Modular Asymmetric Encryption Project

Introduction

This project demonstrates **asymmetric key encryption** using Python's open-source cryptography library. It follows a **modular architecture**, meaning different files handle different responsibilities like key generation, encryption/decryption, and running the program.

Step 1:Understanding the Concept

Asymmetric encryption uses **two keys**: -

Public Key : Used for encryption (shared with everyone) -

Private Key: Used for decryption (kept secret) This ensures data confidentiality and authenticity.

Common algorithms include RSA and ECC. In this project, we use **RSA** (Rivest–Shamir–Adleman) for its simplicity and wide use.

Step 2: Required Library

We use the **cryptography** library in Python which provides secure implementations of cryptographic algorithms. You can install it with: **pip install cryptography.**

Step 3: Modular Structure

The project follows this modular layout: **asymmetric_encryption:**

1.key_manager.py - Generates & loads RSA keys
2.core.py - Handles encryption, decryption, verification
3.main.py - User interface (run this file)

Step 4: How It Works

1. **Key Manager:** generates a 2048-bit RSA key pair and stores them as **private.pem** and **public.pem**.
2. **Encryption:** converts plaintext into ciphertext using the public key.
3. **Decryption:** uses the private key to recover the original text.
4. **Verification:** ensures data integrity by testing decryption success.

Author:
By: Pragna Jyoti Mandal (Roll No: 25BDA083)
DSAI Department, IIIT Dharwad
Project: Modular Asymmetric Key Encryption System (AES-GCM & RSA Hybrid)