Secure File Transfer – Server Side (Milestone Five)

SERVER.CPP Documentation

1 Purpose of the Server Program

The server program is responsible for securely receiving an encrypted file from a client, verifying its integrity, decrypting it, and saving the recovered plaintext to disk.

This program ensures:

- Confidentiality (through AES encryption)
- Integrity (through SHA-256 hashing)
- Correct reconstruction of the original file

2 Included Libraries and Their Purpose

- `#include <iostream>`
  Used for input and output messages (logging server status).
- `#include <openssl/evp.h>`
  `#include <openssl/rand.h>`
  `#include <openssl/aes.h>`
  Used for cryptographic operations:
  - AES-256-CBC decryption
  - SHA-256 hashing
  - OpenSSL EVP interface
- `#include <string>`
  `#include <vector>`
  Used for dynamic memory handling and string manipulation.

- `#include <sys/socket.h>`
  `#include <netinet/in.h>`
  `#include <arpa/inet.h>`
  `#include <unistd.h>`
  Used for low-level TCP socket programming on Unix-based systems.

3 Cryptographic Parameters

```
unsigned char key[32];
unsigned char iv[16];
```

- `key[32]` defines a 256-bit AES key (32 bytes).
- `iv[16]` defines a 128-bit initialization vector required for AES-CBC.

Note: Keys and IVs are hard-coded for demonstration purposes only.

4 Decryption Function Declaration

```
vector<unsigned char> decryptAES(...);
```

This function handles AES-256-CBC decryption of the received ciphertext using OpenSSL.

5 Socket Setup and Server Initialization

```
int server_create = socket(AF_INET, SOCK_STREAM, 0);
sockaddr_in server_addr{};
```

Configuration:

- `AF_INET`: IPv4
- `INADDR_ANY`: Accept connections on all interfaces
- `htons(6767)`: Listen on port 6767

The following calls initialize the server:

- `bind(...)`
- `listen(...)`
- `accept(...)`

6 Receiving the Filename

```
int flen = 0;
recv(client_create, &flen, sizeof(flen), 0);

vector<char> fname(flen+1);
recv(client_create, fname.data(), flen, 0);
```

The server receives:

- The filename length
- The filename itself (null-terminated)

7 Receiving Encrypted File Size

```
long encSize = 0;
recv(client_create, &encSize, sizeof(encSize), 0);
```

This tells the server how many encrypted bytes to expect.

8 Receiving Encrypted File Data

```
vector<unsigned char> cText(encSize);
while (total < encSize) {
    recv(...);
}
```

Since TCP may not deliver all bytes at once, the loop ensures the entire ciphertext is received.

9 Integrity Verification (SHA-256)

```
unsigned char recvDigest[32];
recv(client_create, recvDigest, 32, 0);
```

The server:

1. Receives the SHA-256 digest from the client

2. Computes its own SHA-256 hash using `EVP_Digest(...)`
3. Compares the two using `memcmp(...)`

If the digests do not match, the file is considered corrupted or tampered with.

10 Decryption Process

```
vector<unsigned char> pText = decryptAES(...);
```

The AES-256-CBC decryption uses:

- `EVP_DecryptInit_ex`
- `EVP_DecryptUpdate`
- `EVP_DecryptFinal_ex`

These handle cipher initialization, block processing, and padding removal.

11 Saving the Decrypted File

```
system(”mkdir␣−p␣received”);
fopen(outPath.c_str(), ”wb”);
fwrite(...);
```

The plaintext file is written into a `received/` directory using the original filename.

12 Connection Cleanup

```
close(client_create);
close(server_create);
```

Closes sockets to release system resources.

13 Security Properties Achieved

Confidentiality

File data is encrypted with AES-256 before transmission.

Integrity

SHA-256 hashing ensures data is not altered during transfer.

Correctness

- Decryption occurs only after integrity verification.
- Output file matches the original plaintext.

14 Limitations

- Keys and IVs are hard-coded.
- No authentication or key exchange.
- Intended for educational use only.

15 Conclusion

This server implementation fulfills all Milestone Five requirements by securely receiving, validating, decrypting, and storing encrypted files. It demonstrates correct use of cryptography, networking, and protocol design in a secure file transfer context.