Ain Shams University
Faculty of Engineering

University of East London

Course Assessment Specification (CAS)

Programme Title : **Computer Engineering and Software Systems**
Coursework Title : **Projects**
Module Name (UEL) : **Computer and Network Security (1)**
Course Name (ASU) : **Computer and Network Security (1)**
Module/Course Code : **EG7643 / CSE451**
Level UEL/ASU : **6 / 4**
UEL Credit Rating : **15 Credits** **ASU Credit Rating** : **3 Credits**
Weighting : **25%**

Maximum mark available:

- 25 on software project

Lecturer : Prof. Ayman M. Bahaa-Eldin

Contact : If you have any issues with this coursework you may contact your lecturer.

Contact details are: Email: ayman.bahaa@eng.asu.edu.eg

Hand-out Date : As shown in submission matrix
Hand-in Date : As shown in submission matrix
Hand-in Method : Submission through LMS
Feedback Date : Your work will be marked and returned within two weeks.

Introduction

This coursework is itemized into several parts to get the 60 marks associated to it.
You must use the templates provided by the instructor to prepare your work.
All assignments and projects will be handed-in electronically, while quizzes and exams are written
.

Learning Outcome to be assessed

5. Analyse different problems that may arise during data communication and the impact of different security breaches on computer security?
6. Select suitable ciphers for different applications
7. Implement different ciphers and cryptanalysis techniques
9. Work and communicate effectively in team by effective collaboration and task management, working in a constrained stressful environment, and leading and motivating individuals.

Detail of the task

Attached separate file for each task.

89% and above:

Your work must be of outstanding quality and fully meet the requirements of the coursework specification and learning outcomes stated. You must show independent thinking and apply this to your work showing originality and consideration of key issues. There must be evidence of wider reading on the subject. In addition, your proposed solution should:

- illustrate a professional ability of drafting construction details,
- express a deep understanding of the in-hand problem definition,
- and applying, masterly, the learned knowledge in the proposed solution.

76% - 89%:

Your work must be of good quality and meet the requirements of the coursework specification and learning outcomes stated. You must demonstrate some originality in your work and show this by applying new learning to the key issues of the coursework. There must be evidence of wider reading on the subject. In addition, your proposed solution should:

- illustrate a Good ability of drafting construction details,
- express a very Good understanding of the in-hand problem definition,
- and applying most of the learned knowledge, correctly, in the proposed solution.

67% - 76%:

Your work must be comprehensive and meet all of the requirements stated by the coursework specification and learning outcomes. You must show a good understanding of the key concepts and be able to apply them to solve the problem set by the coursework. There must be enough depth to your work to provide evidence of wider reading. In addition, your proposed solution should:

- illustrate a moderate ability of drafting construction details,
- express a good understanding of the in-hand problem definition,
- and applying most of the learned knowledge, correctly, in the proposed solution.

60% - 67%:

Your work must be of a standard that meets the requirements stated by the coursework specification and learning outcomes. You must show a reasonable level of understanding of the key concepts and principles and you must have applied this knowledge to the coursework problem. There should be some evidence of wider reading. In addition, your proposed solution should:

- illustrate a fair ability of drafting construction details,
- express a fair understanding of the in-hand problem definition,
- and applying some of the learned knowledge, correctly, in the proposed solution.

Below 60%:

Your work is of poor quality and does not meet the requirements stated by the coursework specification and learning outcomes. There is a lack of understanding of key concepts and knowledge and no evidence of wider reading. In addition, your proposed solution would be:

- Illustrate an inability of drafting construction details,
- Failed to define the parameters, limitations, and offerings of the in-hand problem,
- Failed to apply correctly the learned knowledge for proposing a valid solution.

Academic Misconduct

The University defines Academic Misconduct as 'any case of deliberate, premeditated cheating, collusion, plagiarism or falsification of information, in an attempt to deceive and gain an unfair advantage in assessment'. This includes attempting to gain marks as part of a team without making a contribution. The department takes Academic Misconduct very seriously and any suspected cases will be investigated through the University's standard policy. If you are found guilty, you may be expelled from the University with no award.

It is your responsibility to ensure that you understand what constitutes Academic Misconduct and to ensure that you do not break the rules. If you are unclear about what is required, please ask

Secure-Email with Key Distribution

1. **This is a group (3 students) project,**
2. **Groups are announced on the LMS**
3. **Only one member of the group is required to submit.**
4. **Phases and delivery are outlined on the LMS**

Introduction:

In an era dominated by digital communication, the security of electronic messages is of paramount importance. Email, being one of the most widely used communication mediums, demands robust security measures to protect sensitive information from unauthorized access and tampering. The "Secure Email Communication System" project aims to address this critical need by leveraging Python programming and Microsoft Outlook integration to design and implement a comprehensive security protocol for sending and receiving secure emails.

The main idea here is to send and receive emails with attachments,
The email itself will be just plain text, while each email will have 2 attachments,

The first one is used to transfer security information including ID, certificates and wrapped encryption and decryption keys.

The second attached file will be the secured message

The tool will use your installed Office Outlook email client utilizing your university email to send and receive emails, while your software will be used to compose and read secured emails.

Description:

The project unfolds in four distinct phases, each meticulously crafted to build a robust and user-friendly secure email communication system.

Phase 1: Design and Planning In this initial stage, the team will define the project's scope, objectives, and technical requirements. A detailed project plan will be developed, outlining timelines, milestones, and assigned tasks. Potential security threats will be identified, and a risk mitigation strategy will be formulated to guide subsequent development phases.

Objective: Define the scope, objectives, and technical requirements of the secure email communication system.

1. Develop a project proposal outlining the purpose, goals, and expected outcomes.
2. Create a detailed project plan with timelines, milestones, and assigned tasks for each team member.
3. Identify potential security threats related to email communication and outline a risk mitigation strategy.

Phase 2: Implementation of Encryption Algorithms The second phase involves the implementation of essential security measures, including symmetric encryption, hashing, and digital signatures using Python. Algorithms such as Advanced Encryption Standard (AES) for encryption, SHA-256 for hashing, and public-key cryptography for digital signatures will be integrated into the system to ensure confidentiality, data integrity, and sender verification. (Crypto Libraries can be used as a ready implementation, in this case you need to verify the implementation and make sure it is free of attacks)

Objective: Implement symmetric encryption, hashing, and digital signature algorithms using Python for secure email communication.

1. Integrate a symmetric encryption algorithm (e.g., AES) to encrypt email content.
2. Implement a hashing algorithm (e.g., SHA-256) for data integrity verification.

3. Develop a digital signature mechanism using public-key cryptography for sender verification.

Phase 3: Integration with Outlook Building on the secure foundation laid in the previous phase, the team will develop a Python script or add-in to seamlessly integrate the security features into the widely-used Microsoft Outlook email client. This phase is crucial for ensuring practical usability and compatibility with real-world email scenarios. The result will be an enhanced Outlook experience, fortified with encryption, hashing, and digital signature capabilities.

Objective: Integrate the developed security features into the Microsoft Outlook email client.

1. Develop a Python script or add-in to seamlessly integrate encryption, hashing, and digital signatures within the Outlook email environment.
2. Ensure compatibility and usability by conducting thorough testing with various email scenarios.
3. Provide clear documentation and user guides for using the secure email features in Outlook.

Phase 4: Testing and Finalization The final phase focuses on rigorous testing to validate the efficacy of the implemented security measures. Comprehensive security testing will be conducted to identify and address vulnerabilities, ensuring a resilient system. User acceptance testing with real-world scenarios will guarantee that the secure email system is not only secure but also user-friendly. The documentation produced during this phase will provide a comprehensive guide for end-users and serve as a record of the project's development journey.

The project's success will be evaluated based on the effectiveness of the implemented security measures, successful integration with Outlook, thorough testing, clarity of documentation, and the quality of the final presentation. Through this project, the student teams will demonstrate their ability to design, implement, and secure email communication systems, showcasing their proficiency in information security skills and technologies.

Objective: Validate the security measures, conduct testing, and finalize the secure email communication system.

1. Perform comprehensive security testing to identify and address vulnerabilities.
2. Conduct user acceptance testing with real-world scenarios to ensure practical usability.
3. Document the implementation details, challenges faced, and lessons learned during the project.
4. Create a presentation showcasing the secure email communication system, emphasizing the incorporation of encryption, hashing, and digital signatures.

Final Deliverables:

Phase 1:

1. Project proposal
2. Detailed project plan

Phase 2:

3. Implemented Python code for encryption, hashing, and digital signatures

Phase 3:

4. Integration script or add-in for Outlook
5. Testing documentation and results

Phase 4:

6. Final project presentation
7. User guides and documentation for end-users

APPENDIX (A), Simple python

Sample Code to Encrypt and decrypt files:

```
import os, random, struct
from Cryptodome.Cipher import AES

def encrypt_file(key, in_filename, out_filename=None, chunksize=16):
    """ Encrypts a file using AES (CBC mode) with the
        given key.

        key:
            The encryption key - a string that must be
            either 16, 24 or 32 bytes long. Longer keys
            are more secure.

        in_filename:
            Name of the input file

        out_filename:
            If None, '<in_filename>.enc' will be used.

        chunksize:
            Sets the size of the chunk which the function
            uses to read and encrypt the file. Larger chunk
            sizes can be faster for some files and machines.
            chunksize must be divisible by 16.
    """
    if not out_filename:
        out_filename = in_filename + '.enc'

    encryptor = AES.new(key, AES.MODE_ECB)
    filesize = os.path.getsize(in_filename)

    with open(in_filename, 'rb') as infile:
        with open(out_filename, 'wb') as outfile:
            while True:
                chunk = infile.read(chunksize)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    chunk += b' ' * (16 - len(chunk) % 16)

                outfile.write(encryptor.encrypt(chunk))
def decrypt_file(key, in_filename, out_filename=None, chunksize=16):
    """ Decrypts a file using AES (CBC mode) with the
        given key. Parameters are similar to encrypt_file,
        with one difference: out_filename, if not supplied
        will be in_filename without its last extension
        (i.e. if in_filename is 'aaa.zip.enc' then
        out_filename will be 'aaa.zip')
    """
    if not out_filename:
```

```
        out_filename = os.path.splitext(in_filename)[0]

    with open(in_filename, 'rb') as infile:
        decryptor = AES.new(key, AES.MODE_ECB)
        with open(out_filename, 'wb') as outfile:
            while True:
                chunk = infile.read(chunksize)
                if len(chunk) == 0:
                    break
                outfile.write(decryptor.decrypt(chunk))

kk = b"1234567812345678"
encrypt_file(kk,'in.txt','out.txt')
decrypt_file(kk,'out.txt','dec.txt')
```

2. **Automating Outlook with Python**
   You must have outlook installed on your windows machine and configured to send and receive emails through your @eng.asu.edu.eg email.

Step 1: Install Required Libraries

```
pip install pywin32
```

Step 2: Configure Outlook Security Settings

Before you start, you need to configure the Outlook security settings to allow programmatic access. Here are the steps:

1. Open Outlook.
2. Go to "File" -> "Options" -> "Trust Center" -> "Trust Center Settings."
3. In the Trust Center, select "Macro Settings."
4. Enable "Enable all macros" and check "Trust access to the VBA project object model."
5. Click "OK" to save the changes.

Step 3: Create a New Outlook Profile (Optional)

It's recommended to use a dedicated Outlook profile for automation to avoid interference with your personal email. You can create a new Outlook profile by going to "Control Panel" -> "Mail" -> "Show Profiles" -> "Add."

Sample Code to automate outlook send and receive emails with attachments

```
import win32com.client
import os
import datetime

def send_email(to, subject, body, attachment_paths=None):
    outlook_app = win32com.client.Dispatch("Outlook.Application")
    mail_item = outlook_app.CreateItem(0)  # 0 represents olMailItem constant

    mail_item.To = to
    mail_item.Subject = subject
    mail_item.Body = body

    if attachment_paths:
        for attachment_path in attachment_paths:
            attachment = os.path.abspath(attachment_path)
            if os.path.exists(attachment):
                mail_item.Attachments.Add(attachment)
            else:
                print(f"Attachment file not found: {attachment_path}")

    mail_item.Send()

def read_emails(save_attachment_path="attachments/"):
    outlook_app = win32com.client.Dispatch("Outlook.Application")
    namespace = outlook_app.GetNamespace("MAPI")
```

```
    inbox = namespace.GetDefaultFolder(6)  # 6 represents olFolderInbox constant
    items = inbox.Items

    if not os.path.exists(save_attachment_path):
        os.makedirs(save_attachment_path)

    for item in items:
        print("Subject:", item.Subject)
        print("Sender:", item.SenderName)
        print("Received Time:", item.ReceivedTime)
        print("Body:", item.Body)

        # Check for attachments
        if item.Attachments.Count > 0:
            print("Attachments:")
            for attachment in item.Attachments:
                print(attachment.FileName)

                # Save attachment to local storage
                save_path = os.path.join(save_attachment_path, attachment.FileName)
                attachment.SaveAsFile(save_path)
                print(f"Attachment saved to: {save_path}")

        print("\n")

# Example usage
if __name__ == "__main__":
    # Specify the path to save attachments
    save_attachment_path = "path/to/save/attachments/"

    # Send an email with multiple attachments
    send_email("recipient@example.com", "Test Subject with Attachments", "This is the
body of the email with attachments.",
               ["path/to/attachment1.txt", "path/to/attachment2.pdf"])

    # Wait for a moment to allow the sent email to appear in the Inbox
    print("Waiting for email to be received...")
    input("Press Enter when the email is received.")

    # Read emails from the Inbox and save attachments locally
    print("Reading emails from Inbox and saving attachments:")
    read_emails(save_attachment_path)
```

Sample Multithreaded Socket Server

```
import socket, threading
class ClientThread(threading.Thread):

    def __init__(self,ip,port,clientsocket):
        threading.Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.csocket = clientsocket
        print ("[+] New thread started for ",ip,":",str(port))

    def run(self):
        print ("Connection from : ",ip,":",str(port))

        clientsock.send("Welcome to the multi-thraeded server".encode())

        data = "dummydata"

        while len(data):
            data = self.csocket.recv(2048)
            print("Client(%s:%s) sent : %s"%(self.ip, str(self.port), data.decode()))
            self.csocket.send(str.encode("You sent me : "+data.decode()))
            if data.decode()=="quit":
              self.csocket.send(str.encode("Ok By By"))
              self.csocket.close()
              data=''
        print ("Client at ",self.ip," disconnected...")

host = "0.0.0.0"
port = 10000

tcpsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

tcpsock.bind((host,port))

while True:
    tcpsock.listen(4)
    print ("Listening for incoming connections...")
    (clientsock, (ip, port)) = tcpsock.accept()
    #pass clientsock to the ClientThread thread object being created
    newthread = ClientThread(ip, port, clientsock)
    newthread.start()
```