PLATEGUARD

Heavy Machinery

Nathan Burgess, Michael Nutt
nathanburgess@my.unt.edu, michaelnutt2@my.unt.edu

Version 1.0
October 24th , 2018

Specifications Document

Version 1.0

Contents

A. Project Overview

Our system is a security system that has been built to recognize plates in saved videos and encrypt the plate area with the license plate as the key to allow for authorized decryption. The system is comprised of four programs that make up the PlateGuard system, ImageEncrypt, ImageDecrypt, PlateEncrypt, PlateDecrypt. The ImageEncrypt captures video from an embedded system and encrypts the image using Stream Cipher and then transmits it to a server. ImageDecrypt receives the encrypted image and decrypts it, then transfers it to PlateEncrypt. PlateEncrypt isolates the plate area of the image and encrypts it using the plate number. PlateDecrypt reads in the saved encrypted video and decrypts the plate area.

B. Current Problems and Proposed Solutions

With existing technology individual privacy is at risk through the mass gathering of unsecured footage of license plates of cars that pass red light and other traffic cameras on a daily basis. With no encryption currently in place that data could easily be gathered by malicious actors and searched by computers to identify the owners of the cars recorded. Our goal is to make this task harder for malicious actors while still keeping the data usable by authorized parties by identifying the license plate area for each car in the stored videos and encrypting those areas using the license plate as the key.

C. Requirements

1. Functional Requirements

ID	Functional Requirements	Team Member Responsible
1	Read in video stream/saved video	Nathan/Michael
2	Encrypt whole video and transmit to server	Nathan/Michael
3	Decrypt video for plate recognition	Nathan/Michael
4	Recognize license plates	Nathan/Michael
5	Extract plate number	Nathan/Michael
6	Save coordinate and plate meta data to the image	Nathan/Michael
7	Encrypt license plates	Nathan/Michael
8	Save encrypted image as .png	Michael/Nathan
9	Read in encrypted image	Michael/Nathan
10	Decrypt target plate area	Michael/Nathan
11	Search for target plate number	Michael/Nathan
12	Minimum fps of video playback at 15 fps	Michael/Nathan

2. Non-Functional Requirements

ID	Non-Functional Requirements	Team Member Responsible
1	Multi-threading	Michael/Nathan
2	Graphical user interface	Michael/Nathan
3	Track license plate area to reduce image recognition calls	Michael/Nathan
4	Separated Red Light System	Nathan/Michael
5	Separated Security Camera System	Nathan/Michael
6	Combine exported .png files into video	Nathan/Michael
7	Create Installer package	Nathan/Michael
8	Port to C++ for optimization	Nathan/Michael

3. Constraints

a. *Storage space*
   - *file output will be individual .png files, this will likely take up a lot of storage space quickly.*

b. *Processing time*
   - *Image recognition is processor intensive and on embedded systems would take an extended period of time, this will be mitigated by transferring the files to the server prior to license plate encryption*

c. *Embedded CPU clock rate*
   - *Processor of the embedded system will need to be able to efficiently handle reading in video, encrypting the video and sending it to the server*

d. *Image Recognition*
   - *Image recognition is not perfect, there is a chance of mis-recognition that will need to be accounted for during encryption of the plate area.*

e. *Wi-Fi Range*
   - *The embedded system will be placed remote from the server, we will need a method of transmitting the data from the system to the server to allow image recognition and encryption.*

D. Specifications

1. Functional Requirements Specifications

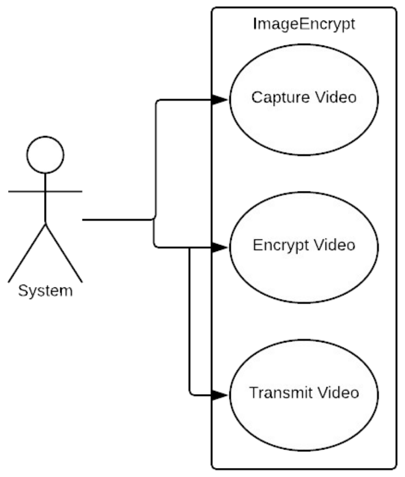
ID	Functional Requirement Specification	Team Member Responsible
1, 8, 9	OpenCV v3	Nathan/Michael
2, 3, 6, 7, 10, 11, 12	Propriety code Python 3	Nathan/Michael
2	Wifi/ethernet connection	Nathan/Michael
4, 5	OpenALPR v2.3.0	Nathan/Michael
1-12	Ubuntu 16.04	Nathan/Michael
1-12	Raspberry Pi 3 model B+	Nathan/Michael
1	USB Camera Module	Nathan/Michael
1-12	Server Software: Ubuntu 16.04 Hardware: i5 CPU, 8 GB RAM, 500 GB HDD	Nathan/Michael

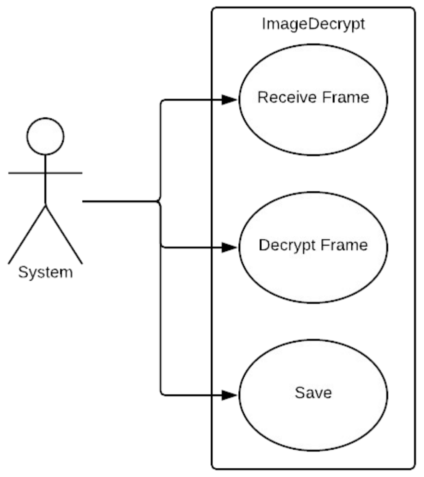
2. Non-Functional Requirements Specifications

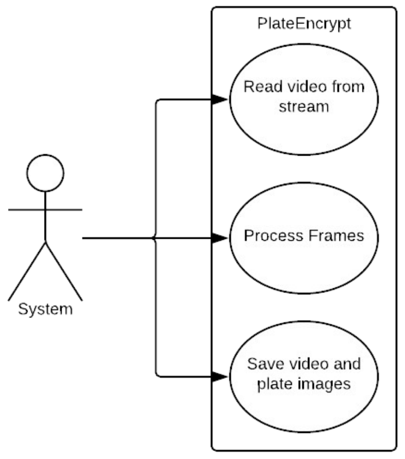
ID	Non-Functional Requirement Specification	Team Member Responsible
3	OpenCV v3	Nathan/Michael
4, 5	Raspberry Pi 3 model B+	Nathan/Michael
4	IR Sensor module	Nathan/Michael
1, 2, 6, 7	Python 3	Nathan/Michael
8	C++	Nathan/Michael

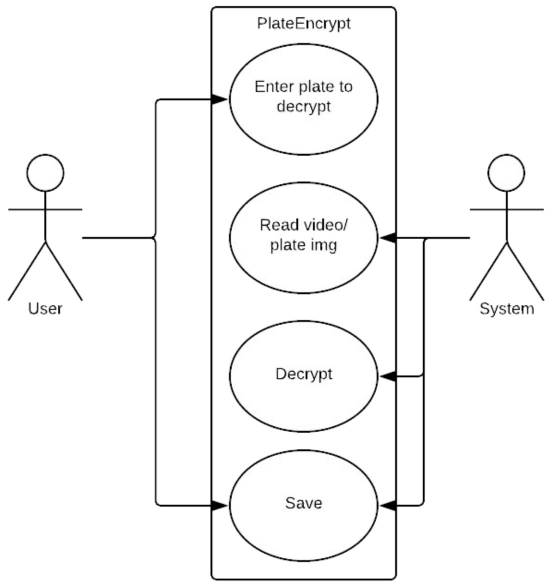
E. Design

1. Use Case

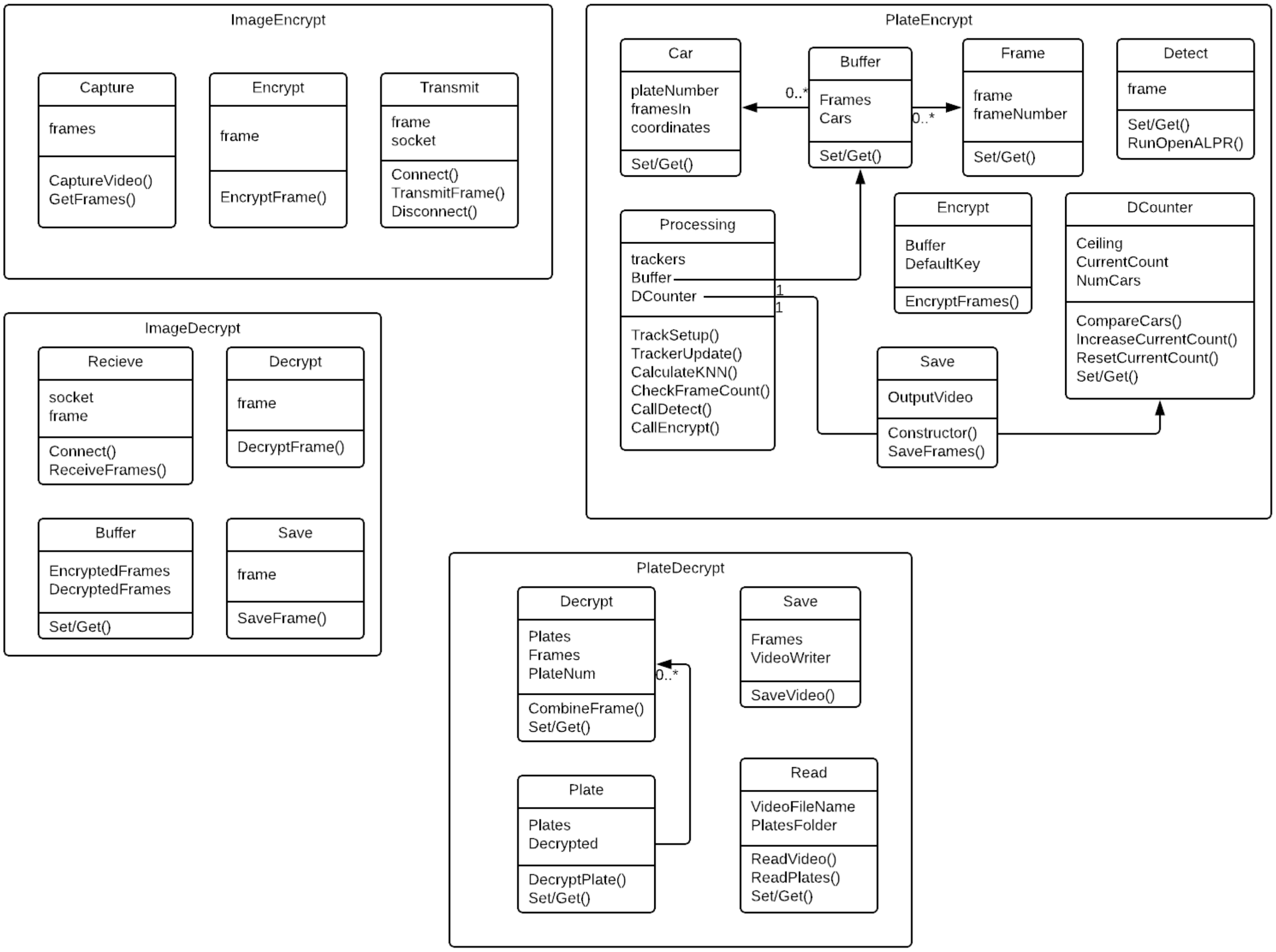




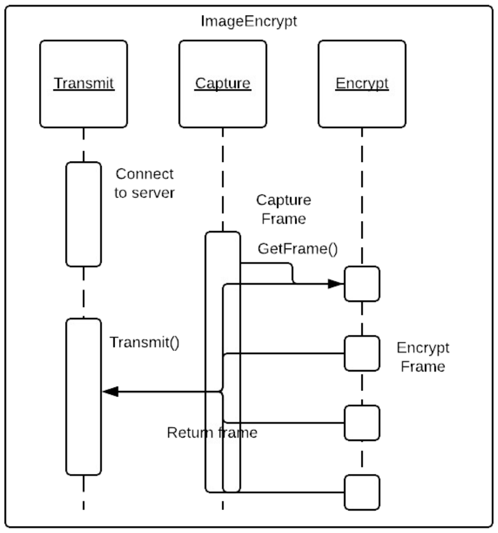


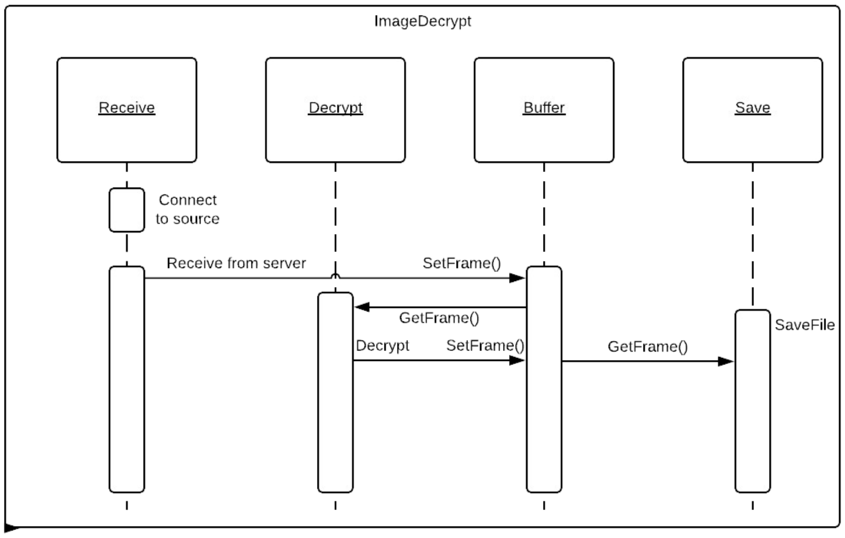


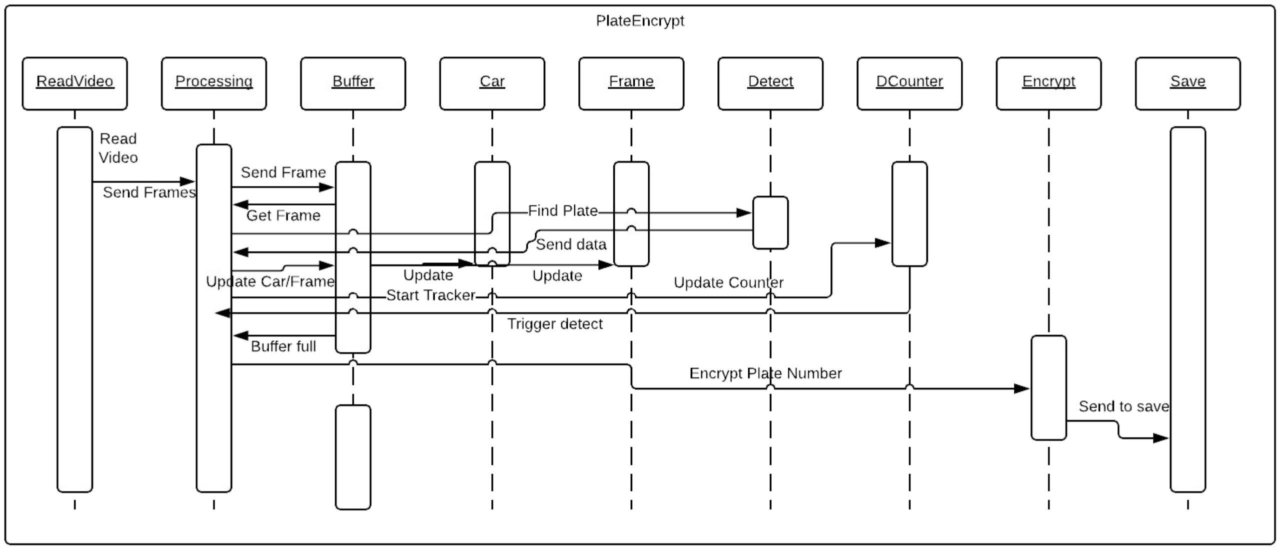
2. Class Diagrams

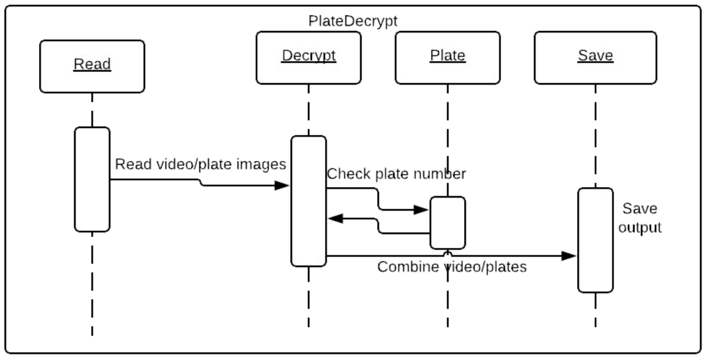


3. Sequence Diagrams

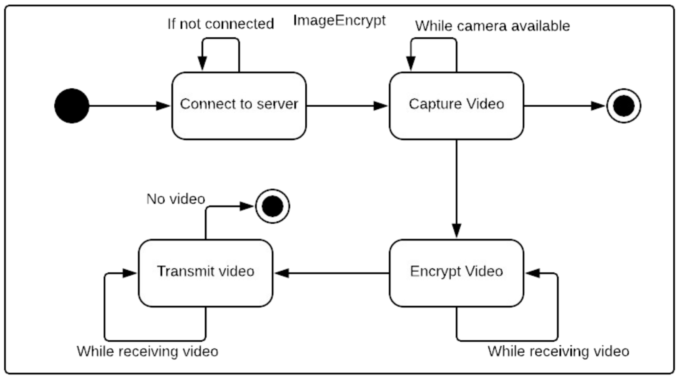


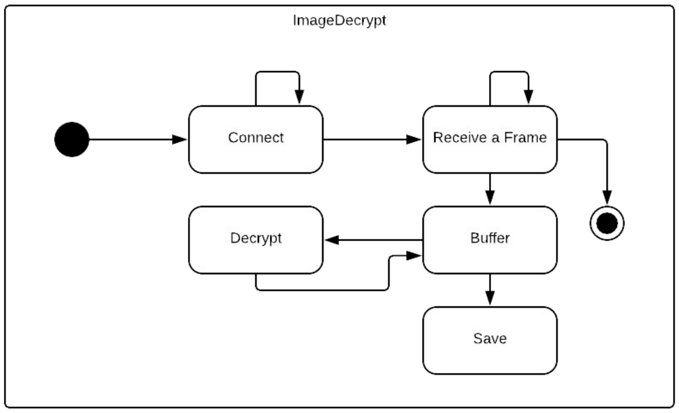


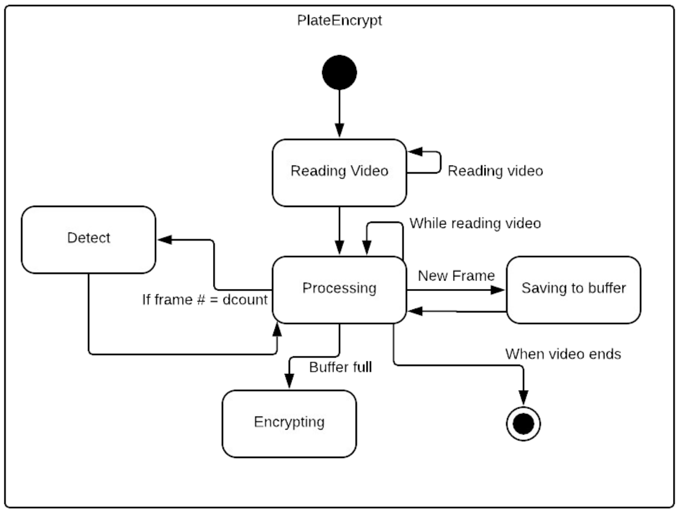


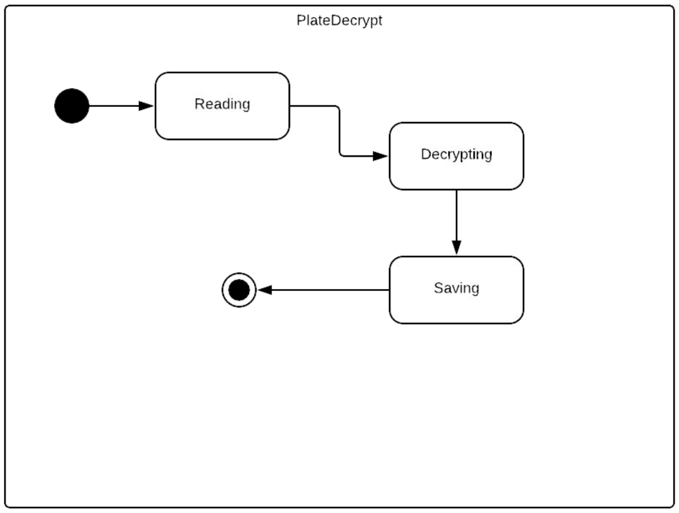


4. State Diagrams

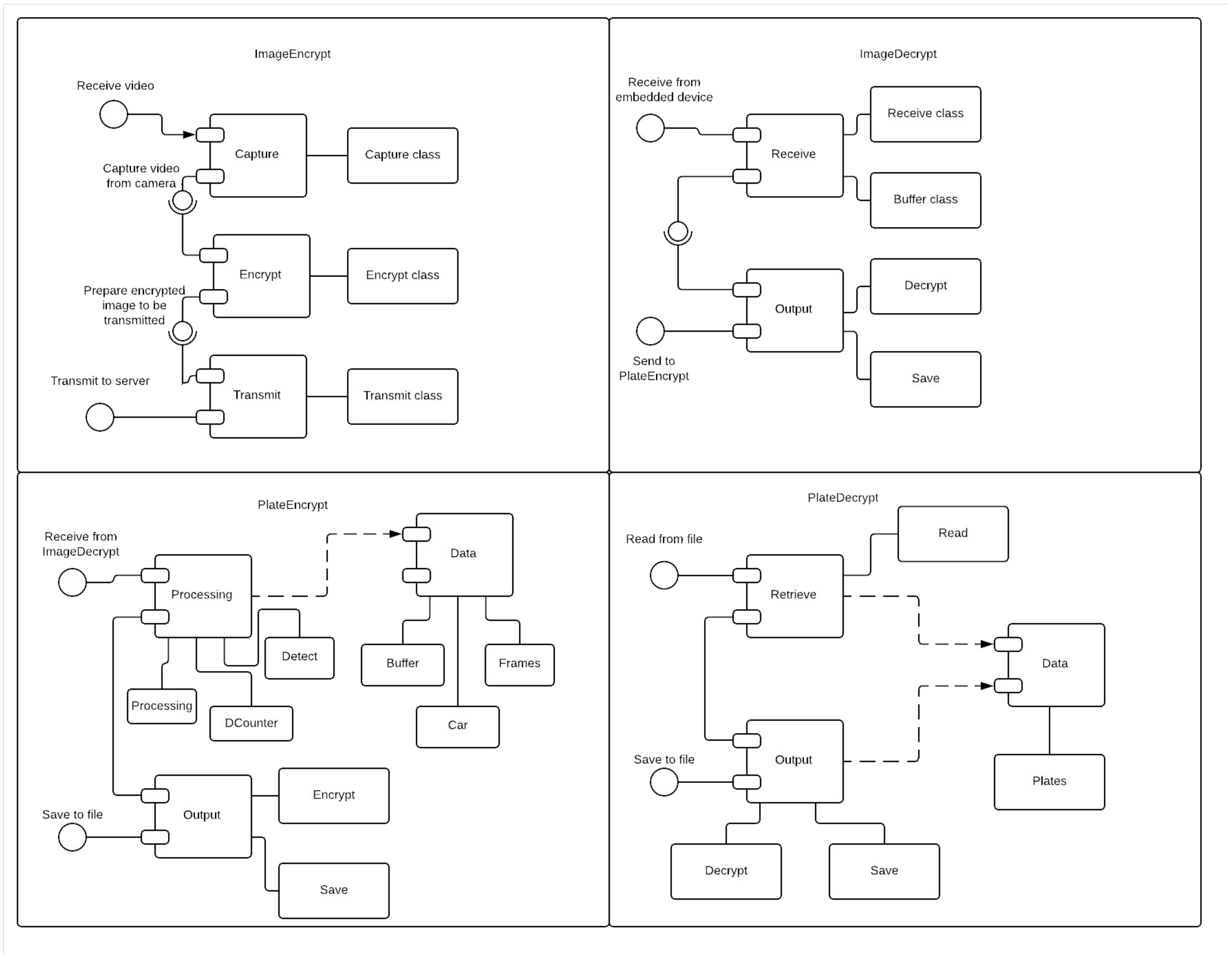




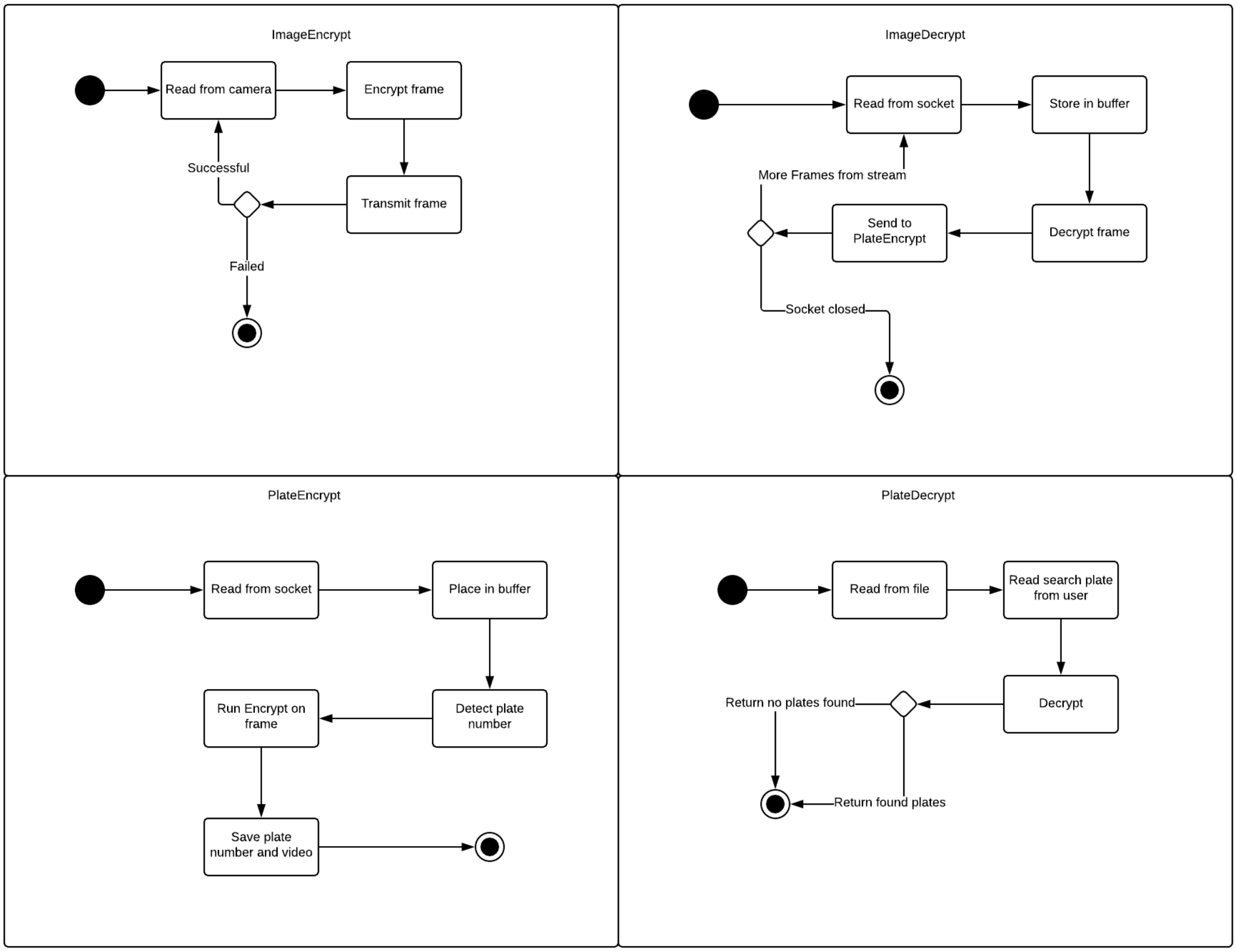




5. Component Diagrams



6. Activity Diagrams


ImageEncrypt
Read from camera
Encrypt frame
Transmit frame
Successful
Failed
ImageDecrypt
Read from socket
Store in buffer
More Frames from stream
Send to PlateEncrypt
Decrypt frame
Socket closed
PlateEncrypt
Read from socket
Place in buffer
Run Encrypt on frame
Detect plate number
Save plate number and video
PlateDecrypt
Read from file
Read search plate from user
Return no plates found
Decrypt
Return found plates