Andrea Marcelli
CPSC 49200 003
December 11th, 2024
Project Manual

Ciphare - A web application

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, shareable features (such as a link to share), and multiple selections of algorithms.

I was inspired by services like EnvShare, Hat.sh, Cyberchef, and OnionShare which emphasize private, temporary file storage, focus on security and encryption, and various tools/methods related to cryptography and cybersecurity. However, this project will be an exhaustive tentative attempt to differentiate by integrating new functionalities, such as the encryption/decryption of any file type, implementing a web interface, and a community space where users can exchange ideas, questions, and more. 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 secure, temporary file sharing tool. It serves anyone needing a secure method for transferring files, 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, a mobile application, the addition of various cybersecurity tools, and more.

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, Cyberchef, and OnionShare, 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 within 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. (Fact Sheet)

While existing secure-sharing platforms like EnvShare, Cybercher, or OnionShare 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 like Cyberchef or OnionShare. 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 also keep in mind certain drawbacks of having such a tool. It can help everyone, which means also those with malicious intentions. Therefore, the website poses a dilemma, which is what do in the case that a criminal uses such a tool? The only way to mitigate such a problem would be to make the security of user privacy less secure, therefore traceable. Making the website vulnerable to exploitation. For this reason, in this project I haven't mitigated this issue, being too challenging for the time frame given to develop this project. The solution will be cited again when presenting possible future adds for the website, such as the integration of personal AI agents.

What is AES-256-bit encryption?

Andrea Marcelli
CPSC 49200 003
December 11th, 2024
Project Manual

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:

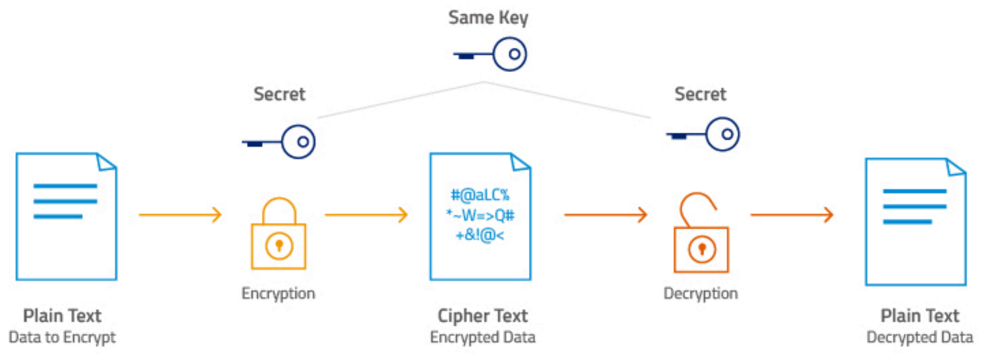
- **High Security:** With a 256-bit key size, AES-256 is incredibly hard to crack through brute force.
- **Versatility:** It works well on both hardware and software, so it's compatible with a wide range of devices and systems.
- **Global Adoption:** Widely used across the world, meaning it's supported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

The following is a diagram of how the AES-256 algorithm works:



(image source: Black Barth)

We have a key generated and has to be 32 bytes used to both encrypt and decrypt, therefore symmetric encryption. The following diagram instead shows the inner processes that happen during encryption and decryption of files:

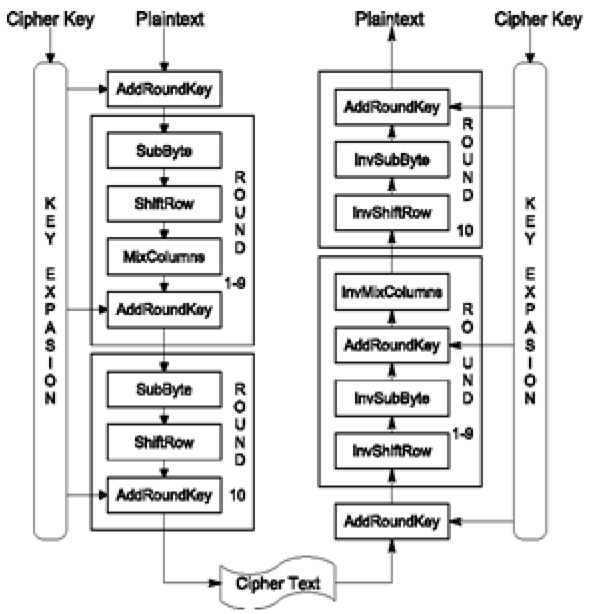


Figure 2: Block diagram AES Encryption and Decryption

AES-256 uses a 256-bit encryption key. The key(which can be 16 or 32 bytes) is expanded into multiple round keys using a process called key scheduling, which involves substitution and permutation.

1. **Initial Round**:
    - **AddRoundKey**: The plaintext (input data) is XORed with the first round key derived from the original key.
2. **Rounds N (14 Rounds for AES-256)**:
    - Each round consists of the following operations:
        1. **SubBytes**: Each byte of the state is replaced with a byte from a predefined substitution box (S-Box) to ensure non-linearity.
        2. **ShiftRows**: Rows of the state matrix are shifted by varying offsets to introduce diffusion.
        3. **MixColumns**: Columns of the state are mixed using linear algebra transformations to increase diffusion (skipped in the final round).
        4. **AddRoundKey**: The current state is XORed with the round key for this round.
3. **Final Round**:
    - The final round omits the **MixColumns** step.
    - It performs **SubBytes**, **ShiftRows**, and **AddRoundKey**.

After the final round, the state matrix is converted back into a linear array to produce the encrypted ciphertext. For the decryption process, we reapply the same but with the inverted functions.

While the AES-256 algorithm is well-known and globally considered the standard for encryption, we might be interested in using different algorithms. Considering the future technologies we are developing, such as the quantum computer, we need to be prepared.

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). The post-quantum algorithms are designed to be resistant to quantum computer attacks by utilizing mathematical problems that are hard for both classical and quantum computers. These algorithms are based on many different mathematical disciplines, such as lattice-based cryptography, which depends on the hardness of problems like the LWE problem, a problem of solving linear equations with added noise. Another approach is hash-based cryptography, which bases its security on the hardness of collision-finding problems in cryptographic hash functions. Code-based cryptography relies on error-correcting codes, such as those underlying the McEliece cryptosystem. Multivariate polynomial cryptography solves systems of multivariate quadratic equations. Isogeny-based cryptography draws its roots from the complexity of problems related to elliptic curve isogenies.

Key establishment and encryption, for instance, make use of lattice-based algorithms such as Kyber and NTRU, providing secure mechanisms of key exchange resistant to quantum attacks. The encryption process in these systems involves a transformation of plaintext into ciphertext with the use of these mathematically intricate problems, ensuring strong resistance against quantum-based decryption attempts.

The security of post-quantum algorithms is meant to be resilient against both classical and quantum attacks. They do this with the aim of backward compatibility and seamless integration into existing systems to ease their adoption. However, many post-quantum algorithms require larger keys or ciphertext sizes than traditional methods, which introduces performance considerations. Ongoing optimizations are being made to these algorithms to make them more practical for real-world applications without compromising their security. This adaptability makes them effective in safeguarding sensitive data in the quantum computing era.

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. While the above-mentioned techniques are a lot more secure and safe, I found steganography to be a cool and effective way to encrypt data without requiring too much computational power.

Libraries and Installation procedures

This project uses various libraries and frameworks 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. Lastly, the backend is coded in Python and operated using Flask servers. The following is a restricted list of dependencies that are crucial to this project:

- **Flask**: Backend framework for API development.
- **Next.js**: Frontend framework for building user interfaces.
- **TypeScript**: Ensures type safety in the frontend code.
- **Redis**: Used for storing encrypted files and managing TTL and READS.
- **Upstash Redis**: Cloud-based Redis solution for scalability.
- **Cryptography**: Python library for implementing encryption algorithms.
- **Tailwind CSS**: Utility-first CSS framework for styling the front end.

The following are the instructions on how to run this project or your local environment:

Clone the repository:

```
git clone https://github.com/TonyMontana-dev/ciphare.git

cd ciphare
```

Install backend dependencies:

```
pip install -r requirements.txt
```

Install frontend dependencies:

```
npm install
```

Set up environment variables:

```
UPSTASH_REDIS_URL

UPSTASH_REDIS_PASSWORD

NEXT_PUBLIC_API_BASE_URL

DOMAIN

PORT
```

Start the backend:

```
flask run or python (or python3 based on your python version) FLASK

INDEX/APP.PY MAIN FILE PATH
```

Start the frontend:

```
npm run dev
```

Features

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

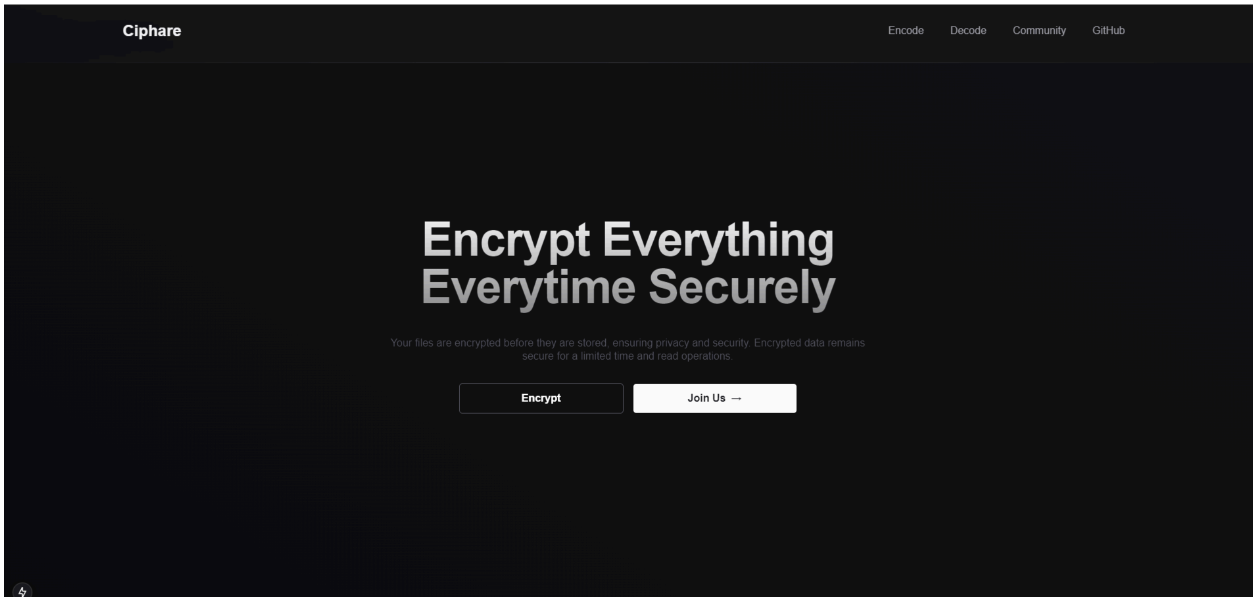
- Store temporarily encrypted files
- Store with a TTL(time to live) the encrypted files.

- Store with a TAR (total amount of Reads). Therefore the file can be read/decrypted only X times
- Use AES-256 encryption algorithm and multiple other algorithms
- 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
- Anonymous Community Posting
- Modern User Interface

User interfaces and User Interactions or Visualization

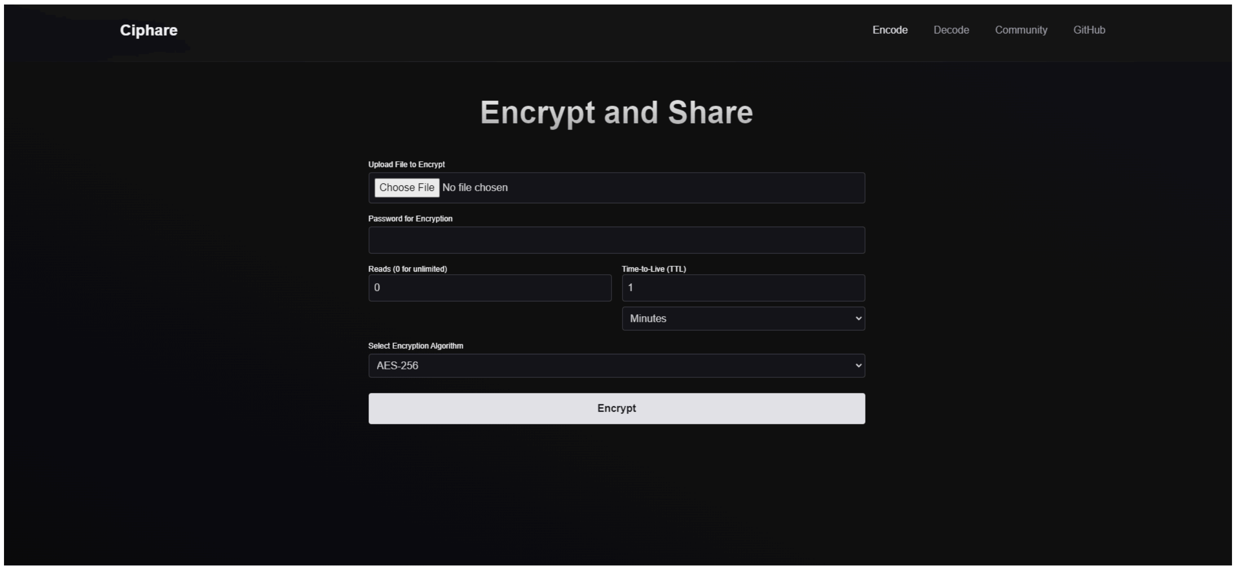
The following is a showcase of the website using screenshots and the step-by-step process of the main scope of this project.

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



Your files are encrypted before they are stored, ensuring privacy and security. Encrypted data remains secure for a limited time and read operations.

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



Encrypt and Share

Upload File to Encrypt: Choose File No file chosen

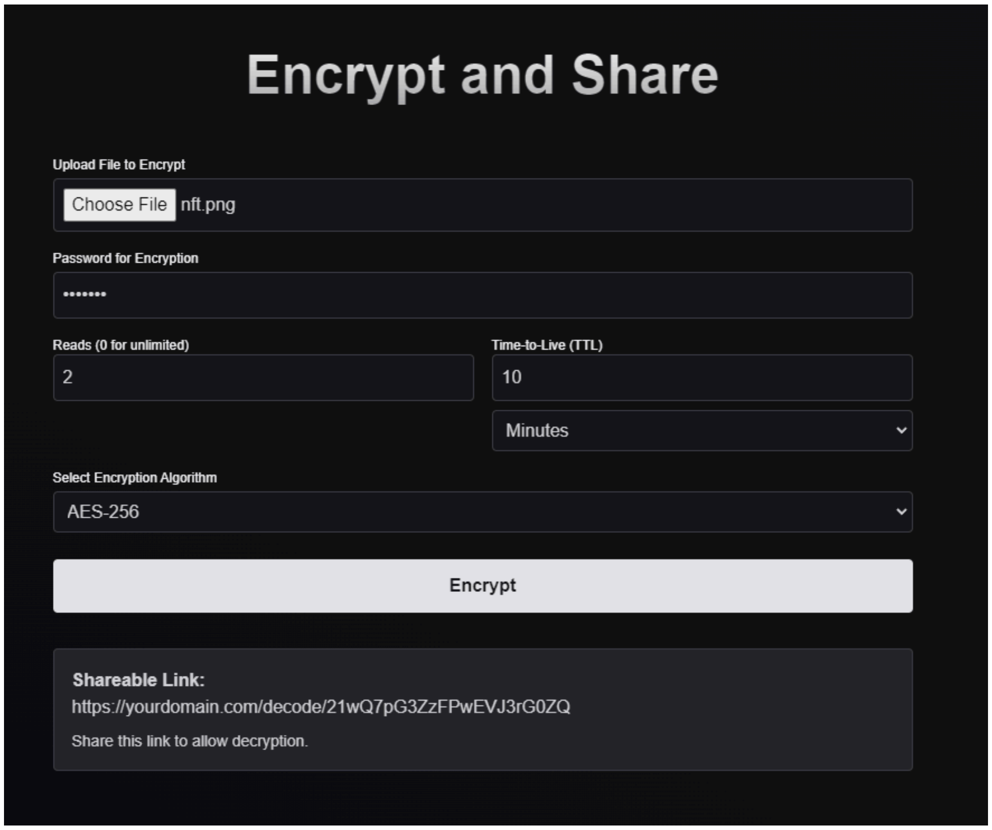
Password for Encryption:

Reads (0 for unlimited): 0

Time-to-Live (TTL): 1 Minutes

Select Encryption Algorithm: AES-256

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



Encrypt and Share

Upload File to Encrypt: Choose File nft.png

Password for Encryption: •••••••

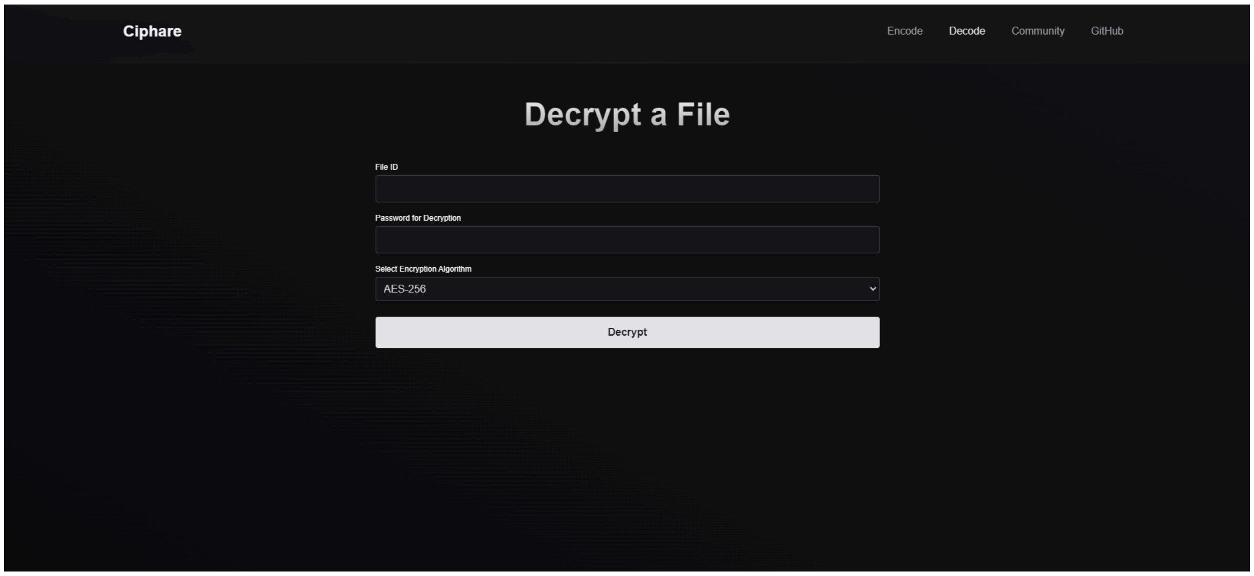
Reads (0 for unlimited): 2

Time-to-Live (TTL): 10 Minutes

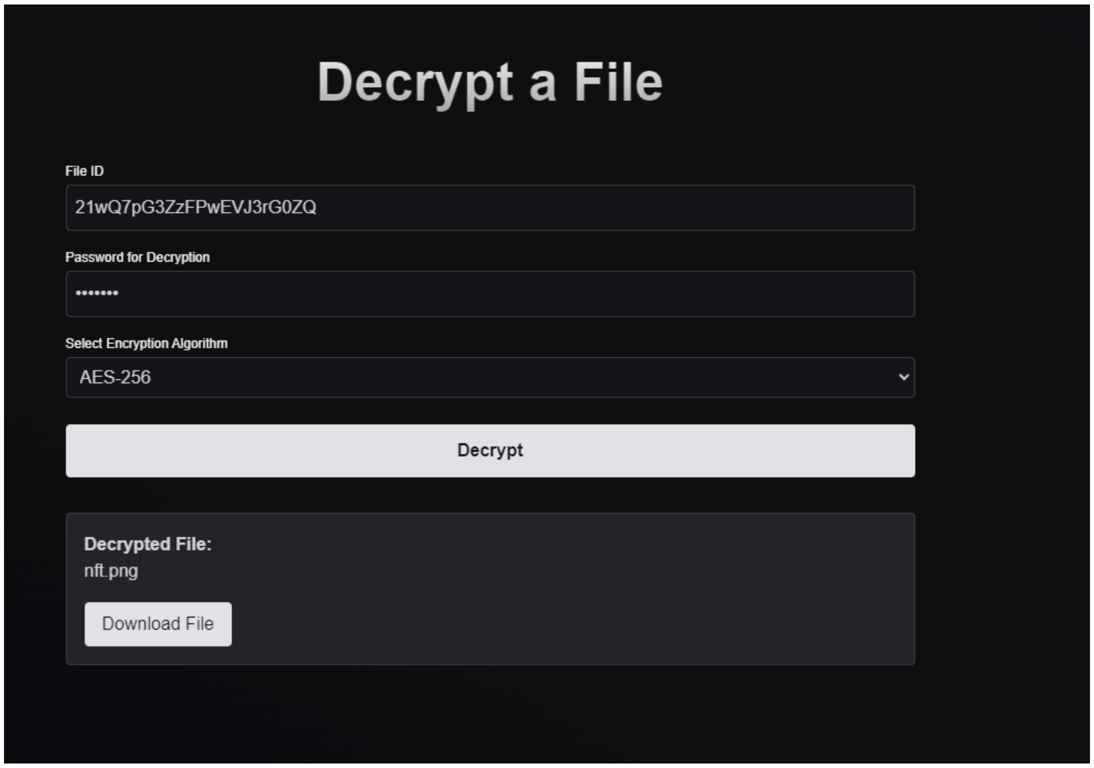
Select Encryption Algorithm: AES-256

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.



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.



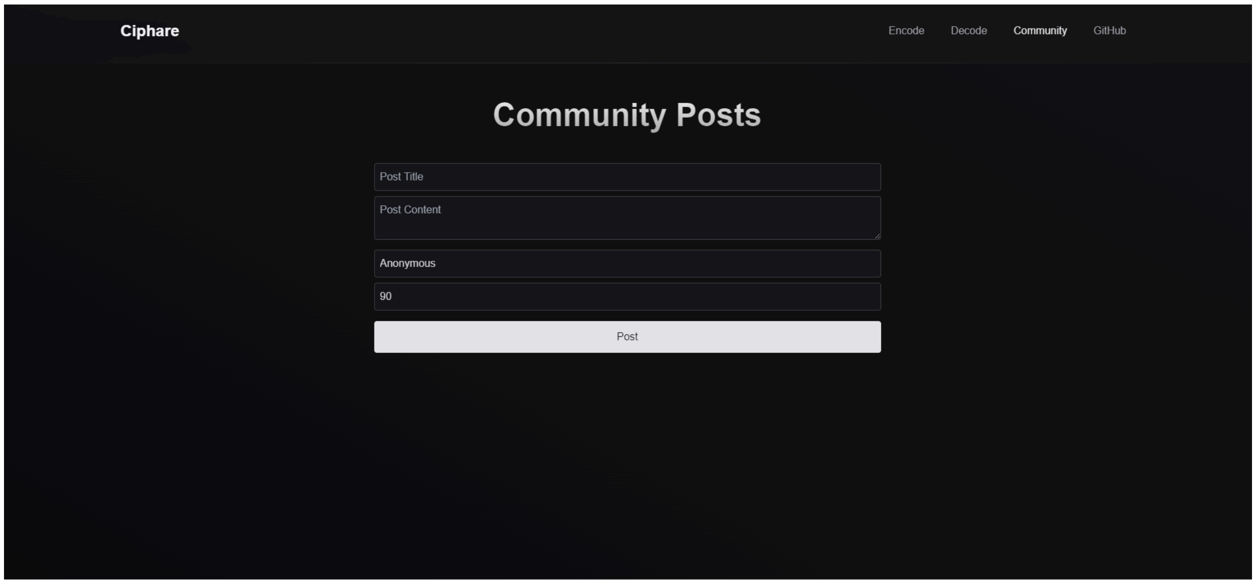
21wQ7pG3ZzFPwEVJ3rG0ZQ

•••••••

AES-256

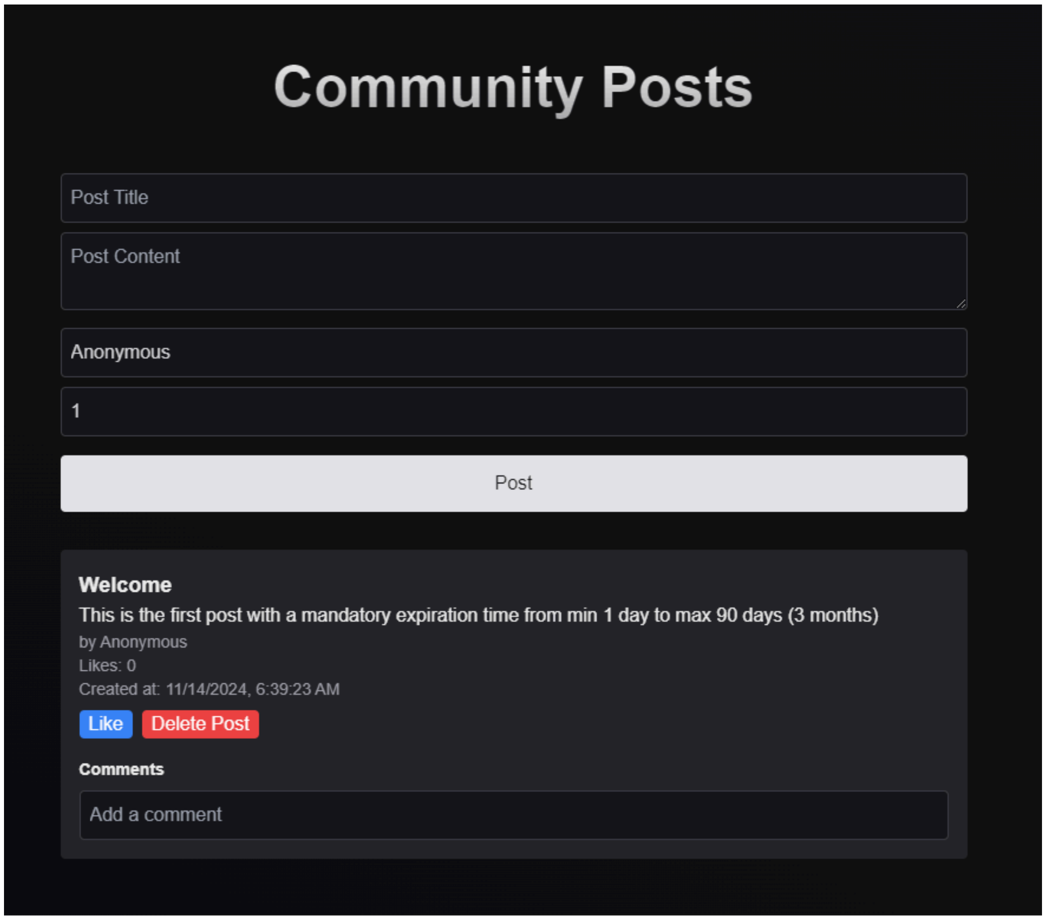
Decrypted File:
nft.png

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



Anonymous
90

Here is what it looks like after creating a 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

Data

The only data that this website works with is the encrypted file's metadata required to store them in the database and for the decryption process. Additionally, there are community posts and their respective comments along with metadata attributes.

Data Relationships:

- File Metadata: Includes file ID, TTL, READS, encryption algorithm, and file type.
- Community Posts: Includes post content, author, creation date, expiration time, and comments.
- Comments on Posts: Includes comments content, creation date, and expiration time.

Data Flow:

- Encrypted files are stored in Redis with metadata.
- Community posts are managed using Redis' key-value storage capabilities.
- API endpoints are created and used to transfer data in and out of the flask server.

Roles

Being the sole member of this project, I was responsible for all requirements and work for the implementation of this website and the research paper.

Backend Development: Design and implementation of APIs for encryption, decryption, and community posts. As well as the configuration of a Flask server dedicated to backend processes and API communication.

Frontend Development: User interface and experience design and integration with backend APIs.

Research: Exploring cryptographic techniques and community needs to inform project design. Research best practices and technologies for front-end and back-end development. Collecting material related to the project, such as cryptography-related content, next documentation, flask, and Vercel.

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. Lastly, an important and interesting implementation is the integration of AI capabilities. The website could use AI to provide a better community space, where users can interact with each other. AI could help mitigate risks for criminal use, and by ensuring safe practices are kept by users.

Conclusion

Building Ciphare has been an excellent learning experience in cryptographic implementation, API design, and user interface development. The challenges were immense, integrating such complex encryption mechanisms, managing Redis-based storage, and the proper implementation of API endpoints taught me many aspects of creating a web application. Even though the current version presents certain errors, the main functionality works efficiently. The encryption and decryption processes are well implemented and work correctly. However, there are still many improvements to be made as well as additional integration in order to expand the scope. Considering that I am a beginner and most of the technologies used were first-time experiences, I am very happy with the final result and eager to continue to work on this project making it more secure and efficient.

Andrea Marcelli
CPSC 49200 003
December 11th, 2024
Project Manual

References (APA citations)

Fact Sheet: How Encryption Can Protect Journalists and the Free Press. (n.d.). Internet Society. https://www.internetsociety.org/resources/doc/2020/fact-sheet-how-encryption-can-protect-journalists-and-the-free-press/

Geeksforgeeks. (2024, July 16). Advanced Encryption Standard (AES). GeeksforGeeks. https://www.geeksforgeeks.org/advanced-encryption-standard-aes/

python - Encrypt and decrypt using PyCrypto AES-256. (n.d.). Stack Overflow. https://stackoverflow.com/questions/12524994/encrypt-and-decrypt-using-pycrypto-aes-256

gaetanserre. (2020). AES_256-Python/src/AES.py at master · gaetanserre/AES_256-Python. GitHub. https://github.com/gaetanserre/AES_256-Python/blob/master/src/AES.py

Bali Coding. (2020, June 7). Encrypt/decrypt any file you want in Python using AES. YouTube. https://www.youtube.com/watch?v=F2av7TaVc5Q

NIST. (2024, August 13). NIST Releases First 3 Finalized Post-Quantum Encryption Standards | NIST. NIST. https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

Abdeladim Fadheli. (2019, September 26). How to Encrypt and Decrypt Files in Python - The Python Code. Thepythoncode.com. https://thepythoncode.com/article/encrypt-decrypt-files-symmetric-python

cryptography. (2019, October 17). PyPI. https://pypi.org/project/cryptography/

Eijs, H. (n.d.). pycryptodome: Cryptographic library for Python. PyPI. https://pypi.org/project/pycryptodome/

NIST. (2022). NIST Announces First Four Quantum-Resistant Cryptographic Algorithms. NIST, 1(1). https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms

Federal Information Processing Standards Publication 197 Announcing the ADVANCED ENCRYPTION STANDARD (AES). (2001). https://csrc.nist.gov/files/pubs/fips/197/final/docs/fips-197.pdf

Image Steganography using Python - Javatpoint. (n.d.). Www.javatpoint.com. https://www.javatpoint.com/image-steganography-using-python

Wagner, L. (2020, February 6). AES-256 Cipher – Python Cryptography Examples. DEV Community. https://dev.to/wagslane/aes-256-cipher-python-cryptography-examples-10b2

Share Environment Variables Securely. (2024). Envshare.dev. https://envshare.dev/

CyberChef. (n.d.). CyberChef tool. Retrieved from https://cyberchef.org/

OnionShare. (n.d.). OnionShare. Retrieved from https://onionshare.org/

Upstash. (n.d.). Upstash Redis. Retrieved from https://upstash.com

GitHub. (n.d.). GitHub. Retrieved from https://github.com

Vercel. (n.d.). Next.js framework. Retrieved from https://nextjs.org/

Vercel. (n.d.). Vercel deployment platform. Retrieved from https://vercel.com/

Pallets Projects. (n.d.). Flask Python framework. Retrieved from https://palletsprojects.com/projects/flask/