UIT2512 - OPERATING SYSTEMS PRACTICES LAB

Cryptographic File Systems

MINI PROJECT

Sruthi R -3122 21 5002 107

Tejaswini R -3122 21 5002 112

Thiruvedhitha S -3122 21 5002 116

Vasundhara B-3122 21 5002 119

Sa Viswavardinii -3122 21 5002 127

AIM :

The aim of this cryptographic file system is to enable users to securely encrypt and decrypt files using web browser-based JavaScript functionalities along with HTML and CSS.

This system allows users to encrypt and decrypt files using a web browser, requiring a password for both encryption and decryption operations.

PROBLEM DESCRIPTION:

The system provides the following functionalities:

File Encryption:

- Allow users to encrypt a selected file using a strong password.
- Use AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode for encryption.
- Derive encryption keys using PBKDF2 (Password-Based Key Derivation Function 2) with a user-provided password and a random salt.
- Enable the download of the encrypted file after encryption.

File Decryption:

- Enable users to decrypt a previously encrypted file using the same password used for encryption.
- Use the password and the salt stored within the encrypted file to derive decryption keys using PBKDF2.
- Utilize AES-CBC decryption to decrypt the file contents.

User Interface:

Offer an intuitive and user-friendly interface to facilitate file selection, password entry, and drag-and-drop file interactions.

Security Considerations:

- Implement strong encryption mechanisms following best practices to ensure data confidentiality.
- Handle errors gracefully and provide clear instructions to users during encryption and decryption processes.

ALGORITHM:

File Encryption:

1.User Inputs:

- Select a file to encrypt.
- Enter a strong password.

2. Generate a Random Salt:

- Generate a random salt.

3. Key Derivation:

- Use PBKDF2 with the user-provided password and the generated salt to derive an encryption key.

4. File Encryption:

- Use AES in CBC mode with the derived key to encrypt the selected file.
- Save the salt along with the encrypted file.

5.Download Encrypted File:

- Provide the option to download the encrypted file.

File Decryption:

1.User Inputs:

- Select the encrypted file.
- Enter the password used for encryption.

2.Extract Salt and Derive Key:

- Extract the salt stored with the encrypted file.
- Use PBKDF2 with the user-provided password and the extracted salt to derive the decryption key.

3. File Decryption:

- Use AES-CBC decryption with the derived key to decrypt the file contents.

User Interface:

1. File Selection:

   - Allow users to select files for encryption or decryption.

2. Password Entry:

   - Provide a secure password entry field.

3. Drag-and-Drop:

   - Support drag-and-drop functionality for file interactions.

Security Considerations:

1. Encryption Mechanism:

   - Use AES in CBC mode for encryption and decryption.

   - Implement PBKDF2 for key derivation.

   - Ensure a strong random salt for each encryption.

2. Error Handling:

   - Gracefully handle errors during encryption and decryption processes.

   - Provide clear instructions to users on how to address errors.

3. Confidentiality:

   - Follow best practices to ensure data confidentiality during both encryption and decryption processes.

CODE:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Web Browser Based File Encryption / Decryption</title>
    </head>
    <style>
```

```
           body {
            font-family: 'Helvetica', 'Arial', 'sans-serif';
            color: white;
            font-size: 15pt;
            /* Add the background image rule here */
            background-image: url('background.jpg');
            background-size: cover;
            background-position: center;
            background-repeat: no-repeat;
            backdrop-filter: blur(5px);
            filter: brightness(0.);
        }
        body, h1, h2, h3, h4, h5, h6, p, a {
    color: #ffffff; /* Set a lighter color for the text */
}

        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;
        }
```

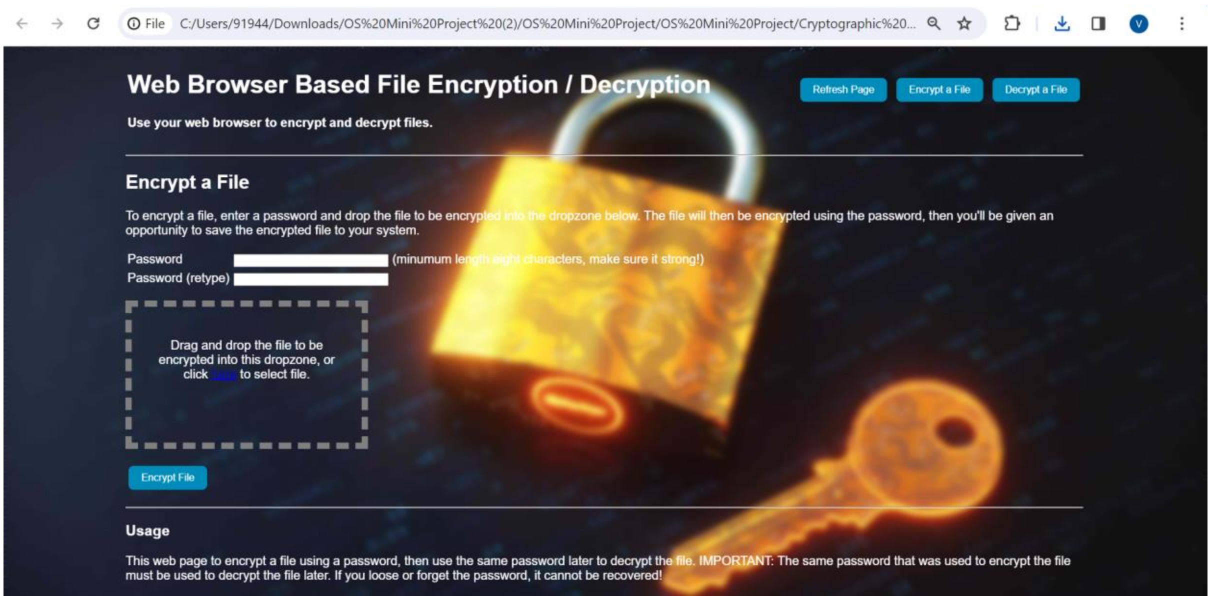
```css
        .divTableRow {
            display: table-row;
        }
        .divTableCell {
            display: table-cell;
            padding: 3px 3px;
        }
        .divTableBody {
            display: table-row-group;
        }

        .greenspan {
            color: green;
        }

        .redspan {
            color: red;
        }
              button {
            background-color: #4CAF50; /* Green background */
            color: white;
            border: none;
            padding: 10px 20px; /* Add some padding to make the buttons
larger */
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin: 4px 2px;
            cursor: pointer;
            border-radius: 8px; /* Rounded corners */
        }

        /* Style for the Refresh Page button */
        #btnRefresh {
            background-color: #008CBA; /* Blue background */
        }

        /* Style for the Encrypt and Decrypt buttons */
        #btnDivEncrypt, #btnDivDecrypt, #btnEncrypt, #btnDecrypt {
            background-color: #008CBA; /* Blue background */
            background-color:#008CBA; /* Red background */
        }

        /* Hover effect for buttons */
        button:hover {
            opacity: 0.8;
        }


        #divEncryptedContent {
            background-color: #f0f0f0;
            padding: 10px;
            border-radius: 8px;
            margin-top: 20px;
            display: none; /* Initially hide the encrypted content div */
        }
    </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=spnCheckretype></span></div>
                    </div>
                </div>
            </div>

            <p> </p>

            <div>
                <div class=dropzone id="encdropzone" 
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>

            <p> </p>

            <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>

            <p> </p>

            <div>
                <a id=aDecsavefile hidden><button>Save Decrypted
File</button></a>
            </div>

            <p> </p>
        </div>



        <div class="container">
            <hr>
            <h3>Usage</h3>
            <p>
            This web page to encrypt a file using a password, then use the
same password later to decrypt the file.  IMPORTANT:  The same password
that was used to encrypt the file must be used to decrypt the file later.
If you loose or forget the password, it cannot be recovered!
            </p>

        </div>
        <BR>
    </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; }
    }
    //drag and drop functions:
    //https://developer.mozilla.org/en-
US/docs/Web/API/HTML_Drag_and_Drop_API/File_drag_and_drop
    function drop_handler(ev) {
        console.log("Drop");
        ev.preventDefault();
        // If dropped items aren't files, reject them
        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=file[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= 'enc_'+objFile.name;

        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='dec_'+objFile.name ;

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

</script>
```

OUTPUT:


File
C:/Users/91944/Downloads/OS%20Mini%20Project%20(2)/OS%20Mini%20Project/OS%20Mini%20Project/Cryptographic%20...
Refresh Page
Encrypt a File
Decrypt a File
Encrypt File

Web Browser Based File Encryption / Decryption

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.

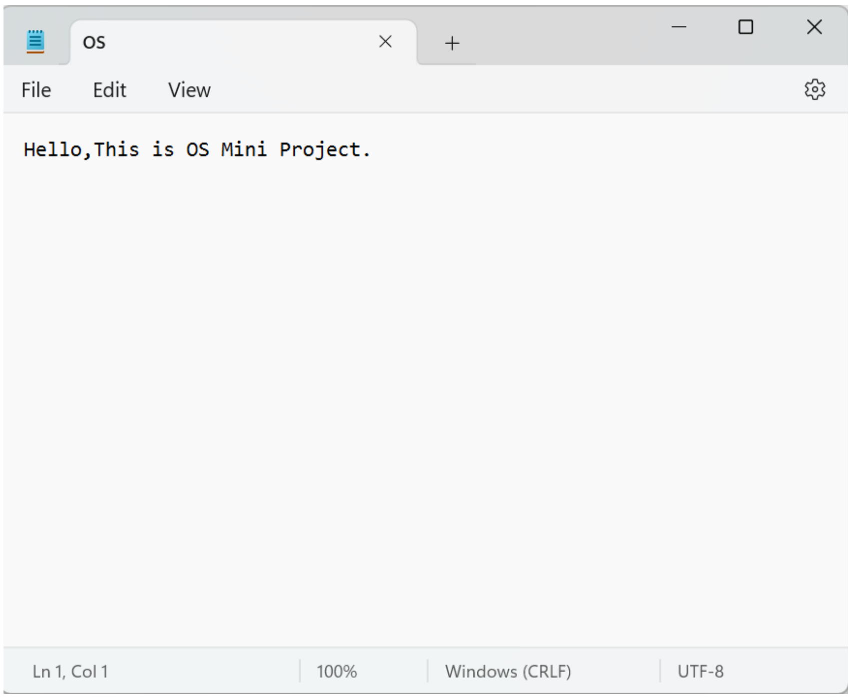
Password (minumum length eight characters, make sure it strong!)
Password (retype)

Drag and drop the file to be encrypted into this dropzone, or click here to select file.

Usage

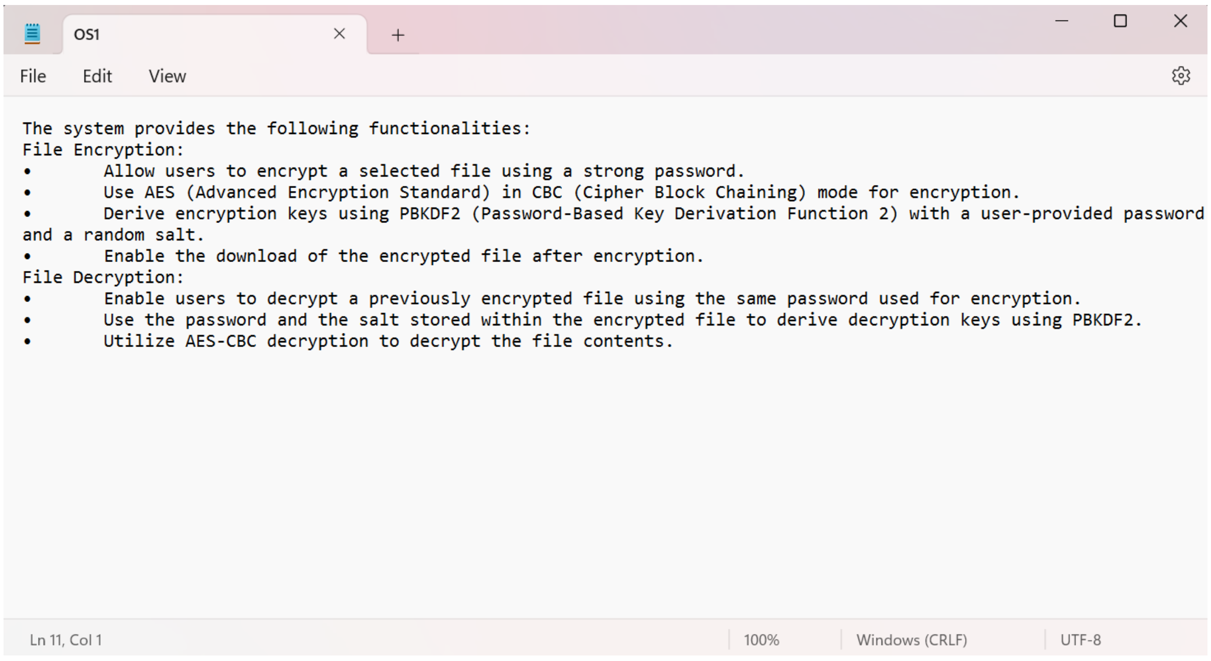
This web page to encrypt a file using a password, then use the same password later to decrypt the file. IMPORTANT: The same password that was used to encrypt the file must be used to decrypt the file later. If you loose or forget the password, it cannot be recovered!

Sample Input 1:



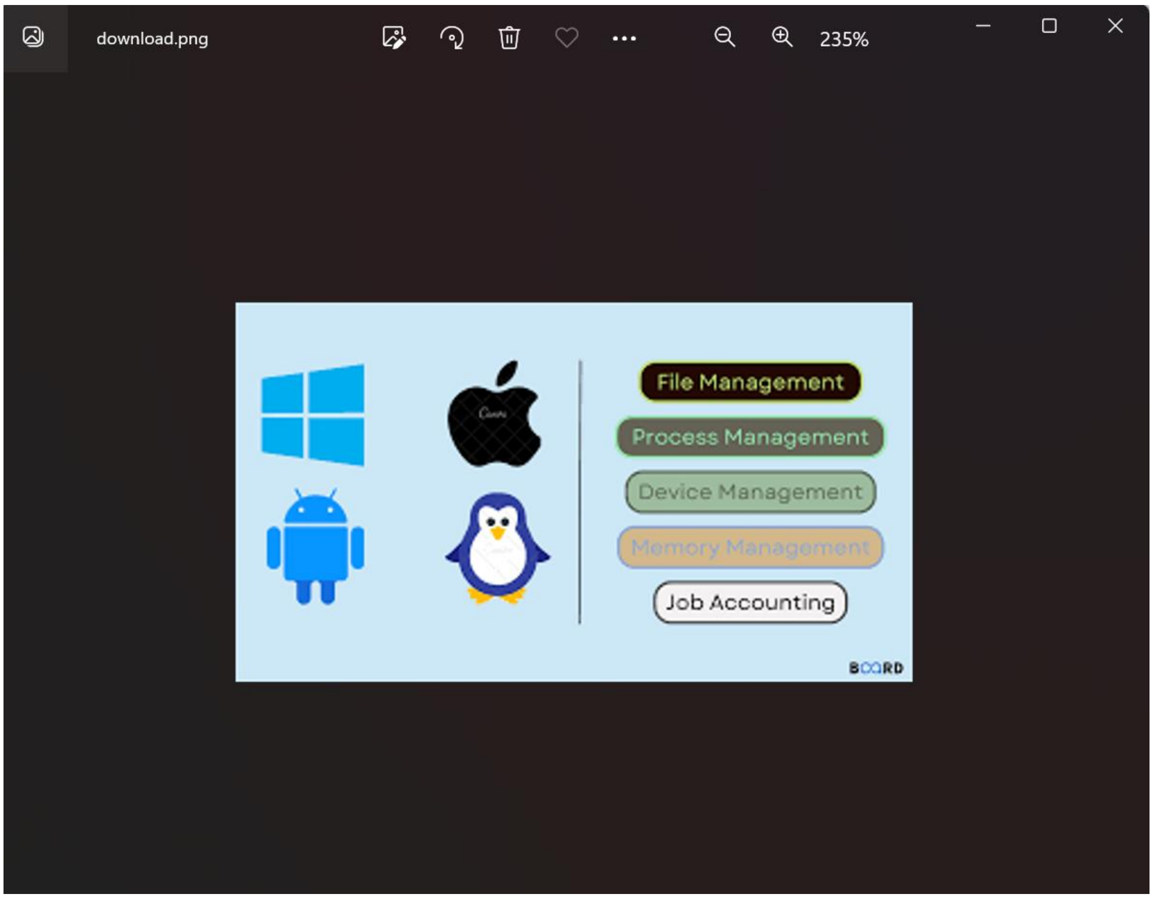
```
Hello,This is OS Mini Project.
```

Sample Input 2:

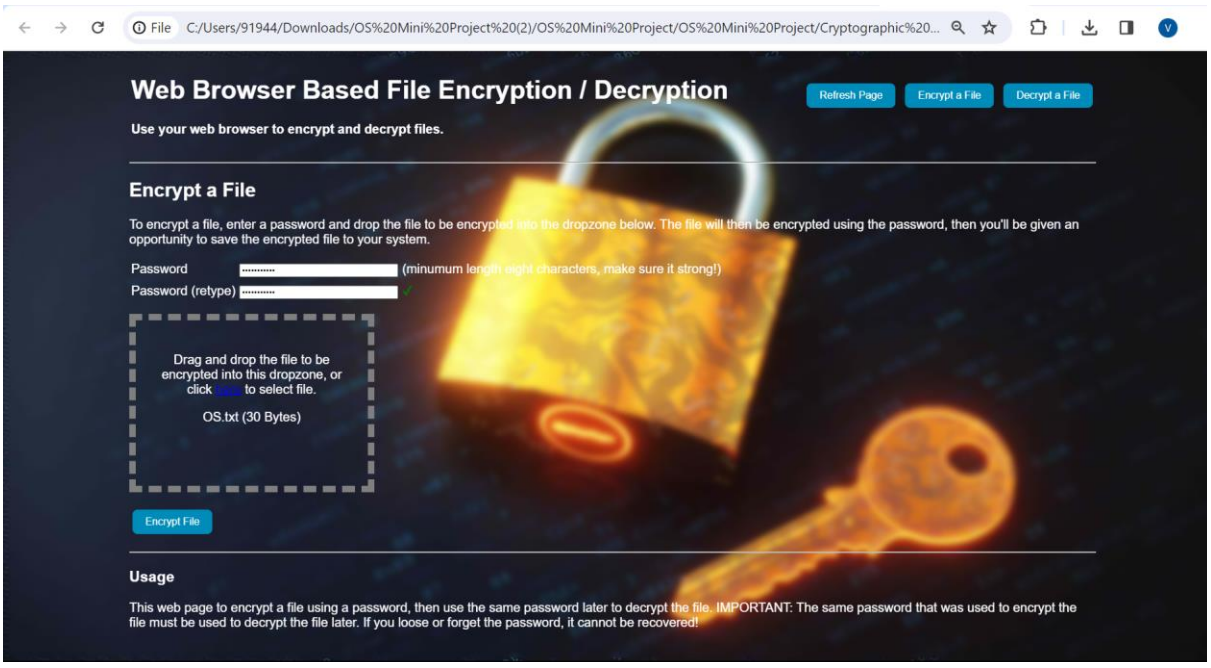


```
The system provides the following functionalities:
File Encryption:
•       Allow users to encrypt a selected file using a strong password.
•       Use AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode for encryption.
•       Derive encryption keys using PBKDF2 (Password-Based Key Derivation Function 2) with a user-provided password
and a random salt.
•       Enable the download of the encrypted file after encryption.
File Decryption:
•       Enable users to decrypt a previously encrypted file using the same password used for encryption.
•       Use the password and the salt stored within the encrypted file to derive decryption keys using PBKDF2.
•       Utilize AES-CBC decryption to decrypt the file contents.
```

Sample Input 3:



ENCRYPTION:


File C:/Users/91944/Downloads/OS%20Mini%20Project%20(2)/OS%20Mini%20Project/OS%20Mini%20Project/Cryptographic%20...
Refresh Page
Encrypt a File
Decrypt a File
Encrypt File

Web Browser Based File Encryption / Decryption

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) ·········· ✓

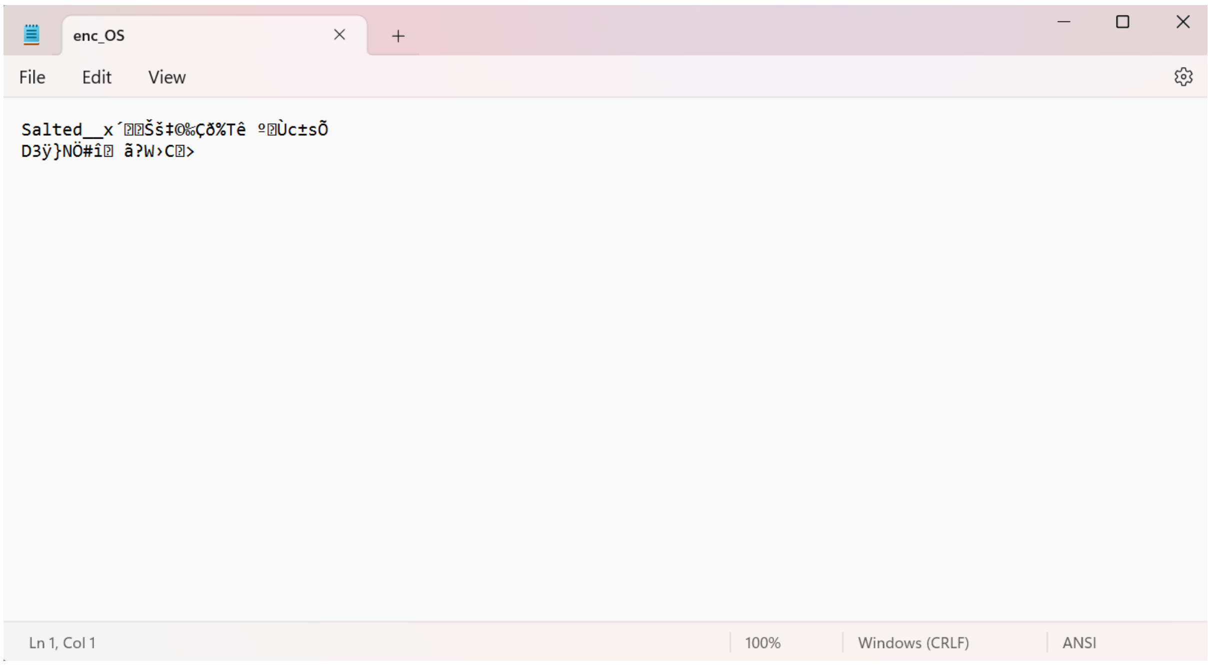
Drag and drop the file to be encrypted into this dropzone, or click here to select file.

OS.txt (30 Bytes)

Usage

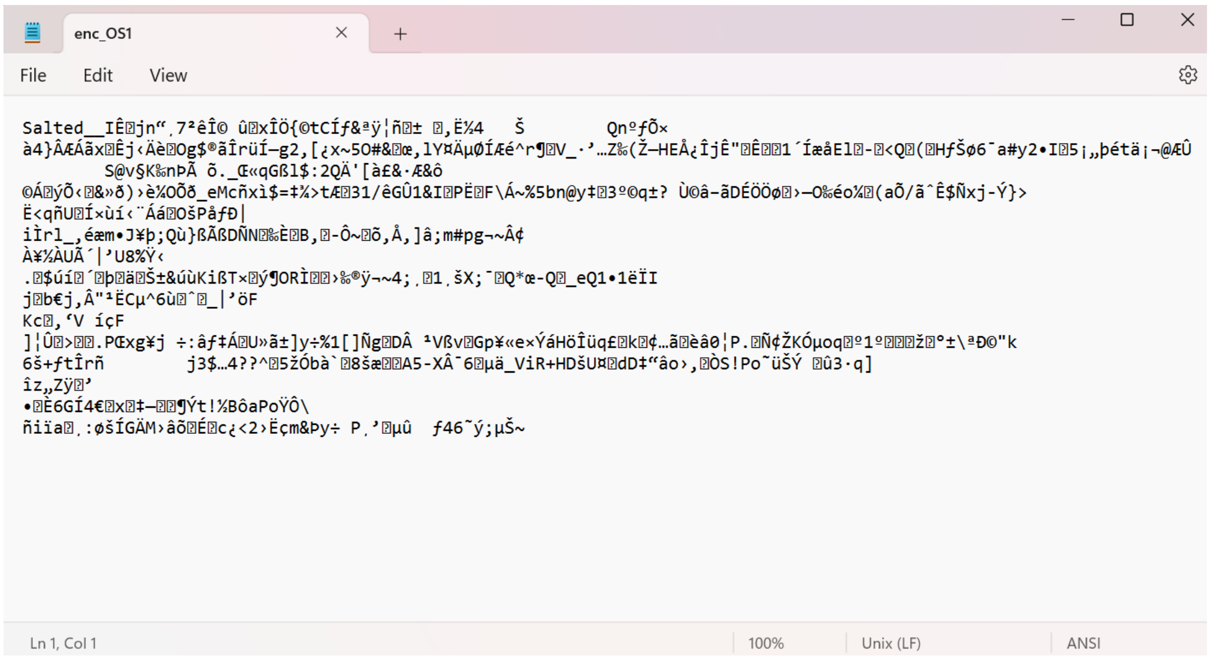
This web page to encrypt a file using a password, then use the same password later to decrypt the file. IMPORTANT: The same password that was used to encrypt the file must be used to decrypt the file later. If you loose or forget the password, it cannot be recovered!

Sample Output 1(Encryption):



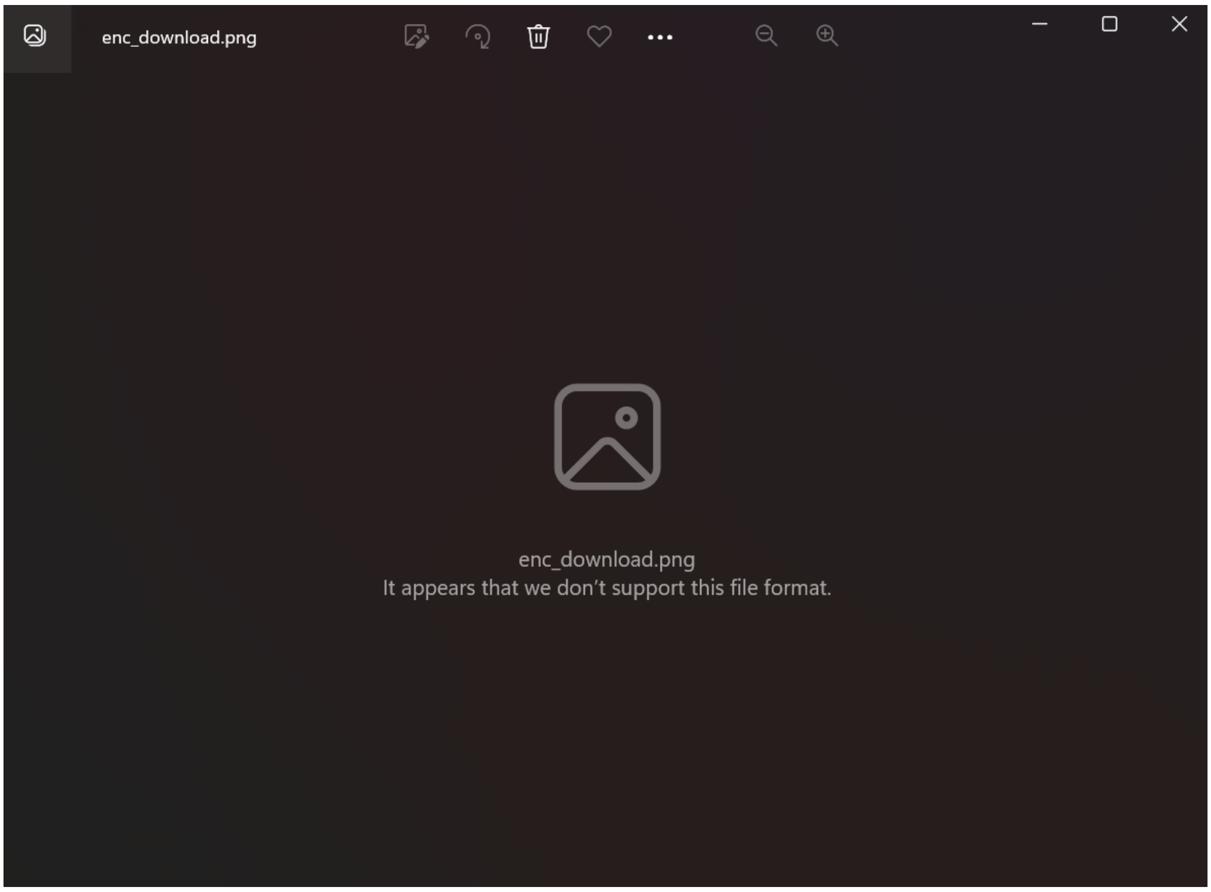
```
Salted__x´⍰⍰Šš‡©‰Çð%Tê º⍰Ùc±sÕ
D3ÿ}NÖ#î⍰ ã?W›C⍰>
```

Sample Output 2(Encryption):



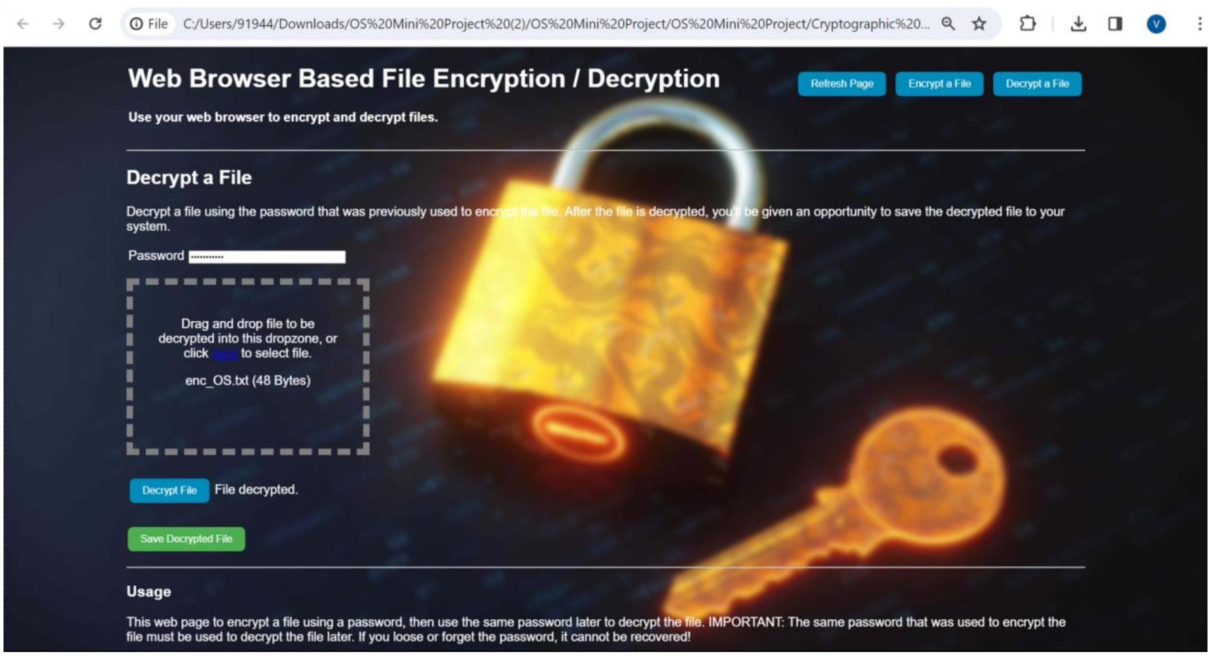
```
Salted__IÊ⍰jn“¸7²êÎ© û⍰xÎÖ{©tCÍƒ&ªÿ¦ñ⍰± ⍰,Ë¼4   Š         QnºƒÕ×
à4}ÂÆÁãx⍰Êj‹Äè⍰Og$®ãÎrüÍ–g2,[¿x~5O#&⍰œ,1Y¤ÄµØÍÆé^r¶⍰V_·’…Z‰(Ž–HEÅ¿ÎjÊ"⍰Ê⍰⍰1´ÍæåEl⍰-⍰<Q⍰(⍰HƒŠø6¯a#y2•I⍰5¡„þétä¡¬@ÆÛ
           S@v§K‰nÞÃ õ._Œ«qGßl$:2QÄ'[à£&·Æ&ô
©Á⍰ýÕ‹⍰&»ð)›è¼OÕð_eMcñxì$=‡¾>tÆ⍰31/êGÛ1&I⍰PË⍰F\Á~%5bn@y‡⍰3º©q±? Ù©â–ãDÉÖÖø⍰›—O‰éo¼⍰(aÕ/ã^Ê$Ñxj-Ý}>
Ë<qñU⍰Í×ùí‹¨Áá⍰OšPåƒÐ|
iÌrl_,éæm•J¥þ;Qù}ßÃßDÑN⍰‰È⍰B,⍰-Ô~⍰õ,Å,]â;m#pg¬~Â¢
À¥½ÀUÃ´|’U8%Ÿ‹
.⍰$úí⍰´⍰þ⍰ä⍰Š±&úùKißT×⍰ý¶ORÌ⍰⍰›‰®ÿ¬~4;¸⍰1¸šX;¯⍰Q*œ-Q⍰_eQ1•1ëÏI
j⍰b€j,Â"¹ËCµ^6ù⍰ˆ⍰_|’öF
Kc⍰,‘V íçF
]¦Û⍰>⍰⍰.PŒxg¥j ÷:âƒ‡Á⍰U»ã±]y÷%1[]Ñg⍰DÂ ¹Vßv⍰Gp¥«e×ÝáHöÎüq£⍰k⍰¢…ã⍰èâ0¦P.⍰Ñ¢ŽKÓµoq⍰º1º⍰⍰⍰ž⍰°±\ªÐ©"k
6š+ƒtÎrñ          j3$…4??^⍰5žÓbà`⍰8šæ⍰⍰A5-XÂ¯6⍰µä_ViR+HDšU¤⍰dD‡“âo›,⍰ÒS!Po˜üŠÝ ⍰û3·q]
îz„Zÿ⍰’
•⍰È6GÍ4€⍰x⍰‡–⍰⍰¶Ýt!½BôaPoŸÔ\
ñiïa⍰¸:øšÍGÄM›âõ⍰É⍰c¿<2›Ëçm&Þy÷ P¸’⍰µû  ƒ46˜ý;µŠ~
```

Sample Output 3(Encryption):



enc_download.png
It appears that we don't support this file format.

DECRYPTION:


C:/Users/91944/Downloads/OS%20Mini%20Project%20(2)/OS%20Mini%20Project/OS%20Mini%20Project/Cryptographic%20...
Refresh Page
Encrypt a File
Decrypt a File

Web Browser Based File Encryption / Decryption

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.

enc_OS.txt (48 Bytes)

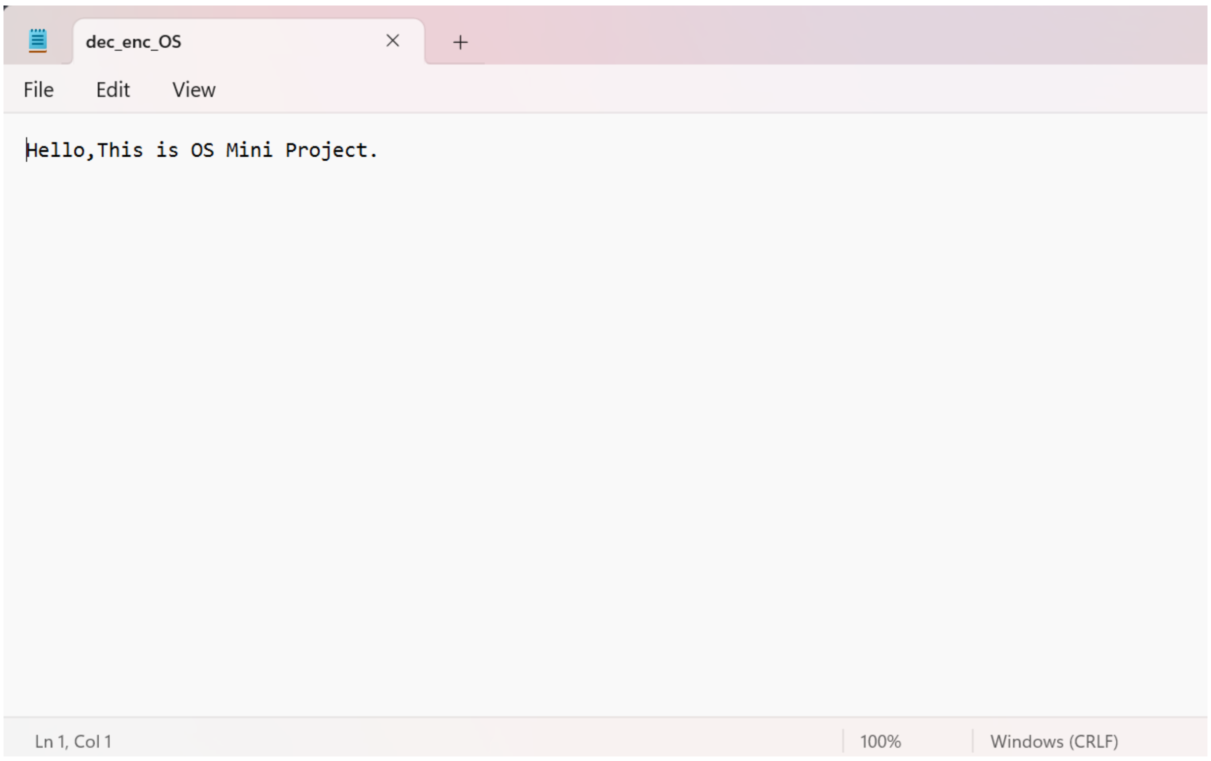
Decrypt File File decrypted.

Save Decrypted File

Usage

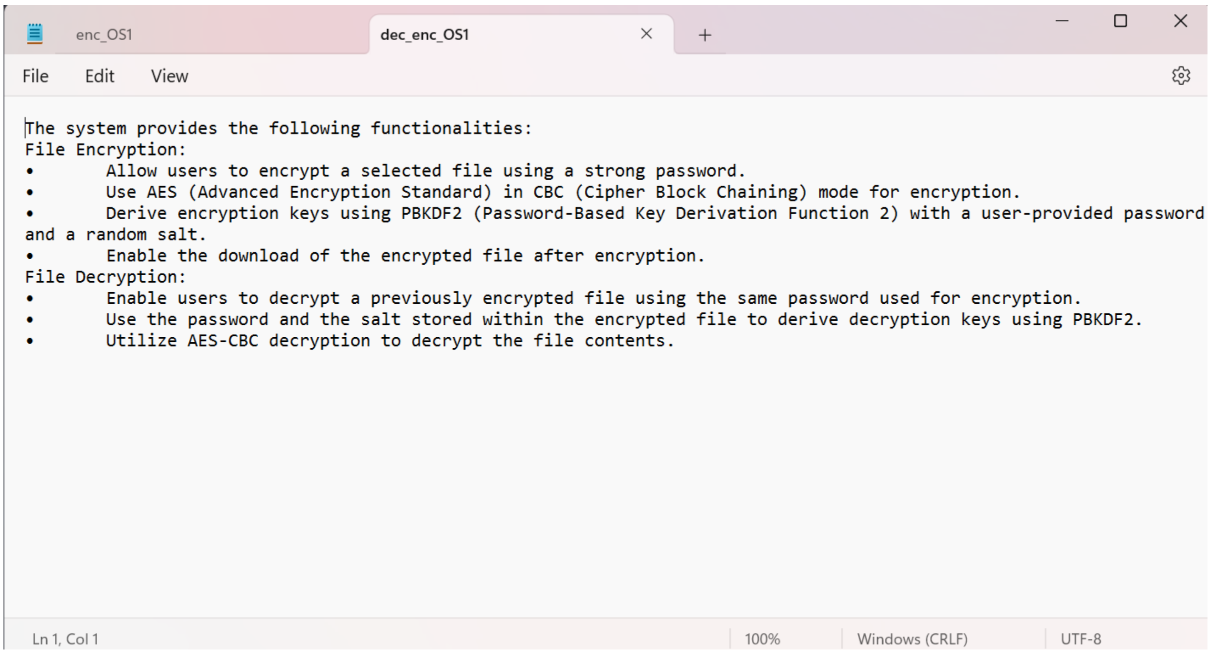
This web page to encrypt a file using a password, then use the same password later to decrypt the file. IMPORTANT: The same password that was used to encrypt the file must be used to decrypt the file later. If you loose or forget the password, it cannot be recovered!

Sample Output 1(Decryption):



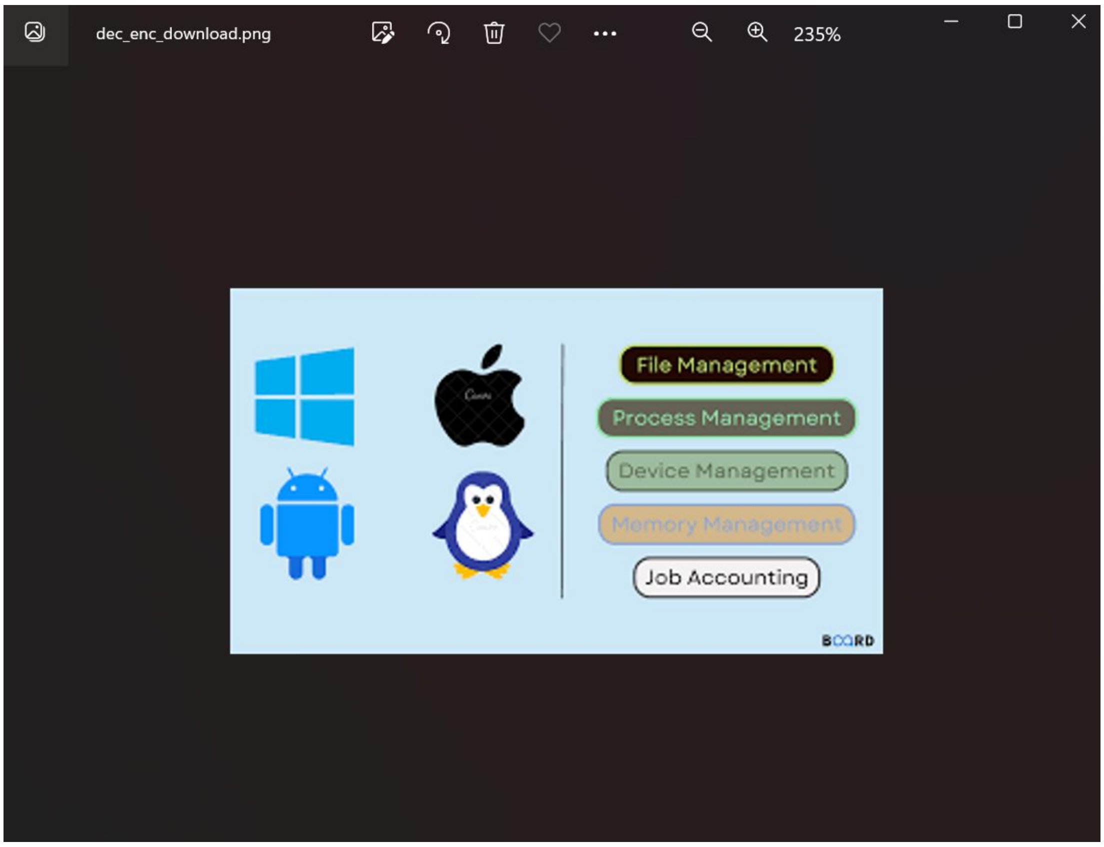
```
Hello,This is OS Mini Project.
```

Sample Output 2(Decryption):



```
The system provides the following functionalities:
File Encryption:
•       Allow users to encrypt a selected file using a strong password.
•       Use AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode for encryption.
•       Derive encryption keys using PBKDF2 (Password-Based Key Derivation Function 2) with a user-provided password
and a random salt.
•       Enable the download of the encrypted file after encryption.
File Decryption:
•       Enable users to decrypt a previously encrypted file using the same password used for encryption.
•       Use the password and the salt stored within the encrypted file to derive decryption keys using PBKDF2.
•       Utilize AES-CBC decryption to decrypt the file contents.
```

Sample Output 3(Decryption):



RESULT:

Thus we have successfully completed the implementation of cryptographic file systems.