PROBLEM STATEMENT 15:

# PROTECTING USER PASSWORD KEYS AT REST (ON THE DISK)

# TABLE OF CONTENTS



# 1.Introduction

The project aims to develop a secure authorization application that protects user password keys at rest on disk. By leveraging advanced encryption techniques and secure storage methods, the application ensures that user data remains confidential and secure.

# 2. PROBLEM STATEMENT

**The project focuses on creating an application that encrypts user-selected files or directories using AES-256 encryption. The generated encryption key (File Encryption Key) must be securely stored and protected by a user passphrase. The challenge is to ensure that neither the user passphrase nor the encryption key is stored in plain text.**

# 3. Application Workflow

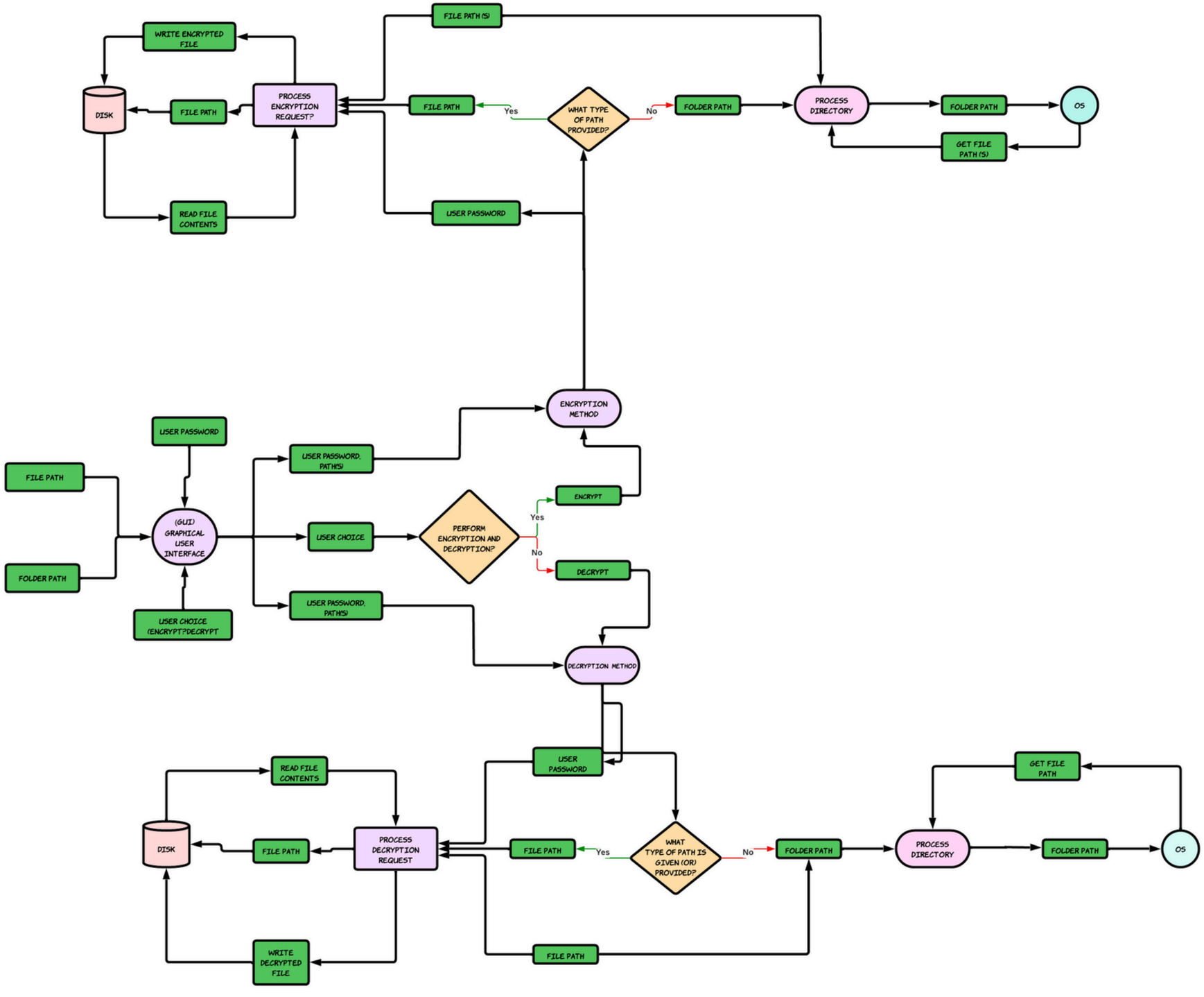
The application workflow is designed to ensure a seamless yet highly secure process for encrypting and decrypting user files. Here’s a detailed look at the steps involved:

## User Authentication:

The user initiates the application and is prompted to enter a secure passphrase. This passphrase will be used as a key component in encrypting the File Encryption Key.
The application employs a Key Derivation Function (KDF) to convert the user passphrase into a cryptographic key.

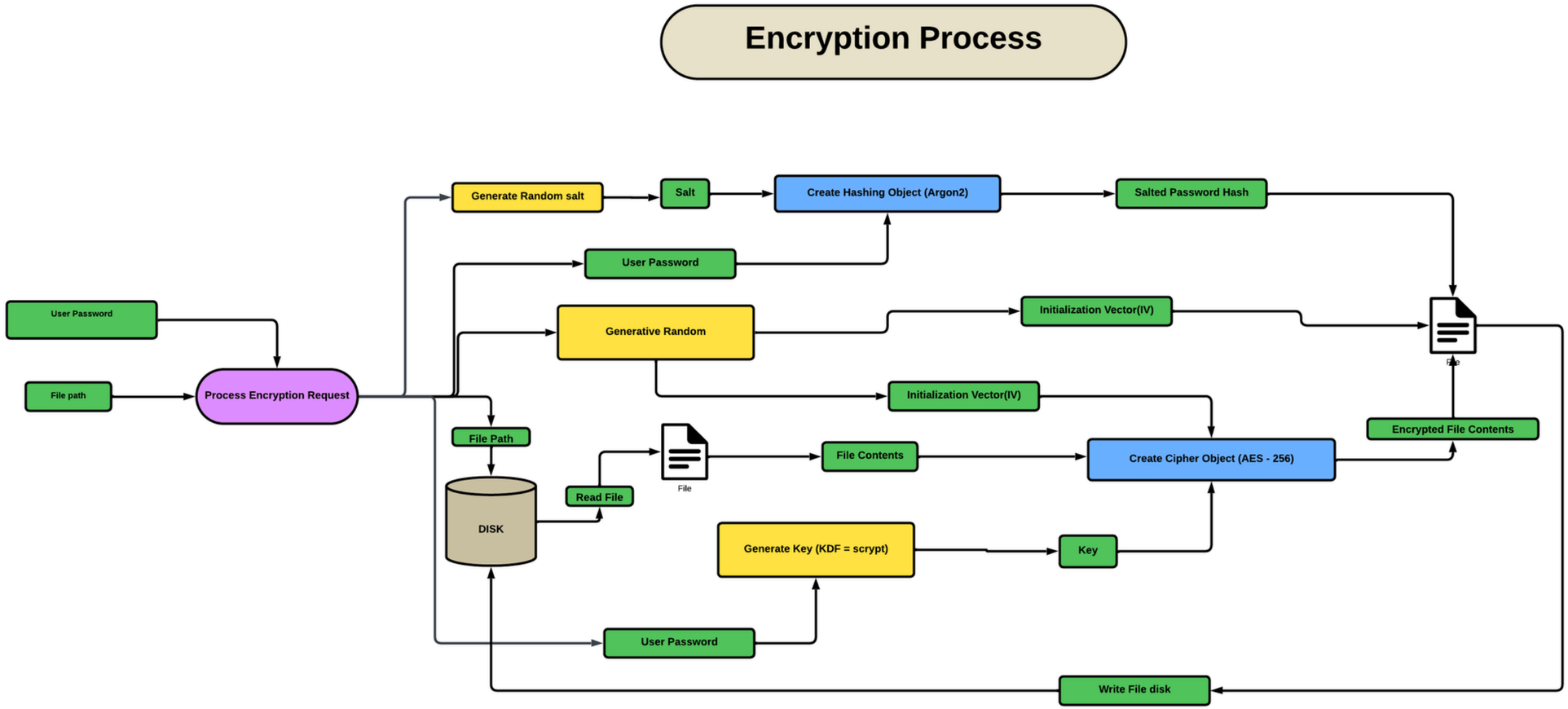
## File/Director Selection:

The user selects the file or directory they wish to encrypt from the file system.
The application allows browsing through directories and choosing multiple files if needed


WRITE ENCRYPTED FILE
FILE PATH (S)
DISK
FILE PATH
PROCESS ENCRYPTION REQUEST?
FILE PATH
Yes
WHAT TYPE OF PATH PROVIDED?
No
FOLDER PATH
PROCESS DIRECTORY
FOLDER PATH
OS
GET FILE PATH (S)
READ FILE CONTENTS
USER PASSWORD
ENCRYPTION METHOD
USER PASSWORD
USER PASSWORD, PATH(S)
FILE PATH
ENCRYPT
(GUI) GRAPHICAL USER INTERFACE
USER CHOICE
PERFORM ENCRYPTION AND DECRYPTION?
Yes
No
FOLDER PATH
DECRYPT
USER CHOICE (ENCRYPT?DECRYPT
USER PASSWORD, PATH(S)
DECRYPTION METHOD
USER PASSWORD
READ FILE CONTENTS
GET FILE PATH
DISK
FILE PATH
PROCESS DECRYPTION REQUEST
FILE PATH
Yes
WHAT TYPE OF PATH IS GIVEN (OR) PROVIDED?
No
FOLDER PATH
PROCESS DIRECTORY
FOLDER PATH
OS
WRITE DECRYPTED FILE
FILE PATH

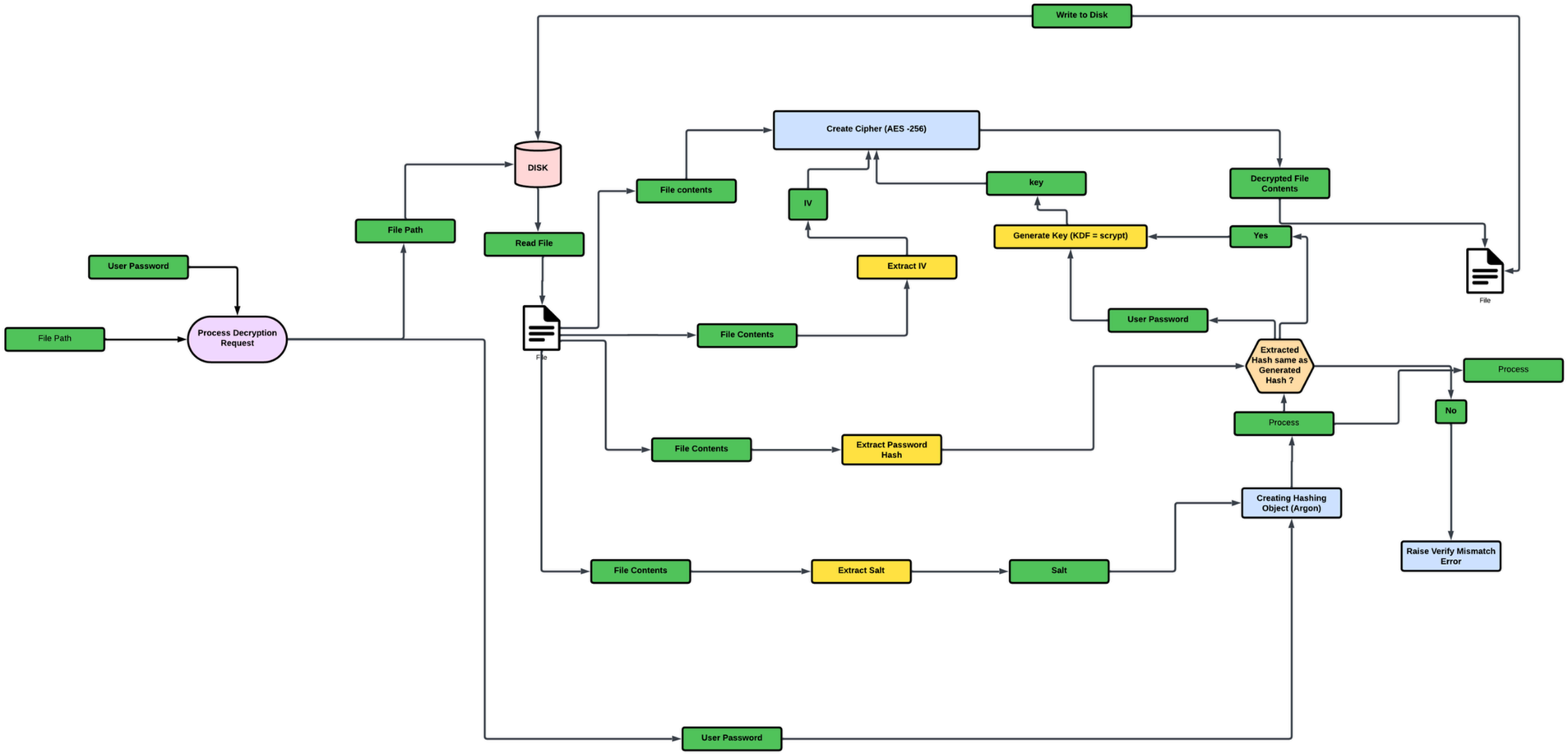
# File Encryption:

**A random File Encryption Key is generated using a secure random number generator.**
**The selected file or directory is encrypted using AES-256 encryption with the generated File Encryption Key.**



# Decryption Process:

**When the user needs to access the encrypted file, they must enter their passphrase.**
**The application uses the KDF to derive the decryption key from the user passphrase.**
**If the passphrase is correct, the File Encryption Key is decrypted.**
**The decrypted File Encryption Key is then used to decrypt the file, making it accessible to the user.**



## Key Storage:

The File Encryption Key is then encrypted using a key derived from the user passphrase via the KDF.
The encrypted File Encryption Key is stored securely in a designated file on the disk.

## Decryption Process:

When the user needs to access the encrypted file, they must enter their passphrase.
The application uses the KDF to derive the decryption key from the user passphrase.
If the passphrase is correct, the File Encryption Key is decrypted.
The decrypted File Encryption Key is then used to decrypt the file, making it accessible to the user.

## Integrity Checks:

The application includes mechanisms to verify the integrity of the encrypted file and the File Encryption Key.
Any discrepancies detected during the decryption process (e.g., wrong passphrase,
corrupted key file) trigger alerts and prevent unauthorized access.

# 4. HIGH-LEVEL ALGORITHM

## Initialize the System:

Start the application and prompt the user for their passphrase.
Utilize a Key Derivation Function (KDF) to convert the passphrase into a cryptographic key (Passphrase Key).

## File Encryption Key Generation:

Generate a random File Encryption Key using a secure random number generator.
This key will be used for encrypting the user-selected file or directory.

## Encrypt the File:

Apply AES-256 encryption to the selected file or directory using the generated File Encryption Key.
Ensure the encryption process is secure and efficient, maintaining the confidentiality of the file contents.

## Store the Encrypted File Encryption Key:

Encrypt the File Encryption Key with the Passphrase Key derived from the user passphrase.
Save the encrypted File Encryption Key in a designated secure file on the disk.
Decryption Process

## User Authentication:

Prompt the user to enter their passphrase when they wish to decrypt an encrypted file.
Use the KDF to derive the Passphrase Key from the user passphrase.

## Retrieve and Decrypt the File Encryption Key:

Access the stored encrypted File Encryption Key file.
Decrypt the File Encryption Key using the Passphrase Key derived from the user passphrase.

## Decrypt the File:

Use the decrypted File Encryption Key to apply AES-256 decryption to the encrypted file or directory.
Ensure the integrity and confidentiality of the file are maintained during the decryption process.

## Error Handling and Security Checks:

Implement error handling for incorrect passphrases, corrupted key files, and other anomalies.
Perform integrity checks to confirm that the decrypted data matches the original data before encryption.

# 5. Test Plan


## Simple Cases:

**Encrypt and decrypt a single file using a user passphrase.**
**Verify the integrity and content of the decrypted file.**

## Corner Cases:

**Handling incorrect passphrase attempts.**
**Ensuring the system's behavior when the passphrase file is corrupted or missing.**

## 6. Source Code Repository

The complete source code, including appropriate comments and documentation,
is archived in a GitHub repository.
The repository also contains instructions for setting up the environment and running the application.


## 7. Conclusion

The project successfully developed a secure application for protecting user password keys at rest.
By employing robust encryption techniques and secure key management practices,
the application ensures the confidentiality and integrity of user data.
The project provided valuable learning experiences in system software development,
cryptographic algorithms, and secure coding practices.

# THANK YOU


***FROM,***

***TEAM BETA,***

TAMMALI KARTHIK(ktammali@gitam.in)

ALLI GOPI(galli@gitam.in)

MARIA PUNYA(mpunya@gitam.in)

**Faculty Mentor-Dr. Avishek Chakraborty(achakrab3@gitam.edu)**

**Assistant Professor, Department of EECE, GITAM University, Bengaluru**