SECURITY BASED MICRO PROJECT REPORT ON

**WAVEFORM ENCRYPTION AND AUDIO SECURITY**

***A Report Submitted to the Partial Fulfillment of the Requirements for the Degree Of***

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

Submitted by,

**ROBERT THOMAS**

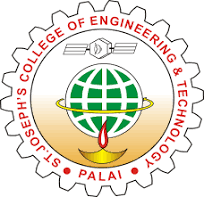
**SJC23CC062**

Under the guidance of

Prof. Sabarinath G Pillai, Assistant Professor <Roll no. 1-23>

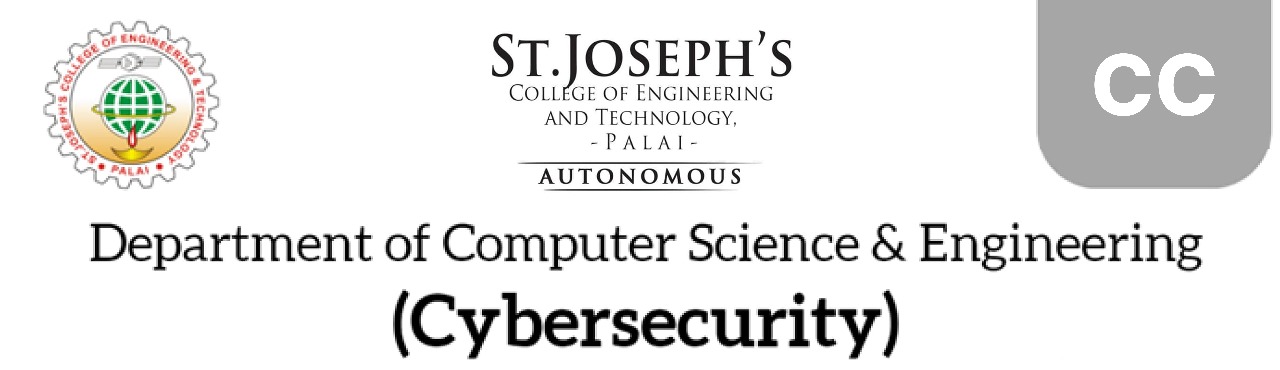
Prof. Revathy A S, Assistant Professor <Roll no. 24-47>

Prof. Devina Vinod, Assistant Professor <Roll no. 48-72>



Approved by AICTE, Affiliated to APJAKTU,  
Bharananganam, Palai, Choondacherry, Kerala – 686579

**MAY 2025**



**Certificate**



*This is to certify that the Micro project report entitled* ***‘WAVE ENCRYPTION AND AUDIO SECURITY’*** *is a Bonafide record of the Micro project done by* ***Mr. Robert Thomas(Register No: SJC23CC062)*** *of Department of Computer Science & Engineering(****Cybersecurity),*** *St. Joseph’s College of Engineering and Technology, Palai, in partial fulfilment of the requirements for the award of Degree of Bachelor of Technology in Computer Science & Engineering(****Cybersecurity****) as part of* ***CCL 202 Scripting Languages for Security*** *from APJ Abdul Kalam Technological University, Kerala, under our supervision and guidance*.

**Prof. Sabarinath G. Pillai Prof. Devina Vinod Prof. Revathy A S**

Asst. Professor, CC Asst Professor, CC Dept Asst Professor, CC

Project Guide

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Name** | | **Page Number** |
|  |  |  | |

**1 Abstract 1**

**2 Introduction 2**

**3 Theoretical Study 3**

**4 Source Code 5**

**5 Observation And Output 8**

**6 Conclusion 10**

**7 Reference 11**

**WAVEFORM ENCRYPTION AND AUDIO SECURITY**

**Project Description:**

This project implements a secure audio cryptography system that encrypts and decrypts audio files using symmetric encryption. Utilizing the cryptography library's Fernet method, it ensures the confidentiality and integrity of audio data. The project supports WAV filehandling, allowing seamless encryption and decryption processes.

**Key Features:**

**Symmetric Encryption:** Single key for both encryption and decryption.File Handling: Supports WAV format for easy audio manipulation.

**Key Management:** Generates and stores a secure encryption key.

**Technologies Used:**

* **Python**
* **Cryptography Library**

**Keywords:**

Audio Cryptography, Symmetric Encryption, Data Security, Python, Cryptography, AudioFile Handling, Key Management, WAV Format.

**INTRODUCTION TO AUDIO CRYPTOGRAPHY**

Audio cryptography is a fascinating intersection of signal processing and cryptography, focusing on the security and privacy of audio data. In a world where digital communications and multimedia content are essential, securing sensitive audio information—whether in voice conversations, recordings, or other audio formats—has become increasingly important. Audio cryptography explores methods for protecting this data from unauthorized access or tampering, ensuring that only authorized parties can decode and interpret it.

At its core, audio cryptography involves applying cryptographic techniques to audio signals, allowing for secure transmission, storage, and retrieval of information. While traditional cryptography typically focuses on text or binary data, audio cryptography addresses the unique challenges posed by audio data, which are continuous, high-dimensional, and perceptible to human ears. This requires specialized encryption and encoding methods that can transform audio into an unintelligible form, ensuring confidentiality while preserving the quality and integrity of the original sound.

The most common cryptographic techniques used in audio cryptography are symmetric and asymmetric encryption algorithms, digital watermarking, and steganography. Symmetric encryption relies on the use of a single key for both encryption and decryption, while asymmetric encryption uses a pair of public and private keys. Digital watermarking embeds hidden data within an audio file, enabling the tracking of ownership or detecting tampering. Steganography, on the other hand, hides messages within the audio data itself, making the presence of the message undetectable to the human ear.

Audio cryptography also includes the challenge of minimizing the degradation of audio quality when applying encryption algorithms. Unlike text, which can be easily encrypted without perceptible loss of information, audio encryption often results in some degree of distortion or degradation due to the complexity of maintaining intelligible sound. Advanced algorithms aim to strike a balance between encryption strength and the preservation of audio quality, making sure that the encrypted audio remains usable and audibly acceptable to the listener.

**THEORETICAL STUDY ON FILE ENCRYPTION AND DECRYPTION**

### Principles of Encryption and Decryption:

Encryption is the process of converting plain text or files into an unreadable format using an algorithm and a key. Only an authorized user with the correct key can decrypt and access the original data. Decryption is the reverse process, restoring the encrypted content back to its original form.

The encryption process enhances data security by preventing unauthorized access, ensuring that sensitive files remain confidential even if intercepted. Modern cryptographic algorithms, such as AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), and symmetric key encryption, are commonly used for secure file encryption.

### Methodology Used in the Code:

The provided code implements a web-based encryption and decryption system using JavaScript and backend support for file processing. The methodology includes the following steps:

#### a. User Interface (UI)

* A dynamic web interface is provided to allow users to select encryption or decryption operations.
* Forms are used to upload files for encryption and decryption.

#### b. Encryption Process

* The user selects a file to encrypt and uploads it.
* The file is sent to the server using fetch and FormData.
* The server encrypts the file and generates a key for decryption.
* A download link is provided for both the encrypted file and the key.

#### c. Decryption Process

* The user selects both the encrypted file and the corresponding key file.
* The files are sent to the server for processing.
* The server decrypts the file using the provided key.
* A link to download the decrypted file is made available.

### Security Considerations

The encryption and decryption system ensures data security, but several factors must be considered:

* **Key Management:** Proper storage and handling of encryption keys are crucial. Losing the key results in the inability to decrypt the file.
* **Algorithm Strength:** The strength of encryption depends on the algorithm used. Modern standards like AES-256 provide high levels of security.
* **Transmission Security:** Using HTTPS ensures secure transmission of files between the client and server.
* **Error Handling:** Implementing robust error handling prevents vulnerabilities and enhances user experience.

### Real-World Applications

Encryption and decryption techniques are widely used in various domains:

* **Data Protection:** Secure storage of sensitive data such as financial records and personal information.
* **Communication Security:** End-to-end encryption in messaging apps to prevent eavesdropping.
* **Cloud Security:** Protecting files stored in cloud storage services from unauthorized access.
* **Legal and Compliance Requirements:** Ensuring compliance with regulations such as GDPR and HIPAA for data security.

**SOURCE CODE**

from flask import Flask, render\_template, request, send\_from\_directory, jsonify

import os

from cryptography.fernet import Fernet

app = Flask(\_\_name\_\_)

UPLOAD\_FOLDER = "uploads"

ENCRYPTED\_FOLDER = "encrypted"

DECRYPTED\_FOLDER = "decrypted"

KEY\_FOLDER = "keys"

for folder in [UPLOAD\_FOLDER, ENCRYPTED\_FOLDER, DECRYPTED\_FOLDER, KEY\_FOLDER]:

    os.makedirs(folder, exist\_ok=True)

@app.route("/")

def index():

    return render\_template("index.html")

@app.route("/encrypt", methods=["POST"])

def encrypt():

    file = request.files["file"]

    filepath = os.path.join(UPLOAD\_FOLDER, file.filename)

    file.save(filepath)

    key = Fernet.generate\_key()

    key\_filename = "key.key"

    key\_path = os.path.join(KEY\_FOLDER, key\_filename)

    with open(key\_path, "wb") as key\_file:

        key\_file.write(key)

    fernet = Fernet(key)

    with open(filepath, "rb") as f:

        encrypted\_data = fernet.encrypt(f.read())

    encrypted\_filename = f"encrypted\_{file.filename}"

    encrypted\_path = os.path.join(ENCRYPTED\_FOLDER, encrypted\_filename)

    with open(encrypted\_path, "wb") as enc\_file:

        enc\_file.write(encrypted\_data)

    return jsonify({

        "encrypted\_url": f"/download/encrypted/{encrypted\_filename}",

        "key\_url": f"/download/keys/{key\_filename}"

    })

@app.route("/decrypt", methods=["POST"])

def decrypt():

    file = request.files["file"]

    key\_file = request.files["key"]

    encrypted\_filepath = os.path.join(UPLOAD\_FOLDER, file.filename)

    key\_filepath = os.path.join(KEY\_FOLDER, key\_file.filename)

    file.save(encrypted\_filepath)

    key\_file.save(key\_filepath)

    with open(key\_filepath, "rb") as kf:

        key = kf.read()

    fernet = Fernet(key)

    with open(encrypted\_filepath, "rb") as enc\_file:

        decrypted\_data = fernet.decrypt(enc\_file.read())

    decrypted\_filename = f"decrypted\_{file.filename}"

    decrypted\_path = os.path.join(DECRYPTED\_FOLDER, decrypted\_filename)

    with open(decrypted\_path, "wb") as dec\_file:

        dec\_file.write(decrypted\_data)

    return jsonify({"decrypted\_url": f"/download/decrypted/{decrypted\_filename}"})

@app.route("/download/<folder>/<filename>")

def download(folder, filename):

    """Serve files from the correct folders."""

    if folder == "encrypted":

        return send\_from\_directory(ENCRYPTED\_FOLDER, filename, as\_attachment=True)

    elif folder == "keys":

        return send\_from\_directory(KEY\_FOLDER, filename, as\_attachment=True)

    elif folder == "decrypted":

        return send\_from\_directory(DECRYPTED\_FOLDER, filename, as\_attachment=True)

    return "File not found", 404

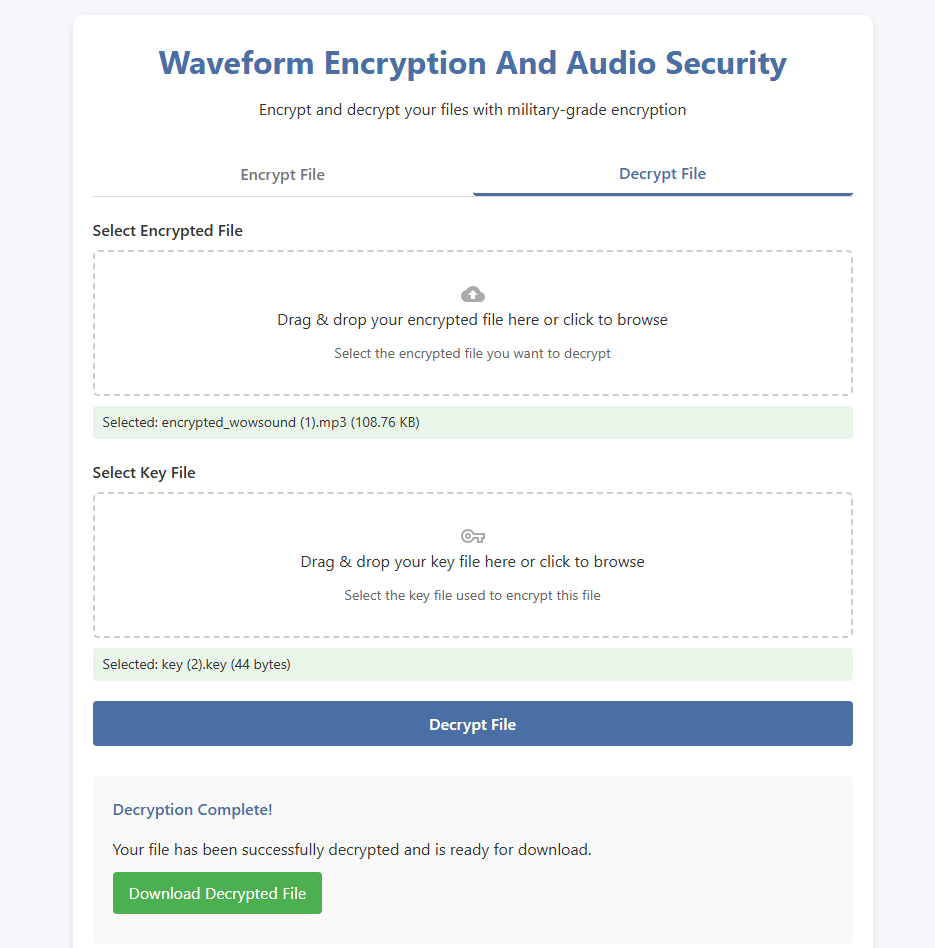
if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

**OBSERVATIONS AND OUTPUT**



Webapp deployment of the project by default selects the option Encryption.The user is allowed to upload the file which they want to securely encrypt. After choosing the file to encrypt ,pressing the ‘Encrypt file’ button will generate the encrypted file and the key used for encryption for the user to download and save with themselves.



Decryption part of the webapp have the functionality to uploadany or the previously encrypted file and the key file used for encryption. Further clicking on the ‘Decrypt File’ button will decript the encrypted file using the specific key the user have given.The Decripted original file is now available to be downloaded.

## CONCLUSION

In this project, we successfully developed a web-based file encryption and decryption system that ensures secure file transfer and storage. The implementation utilizes a simple yet effective approach to encrypt and decrypt files using a key-based method, allowing users to protect sensitive information efficiently.

Through the use of HTML, CSS, JavaScript, and a backend server, we created an intuitive user interface that enables seamless encryption and decryption operations. The project highlights the importance of data security in digital communication and demonstrates how cryptographic techniques can be integrated into web applications to enhance security.

Overall, this project serves as a foundational implementation of encryption, which can be expanded in the future with stronger encryption algorithms, user authentication mechanisms, and cloud storage integration. By continuing to refine and improve the system, we can further enhance data protection and privacy in digital environments.

**REFERENCE**

1. <https://cyberpedia.reasonlabs.com/EN/audio%20encryption.html>
2. <https://ieeexplore.ieee.org/document/8365185>
3. <https://www.ijcit.com/archives/volume5/issue5/Paper050509.pdf>
4. Research and Extra work done with: Chatgpt,cloudai.