

Introduction

With loads of files exchanged every day, data privacy and secure file sharing have become crucial in today's world. As technology evolves, there's an increasing need for people and organizations to share sensitive information without risking data breaches or unauthorized access. This Capstone project aims to meet this need by providing a secure, web-based platform that allows users to encrypt and decrypt various file types including documents, photos, and audio files using the military-grade AES-256 encryption algorithm. Launching this project will offer a reliable, user-friendly solution for sharing files temporarily, with added security features like expiration times, limits on decryption, and shareable features (such as a link to share).

I was inspired by services like [EnvShare](#), [Hat.sh](#), and [Cyberchef](#) which emphasize private, temporary file storage, focus on security and encryption, and various tools/methods. However, this project will be an exhaustive tentative attempt to differentiate by integrating new functionalities, such as the encryption/decryption of any file type. Furthermore, this project intends to be more versatile by offering users a choice of multiple encryption algorithms, making it adaptable to a wider variety of security needs.

This project aims to support the larger cybersecurity and privacy community by providing a dependable tool for secure, temporary file sharing. It serves anyone needing a secure method for file transfer, including journalists, legal professionals, and NGOs operating in high-risk settings where secure information transmission is essential. By offering temporary file storage and self-destructing download links, this platform minimizes the long-term exposure of sensitive data, giving users better control over their information.

Ultimately, this project empowers individuals and organizations to communicate securely, with peace of mind that their data is protected by advanced encryption standards. Through this research and development, I hope to contribute a valuable tool to the cybersecurity community and lay a solid foundation for future work in secure digital communication. It also encompasses possible further implementation to expand this project including community-focused features related to privacy and security.

Research

Nowadays with the extremely fast evolution of technology, we must keep up with privacy and security. As we have learned in our book Code v2 by Lawrence Lessig, it is important to be careful with cyberspace as we could lose complete control of our identity and privacy. This project addresses this need by creating a platform that provides strong encryption for various file types through AES-256 encryption (and possibly more), with the potential to support other cryptographic algorithms in the future. Inspired by EnvShare, Hat.sh, and Cyberchef this project goes beyond simple text encryption to support multiple file formats, such as PDFs, images, audio files, and more. Moreover, it includes options for access control, like expiration times and decryption limits (Total amount of reads. Which means you can decrypt a file a certain amount of time specified during the encryption process).

The aim is to provide a short-term, secure file storage solution that prioritizes both user security and privacy, utilizing AES-256, a well-regarded encryption algorithm known for its durability.

Many professional fields recognize the ethical responsibility to ensure secure file sharing. For example, the International Center for Journalists highlights the importance of secure digital tools to protect journalists and their sources in high-risk areas. (source reference Fact Sheet *)

While existing secure-sharing platforms like EnvShare* or Cyberchef* offer valuable frameworks, they have limitations. EnvShare restricts encryption options to text and is built in TypeScript, which can limit flexibility in using diverse encryption libraries and algorithms. By developing this project's encryption and decryption in Python, we open the door to a range of cryptographic methods, including lattice-based and post-quantum algorithms (two of the few algorithms currently on stake for the quantum technology battle).

Additionally, features like decryption limits and customizable expiration times set this platform apart from others. These improvements allow for more precise control over data access, particularly for users who need strict policies on how many times a file can be decrypted or when its access should expire.

The value of secure file sharing extends beyond individual users, as I have already noted before there are also specific roles in which privacy is crucial such as journalism, law, healthcare, and NGOs. For example:

Journalism: Journalists often operate in environments with limited privacy protections, making secure data transfer critical. This platform allows journalists to share sensitive documents with sources, even in high-surveillance areas.

Legal Services: Law firms need tools that protect client confidentiality. By incorporating features like expiration times and decryption limits, this project helps legal professionals manage sensitive files responsibly, reducing risks of unintended data exposure.

Nonprofits and NGOs: Nonprofits operating in politically sensitive regions or under close surveillance can benefit from this secure file-sharing system to safeguard sensitive information about their work or collaborators, as recommended by secure communication resources like the Global Investigative Journalism Network (GIJN).

However, we must keep in mind also certain drawbacks of having such a tool. It can help everyone, which means also those with malicious intentions.

What is AES-256-bit encryption?

The Advanced Encryption Standard, or AES especially the AES-256 version (there are three versions, 128, 192, or 256 bits based on key length) is one of the strongest encryption methods out there. It was officially adopted in 2001 by the National Institute of Standards and Technology (NIST). To find the best encryption method, NIST held a worldwide competition, and a cipher called "Rijndael" came out on top. This cipher, designed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, became the new standard, replacing the older Data Encryption Standard (DES). AES brought a major upgrade in data security for U.S. federal agencies and quickly spread around the world as the go-to choice for protecting sensitive data in both public and private sectors.

AES-256 is what's known as a "symmetric key block cipher." This means it uses the same key to lock (encrypt) and unlock (decrypt) information. It processes data in 128-bit chunks and uses a 256-bit key, which makes it extremely secure and tough to crack, even with brute-force attacks (where attackers try every possible key until they find the right one). Today, AES-256 is used for all kinds of security tasks, including:

- Securing file storage (keeping stored data private)
- Encrypting network data (protecting data while it's being transmitted)
- Enabling secure browsing over HTTPS/TLS (keeping online transactions and communications safe)

Here's a quick look at the pros and cons of AES-256:

Pros:

- **High Security:** With a 256-bit key size, AES-256 is incredibly hard to crack through brute force.
- **Versatility:** It works well on both hardware and software, so it's compatible with a wide range of devices and systems.
- **Global Adoption:** Widely used across the world, meaning it's supported by many platforms and devices.

Cons:

- **Resource-Intensive:** AES-256 uses more computing power than some other algorithms, which can slow down devices with limited resources.

- **Key Management Challenges:** Managing keys is crucial, if a key is weak or compromised, the encryption can be broken. To date, any vulnerabilities have come from how AES was implemented, not the algorithm itself.

Thanks to its strength, AES-256 has become a standard for military-grade encryption and is trusted worldwide. Banks, hospitals, social media platforms, and many other organizations use it to protect sensitive information and keep data safe in today's digital world. (*source NIST document*)

Here a diagram of how AES-256 algorithm works will be inserted

Post-Quantum Cryptography: As quantum computing advances, traditional algorithms like RSA (comes from the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977.) and ECC (Elliptic-curve cryptography) may become vulnerable. AES-256 is considered quantum-resistant until at least 2050, but integrating lattice-based or post-quantum cryptographic (PQC) standards, as recommended by NIST, could future-proof this platform. (NIST article on Post-Quantum standards)

Steganography: In addition to conventional encryption, techniques like steganography could offer an additional layer of security. This method, which hides sensitive information within other media, can be particularly useful in adversarial environments where discrete data transfer is needed.

Libraries and Installation procedures

This project uses various libraries and frameworks in order to include all the features explained above. First, for the Front-end side of the project, Next.js is the choice to handle user interface and experience. Next.js being a framework built on top of the React framework allows for many benefits in the aesthetical and maintainability aspects. The back-end is built using Python, which ensures efficiency for the algorithm, database, and API communication. The database of choice for this project is Redis provided by Upstash.com. This choice was because Redis databases provide a smooth and lightweight option to transfer data while maintaining and ensuring security and privacy.

To be expanded and corrected

Features

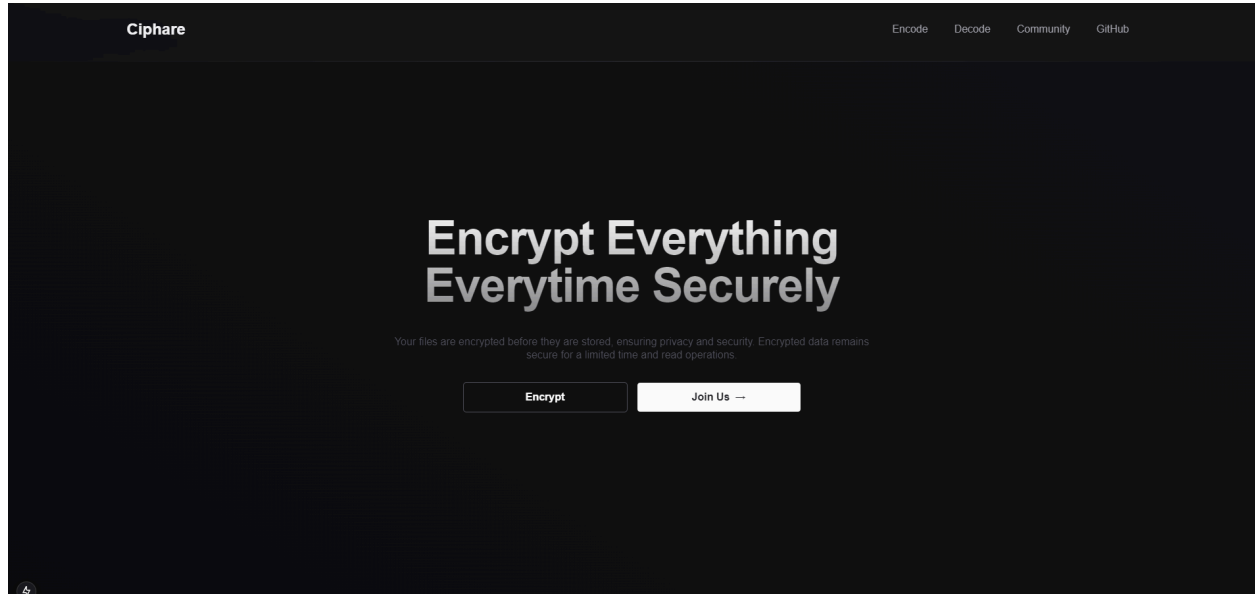
The features included in this project are but not limited to the following:

- Store temporarily encrypted files
- Store with a TTL(time to live) the encrypted files.
- Store with a TAR (total amount of Reads). Therefore the file can be read/decrypted only X times
- Use AES-256 encryption algorithm
- Input files of any type (PDF, JPEG, PNG, MP4, etc...)
- Download the files aside from only sharing
- Provide a sharable link for the encrypted data

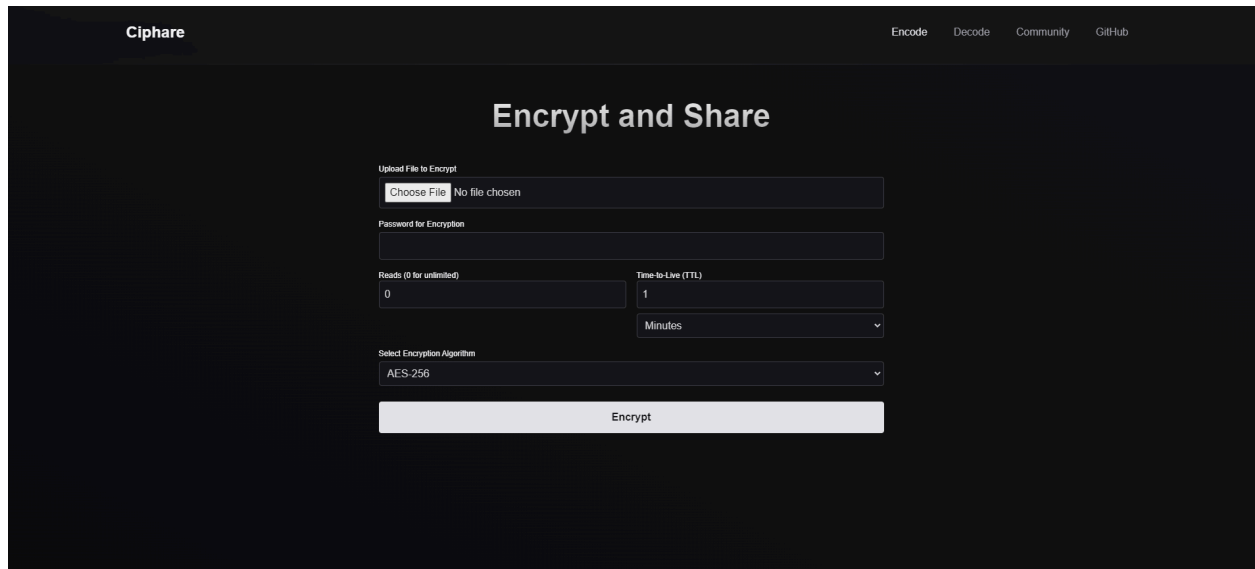
User interfaces and User Interactions or Visualization

The following is a showcase of the website using screenshots and the step-by-step process of the main scope of this project.

Homepage (page.tsx) (from live environment development deployment)



Encode page (encode/page.tsx) Where we can encrypt our files



The link to share with the file ID needed to decrypt the file. This is the screen when we successfully encrypted a file (in our case nft.png)

The screenshot shows a web interface titled "Encrypt and Share" on a dark background. It contains several input fields and a button. The "Upload File to Encrypt" section has a "Choose File" button and the filename "nft.png". The "Password for Encryption" section has a password input field with masked characters. The "Reads (0 for unlimited)" section has a numeric input field with the value "2". The "Time-to-Live (TTL)" section has a numeric input field with the value "10" and a dropdown menu set to "Minutes". The "Select Encryption Algorithm" section has a dropdown menu set to "AES-256". A large "Encrypt" button is centered below these fields. At the bottom, a "Shareable Link:" section displays the URL "https://yourdomain.com/decode/21wQ7pG3ZzFPwEVJ3rG0ZQ" and the instruction "Share this link to allow decryption."

Encrypt and Share

Upload File to Encrypt

Choose File nft.png

Password for Encryption

Reads (0 for unlimited)

2

Time-to-Live (TTL)

10

Minutes

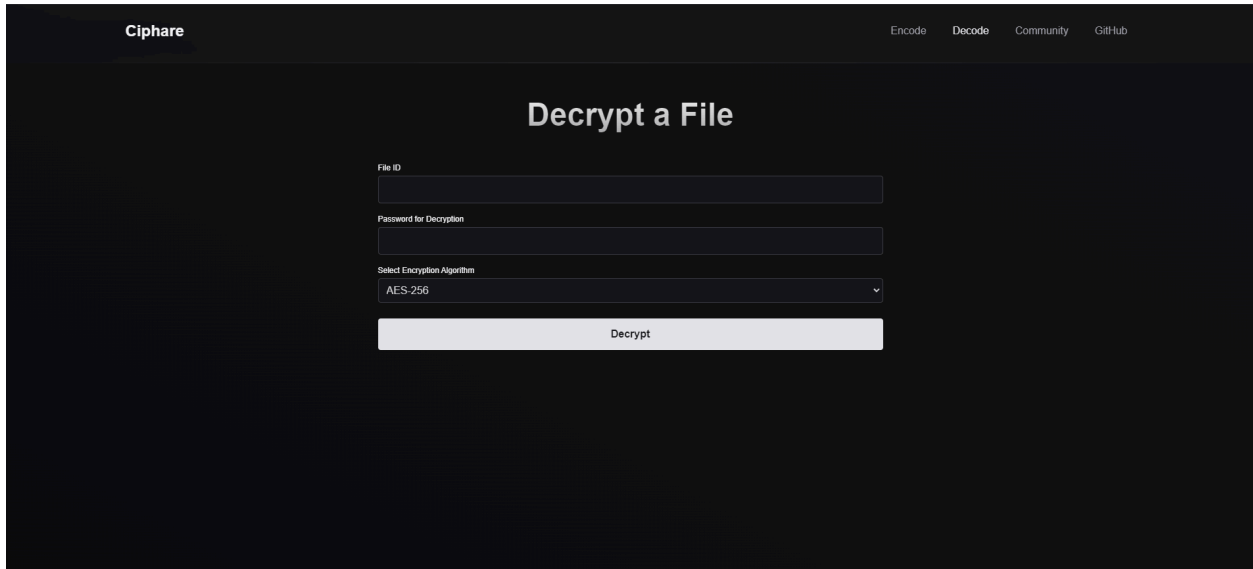
Select Encryption Algorithm

AES-256

Encrypt

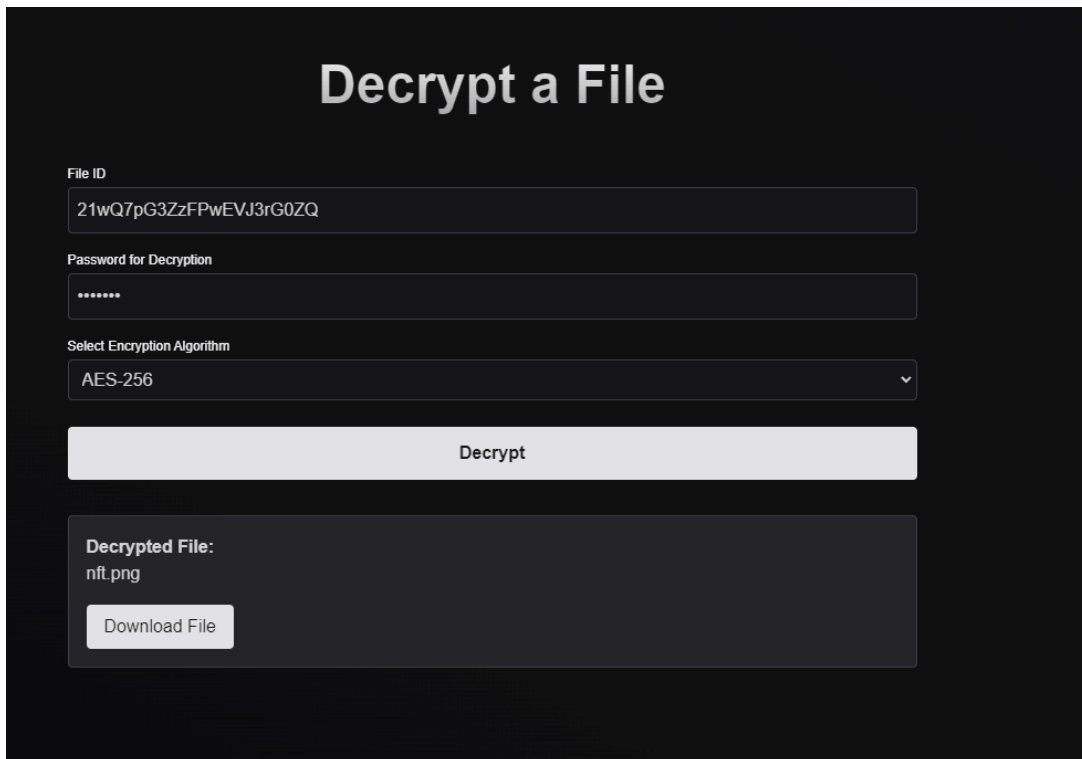
Shareable Link:
<https://yourdomain.com/decode/21wQ7pG3ZzFPwEVJ3rG0ZQ>
Share this link to allow decryption.

Decode page (decode/page.tsx) Where we can decrypt our files using the given file ID or shared link



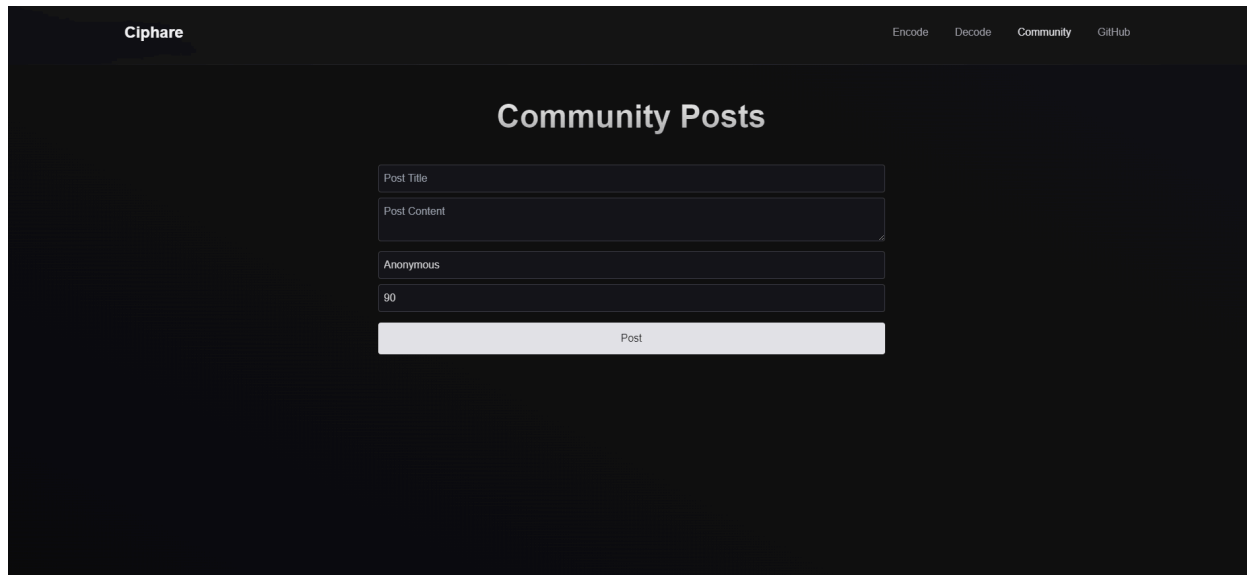
The screenshot shows the 'Ciphare' application interface. At the top, there is a navigation bar with links for 'Encode', 'Decode', 'Community', and 'GitHub'. The main heading is 'Decrypt a File'. Below this, there is a form with three input fields: 'File ID', 'Password for Decryption', and 'Select Encryption Algorithm'. The 'Select Encryption Algorithm' dropdown is currently set to 'AES-256'. A 'Decrypt' button is located at the bottom of the form.

Using the ID we were given and the password used to encrypt the file we can decrypt it. This is the screen when we successfully decrypt.



This screenshot shows the 'Decrypt a File' form after a successful decryption. The 'File ID' field now contains the value '21wQ7pG3ZzFPwEVJ3rG0ZQ'. The 'Password for Decryption' field is masked with dots. The 'Select Encryption Algorithm' dropdown remains set to 'AES-256'. The 'Decrypt' button is still present. Below the form, a section titled 'Decrypted File:' shows the filename 'nft.png' and a 'Download File' button.

Lastly, the Community page where you can create posts, comment, and like.



The screenshot shows a web interface for 'Ciphare' with a dark theme. The top navigation bar includes links for 'Encode', 'Decode', 'Community', and 'GitHub'. The main heading is 'Community Posts'. Below this is a form with four input fields: 'Post Title', 'Post Content', 'Anonymous', and a character count '90'. A 'Post' button is located at the bottom of the form.

Ciphare

Encode Decode Community GitHub

Community Posts

Post Title

Post Content

Anonymous

90

Post

Here is what it looks like after creating a post:

Community Posts

Post

Welcome
This is the first post with a mandatory expiration time from min 1 day to max 90 days (3 months)
by Anonymous
Likes: 0
Created at: 11/14/2024, 6:39:23 AM

[Like](#) [Delete Post](#)

Comments

Data

Talk about the data that you are collecting and all of the relationships between the data. A E-R diagram might be helpful with this depending on if you have record-based data. For research, talk about your dataset and what are the transitions/ or transformations of your dataset to its final output.

Testing, evaluation

In order to test this project there are various ways. First, it is required to test the algorithm for encryption and decryption. Since the project is focused on any kind of file, then the test should

be on multiple files. The test will begin with the scripts for encryption and decryption. Once tested and debugged, the test will proceed on the website in production or online through dev mode. The test will be the same as before, by encrypting and decrypting multiple files and fixing any bugs that we may encounter. Lastly, the most important tests are those related to the security and defense of the website itself. Does it have any vulnerability to be exploited? Does it have holes where confidential data can be exfiltrated? These are all important questions where the answer lies in testing by applying penetration testing techniques. Through offensive attacks, we can check for any vulnerabilities mitigating risks for security failures.

Future adds

The improvements that can be made to this project are many. The first step was to create an environment focused on cybersecurity providing more knowledge and tools for privacy and security. However, we must keep in mind that privacy and security in cyberspace require constant improvement and upgrade. Ideas to make the project, even more, community-oriented, are to integrate a forum dedicated to cybersecurity topics such as privacy and security. People will be able to either access a personal account or a temporary account in order to communicate with the community of experts, enthusiasts, or people who need support. Furthermore, the project will integrate many more algorithms for encryption and decryption, especially those built with post-quantum technology in mind. New tools related to security and privacy will be implemented, such as vulnerability analysis, scans, or monitoring.

Conclusion

To be finished

References (APA citations)

<https://www.internetsociety.org/resources/doc/2020/fact-sheet-how-encryption-can-protect-journals-and-the-free-press/>

<https://www.geeksforgeeks.org/advanced-encryption-standard-aes/>

<https://stackoverflow.com/questions/12524994/encrypt-and-decrypt-using-pycrypto-aes-256>

https://github.com/gaetanserre/AES_256-Python/blob/master/src/AES.py

<https://www.youtube.com/watch?v=F2av7TaVc5Q>

<https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>

<https://thepythoncode.com/article/encrypt-decrypt-files-symmetric-python>

<https://pypi.org/project/cryptography/>

<https://pypi.org/project/pycryptodome/>

<https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms>

<https://csrc.nist.gov/files/pubs/fips/197/final/docs/fips-197.pdf>

<https://www.javatpoint.com/image-steganography-using-python>

<https://envshare.dev/>

<https://cyberchef.org/>