Remote File Storage System

Bakr Alkaanbari
North Central college
Balkanbari@noctrl.edu
Richard Roa
North Central College
rroa@noctlr.rdu

Abstract— Peer-to-peer (P2P) Systems is a way to distribute systems and are independent nodes that are joined together. For this assignment, we needed to find a way to implement a P2P system that acts as a secure storage for files. The main goal of this assignment is to use P2P to store files on a remote system while also making sure that the data is secure while the server does not know what is in the plaintext to ensure that the integrity and confidentiality are safe. The only way to access the file's content will be via client verification to ensure the data is not tampered with.

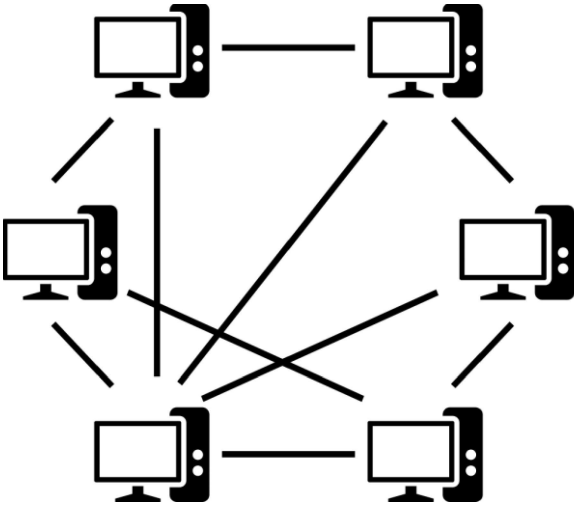
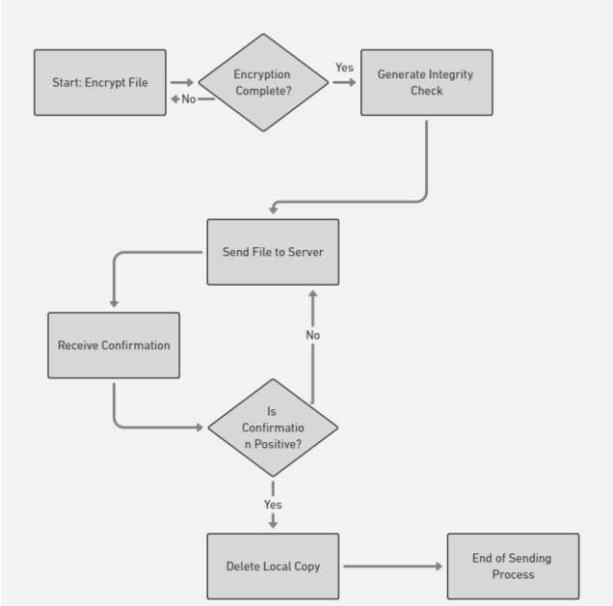


Fig.1 Peer-to peer

I. Introduction

Peer-to-peer is a way in which we can send and distribute information to different clients. In the beginning, we were looking for a way to find the most efficient way to send files to the correct client. We first thought about implementing a linked list to organize and manage information from a client to a server and server to a client. In figure 1, it shows how each client can be a node where they can request information and the server can later transfer the data to the correct client by locating them in the LinkedList. This can also help us since it could be another security layer that will ensure that the data is getting sent to the right person. We then use Pycryptodome.com to help us find an AES scheme to properly encrypt and decrypt the file.

II. Storing files



A. *Encrypt the file*

In order to begin the encryption, process we decided to use AES SHA3-512. We then use the `encrypt()` function to encrypt the file. This function will take the file (as plaintext) and a key. The encryption should ensure confidentiality. You could use symmetric encryption (AES) for the file itself and asymmetric encryption (RSA) to encrypt the symmetric key.

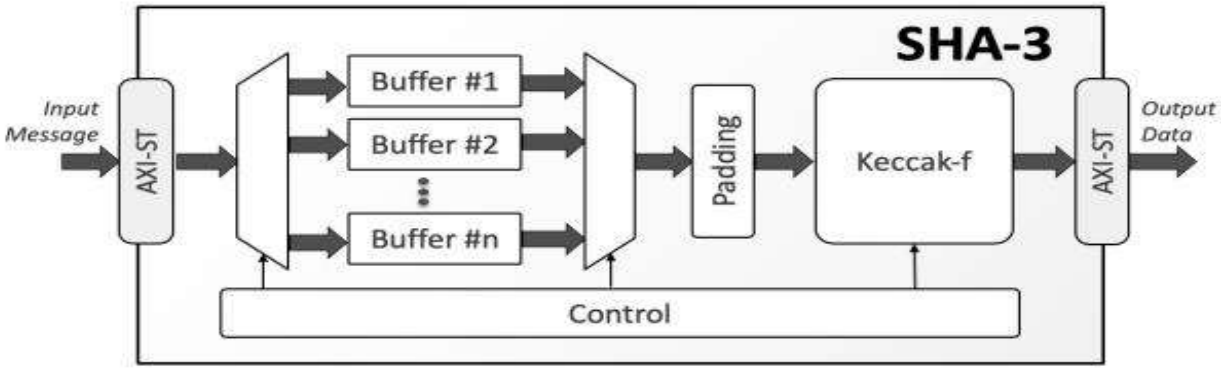


Fig.2 visual representation of SHA-3

B. *Generate integrity check*

In order to verify the file has the correct data before it is sent to the server. We will create a small code that will help us calculate the hash of the encrypted file.

C. *Send files to server*

To send the files to the server we will first need to create and utilize a `send()` function to transmit the encrypted file along with its hash to the server. The function would require having the encrypted file, hash, and recipient's (server's) identifier to ensure its authentication and it gets sent to the right person.

D. *Receive confirmation and delete local copy*

Upon receiving confirmation from the server and the encrypted file having the same hash as the original hash, we then go ahead and delete the file from the user's local storage.

III. Retrieving files

A. *Authentication client-to-server*

The client requests the file from the server. This step involves proving the client's identity, using a digital signature or a pre-shared key if symmetric encryption is used.

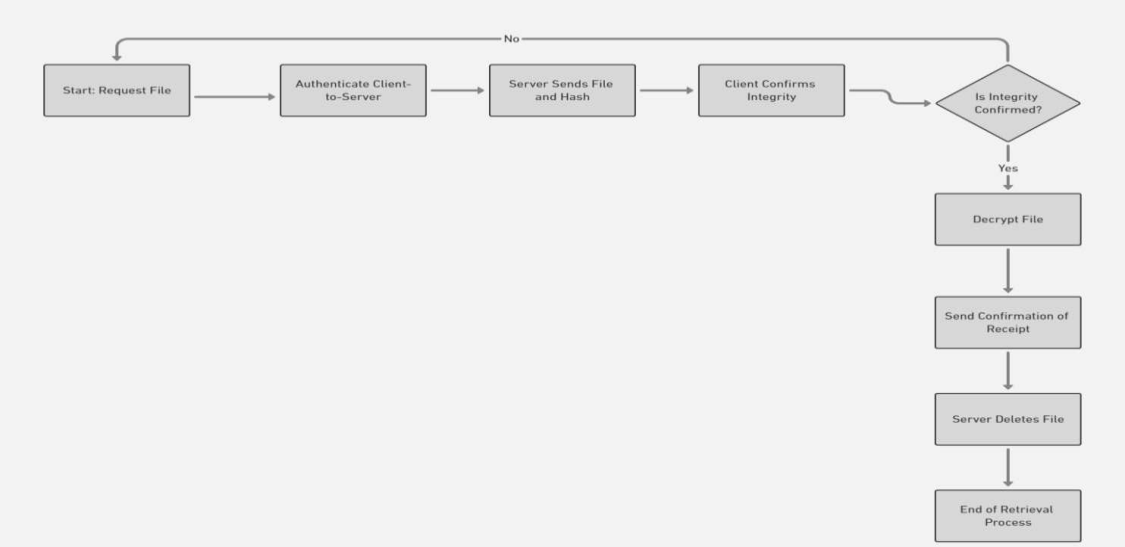
B. *Server send file*

Once the client has verified to have the same credentials as the public key then the server will send the encrypted file along with the hash to the client

C. *Client confirms integrity*

Upon the client receiving the file and hash, the client will then be able to calculate the hash of the encrypted file and compare it with the received hash to confirm integrity.

D. *Decrypt the file*



In order for the client to decrypt the file the client will then use the `decrypt()` function, which is the inverse of `encrypt (, to decrypt the file. in the decrypt () function the client will have to loads their private key and the session key to decrypt the file

E. *Confirmation of receipt*

Once data has been decrypted and verified to have the correct data, the client will send a confirmation of receipt and successful decryption to the server, prompting them to delete the file from its storage.

IV. Key Components

A. *Encrypt and decrypt*

These functions handle the encryption and decryption of files, respectively. They ensure the confidentiality of the data during transmission and storage.

B. *Send*
Facilitates the transmission of data between client and server, ensuring that data is sent securely and to the correct recipient.

C. *Authentication server-to-client*
A mechanism to verify the identity of the client and server, which can be based on public-key cryptography or a shared secret.

D. *Integrity check*
Utilizing hash functions to generate and verify the integrity of the files during transmission and storage.

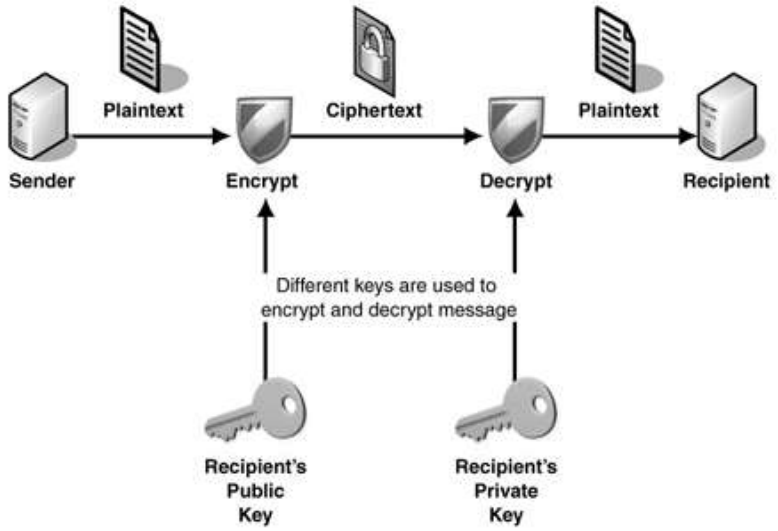


Fig.3 public key infrastructure

V. Visualization

A. *Encryption and sending process*
A flowchart showing the steps from file encryption, and integrity check generation, to sending the file to the server. In Figure 3 it shows how the sender encrypts the plain text before sending it to server to ensure that the data in plaintext will have its integrity and the correct information as it reaches the recipient. Once it is encrypted the sender sends the plaintext to the server using the recipient public key.

B. *Retrieval and Decryption Process*
A flowchart illustrating the file request, authentication, file and hash receipt, integrity check, decryption, and confirmation of receipt Figure 3 demonstrates that once the server has found the recipient, the recipient will then use its private key to decrypt the plain text and ensuring that the plain text is not tampered with.

References

[1] "P2P basics A brief look at Peer-to-Peer Systems," Sign In, https://canvas.noctrl.edu/courses/10559 (accessed Mar. 8, 2024).
[2] "Hashing,MAC," Sign In, https://canvas.noctrl.edu/courses/10559 (accessed Mar. 8, 2024).
[3] "Examples¶," Examples - PyCryptodome 3.210b0 documentation, https://www.pycryptodome.org/src/examples#encrypt-data-with-aes (accessed Mar. 8, 2024).

[4] “Peer-to-peer,” Wikipedia, https://en.wikipedia.org/wiki/Peer-to-peer (accessed Mar. 10, 2024).

[5] “SHA-3: SHA-3 secure hash crypto engine Ip core,” CAST, https://www.cast-inc.com/security/encryption-primitives/sha-3 (accessed Mar. 10, 2024).

[6] “Intro to cryptography on the web,” SlideShare, https://www.slideshare.net/mwynholds/intro-to-cryptography-on-the-web (accessed Mar. 10, 2024).