Secure File Transfer Protocol - Security Architecture

Client A (Sender)

+ RSA Key Pair (Public/Private)
+ HMAC Key
+ Transfer History
+ User Interface

+ Generate File Transfer Request()
+ Generate Symmetric/HMAC Keys()
+ Encrypt Keys with Recipient's Public Key()
+ Split File into Chunks()
+ Encrypt Chunks with AES-CBC()
+ Generate HMAC for Chunks()
+ Send Encrypted Chunks()
+ Sign Messages()

1. Authentication & Public Key Exchange

Server

+ User Directory (with Public Keys)
+ Active Connections
+ Transfer Request Registry

+ Authenticate Users()
+ Distribute Public Keys()
+ Relay Transfer Requests()
+ Forward Encrypted Chunks()
+ Notify Transfer Status()

1. Authentication & Public Key Exchange

Client B (Receiver)

+ RSA Key Pair (Public/Private)
+ HMAC Key
+ Transfer History
+ User Interface

+ Decrypt Symmetric Key with Private Key()
+ Decrypt HMAC Key with Private Key()
+ Verify HMAC of Each Chunk()
+ Decrypt File Chunks with AES-CBC()
+ Reassemble File from Chunks()
+ Store Completed Files()
+ Verify Message Signatures()
+ Update Transfer History()

2. File Transfer Request   **3. Encrypted File Chunks**

Cryptographic Operations

+ RSA-2048: Key Exchange

+ AES-256-CBC: File Encryption

+ HMAC-SHA256: Integrity Verification

+ Secure Random: IV & Nonce Generation

+ Digital Signatures: Message Authentication

+ SHA-256: Hash Functions

File Transfer Protocol

+ FileTransferRequest
+ SecureMessage
+ TransferHistory
+ CryptoUtils

Key Security Features:
- End-to-end encryption
- RSA for key exchange
- AES-CBC for data encryption
- HMAC-SHA256 for integrity
- Chunk-based file transfer
- Unique keys per transfer
- Pause/Resume capability
- Transfer history tracking
- Separate log/data storage
- Anti-replay protection
- Message signatures

4. Storage & Logging

2. Transfer Request   **3. Encrypted File Chunks**

Message Types

+ FileTransferRequest: Initiates transfer

+ SecureMessage: Encrypted data chunks

+ SignedSecureMessage: Authenticated messages

+ UserAuthMessage: Login & authentication

+ TransferStatusMessage: Progress updates

File System

+ /logs: Security and transfer logs
+ /data: Transfer history and metadata
+ /downloads: Received files
+ /keys: Local key storage

Transfer Sequence:
1. User Authentication & Key Exchange
2. Sender initiates transfer request
3. Server relays request to recipient
4. Recipient accepts transfer
5. Sender encrypts file chunks
6. Server relays encrypted chunks
7. Recipient verifies & decrypts chunks
8. Recipient reassembles file
9. Transfer completion notification
10. History updated on both sides

Security Considerations:
1. No perfect forward secrecy
2. Limited user authentication
3. No certificate authority
4. Local key storage risks
5. Limited DoS protection
6. No built-in key backup
7. Relies on system file permissions
8. Network-level security not addressed