Practical Cryptography

Ramy Naiem

Table of Content

AES Encryption Method

- AES Key Generation: 'aes_key = os.urandom(32)'
- AES Encryption Function: 'encrypt_aes(data)' uses PKCS7 padding and CBC mode.
- AES Decryption Function: 'decrypt_aes(encrypted_data)' reverses the encryption process.

RSA Encryption

- RSA Key Generation:
  'private_key = rsa.generate_private_key(...)
- Public Key Derivation:
  'public_key = private_key.public_key()'
- Functions 'encrypt_data(public_key, data)' and 'decrypt_data(private_key, encrypted_data)' for RSA encryption and decryption.

Fernet for Secure File Handling

- Fernet Key Generation: 'key = Fernet.generate_key()'
- Cipher Suite Creation: 'cipher_suite = Fernet(key)'
- Use in File Upload and Download: Encrypting and decrypting file content.

Bcrypt (Password Hashing)

- @app.route('/register', methods=['GET', 'POST'])
- def register():
- if request.method == 'POST': username = request.form['username'] password = request.form['password'].encode('utf-8') password_hash = bcrypt.hashpw(password, bcrypt.gensalt()) # ... [omitted code for saving the user]

Integrating Encryption in the Application

- User Registration: Encrypting user passwords using bcrypt.
- Data Handling: Using AES and RSA for sensitive data encryption.
- File Management: Secure file upload and download with Fernet encryption.

WEBSITE SHOWCASE