**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, 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

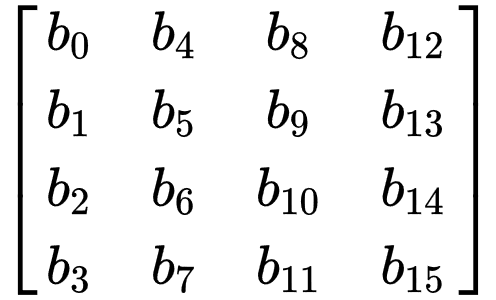
D: Decrypt

k: key

m: message

C: Ciphertext

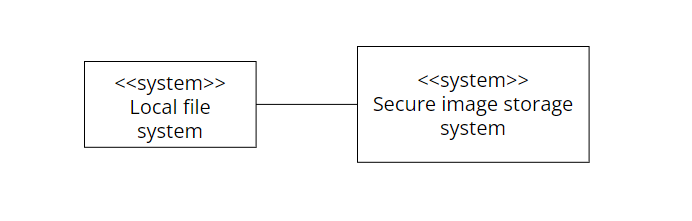
Input and Output Form of AES

****

*4 by 4 2D array of 16 bytes*

**System Model**

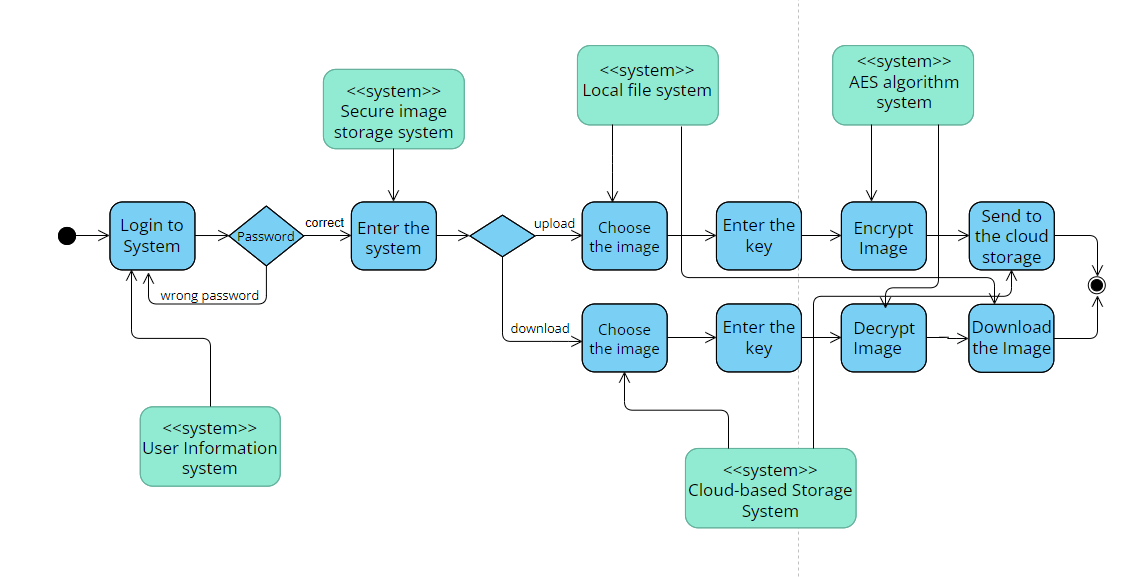
**Context model** are used to illustrate the operational context of a system, they show what lies outside the system boundaries. For our project, the picture below is the context of the Secure image storage system. Local file system is outside the system boundaries, so the Local file system is associated with the Secure image storage system. The local file system contains the files in your local storage.



Context of the Secure image storage system

For the Secure image storage system, it has three subsystems: User information system, AES algorithm system and Cloud-based storage system. For the User information system, when a user signup in the Secure image storage system, this system will record the user’s information. And also, this system will do authentication. For the AES algorithm system, it is used to provide algorithms for encryption and decryption. For the Cloud-based storage system, it is used to store and manage files uploaded by users.

**Process models** reveal how the system being developed. The picture below is the process model of encryption and decryption.

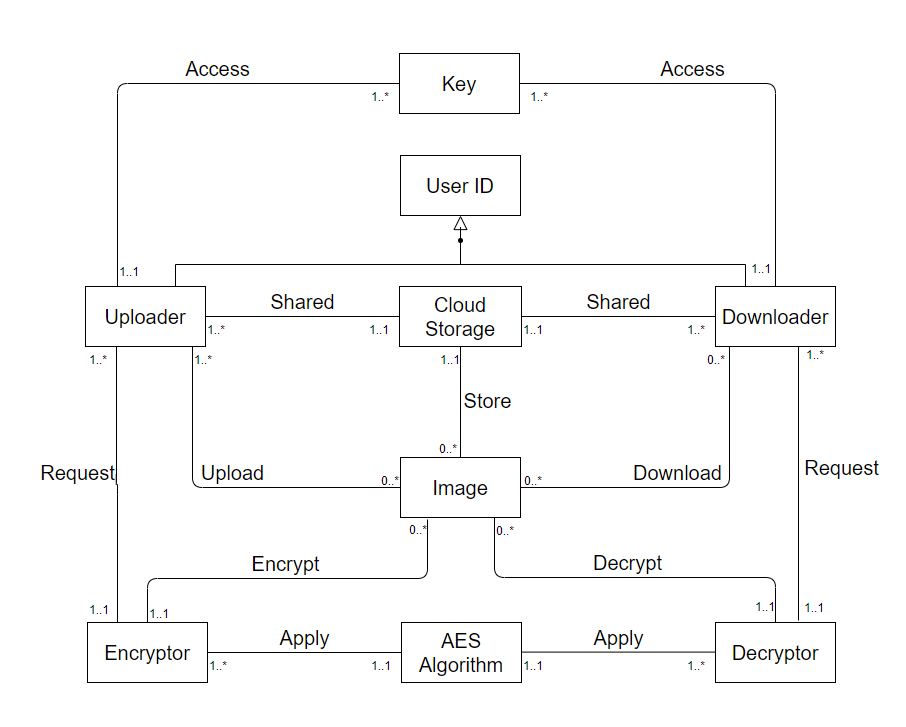


Activity diagram of encryption and decryption

Firstly, users need to enter correct password to log in to the Secure image storage system. Next, if the users want to upload the images, they must choose the file from the local file system, and then enter the key to encrypt the images, after encryption, the images will be sent to the cloud storage. If the users want to download the images, they should choose the files in the cloud storage, and then enter the key to decrypt the images. After decryption, you can download the images.

**Structural models** of software display the organization of a system in terms of the components that make up that system and their relationships.

**Class diagrams** are used when developing an object-oriented system model to show the classes in a system and the associations between these classes. The picture below is the class diagram of Secure image storage system.

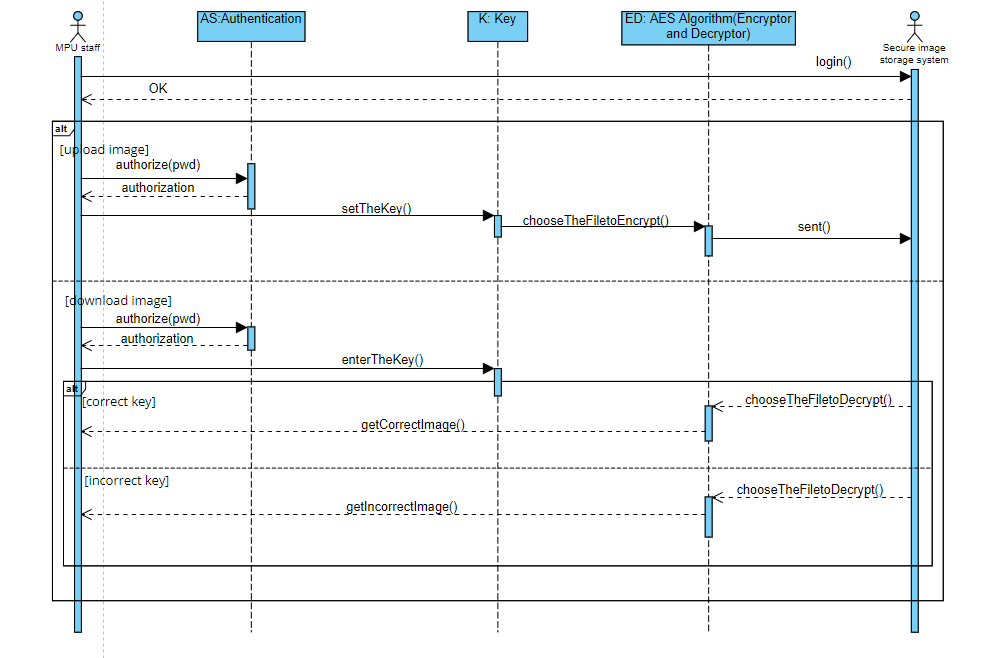


Class diagram of Secure image storage system

1. In class diagram, there are two kinds of users: uploader and downloader. They all share the same cloud storage.
2. The uploader can upload 0 to many images, because if the users just login and don’t upload any files. For the images in the cloud storage, they must be uploaded by 1 to many uploaders. And for downloaders, they can download 0 to many images, but one image which in the storage can be downloaded by 0 to many downloaders. Because one user maybe just uploads the image, and doesn’t download it.
3. Each user can assess(评估评定？) 1 to many keys if they want to get files from different uploaders. And one key can be accessed by one and only one uploader or downloader.
4. The encryptor and decryptor can be requested by 1 to many users, and encrypt or decrypt 0 to many images, because the file format is not image or no users request, it can be 0 image.
5. Both encryptor and decryptor need to apply the AES algorithm.

**Interaction model**

**Sequence diagrams** are part of the UML and are used to model the interactions between the actors and the objects within a system. The picture below is the sequence diagram of Secure image storage system.

****

Sequence diagram of Secure image storage system

In this sequence diagram, there are 2 actors and 3 objects. The 2 actors are MPU staff and Secure image storage system. There are 3 objects in the Secure image storage system. They are Authentication, Key and AES algorithm (Encryptor and Decryptor).

Firstly, MPU staff need to login to the system, and the system will return the OK message.

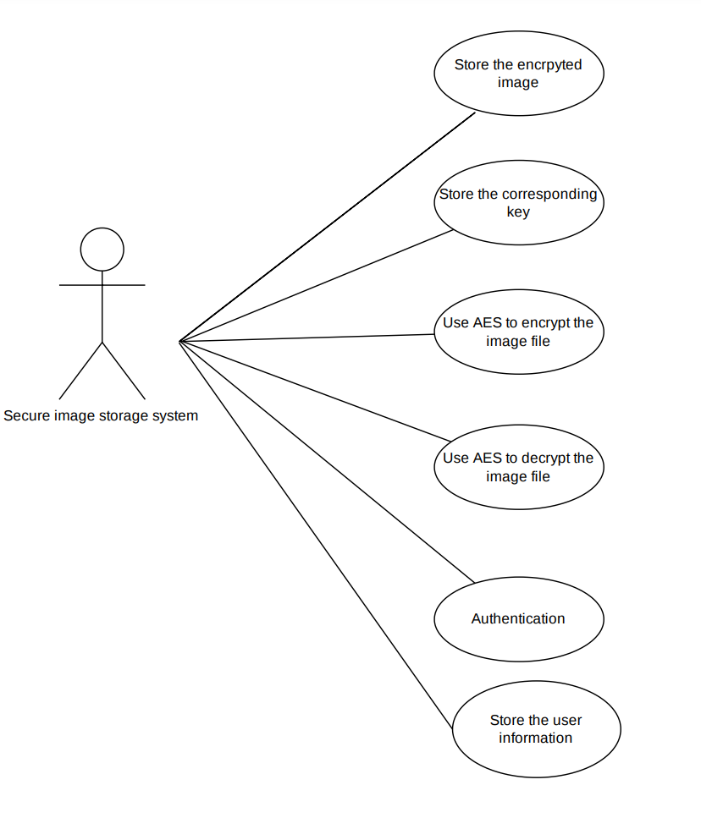
If a staff want to upload the images, the staff should first do the authentication. After that, he/she can set the key, and choose the files to encrypt, the images will be sent to the Secure image storage system.

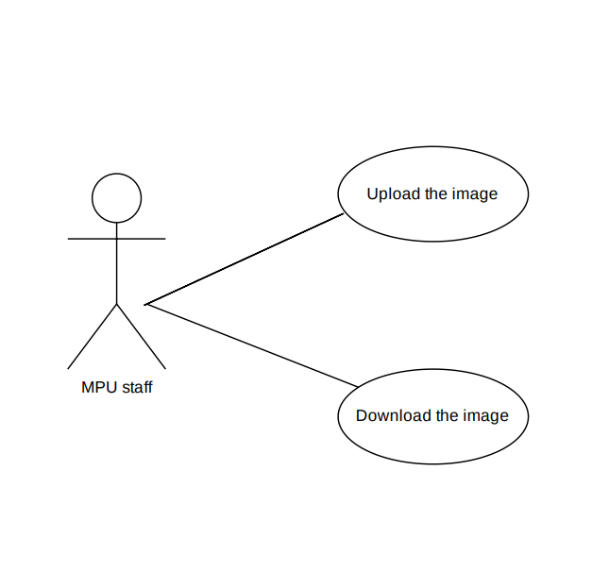
If a staff want to download the images, the staff should first do the authentication. After that, he/she need to enter the key, and choose the files from the Secure image storage system. If the key is correct, he/she can download the images correctly. If he/she enter the wrong key, then only the mess code will be given.

**Use case diagrams**

There are 2 actors in this system.

The first one is MPU staff. Because they use this system to store the examination paper pictures. The MPU staff can upload and download the image.





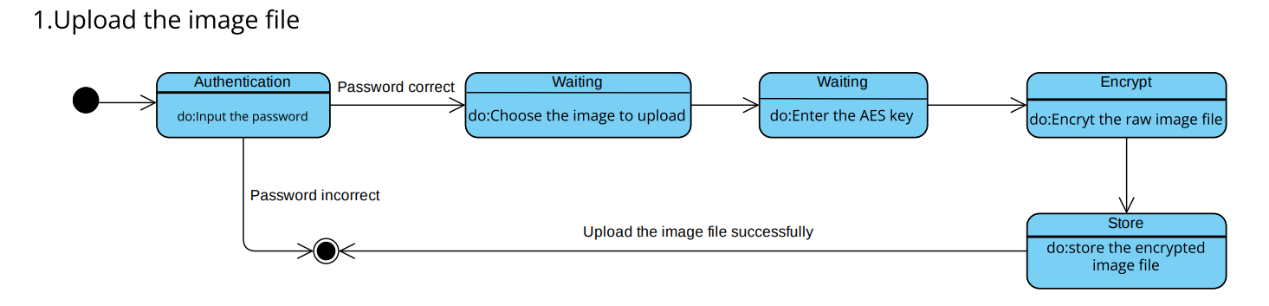
The second actor is the system itself (secure image storage system). The first role the system can play is authentication. Because in the requirement we define only authenticated users can access this system. And it also means that the user needs to input the username and the corresponding password. So, this system also stores the users’ information. The core of this system is about using AES algorithm to encrypt and decrypt the image file. When uploading the image file, the user also needs to enter the key to encrypt the image file. To download the file, the user also needs to enter the key. So, this system not only stores the encrypted image file but also the corresponding key value.

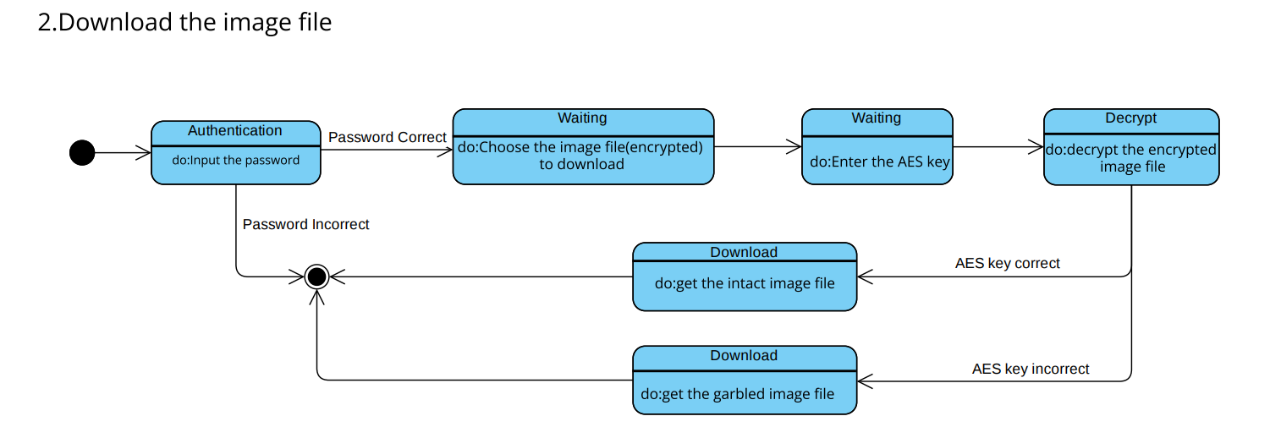
**State machine diagram**

State machine diagram shows the behaviors of the system in response to external and internal events.

Here we divided this model into 2 parts based on user’s actions.

At the beginning, the user needs to get through the authentication. When they input the correct password, they will enter the system. After they choose the image to upload and enter the key whose format is legal, the system will automatically encrypt the raw image file. In the end, it will store this encrypted image file.

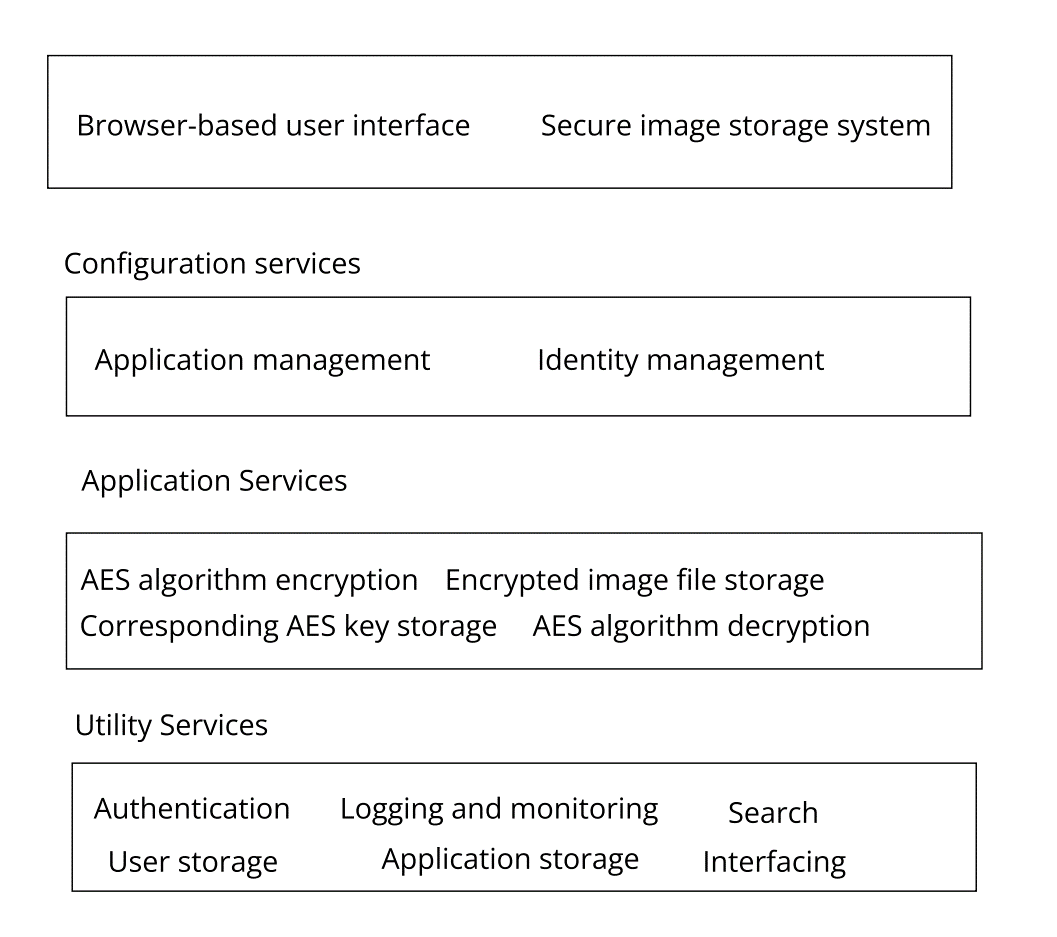




Because download and upload are symmetrical. These first few steps are the same. The major difference is concentrated on the decryption. When the user inputs the wrong key, he will get the garbled image file without being reminded that he made a mistake. Imagine the situation that one malicious user gets through the authentication by accident, he tries inputting the key to get the examination paper image. Without reminding, he will get the gabled file. Maybe he thinks he gets the result and exits the system. This behavior of the system benefits the security a lot.

**System architecture**

This system can be divided into 3 subsystems based on the functions (cloud-based storage system, AES algorithm system, user information storage system). And as we know, layered architecture organizes the system into a set of layers (or abstract machines) each of which provide a set of services. The architecture is also changeable and portable. Because if its interface is unchanged, a new layer with extended functionality can replace an existing layer without changing other parts of the system. So, we choose the layered architecture for this system. This system is divided into 4 layers. The pattern is shown below.



1. The first layer is user interface. This system will have a browser-based user interface. This will give the user the entry to this system and make a connection between these two.

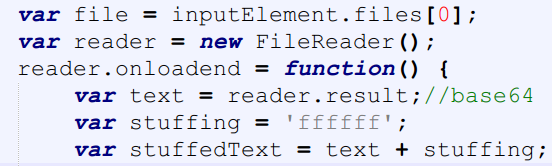
2.The second layer is configuration services. This layer consists of 2 parts: application management and identity management. Application management is about managing and monitoring the application services to make sure them work normally and correctly. Identity management is about managing the user’s identity such as sign up, change the password and so on.

3.The third layer is application services. It contains all the functions that the system provides to the users.

4.The lowest layer is utility services. This is the deepest of this system. It supports above layers.

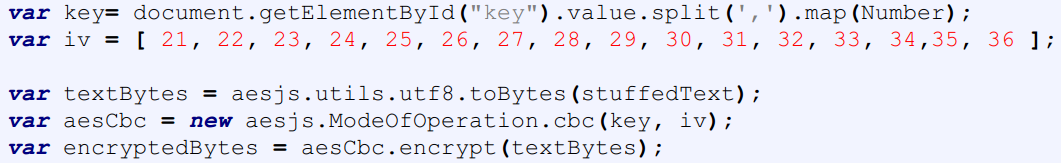
**Demo**

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.

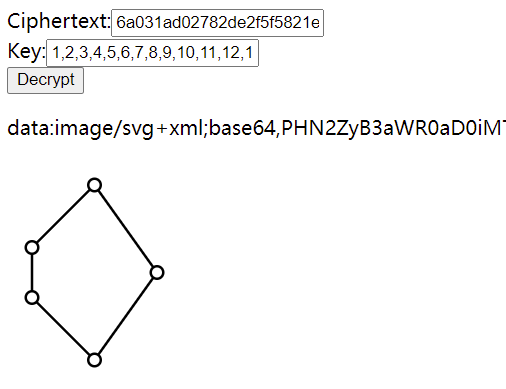


Encryption

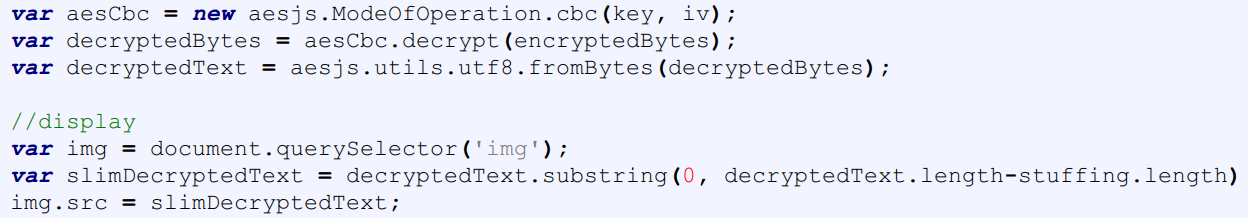
As for encryption, firstly, the image is converted to base64 string after being uploaded. And there should be stuffing bytes if the length of the string is not multiple of 16 because each block AES takes as input is a 4 by 4 2D array containing 16 bytes.

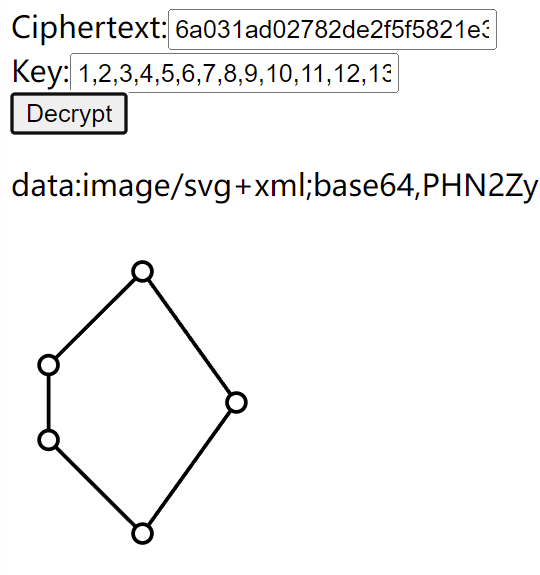
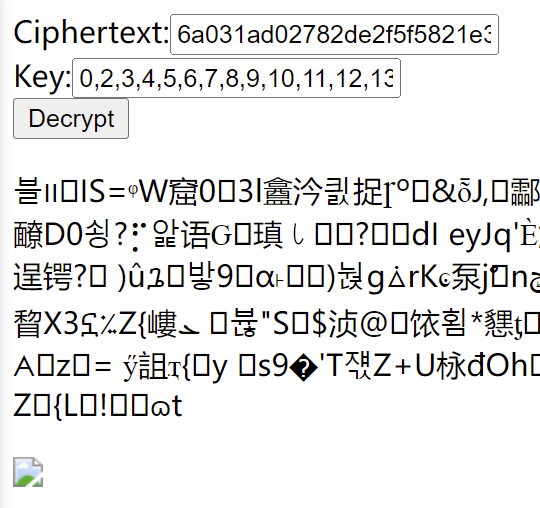


Key (128 bits, but in the form of in 16 decimal numbers separated by commas for user friendliness) is specified by the user, but the initial vector is set as default and not visible to the user for security. The output of encryption would also be a base64 string.



Decryption

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

**References**

[1] <https://github.com/ricmoo/aes-js>

[2] <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard>