November 17, 2020

Introduction to cyber security 156360

semester a 2020-2021

hw # \_3&4\_

machon tal english speakers

|  |  |
| --- | --- |
| Teudat zehut | Student Name |
| 209927128 | Tamar Harizy |
| 006798775 | Gila Odes |

**Question 1 – cars challange**

The code we used to decrypt the code in the secret file was:

from \_\_future\_\_ import print\_function

import Crypto.Cipher.AES as AES

import hashlib

import binascii

message = "cd5074fe4adb4c0f61a3b5d0d60d3c7cab751dcff53d82f3c627dd1e5b5f63c0"

secret = binascii.unhexlify(message)

carFile = open("ListOfCars.txt","r")

for line in carFile:

#car = carFile.readline()

print(line)

line = line.rstrip()

carBackwards = line[::-1]

print(line + carBackwards + "JCT2020")

sec = hashlib.sha256(line + carBackwards + "JCT2020").digest()

print(sec)

# cipher = AES.new(AES.MODE\_ECB, sec)

cipher = AES.new(sec, AES.MODE\_ECB)

decMessage = cipher.decrypt(secret)

print("Decrypted message is %s" % decMessage)

The car used for encryption was **Honda** and the secret message is **!! Cyb3r S3cur1ty @ JCT r0cks !!**

The output code that is correct is as follows:

HondaadnoHJCT2020 #the hash

aS6%qP Hk?c #aes encryption

Decrypted message is !! Cyb3r S3cur1ty @ JCT r0cks !! #answer

**Question 2**

In order to work out which was the untampered message file we used ”diff ./message ./message\_dec”

Graphical user interface, application

Description automatically generated

This is our Verified file which contains the code save in message7 which was the correct message file that was not destroyed – it contained the message 3985 to which we added our names and ID’s as instructed.

T

Text

Description automatically generated

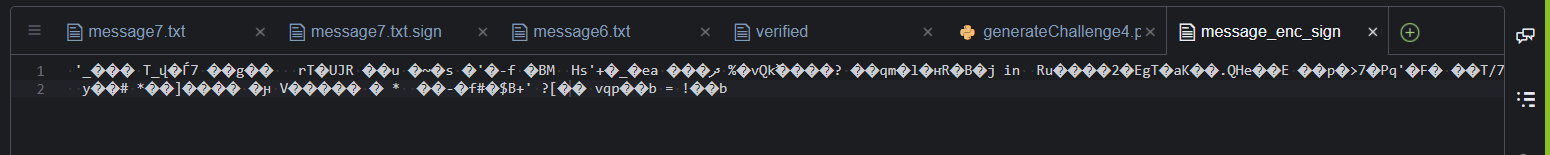
This is our public Key which we used to help decrypt our verified file

Graphical user interface

Description automatically generated

Below is our verified encrypted file with an added signature 🡪 message\_enc\_sign

Above is our verified file which has been encrypted – verified\_encryted.



Question 3

In order to decrypt the message we first calculated the hash of the standard (each time substituting the different digest options for “DIGESTALGO” & note ‘standard.txt’ is the standard file from ctf):

$ openssl -DIGESTALGO ./standard.txt

we then used the following to decrypt the secret [HASH is the hash resulted from trying each digest algorithm, therefore we ran this 4 times each time outputting to a different ‘result.txt’ file]:

$ openssl aes-128-cbc -K HASH -iv HASH -d -in ./secret.txt -out ./result.txt

The algorithm that resulted in a readable message (and therefore revealed the secret) was: -error all files appeared illegible

the secret decrypted is: - error all files appeared illegible

[below are snapshots of our work]

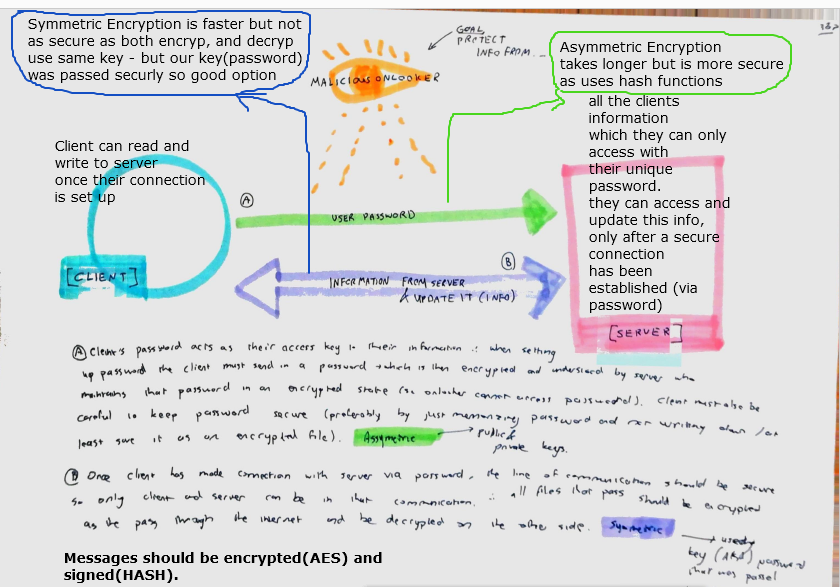
Text

Description automatically generatedText

Description automatically generated

Question 4

a) Describe the protocol and cryptography you will use to protect the client-server communication of the students’ confidential data:

Assuming the intention of this question was for us to suggest our own system using concepts from existing protocols and cryptographic methods – we suggest the following layout of communication:

There are two main stages to every communication between client (student) and the server (this website which contains all their information). The first stage being the client entering their password which allows initial connection and access to the server (as mentioned in question); The Second stage is the actual communication between client and server when in use – aka the student’s ability to read from and write to the server (as the question stated that the student can both see and update their information.) This Second stage can further be divided into student’s reading the information and student changing the saved information on the server. Seeing as our main concern (at the moment) is preventing a malicious onlooker from seeing and accessing information, we propose the following:

We would ideally like the main communication between client and server in stage two to be both secure and efficient – so a symmetric encryption would be ideal as this is the fastest encryption method. However. it presents the issue of both client and server needing the same key and passing that key can be unsecure. Therefore, we would pass the key for symmetric encryption in the first stage of the process – the password. We would like the Hash of the student password using a standard digest and AES encryption to be the key of our symmetric encryption process. This would mean that every communication between server and client is encrypted but both sides can decrypt it using the password (in the agreed upon form) as the key. The most secure way to pass this password would be when the student first creates an account to do so via asymmetric encryption. Asymmetric encryption is far slower but perfect for passing keys to be later used for symmetric encryption – as is the case here.

We would suggest that in stage two if a client is simply reading their information, then putting in their password is enough. However, when it comes to editing what is there on the main server it is advisable that the client be asked to renter their password as an extra security measure – or be sent a temporary one time password for this purpose.

b) How can you protect this protocol against Man-In-The-Middle? Assume the clients will be installed on non-trusted computers and any secrets stored in installation files may be stolen.

In order for the above-mentioned system to work both the client and the server have to be very careful to keep the client’s password information secure. The client should preferably memorize their password and not write it down or tell it to anyone else – is they need to they should write it down in such a way that they are reminded what it is as opposed to writing the password itself. The Server needs to authenticate the password and would need to have the passwords “saved on file” it is vital that these files are securely encrypted and that a specific level of clearance is needed for one to access those files. If the client is using an insecure computer the server should have a system whereby it operate on site only. The client should only be allowed to change things on the site and not download them. If one is reading and downloading information the file should be encrypted too to avoid information being understood if copied.