

UIT2512---Operating Systems Practices Lab

Cryptographic File Systems – Mini Project

Team Members:

Sruthi R
Tejaswini R
Thiruvadhitha S
Vasundhara B
Viswavardinii S

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.

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: 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;
    }
  </style>
</html>
```

```
.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>
    </body>
</html>
```

```

        </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=txtEncpassphaseretype
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(pbkd2salt, 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 pbkd2iterations=10000;
        var passphrasebytes=new TextEncoder("utf-
8").encode(txtDecpassphrase.value);
        var pbkd2salt=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;
        }

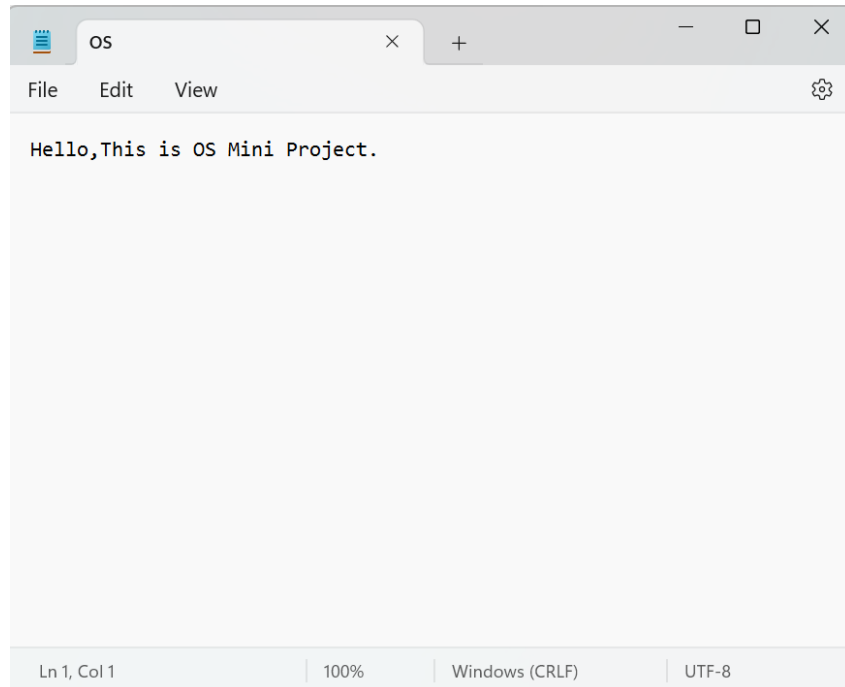
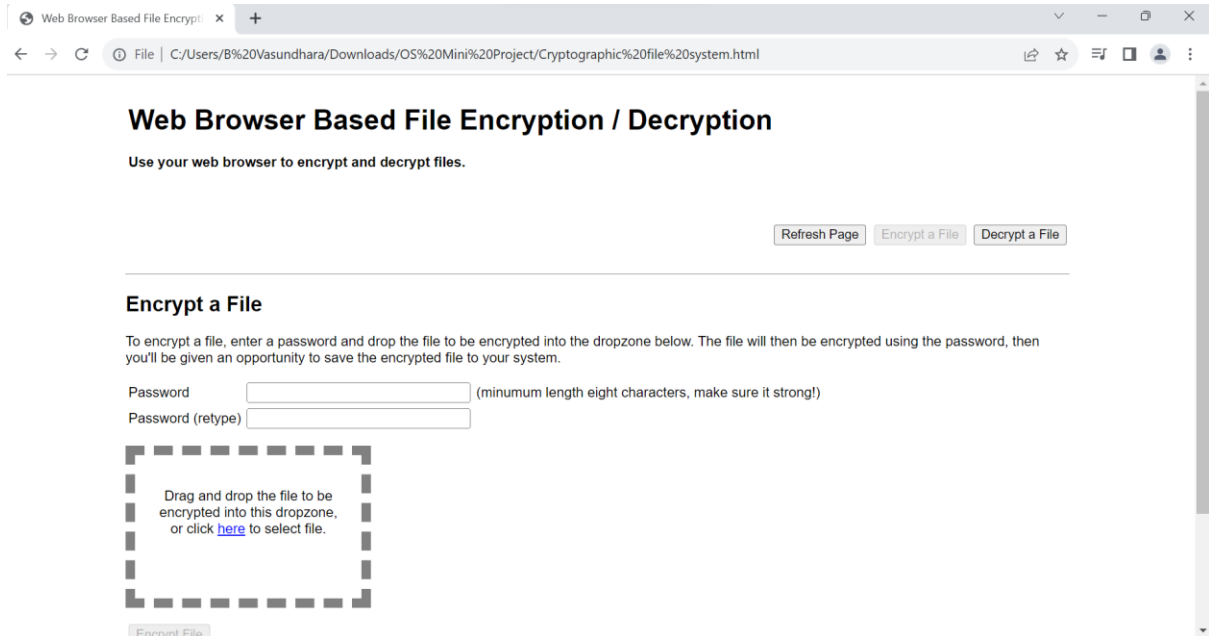
        console.log('ciphertext decrypted');
        plaintextbytes=new Uint8Array(plaintextbytes);

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

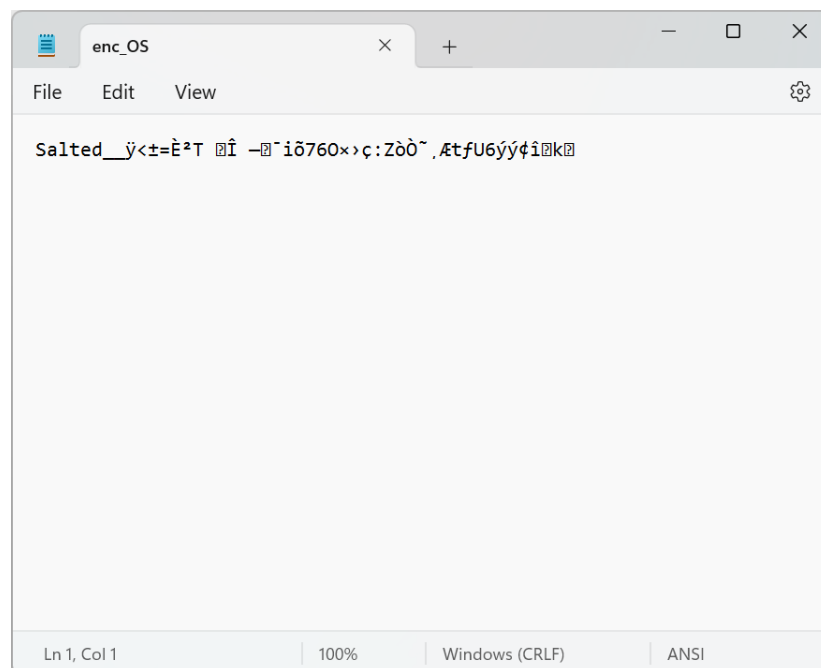
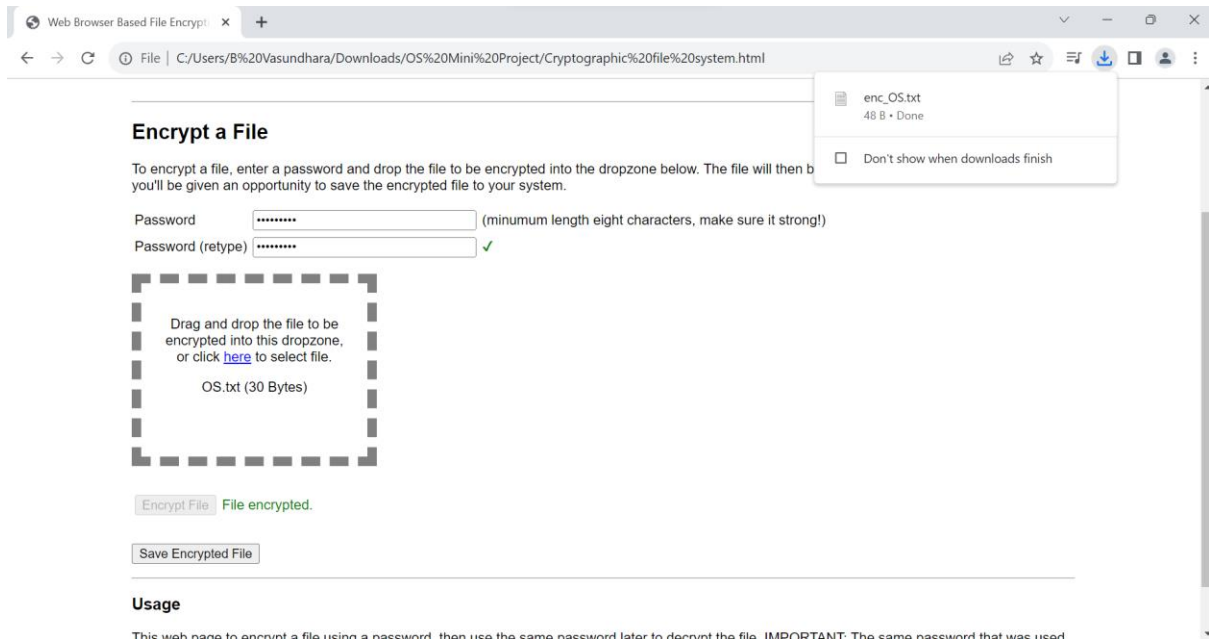
        spnDecstatus.classList.add("greenspan");
        spnDecstatus.innerHTML='<p>File decrypted.</p>';
        aDecsavefile.hidden=false;
    }
</script>

```

OUTPUT:



ENCRYPTION:



DECRYPTION:

