File Integrity Sender & Receiver Applications

Overview

This project consists of two applications: a **Sender** and a **Receiver**. The Sender application allows you to select a file, generate a hash of its content, sign it using RSA encryption, and send it to the Receiver. The Receiver application verifies the file's integrity using the provided signature and hash, ensuring that the file was not modified during transit.

Technologies Used

- **Flask**: Python web framework used for both the sender and receiver applications.
- **Cryptography (PyCryptodome)**: Used for generating RSA keys, creating digital signatures, and hashing file content.
- **HTML & CSS**: Used for creating the user interface in both applications.
- **Requests (Python library)**: Used in the sender application to send files to the receiver.

Features

- **RSA Encryption**: The sender generates an RSA key pair to sign the file's hash before sending it to the receiver.
- **Digital Signature**: The file is signed with the sender's private key using the PKCS1 v1.5 signature algorithm.
- **SHA-256 Hashing**: The file content is hashed using the SHA-256 algorithm to ensure its integrity.
- **Signature Verification**: The receiver verifies the digital signature using the sender's public key to confirm the file's authenticity.
- **File Integrity Check**: The receiver also compares the calculated hash of the received file with the original hash to ensure that the file has not been modified in transit.

Setup & Installation

Prerequisites

- Python 3.x
- `pip` (Python package installer)

- Install the required Python libraries:

```
pip install flask pycryptodome requests
```

Detailed Process

1. **Sender Application**:
    - The user selects a file and specifies the receiver's URL.
    - The file is read and hashed using the SHA-256 algorithm.
    - The sender signs the hash using its private RSA key (PKCS#1 v1.5).
    - The file content, signature, public key, and hash are sent to the receiver as JSON data.
2. **Receiver Application**:
    - Upon receiving the JSON data, the receiver decodes the file content, signature, and public key.
    - The receiver recalculates the file hash using SHA-256.
    - The signature is verified using the public key. If the signature and hash match, the receiver confirms that the file is authentic and was not modified.
    - The result is displayed in the receiver's UI and returned as a JSON response.

Security Considerations

- **RSA Key Pair**: The sender generates a 2048-bit RSA key pair. The private key is stored locally, while the public key is sent to the receiver for signature verification.
- **Data Transmission**: The file content, hash, and signature are transmitted over HTTP. For production environments, HTTPS should be used to ensure secure data transmission.

Algorithms Used

- **RSA (2048-bit)**: Used for generating the private and public key pairs.
- **SHA-256**: Used for creating a unique hash of the file content.
- **PKCS#1 v1.5**: The digital signature algorithm used to sign the file hash.

Error Handling

- The sender checks for file selection and receiver URL before processing.
- The receiver validates the incoming JSON data and checks for all required fields (file content, signature, public key, hash).
- If any errors occur during transmission or verification, appropriate error messages are displayed.