**Requirement Specification**

**User Requirement (Natural Language)**

* A secure image system for MPU staffs to store students' examination paper images. The images shall be stored safely and protected from attacks from hackers.
* Users need to log-in before using the system.
* Users who want to download or upload an image owns a key (by default, the user knows). They can upload and download images with the key entered. The system shall automatically *encrypt* and *decrypt* the images with the key. If someone use wrong key to download an image, he or she shall not get the initial image.
* The system should only for storing images.

**System Requirement (Structured Specification)**

Secure image storage system based on AES

**Function** Store image in shared filesystem securely: encrypt and decrypt using Advance Encryption Standard.

**Description**

Stores encrypt images from being hacked and decrypt images automatically before download.

**Inputs** Raw images to be encrypted; Download request of a specific encrypted image.

**Outputs** Encrypted images /

**Destination** Cloud storage / Staffs' web browser.

**Action**

Before access the cloud storage, users should be authenticated. The passwords and accounts shall be encrypted and stored.

For upload, the system should check the size of images (less than 10MB per file) and the upload times of this user that day (each user shall not upload more than 100 times per day for system safety). After that, the system will prompt the user to enter 128-bit key for encryption. Once the key is entered and submitted, the system will automatically encrypt and upload images to the cloud storage.

For download, after images being selected, the system will prompt the user to enter 128-bit key for decryption. Once the key is entered and confirmed, the images (*ciphertext*) will be decrypted (shall be garbled file if the key is wrong) and downloaded to the user's web browser from the cloud storage.

**Requires**

Symmetric key user entered and the raw / encrypted images.

**Pre-condition**

The staffs have already possessed a key safely.

**Post-condition**

The stuffing bytes of the image should be removed after decryption

Side effects None.

**Mathematical Specifications**

Brief Formula of AES

**E (k, m) = C**

**D (k, C) = m**

E: Encrypt

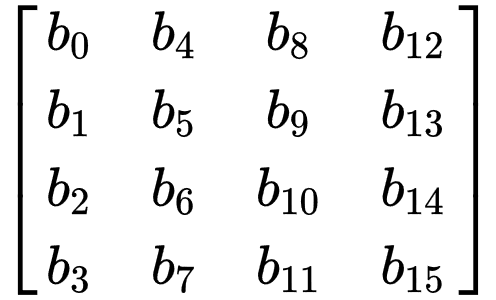
D: Decrypt

k: key

m: message

C: Ciphertext

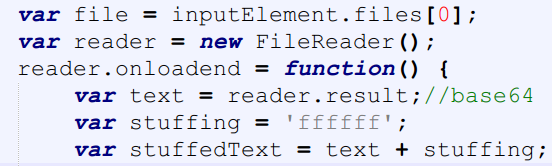
Input and Output Form of AES

****

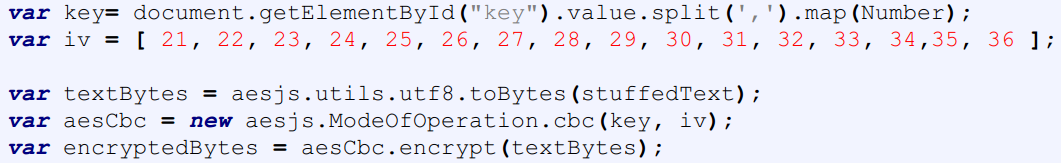
*4 by 4 2D array of 16 bytes*

**Demo**

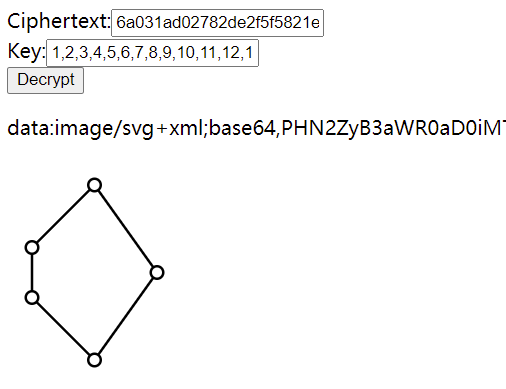
We’ve developed a simple JavaScript to demonstrate core functionality of AES system for encryption and decryption. We’ve reused existing AES algorithm (in CBC operation mode) and made configuration for image encryption and decryption.

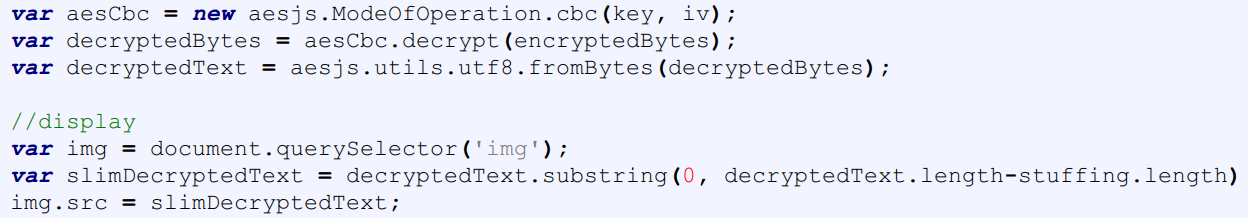


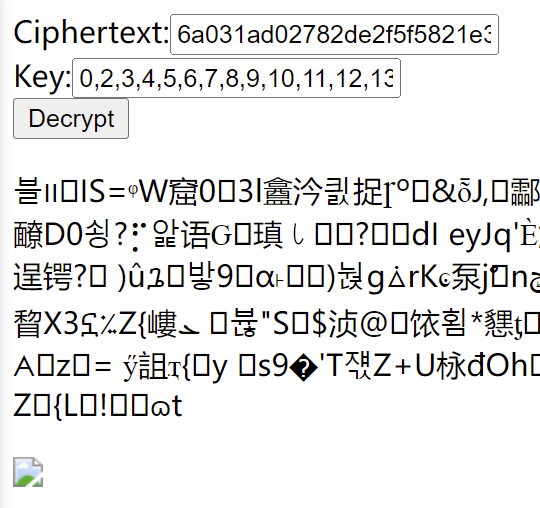
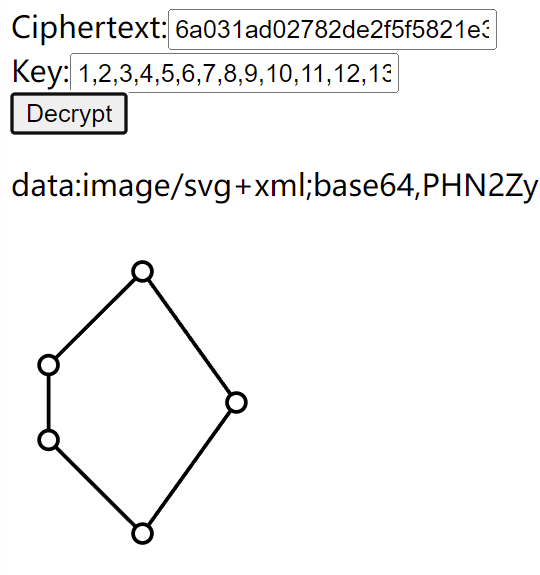
As for encryption, firstly, the image is converted to base64 string after being uploaded. And there should be stuffing bytes if the length of the string is not multiple of 16 because each block AES takes as input is a 4 by 4 2D array containing 16 bytes.



Key (128 bits, but in the form of in 16 decimal numbers separated by commas for user friendliness) is specified by the user, but the initial vector is set as default and not visible to the user for security. The output of encryption would also be a base64 string.



As for decryption, the user needs to enter the correct key otherwise a garbled file (or some messy code) would be the output. 



*Successfully Decrypted Image*

*Messy Code (wrong key to decrypt)*