File encryption, decryption, and hashing application

Introduction

File encryption, decryption, and hashing are essential security measures for protecting sensitive data from unauthorised access and tampering. This document aims to provide a clear understanding of these concepts, their main components, and the functional flow of a file encryption, decryption, and hashing application. Additionally, we will discuss existing solutions, technologies used, and the rationale behind choosing Python for this project.

The usefulness of file encryption, decryption, and hashing application:

The encryption, decryption, and hashing application would ensure:

1. *Data confidentiality*: The application can provide a secure way to store and transfer sensitive data. Encrypting the data, becomes unreadable to unauthorized parties, thus protecting against data theft.
2. *Authenticity*: The application can use hashing algorithms to ensure data integrity and authenticity. This helps to prevent data tampering or unauthorized changes.
3. *Compliance*: Many industries, such as healthcare and finance, have regulations that require data encryption to ensure compliance.

Who would use the application?

1. *Financial institutions*: Banks and financial institutions often deal with sensitive financial information and personal data that must be kept secure.
2. *Healthcare providers:* Healthcare providers must comply with strict regulations that require them to protect patient information.
3. *E-commerce businesses:* Online businesses that handle sensitive customer data, such as credit card numbers, must ensure that this data is kept secure.
4. *Government agencies:* Government agencies often deal with sensitive data that must be kept confidential. Encryption, decryption, and hashing applications can help them to protect this data and ensure that it is not accessed by unauthorized parties.
5. *Individuals*: Individuals who want to protect their personal data, such as financial records or sensitive documents, can use encryption, decryption, and hashing applications to keep their information secure. This is especially important when sharing data over unsecured networks or storing data on cloud services.

Main Concepts

File Encryption

File encryption is the process of converting a file's contents into unreadable gibberish using a cryptographic algorithm. This ensures that only the intended recipient (who has the decryption key) can access the file's original contents. There are two main types of encryption algorithms:

- Symmetric Encryption: The same key is used for both encryption and decryption. The most common symmetric encryption algorithms include AES (Advanced Encryption Standard) and DES (Data Encryption Standard).

Examples of software that use symmetric-key encryption include Microsoft BitLocker, 7-Zip, and WinZip.

1. **Microsoft BitLocker** is a full-disk encryption tool that is built into the Windows operating system. It uses symmetric-key encryption to encrypt the entire contents of a hard drive, protecting the data from unauthorized access.
2. **7-Zip** is a file archiving and compression tool that uses symmetric-key encryption to protect data in compressed archives.

- Asymmetric Encryption: Different keys are used for encryption and decryption. The encryption key is public, while the decryption key is private. RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) are widely used asymmetric encryption algorithms.
  Examples of software that use asymmetric-key encryption include Pretty Good Privacy (PGP) and GNU Privacy Guard (GnuPG).
  - **Pretty Good Privacy (PGP):** PGP is a data encryption and decryption software that provides cryptographic privacy and authentication for data communication. It is commonly used for email encryption and digital signatures.

There is also:

- *Disk encryption:* Disk encryption is a method of encrypting entire disks or partitions, protecting the data from unauthorized access if the device is lost or stolen. Examples of software that use disk encryption include Microsoft BitLocker, VeraCrypt, and Apple FileVault.
  - **VeraCrypt:** is a free and open-source disk encryption tool that can be used on Windows, macOS, and Linux. It can encrypt entire disks, partitions, and even virtual disks. VeraCrypt uses a combination of symmetric-key and asymmetric-key encryption algorithms to provide strong security for data.
  - **Apple FileVault:** is a full-disk encryption tool that is built into macOS. It encrypts the entire contents of the Mac's startup disk, protecting the data from unauthorized access. FileVault can also be used to encrypt external drives connected to the Mac.

- *Network encryption:* Network encryption is a method of encrypting data that is transmitted over a network, such as the Internet. It is commonly used to secure online transactions, online banking, and secure communication. Examples of technologies that use network encryption include Transport Layer Security (TLS), Secure Sockets Layer (SSL), and Virtual Private Networks (VPNs).

  - **Transport Layer Security (TLS):** TLS is a cryptographic protocol that provides secure communication over a network. It encrypts data in transit between two endpoints, such as a web server and a client's web browser. This ensures that the data being transmitted cannot be intercepted and read by unauthorized parties.
  - **Secure Sockets Layer(SSL):** uses encryption algorithms to encrypt data transmitted between a web browser and a web server, preventing unauthorized access and tampering. SSL works by establishing a secure connection between the client and server through a process called a

"handshake". During the handshake, the client and server agree on a common encryption algorithm and exchange cryptographic keys to establish a secure connection.
    - **VPN (Virtual Private Network):** is a technology that allows users to create a secure and private network connection over a public network, such as the Internet. It does this by encrypting the data that is transmitted between the user's device and the VPN server, which is usually located in a different geographic location.

- *Cloud encryption:* Cloud encryption is a method of encrypting data that is stored in the cloud, protecting it from unauthorized access. Examples of software that use cloud encryption include Cryptomator, Boxcryptor, and VeraCrypt.
    - **Cryptomator** is a free and open-source software program that can be used on Windows, macOS, Linux, Android, and iOS devices. It also allows users to encrypt files and folders stored in cloud storage services such as Dropbox, Google Drive, and OneDrive. Cryptomator uses AES encryption with 256-bit keys to protect data and also allows for two-factor authentication and integration with other security tools such as Keybase.
      Both Boxcryptor and Cryptomator offer similar functionality, but Boxcryptor is a paid program with more advanced features and support options

File Decryption
File decryption is the reverse process of encryption. It involves converting the encrypted data back into its original, readable form using the appropriate decryption key. In symmetric encryption, the same key used for encryption is used for decryption. In asymmetric encryption, the private key is used for decryption.

Hashing
Hashing is a one-way function that takes an input (or "message") and returns a fixed-size string of bytes. The output is typically a "digest" that is unique to each unique input. Hashing is commonly used for data integrity verification, password storage, and digital signatures.

Some popular hashing algorithms include SHA-256 (Secure Hash Algorithm), SHA-3, and BLAKE2.

Functional Flow

…..

Existing Solutions and Technologies

There are several existing solutions for file encryption, decryption, and hashing, such as OpenSSL, GnuPG, and VeraCrypt. These tools offer different features and support various cryptographic algorithms.

Most solutions are developed using languages like C, C++, and Python. Python is an excellent choice for this project due to its simplicity, readability, and a vast library of cryptographic modules, such as cryptography, pycrypto, and hashlib.

Why Python?

We have chosen Python for the following reasons:

Ease of use: Python's simplicity and readability make it easy to understand and maintain the code.

Library support: Python has a rich ecosystem of libraries for cryptography, such as cryptography, pycrypto, and hashlib.

Platform independence: Python is platform-independent, which means our application can run on various operating systems without modification.

Community support: Python has a large and active community, which ensures access to up-to-date resources and support.