Secure File Transfer with RSA and Hashing in Python

Implement a secure file transfer system in Python that utilizes RSA encryption and hashing algorithms to ensure data confidentiality and integrity.

You need to:

- Design and implement a Python program for:
  - Generating RSA key pairs (public and private).
  - Encrypting a file using the recipient's public key.
  - Decrypting the encrypted file using the corresponding private key.
  - Generating a hash of the original file using a secure hashing algorithm (e.g., SHA-256).
  - Verifying the integrity of the received file using the original hash.
- Document the code and design decisions clearly.
- Create a simple user interface for interacting with the system (input and output).
- Test the functionality thoroughly with various test cases.

In the following steps:

1. Key Generation: Implement functions to generate RSA key pairs using the `cryptography` library.
2. File Encryption: Write functions to encrypt a file using the public key of the recipient and the RSA encryption scheme.
3. File Decryption: Create functions to decrypt an encrypted file using the corresponding private key and the RSA decryption scheme.
4. Hashing: Implement functions to generate a hash (e.g., SHA-256) of a file using the `hashlib` library.
5. Integrity Verification: Develop functions to verify the integrity of the received file by comparing its hash with the original hash.
6. User Interface: Create a simple user interface for interacting with the system.
7. Testing: Optionally, write comprehensive unit tests for each functionality using a testing framework like `unittest`.

Project Deliverables:

- Python code for the secure file transfer system.
- Documentation covering:
  - Design overview and key decisions.
  - Testing strategy and coverage.
- A presentation demonstrating the functionality and security aspects of the system (if time permits).

Additional Considerations (Optional):

- Explore different RSA padding schemes and their security implications.
- Investigate the performance trade-offs of different cryptographic algorithms and key sizes.
- Consider integrating the system with a file transfer protocol (e.g., Secure FTP) for real-world applications.
- Discuss the limitations of your implementation and potential areas for improvement.