDF2 derivation with 10000 iterations of SHA256 hashing. To implement these steps, we will be using JavaScript and html.

Code

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>File Encryption / Decryption</title>
  </head>
  <style>
    body {
      font-family: 'Helvetica', 'Arial', 'sans-serif';
      color: black;
      font-size: 11pt;
    }

    a, a:link, a:visited, a:active {
      color: blue;
```

```
        text-decoration: underline;
    }
```

```
a:hover {
    cursor:pointer;
    color: red;
}
```

```
.black10pointcourier {
    font-family: 'courier';
    color: black;
    font-size: 10pt;
}
```

```
.container {
    width: 80%;
    margin: 0 auto;
}
```

```
.dropzone {
    border: 10px dashed gray;
    width: 20%;
    padding: 2% 2% 5% 2%;
    text-align: center;
    margin: 5px 0 5px 0;
}
```

```
.divTablefullwidth{
    display: table;
    width: 100%;
}
```

```
.divTable{
    display: table;
```

```

    }

    .divTableRow {
        display: table-row;
    }
    .divTableCell {
        display: table-cell;
        padding: 3px 3px;
    }
    .divTableBody {
        display: table-row-group;
    }

    .greenspan {
        color: green;
    }

    .redspan {
        color: red;
    }
</style>
<body>
    <div class=container>
        <div class="divTablefullwidth">
            <div class="divTableBody">
                <div class="divTableRow">
                    <div class="divTableCell" style="float: left;">
                        <h1>Web Browser Based File
Encryption / Decryption</h1>

                        <h4>Use your web browser to encrypt
and decrypt files.</h4>

                    </div>
                    <div class="divTableCell" style="float: right;">
                        <h1>

```



```

        <button id="btnRefresh"
onClick="javascript:location.reload();">Refresh Page</button>
        <button id="btnDivEncrypt"
onClick="javascript:switchdiv('encrypt');">Encrypt a File</button>
        <button id="btnDivDecrypt"
onClick="javascript:switchdiv('decrypt');">Decrypt a File</button>
    </h1>
</div>
</div>
</div>
</div>
</div>
<div class=container>
    <hr>
</div>
<div class="container" id=divEncryptfile>
    <h2>Encrypt a File</h2>
    <p>To encrypt a file, enter a password and drop the file to be
encrypted into the dropzone below. The file will then be encrypted using the password, then
you'll be given an opportunity to save the encrypted file to your system.</p>
    <div class="divTable">
        <div class="divTableBody">
            <div class="divTableRow">
                <div class="divTableCell">Password</div>
                <div class="divTableCell"><input
id=txtEncpassphrase type=password size=30 onkeyup=javascript:encvalidate();
value=""></div>
                <div class="divTableCell">(minumum length
eight characters, make sure it strong!)</div>
            </div>
            <div class="divTableRow">
                <div class="divTableCell">Password
(retype)</div>

```

```

<div class="divTableCell"><input
id=txtEncpassphraseretype type=password size=30 onkeyup=javascript:encvalidate();
value=""></div>

<div class="divTableCell"><span
class=greenspan id=spnCheckretry></span></div>

</div>

</div>

</div>

<p> </p>

<div>

<div class=dropzone id="endropzone"
ondrop="drop_handler(event);" ondragover="dragover_handler(event);"
ondragend="dragend_handler(event);">

<p>Drag and drop the file to be encrypted into this
dropzone, or click <a onclick=javascript:encfileElem.click();>here</a> to select file.</p>

<p><span id=spnencfilename></span></p>

</div>

<input type="file" id="encfileElem" style="display:none"
onchange="selectfile(this.files)">

</div>

<p> </p>

<div class="divTable">

<div class="divTableBody">

<div class="divTableRow">

<div class="divTableCell"><button
id=btnEncrypt onclick=javascript:encryptfile(); disabled>Encrypt File</button></div>

<div class="divTableCell"><span
id=spnEncstatus></span></div>

</div>

</div>

</div>

<p> </p>

```

```

        <div>
            <a id=aEncsavefile hidden><button>Save Encrypted
File</button></a>
        </div>
    <p> </p>
</div>
<div class="container" id=divDecryptfile>
    <h2>Decrypt a File</h2>
    <p>Decrypt a file using the password that was previously used to
encrypt the file. After the file is decrypted, you'll be given an opportunity to save the
decrypted file to your system.</p>
    <div class="divTable">
        <div class="divTableBody">
            <div class="divTableRow">
                <div class="divTableCell">Password</div>
                <div class="divTableCell"><input
id=txtDecpassphrase type=password size=30 onkeyup=javascript:decvalidate();
value=""></div>
            </div>
        </div>
    </div>
    <div>
        <div class=dropzone id="decdropzone"
ondrop="drop_handler(event);" ondragover="dragover_handler(event);"
ondragend="dragend_handler(event);">
            <p>Drag and drop file to be decrypted into this
dropzone, or click <a role=button onclick=javascript:decfileElem.click();>here</a> to select
file.</p>
            <p><span id=spndecfilename></span></p>
        </div>
        <input type="file" id="decfileElem"
style="display:none" onchange="selectfile(this.files)">
    </div>

```

```

        <p> </p>

        <div class="divTable">
            <div class="divTableBody">
                <div class="divTableRow">
                    <div class="divTableCell"><button
id=btnDecrypt onclick=javascript:decryptfile(); disabled>Decrypt File</button></div>
                    <div class="divTableCell"><span
id=spnDecstatus></span></div>
                </div>
            </div>
        </div>

        <div>
            <p> </p>
            <div>
                <a id=aDecsavefile hidden><button>Save Decrypted
File</button></a>
            </div>
        </div>
    </div>
</div>
</body>
</html>
<script type="text/javascript">
    var mode=null;
    var objFile=null;
    switchdiv('encrypt');
    function switchdiv(t) {
        if(t=='encrypt') {
            divEncryptfile.style.display='block';
            divDecryptfile.style.display='none';
            btnDivEncrypt.disabled=true;
            btnDivDecrypt.disabled=false;
            mode='encrypt';
        } else if(t=='decrypt') {

```

```

        divEncryptfile.style.display='none';
        divDecryptfile.style.display='block';
        btnDivEncrypt.disabled=false;
        btnDivDecrypt.disabled=true;
        mode='decrypt';
    }
}

function encvalidate() {
    if(txtEncpassphrase.value.length>=8 &&
txtEncpassphrase.value==txtEncpassphraseretype.value) {
        spnCheckretype.classList.add("greenspan");
        spnCheckretype.classList.remove("redspan");
        spnCheckretype.innerHTML='&#10004;';
    } else {
        spnCheckretype.classList.remove("greenspan");
        spnCheckretype.classList.add("redspan");
        spnCheckretype.innerHTML='&#10006;';
    }

    if( txtEncpassphrase.value.length>=8 &&
txtEncpassphrase.value==txtEncpassphraseretype.value && objFile ) {
        btnEncrypt.disabled=false; } else { btnEncrypt.disabled=true; }
    }

function decvalidate() {
    if( txtDecpassphrase.value.length>0 && objFile ) {
        btnDecrypt.disabled=false; } else { btnDecrypt.disabled=true; }
    }

function drop_handler(ev) {
    console.log("Drop");
    ev.preventDefault();
    var dt = ev.dataTransfer;

```

```

    if (dt.items) {
        // Use DataTransferItemList interface to access the file(s)
        for (var i=0; i < dt.items.length; i++) {
            if (dt.items[i].kind == "file") {
                var f = dt.items[i].getAsFile();
                console.log("... file[" + i + "].name = " + f.name);
                objFile=f;
            }
        }
    } else {
        // Use DataTransfer interface to access the file(s)
        for (var i=0; i < dt.files.length; i++) {
            console.log("... file[" + i + "].name = " + dt.files[i].name);
        }
        objFile=dt.files[0];
    }
    displayfile()
    if(mode=='encrypt') { encvalidate(); } else if(mode=='decrypt') {
decvalidate(); }
}

```

```

function dragover_handler(ev) {
    console.log("dragOver");
    // Prevent default select and drag behavior
    ev.preventDefault();
}

```

```

function dragend_handler(ev) {
    console.log("dragEnd");
    // Remove all of the drag data
    var dt = ev.dataTransfer;
    if (dt.items) {
        // Use DataTransferItemList interface to remove the drag data
        for (var i = 0; i < dt.items.length; i++) {

```

```

        dt.items.remove(i);
    }
} else {
    // Use DataTransfer interface to remove the drag data
    ev.dataTransfer.clearData();
}
}

function selectfile(Files) {
    objFile=Files[0];
    displayfile()
    if(mode=='encrypt') { encvalidate(); } else if(mode=='decrypt') {
decvalidate(); }
}

function displayfile() {
    var s;
    var sizes = ['Bytes', 'KB', 'MB', 'GB', 'TB'];
    var bytes=objFile.size;
    var i = parseInt(Math.floor(Math.log(bytes) / Math.log(1024)));
    if(i==0) { s=bytes + ' ' + sizes[i]; } else { s=(bytes / Math.pow(1024,
i)).toFixed(2) + ' ' + sizes[i]; }

    if(mode=='encrypt') {
        spnencfilename.textContent=objFile.name + ' (' + s + ')';
    } else if(mode=='decrypt') {
        spndecfilename.textContent=objFile.name + ' (' + s + ')';
    }
}

function readfile(file){
    return new Promise((resolve, reject) => {
        var fr = new FileReader();
        fr.onload = () => {

```

```

        resolve(fr.result )
    };
    fr.readAsArrayBuffer(file);
});
}

async function encryptfile() {
    btnEncrypt.disabled=true;

    var plaintextbytes=await readFile(objFile)
    .catch(function(err){
        console.error(err);
    });
    var plaintextbytes=new Uint8Array(plaintextbytes);

    var pbkdf2iterations=10000;
    var passphrasebytes=new TextEncoder("utf-
8").encode(txtEncpassphrase.value);
    var pbkdf2salt=window.crypto.getRandomValues(new Uint8Array(8));

    var passphrasekey=await window.crypto.subtle.importKey('raw',
passphrasebytes, { name: 'PBKDF2'}, false, ['deriveBits'])
    .catch(function(err){
        console.error(err);
    });
    console.log('passphrasekey imported');

    var pbkdf2bytes=await window.crypto.subtle.deriveBits({ "name": 'PBKDF2',
"salt": pbkdf2salt, "iterations": pbkdf2iterations, "hash": 'SHA-256'}, passphrasekey, 384)

    .catch(function(err){
        console.error(err);
    });
    console.log('pbkdf2bytes derived');

```



```
pbkdf2bytes=new Uint8Array(pbkdf2bytes);

keybytes=pbkdf2bytes.slice(0,32);
ivbytes=pbkdf2bytes.slice(32);

var key=await window.crypto.subtle.importKey('raw', keybytes, { name: 'AES-
CBC', length: 256}, false, ['encrypt'])
.catch(function(err){
    console.error(err);
});
console.log('key imported');

var cipherbytes=await window.crypto.subtle.encrypt({ name: "AES-CBC", iv:
ivbytes}, key, plaintextbytes)
.catch(function(err){
    console.error(err);
});

if(!cipherbytes) {
    spnEncstatus.classList.add("redspan");
    spnEncstatus.innerHTML='<p>Error encrypting file. See console
log.</p>';
    return;
}

console.log('plaintext encrypted');
cipherbytes=new Uint8Array(cipherbytes);

var resultbytes=new Uint8Array(cipherbytes.length+16)
resultbytes.set(new TextEncoder("utf-8").encode('Salted__'));
resultbytes.set(pbkdf2salt, 8);
resultbytes.set(cipherbytes, 16);

var blob=new Blob([resultbytes], { type: 'application/download'});
```

```

        var blobUrl=URL.createObjectURL(blob);
        aEncsavefile.href=blobUrl;
        aEncsavefile.download=objFile.name + '.enc';

        spnEncstatus.classList.add("greenspan");
        spnEncstatus.innerHTML='<p>File encrypted.</p>';
        aEncsavefile.hidden=false;
    }

    async function decryptfile() {
        btnDecrypt.disabled=true;

        var cipherbytes=await readfile(objFile)
        .catch(function(err){
            console.error(err);
        });
        var cipherbytes=new Uint8Array(cipherbytes);

        var pbkdf2iterations=10000;
        var passphrasebytes=new TextEncoder("utf-
8").encode(txtDecpassphrase.value);
        var pbkdf2salt=cipherbytes.slice(8,16);

        var passphrasekey=await window.crypto.subtle.importKey('raw',
passphrasebytes, { name: 'PBKDF2'}, false, ['deriveBits'])
        .catch(function(err){
            console.error(err);
        });

        console.log('passphrasekey imported');
    }

```

```
var pbkdf2bytes=await window.crypto.subtle.deriveBits({"name": 'PBKDF2',  
"salt": pbkdf2salt, "iterations": pbkdf2iterations, "hash": 'SHA-256'}, passphrasekey, 384)
```

```
.catch(function(err){  
    console.error(err);  
});  
console.log('pbkdf2bytes derived');  
pbkdf2bytes=new Uint8Array(pbkdf2bytes);
```

```
keybytes=pbkdf2bytes.slice(0,32);  
ivbytes=pbkdf2bytes.slice(32);  
cipherbytes=cipherbytes.slice(16);
```

```
var key=await window.crypto.subtle.importKey('raw', keybytes, {name: 'AES-  
CBC', length: 256}, false, ['decrypt'])
```

```
.catch(function(err){  
    console.error(err);  
});  
console.log('key imported');
```

```
var plaintextbytes=await window.crypto.subtle.decrypt({name: "AES-CBC",  
iv: ivbytes}, key, cipherbytes)
```

```
.catch(function(err){  
    console.error(err);  
});
```

```
if(!plaintextbytes) {  
    spnDecstatus.classList.add("redspan");  
    spnDecstatus.innerHTML=<p>Error decrypting file. Password may  
be incorrect.</p>;  
    return;  
}
```

```
console.log('ciphertext decrypted');
```

```

        plaintextbytes=new Uint8Array(plaintextbytes);

        var blob=new Blob([plaintextbytes], {type: 'application/download'});
        var blobUrl=URL.createObjectURL(blob);
        aDecsavefile.href=blobUrl;
        aDecsavefile.download=objFile.name + '.dec';

        spnDecstatus.classList.add("greenspan");
        spnDecstatus.innerHTML='<p>File decrypted.</p>';
        aDecsavefile.hidden=false;
    }

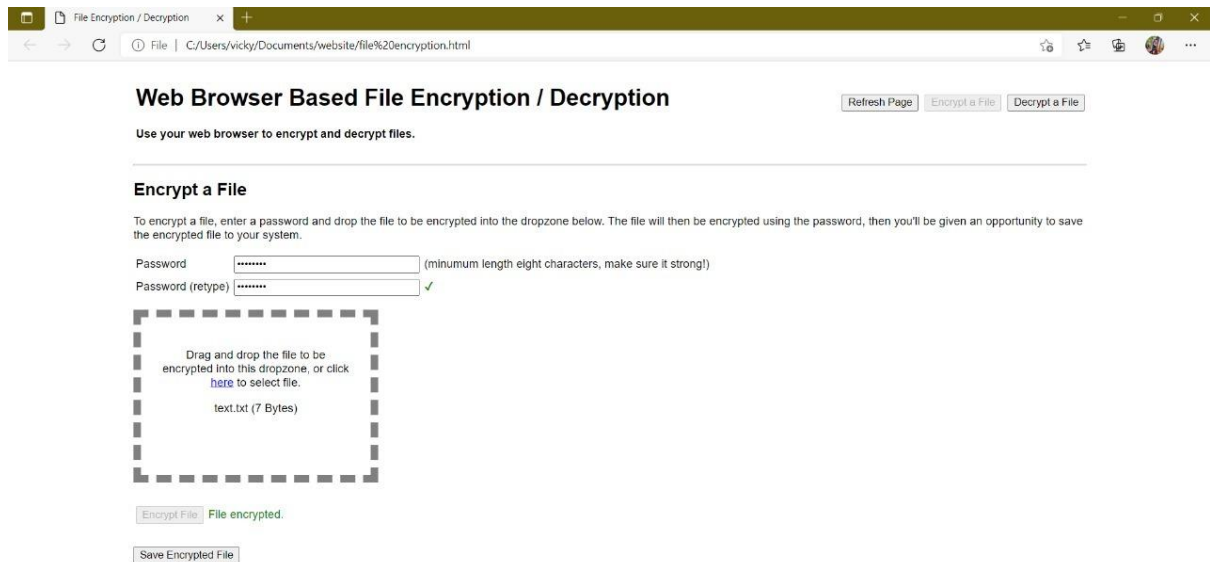
</script>

```

Overview of the Web Application

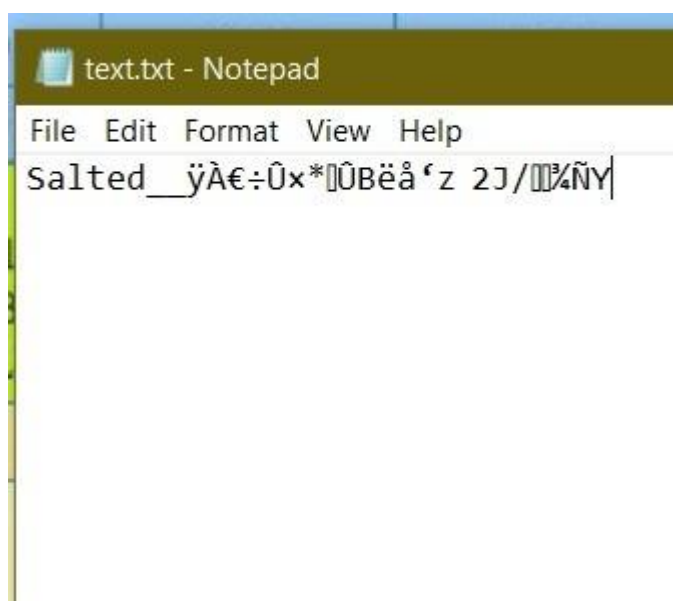


Text File Encryption process



The screenshot shows a web browser window with the title "File Encryption / Decryption". The address bar shows the URL "C:/Users/vicky/Documents/website/file%20encryption.html". The page content includes a title "Web Browser Based File Encryption / Decryption" and a subtitle "Use your web browser to encrypt and decrypt files." There are three buttons: "Refresh Page", "Encrypt a File", and "Decrypt a File". Below the subtitle is a section titled "Encrypt a File" with instructions: "To encrypt a file, enter a password and drop the file to be encrypted into the dropzone below. The file will then be encrypted using the password, then you'll be given an opportunity to save the encrypted file to your system." There are two password input fields: "Password" and "Password (retype)". The "Password" field has a hint "(minimum length eight characters, make sure it strong!)". The "Password (retype)" field has a green checkmark. Below the password fields is a dashed box representing a dropzone. Inside the dropzone, it says "Drag and drop the file to be encrypted into this dropzone, or click [here](#) to select file." and "text.txt (7 Bytes)". Below the dropzone are two buttons: "Encrypt File" and "File encrypted.". Below these buttons is a "Save Encrypted File" button.

Encrypted Text File



Text File Decryption process



Decrypted Text File

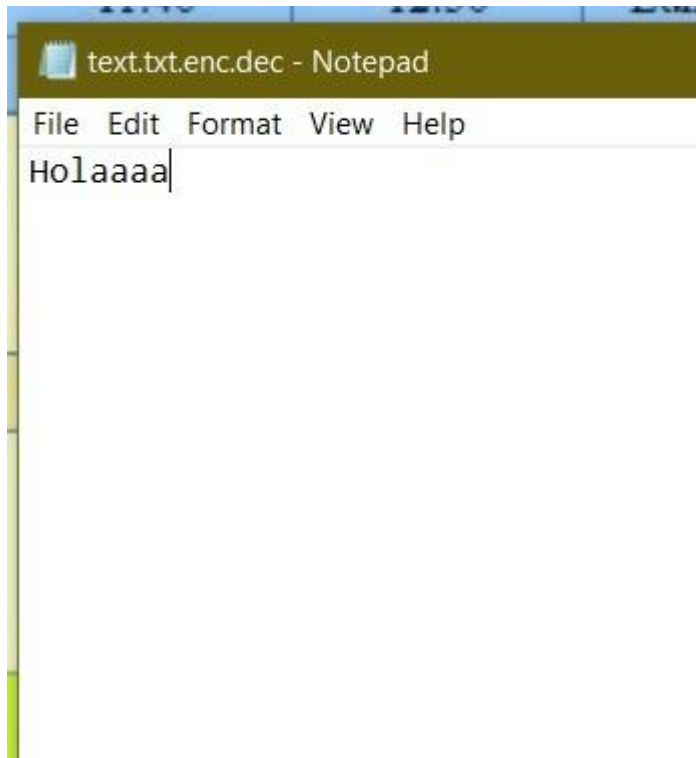


Image Encryption

File Encryption / Decryption

C:/Users/vicky/Documents/website/file%20encryption.html

Refresh PageEncrypt a FileDecrypt a File

Use your web browser to encrypt and decrypt files.

Encrypt a File

To encrypt a file, enter a password and drop the file to be encrypted into the dropzone below. The file will then be encrypted using the password, then you'll be given an opportunity to save the encrypted file to your system.

Password (minimum length eight characters, make sure it's strong!)

Password (retype) ✓

Drag and drop the file to be encrypted into this dropzone, or click [here](#) to select file.

sunset.jpg (537.85 KB)

Encrypt FileFile encrypted.

Save Encrypted File

Encrypted Image

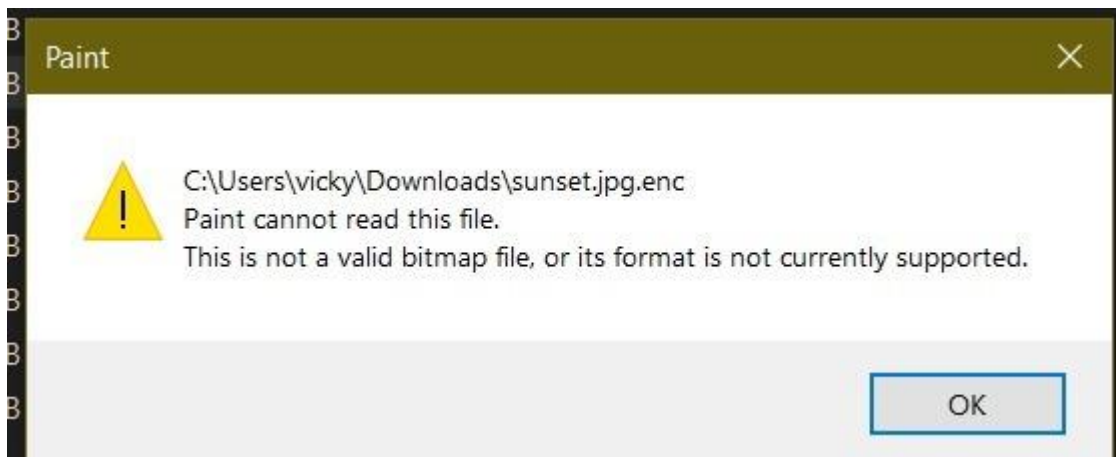
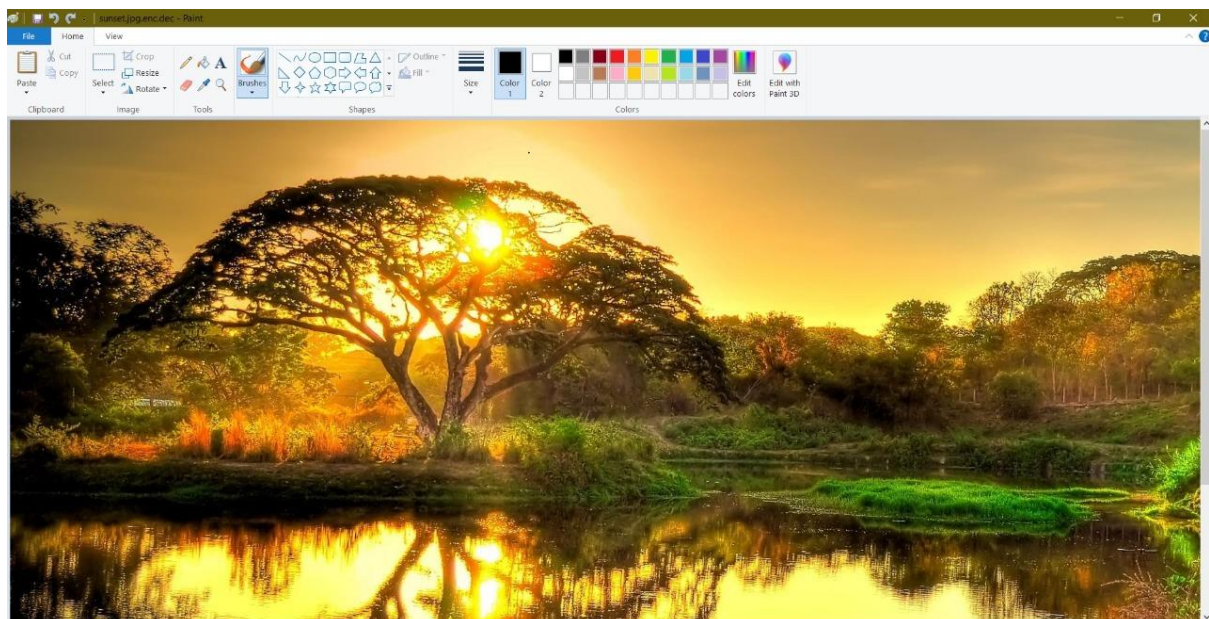


Image Decryption



Decrypted Image



Video Encryption

File Encryption / Decryption

File | C:/Users/vicky/Documents/website/file%20encryption.html

Refresh Page Encrypt a File Decrypt a File

Use your web browser to encrypt and decrypt files.

Encrypt a File

To encrypt a file, enter a password and drop the file to be encrypted into the dropzone below. The file will then be encrypted using the password, then you'll be given an opportunity to save the encrypted file to your system.

Password (minumum length eight characters, make sure it strong!)

Password (retype) ✓

Drag and drop the file to be encrypted into this dropzone, or click [here](#) to select file.

WhatsApp Video 2021-12-01 at 11.45.29.mp4 (228.52 KB)

Encrypt File File encrypted.

Save Encrypted File

Video Decryption

File Encryption / Decryption

File | C:/Users/vicky/Documents/website/file%20encryption.html

Refresh Page Encrypt a File Decrypt a File

Use your web browser to encrypt and decrypt files.

Decrypt a File

Decrypt a file using the password that was previously used to encrypt the file. After the file is decrypted, you'll be given an opportunity to save the decrypted file to your system.

Password

Drag and drop file to be decrypted into this dropzone, or click [here](#) to select file.

WhatsApp Video 2021-12-01 at 11.45.29.mp4.enc (228.55 KB)

Decrypt File File decrypted.

Save Decrypted File

Decrypted Video

WhatsApp Video 2021-12-01 at 11.45.29.mp4.enc.dec - VLC media player
Media Playback Audio Video Subtitle Tools View Help



CONCLUSION:

Protecting data from attacks is difficult. One way to secure data from attacks is to use encryption, One of them by using the AES encryption method. Based on the results of the implementation that has been carried out that the application of the AES (Advanced Encryption Standard) algorithm for information security both web-based text and image files runs well. Web-based applications are able and can help users to encrypt and decrypt text and image files.

REFERENCES:

- [1] K. k. R. Saraf, V. P. Jagtap, and A. K. Mishra, "Text and Image Encryption Decryption Using Advanced Encryption Standard," International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 3, Issue 3, pp. 118-126, May – June 2014.
- [2] R. Pakshwar, V. K. Trivedi, and V. Richhariya, "A Survey on Different Image encryption & Decryption Techniques," International Journal of Computer Science and Information Technology, vol. 4, no. 1, pp. 113-116, 2013.
- [3] Dr. P. Mahajan, A. Sachdeva, "A Study of Encryption Algorithms AES, DES and RSA for Security", Global Journal of Computer Science and Technology Network, Web & Security (GJCSTNWS), vo. 13 Issue. 15 Version 1.0 Year 2013.
- [4] L. Singh, Dr. R.K. Bharti, "Comparative performance analysis of Cryptographic Algorithms," International Journal of Advanced Research and Computer Science and Software Engineering (IJARCSSE), vol. 3, issue. 11, November 2013.
- [5] W. Stallings, "Network Security Essentials: Applications and Standards," Prentice Hall, 2007.
- [6] S. B. Sasi, D. Dixon, and J. Wilson, "A General Comparison of Symmetric and Asymmetric Cryptosystems for WSNs and an Overview of Location Based Encryption Technique for Improving Security," IOSR Journal of Engineering (IOSRJEN) www.iosrjen.org ISSN (e): 2250-3021, ISSN (p): 2278-8719, vol. 04, Issue. 03, PP. 01-04, (March. 2014)
- [7] J. Katz, Y. Lindell, "Introduction to Modern Cryptography," 2nd edition, Boca Raton : CRC Press/Taylor & Francis, pages 583, 2015.
- [8] P. K. Das, Mr. P. Kumar and M. Sreenivasulu, "Image Cryptography: A Survey towards its Growth," Advance in Electronic and Electric Engineering, vol. 4, no. 2, pp. 179-184, 2014.
- [9] E. Fathi et al, "Image Encryption: A Communication Perspective," by CRC Press, Reference - 418 Pages - 232 B/W Illustrations, ISBN