Applied Cryptography ST2504 CA2

Class: DISM/1B06

Full Name	Student Number
Koh Kai En	P2104175
Sathiah Elamaran	P2129017
Lee Pin	P2128610

Submitted to: Mr Casey How

Submission Date: 13th Feb

Table of contents

Introduction

Singapore is a first world country, one that is extremely advanced in terms of its technology. Using these improvements in technology, the team has developed an automated severance system, which involves several thousands of cameras across different parts of Singapore where crimes frequently occur. The cameras use the current public 4G network or Wireless@SG to connect to the internet.

The cameras installed are placed in different locations in the public and are vulnerable to risks. Such risks include the camera being physically stolen with the public and private keys and program extracted after accessing the stolen camera. Hence, the team will take necessary steps to ensure that in the event of a stolen camera such as encrypting the private key.

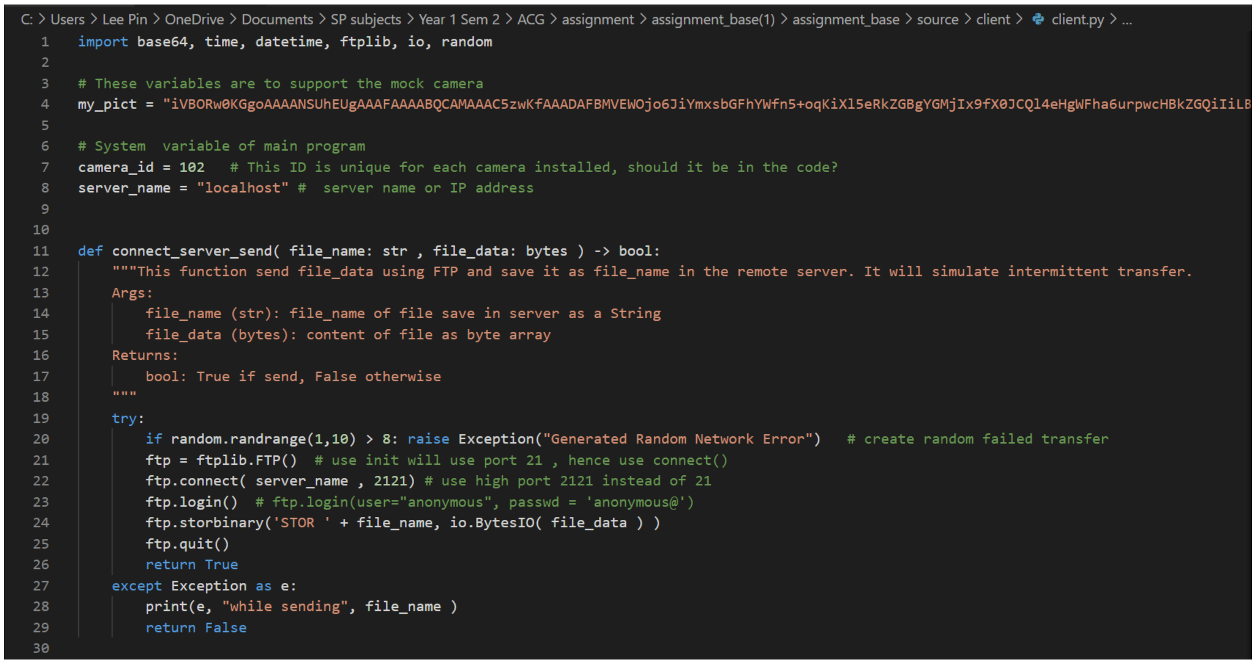
As for the camera images, after an image is taken, it is immediately sent to the server. Because such images can be used in court during criminal cases, the images must ensure authenticity and non-repudiation. Additionally, the confidentiality, integrity and availability of the images must be valid. Therefore, the report will show one how the system can be upgraded in order for securing the images during its exchange between the client camera and the server.

Illustration of current system

In the system, the client.py file is the camera's program and code which would connect to the server via the server.py code.

The current client.py and server.py is a very basic system which allows the transfer of pictures without the use of any encryption, hashing and digital signatures.

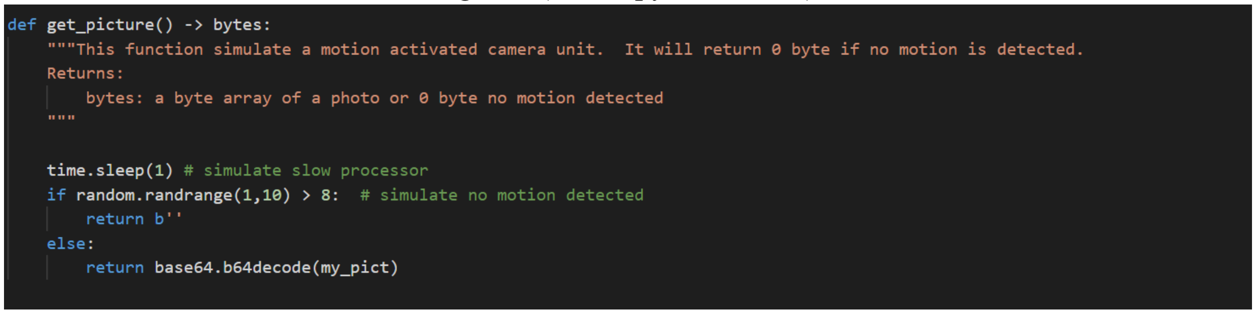
Client.py



C: > Users > Lee Pin > OneDrive > Documents > SP subjects > Year 1 Sem 2 > ACG > assignment > assignment_base(1) > assignment_base > source > client > client.py > ...

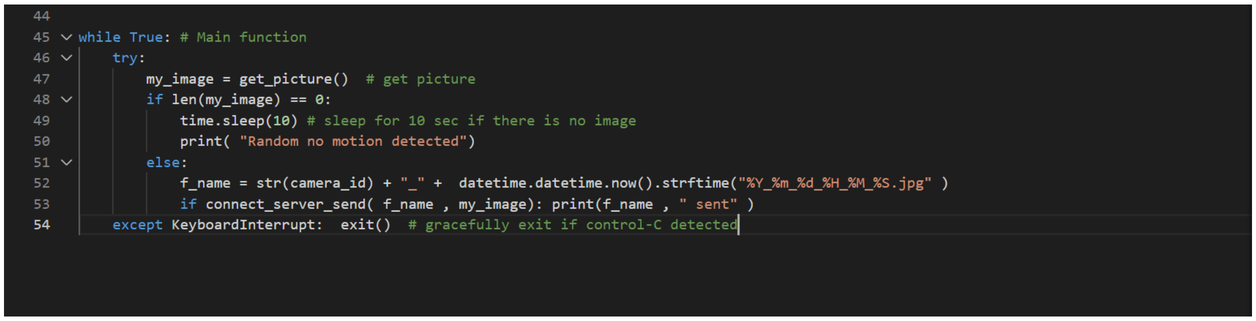
```python
import base64, time, datetime, ftplib, io, random

# These variables are to support the mock camera
my_pict = "iVBORw0KGgoAAAANSUhEUgAAAFAAAABQCAMAAAC5zwKfAAADAFBMVEWOjo6JiYmxsbGFhYWfn5+oqKiXl5eRkZGBgYGMjIx9fX0JCQl4eHgWFha6urpwcHBkZGQiIiLB

# System  variable of main program
camera_id = 102   # This ID is unique for each camera installed, should it be in the code?
server_name = "localhost" #  server name or IP address


def connect_server_send( file_name: str , file_data: bytes ) -> bool:
    """This function send file_data using FTP and save it as file_name in the remote server. It will simulate intermittent transfer.
    Args:
        file_name (str): file_name of file save in server as a String
        file_data (bytes): content of file as byte array
    Returns:
        bool: True if send, False otherwise
    """
    try:
        if random.randrange(1,10) > 8: raise Exception("Generated Random Network Error")   # create random failed transfer
        ftp = ftplib.FTP()  # use init will use port 21 , hence use connect()
        ftp.connect( server_name , 2121) # use high port 2121 instead of 21
        ftp.login()  # ftp.login(user="anonymous", passwd = 'anonymous@')
        ftp.storbinary('STOR ' + file_name, io.BytesIO( file_data ) )
        ftp.quit()
        return True
    except Exception as e:
        print(e, "while sending", file_name )
        return False

```

Figure 1(Client.py function 1)



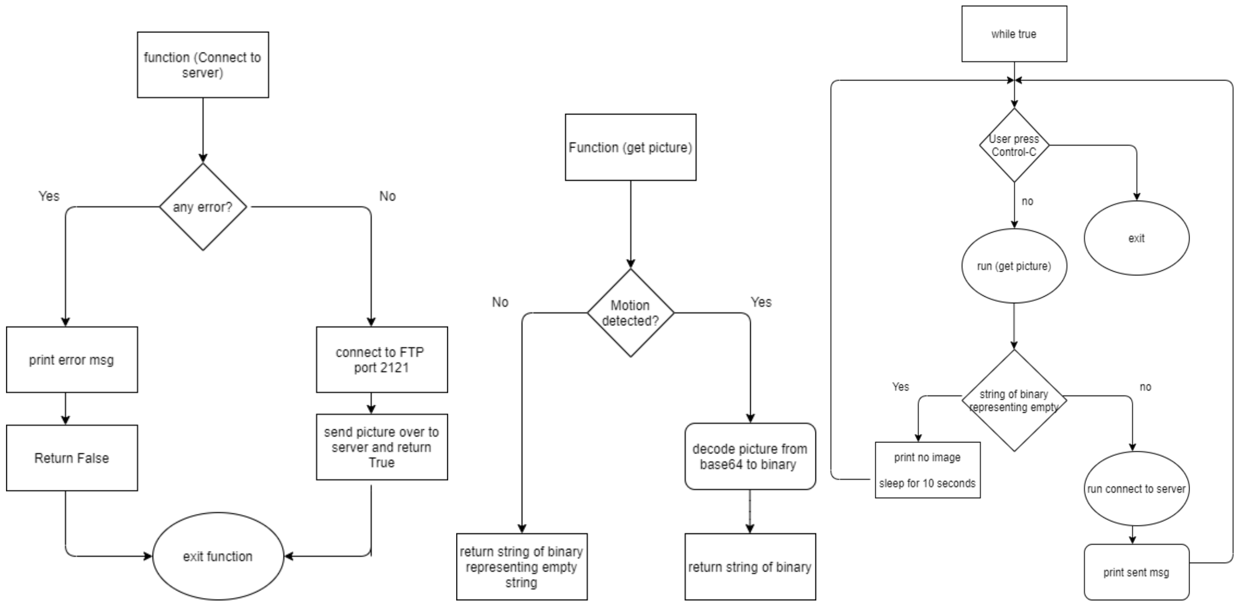
```python
def get_picture() -> bytes:
    """This function simulate a motion activated camera unit.  It will return 0 byte if no motion is detected.
    Returns:
        bytes: a byte array of a photo or 0 byte no motion detected
    """

    time.sleep(1) # simulate slow processor
    if random.randrange(1,10) > 8:  # simulate no motion detected
        return b''
    else:
        return base64.b64decode(my_pict)
```

Figure 2(Client.py function 2)



```python

while True: # Main function
    try:
        my_image = get_picture()  # get picture
        if len(my_image) == 0:
            time.sleep(10) # sleep for 10 sec if there is no image
            print( "Random no motion detected")
        else:
            f_name = str(camera_id) + "_" +  datetime.datetime.now().strftime("%Y_%m_%d_%H_%M_%S.jpg" )
            if connect_server_send( f_name , my_image): print(f_name , " sent" )
    except KeyboardInterrupt:  exit()  # gracefully exit if control-C detected
```

Figure 3(Client.py main code)



Function 1(connect to server):

The first function of the client is to connect to the server using FTP, file transfer protocol, and to connect to port 2121. The function would use an anonymous login to the FTP service, and then send over the file(Image) as a string of bytes. There is no use of encryption and hashing to protect the images as it is being transferred over FTP, hence allowing hackers to use wireshark to capture the files(Images) sent over and view it immediately.

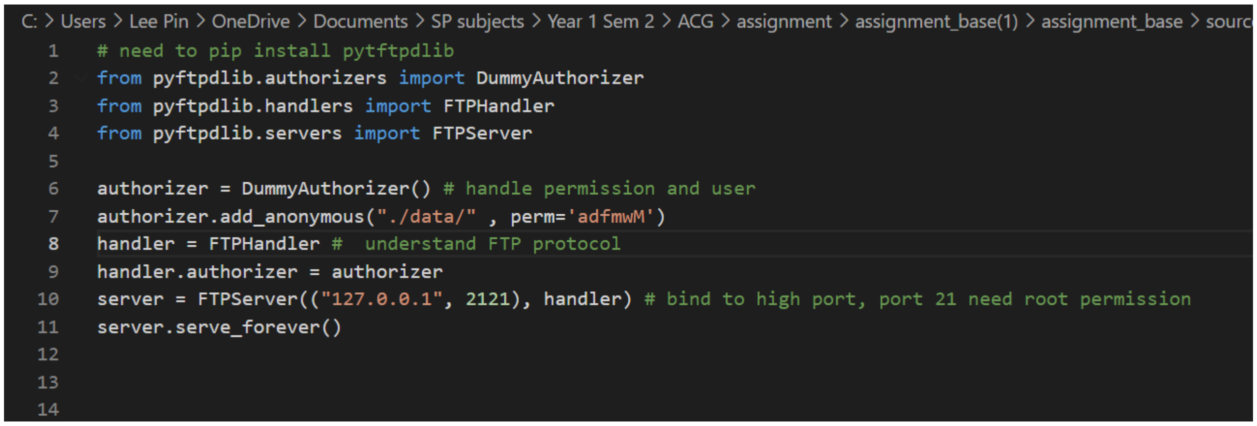
Function 2(Get picture):

The second function is used to stimulate the obtaining of pictures when motion is detected along the surveillance area. The picture taken is in base64 and it is decoded in the function into a string of bytes that is suitable for transmission in FTP. If no motion is detected, a string of bytes representing an empty string is returned.

Main Code Outline:

While running, the client.py program would run forever, getting images continuously using the function 2(Get picture) and sending it to the server using the function 1(connect to server) by connecting to server using port 2121 FTP service. If no motion is detected, the code will receive the string of bytes representing the empty string. It will then sleep for 10 seconds and print a no motion message, only running again after the 10 seconds to check for new motion. Upon the user using the keyboard interruption cue, Control-C, on his or her computer, the client.py would stop running and would exit, stopping any further pictures from being sent to the server.

Server.py



C: > Users > Lee Pin > OneDrive > Documents > SP subjects > Year 1 Sem 2 > ACG > assignment > assignment_base(1) > assignment_base > sourc

```python
# need to pip install pytftpdlib
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer

authorizer = DummyAuthorizer() # handle permission and user
authorizer.add_anonymous("./data/" , perm='adfmwM')
handler = FTPHandler #  understand FTP protocol
handler.authorizer = authorizer
server = FTPServer(("127.0.0.1", 2121), handler) # bind to high port, port 21 need root permission
server.serve_forever()
```

Figure 4(Server.py main code)

Main Code Outline:

In a realistic situation, the server is run forever and continuously to receive images from all cameras at any time. The server would run the FTP service at port 2121, and allow for multiple cameras to connect to port 2121 to send the images as strings of bytes over and will be stored in the database.

Assumptions made on implementation

1) Assumption 1 is that a certificate authority is not present to sign the public key of the server program and the client program. This would result in hackers being able to pose as the client program to start an exchange of files with the server as there is no need to validate whether the public key of the client is valid and from the official client. Hence, the hackers would be able to start a public-key exchange with the server, and send over infectious files to the server which could potentially lead to virus installation.

2) Assumption 2 is that images are automatically taken upon motion sensors activated.

Attack Scenarios

An attacker could be a criminal and the security cameras could be in the vicinity of the crime scene where the attacker carried out a criminal activity, thus the cameras could have captured valuable information on the crime and who did it. Therefore the attacker could be trying to access the information and data stored on the camera in order to delete the information or alter the pictures and change the pictures with a picture of someone else, this would allow them to get off the crime scotch free.

Risks regarding MITM Interception

Attacker tries to delete or change evidence.
- The attacker can intercept the transmission between the camera and the server. This allows the attacker to have access to the information being transmitted and they can alter, delete or have useful insight to what is being transmitted. This will happen during the transmission of the image to the server.

Impact - The integrity and the confidentiality of the information being sent from the server is compromised. Therefore if information is used as evidence, it may not be liable.

Capabilities - Need to be skilled in hacking in order to know how to intercept transmission between the camera and the server and to alter or delete the information being transmitted.

Countermeasure - Use Certificate authority in order to ensure that the information received is from the actual camera and that it is not sent from elsewhere. Moreover the usage of digital signature ensures information authentication. Not only that, implementing an encryption mechanism also ensures that the data is encrypted.

Anonymous ftp Login

Attacker logs in to the server using anonymous ftp login.
- If anonymous ftp login is enabled the attacker will then login to the server and they will have access to all the pictures and information that was sent by the cameras as it does not require authentication and the user id will be "anonymous" and the password will be a default password. This will allow them to alter or delete certain pictures and information without the need to have any authority.

Impact - The integrity and the confidentiality of any information stored in the server is compromised.

Capabilities - Need to have experience is using ftp and need to know how to login using anonymous ftp login

Countermeasure - Restrict anonymous login and instead user a user and password to login

Brute Force Login

The Attacker can login to the server using brute force and eventually get in.
- The attacker could brute force the login and gain access to the server and can view all the pictures and information sent from the camera.This will allow them to alter or delete certain pictures and information without the need to have any authority.

Impact - The integrity and the confidentiality of any information stored in the server is compromised.
Capabilities - Need a lot of time and need to know how to use or make a programme that can brute force a login.
Countermeasure - Implement Fail2ban so that if the password entered is wrong too many times, it bans users and doesn't allow them to try to login again.

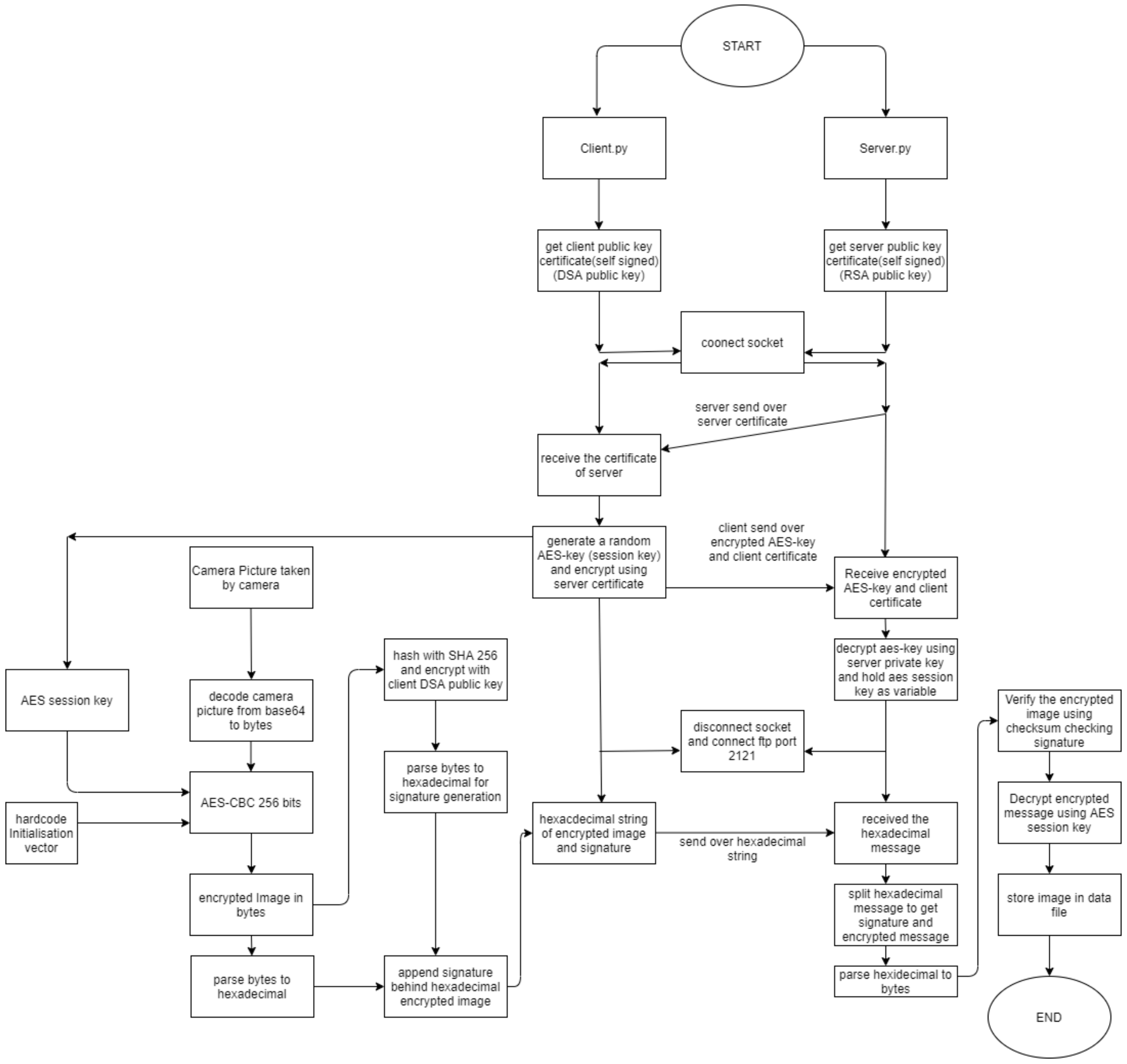
Stolen Camera

Attacker steals the camera to gain insight into the code of the camera.
- An attacker could steal the camera in order to figure out how the program works. This would allow the attacker to gain insight on how the encryption of the images works and it would also allow the attacker to develop any programs to attack the vulnerabilities of the camera. If there are any images stored on the camera they will also have access to those images.

Impact - The integrity of the software is compromised as programs that take advantage of the vulnerabilities of the software can be developed. This may cause more attacks to take place.
Capabilities - In order to steal the camera, the attacker does not require much skill, however to interpret the code the attacker needs to have knowledge in coding.
Countermeasure - Add additional physical security to ensure that theft does not occur. Have an alarm for when the camera senses that someone is attempting to move it out of place. The camera can also send an alert to the authorities.

Proposed solution

Illustration of new system


START
Client.py
Server.py
get client public key certificate(self signed) (DSA public key)
get server public key certificate(self signed) (RSA public key)
coonect socket
server send over server certificate
receive the certificate of server
generate a random AES-key (session key) and encrypt using server certificate
client send over encrypted AES-key and client certificate
Receive encrypted AES-key and client certificate
decrypt aes-key using server private key and hold aes session key as variable
Camera Picture taken by camera
AES session key
decode camera picture from base64 to bytes
hash with SHA 256 and encrypt with client DSA public key
parse bytes to hexadecimal for signature generation
AES-CBC 256 bits
hardcode Initialisation vector
encrypted Image in bytes
parse bytes to hexadecimal
append signature behind hexadecimal encrypted image
disconnect socket and connect ftp port 2121
hexacdecimal string of encrypted image and signature
send over hexadecimal string
received the hexadecimal message
split hexadecimal message to get signature and encrypted message
parse hexidecimal to bytes
Verify the encrypted image using checksum checking signature
Decrypt encrypted message using AES session key
store image in data file
END

New Features

Self-signed certificate

The self the client are, although not as secure as using an official Certificate Authority to sign the public key, enough to prove that the client's and server's public key are official because all the client cameras have their public key signed into a certificate already. This would ensure that no hacker would try to get a public key certificate and try to connect to the server.

Exchanging of certificates and AES-key

Upon the start up of the client and server, the client would connect to the server using socketing. Then the transfer of certificates of the public keys would be facilitated. On the client side, upon the receival of the server's public key certificate, a random AES session key would be generated and encrypted using the server's certificate. Then, the session key is sent over to the server program. The server then decrypts the encrypted session key using its private key. This session key is then used later on for verification of the encrypted image to ensure the validity of the image coming from a client and the non-repudiation.

Generating encrypted image

Assuming that images are taken upon motions are captured, the images are stored in base64. The client program would convert the base64 code to binary bytes via base64 decoding. Then, the AES session key and a hardcoded initialisation vector is used in an AES-CBC256 bits encryption together with the bytes of the image. Hence the encrypted image in binary is obtained. This ensures that the image cannot be read even if the image is captured over wireshark as it requires the AES session key and IV to be decrypted and read.

Note: The initialisation vector(IV) is hardcoded into both the client and server to be used during the AES-CBC 256 bit encryption process and the reverse during the decryption.

After getting the encrypted string of bytes, the bytes are hashed using SHA-256 and then encrypted using the public key of the DSA public key of the client. This final string of bytes obtained is then used as the signature.

The string of bytes representing the encrypted image is parsed to hexadecimal and so is the signature. Then the signature is appended behind the encrypted message. This is then sent over as one message to the server using the FTP service over port 2121.

Image verification

The received message is split and the signature and encrypted message in hexadecimal is obtained. Then the two hexadecimal strings are parsed into binary strings of bytes. Firstly, the image is verified against the signature by hashing and encrypting the message to match the signature. This would ensure that the image received is from the client. Then, the encrypted image is decrypted using the hardcoded IV and the AES session key. This decrypt process also serves as a method to verify the validity of the image coming from the client. After the image is decrypted and obtained as its original string of bytes, it is saved into a file and the exchange process is complete. Hence the server would wait for a new connection to the client to receive a new image, until the admin of the program force shut down the client or server program using Control-C command.

Conclusion and additional considerations

All in all, in order to ensure the safe and secure transmission of files from the camera to the server without jeopardising the integrity, confidentiality, authenticity and ensuring non-repudiation of the images there has to be many improvements made to the current system.

If improvements are not made, there are many vulnerabilities in the system that could be taken advantage of. These vulnerabilities pose a significant threat to the confidentiality, integrity, availability, authenticity and non-repudiation of the images. Examples of these threats are anonymous ftp login and MITM interception.

The proposed solution has addressed such issues by disabling anonymous ftp login and requiring a user's username and their password to login. Moreover in order to reduce threats regarding MITM interception the proposed solution also has an encryption method using SHA256 to further verify the authenticity of the message.

One additional consideration is that the server is always running.

Individual Reflection

Koh Kai En

I have learnt how to apply the theory I have learnt in Applied Cryptography to a real-life scenario of securing a client-server connection over an unsecured network. This has solidified my understanding of topics such as asymmetric cryptography and encryption, as well as taught me how to code these implementations into a functional system. Additionally, I have learnt how to source the internet for cryptographic documentation on libraries such as pycryptodome and cryptography, which is an essential programming skill in the future. Overall, this assignment has been a fulfilling and enlightening experience.

Sathiah Elamaran

I have learnt how to work together as a team and communicate effectively. Moreover I have learnt more about how encryption between a client and a server works. I have also learnt how to apply the concept of encryption and digital signature in real life. Not only that, I have learnt how to identify risks effectively and how to identify the risk response strategy.

Lee Pin

Through this assignment, I am able to learn how to effectively understand the process of ensuring data exchange between the server and the client is safe and secure, and how the message received can be verified to prove the validity of the received message or file. Furthermore, I learnt the importance of a CA and how it can be used for key exchanging between a server and client to get certificates which proves the public key of both parties are official.

Lastly, I learnt how to work together in a team to complete this assignment. Without the help of my teammates, it would be impossible for me to complete the assignment, having to do a report and code a program by myself. Thus, I am grateful for their help.

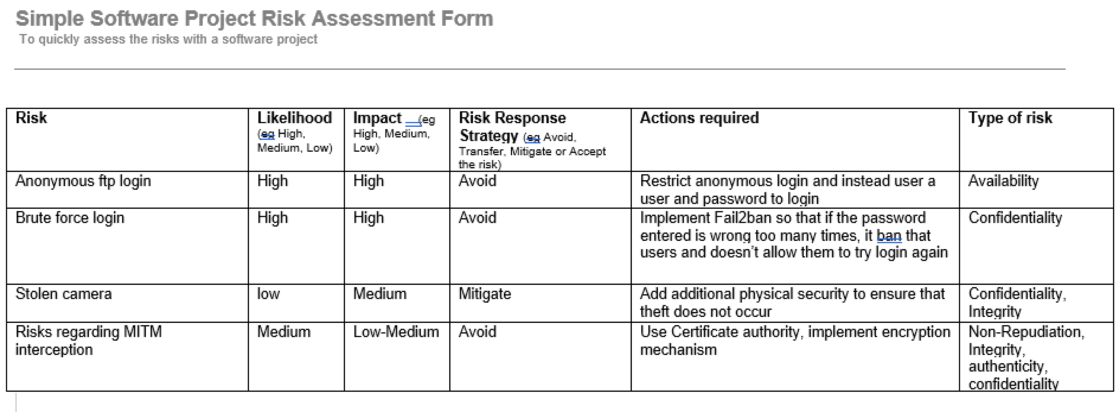
References

Cryptography.io. (2022). Welcome to pyca/cryptography — Cryptography 37.0.0.dev1 documentation. [online] Available at: https://cryptography.io/en/latest/ [Accessed 12 Feb. 2022].

Readthedocs.io. (2022). API documentation — PyCryptodome 3.14.1 documentation. [online] Available at: https://pycryptodome.readthedocs.io/en/latest/src/api.html [Accessed 11 Feb. 2022].

Pygame.org. (2021). Pygame Front Page — pygame v2.1.1 documentation. [online] Available at: https://www.pygame.org/docs/ [Accessed 13 Feb. 2022].

Appendix



Simple Software Project Risk Assessment Form

To quickly assess the risks with a software project

Risk	**Likelihood** (eg High, Medium, Low)	**Impact** (eg High, Medium, Low)	**Risk Response Strategy** (eg Avoid, Transfer, Mitigate or Accept the risk)	**Actions required**	**Type of risk**
Anonymous ftp login	High	High	Avoid	Restrict anonymous login and instead user a user and password to login	Availability
Brute force login	High	High	Avoid	Implement Fail2ban so that if the password entered is wrong too many times, it ban that users and doesn't allow them to try login again	Confidentiality
Stolen camera	low	Medium	Mitigate	Add additional physical security to ensure that theft does not occur	Confidentiality, Integrity
Risks regarding MITM interception	Medium	Low-Medium	Avoid	Use Certificate authority, implement encryption mechanism	Non-Repudiation, Integrity, authenticity, confidentiality