ISEC2000 Fundamental Concepts of Cryptography
Assignment 2, 2025

Curtin University

Weightage: This assignment contains 6 questions, for a total of 100 points, which weights for 25% of the final mark.

Submission:
For the submission of your assignment, you are required to compile all necessary files into **one ZIP archive**. This archive should be uploaded to Blackboard for evaluation. The naming convention for your ZIP file should follow this precise format:
`<studentID>_<FirstName_LastName>_assignment02.zip`.
Please ensure your submission includes the following components:

- **Code Files:** All source code files written as part of the assignment solutions. This may include .py, .java, .c, or .cpp files, depending on the programming language(s) you have used.
- **Report:** A comprehensive report detailing your approach, findings, and any challenges encountered during the assignment. This document should provide insights into the logic behind your code and any conclusions drawn from the assignment tasks. Ensure that the report is in PDF format (Name should be: `StudentID_ReportA2.pdf`).
- **Other Files:** Any plaintext files, related source code, image files used or generated as part of the assignment, or output files from your encryption/decryption algorithms.
- **Cover Page:** The first page of your report should be the `DeclarationOfOriginality_v1.1.pdf`. This document serves as your commitment to submitting original work and adhering to academic integrity guidelines.
  The due date is **13 May 2025 11:59 PM**.

Academic Integrity:
This is an **individual** assignment so that any form of collaboration is not permitted. This is an **open-book** assignment so that you are allowed to use external materials, but make sure you properly **cite the references**. All answers must be your own understanding. It is your responsibility to understand Curtin's Academic Misconduct Rules, for example, post assessment questions online and ask for answers is considered as contract cheating and not permitted.

Conceptual Understanding and Question Answering

1. (5 points) Explain the core principle of public key cryptography and how it addresses the issue of secure communication over untrusted channels.

2. (15 points) Given two prime numbers $p = 61$ and $q = 53$, and public key exponent $e = 17$.

   - Calculate the modulus n, Euler's totient $\phi(n)$.
   - Compute the private key d.
   - Convert "**SECURE**" to ASCII and encrypt using RSA. Show the working steps.

3. (10 points) The Euclidean algorithm is based on the following assertion. Given two integers a, b, $(a < b)$,

$$\gcd(a, b) = \gcd(a, b \bmod a). \tag{1}$$

   Prove the assertion (1) **mathematically**. (Note that proof by example is NOT appropriate here)

4. (10 points) You are given the following parameters for a simplified Diffie-Hellman key exchange:

   - Prime modulus $p = 467$, Generator $g = 2$, Alice's private key $a = 153$, Bob's private key $b = 197$

   Now,

   a. Please compute both the public keys and the shared key.
   b. Describe a possible Man-in-the-Middle (MitM) attack scenario during the key exchange between Alice and Bob. Include the attacker's steps and how the attack compromises security.

5. (15 points) Answer the following questions:

   a. Explain how Elliptic Curve Cryptography (ECC) achieves security with smaller key sizes than RSA.

   b. Compare RSA-2048 vs ECC-256 in terms of speed and memory.

   c. If you are designing encryption for IoT devices, would you choose ECC or RSA? Justify with technical reasoning.

Secure Document Processor (Programming Task)

6. (45 points) Suppose, you are hired as a Junior Cryptography Engineer working for the Department of Secure Digital Infrastructure (DSDI), which oversees encrypted communications across government branches. A breach in document authenticity and confidentiality last year caused a major data leak.

   At DSDI, your role is set to develop a *Secure Document Processor* that ensures:
   - *Confidentiality*: Prevent others from reading the message
   - *Integrity*: Ensure the content has not been tampered with
   - *Authenticity*: Verify the sender's identity via digital signature

   This processor will be used to send classified reports between internal teams (e.g., Defense & Intelligence) in '.txt' format.
   Task Description: You will build a command-line tool using RSA public-key cryptography that performs the following, where the core features are required as:

   1. **Key Generation:** (a) Generate a 2048-bit RSA key pair, Use public exponent 65537. (b) Save the keys as 'private.pem' and 'public.pem'.

Note: This step should be included as part of `generate_keys()` function.

2. **Document Encryption:** (a) Encrypt the contents of 'confidential_message.txt' (this file is available in Blackboard). (b) Save the encrypted output to 'secure_message.enc' (Base64 format).
   Note: This logic should be integrated within the `encrypt_document(input_file,public_key_file,output_file)` function.
3. **Document Decryption:** (a) Decrypt the message from 'secure_message.enc' using the private key. (b) Print the plaintext to terminal for confirmation.
   Note: This logic should be integrated within the `decrypt_document(encrypted_file, private_key_file)` function.
4. **Digital Signature:** (a) Sign the message in 'confidential_message.txt' using SHA-256. (b) Save the signature as 'digital_signature.sig'.
   Note: This logic should be integrated within the `sign_document(input_file,private_key_file,signature_file)` function.
5. **Signature Verification:** (a) Verify the signature using the public key., (b) Clearly output verification success/failure to terminal.
   Note: This logic should be integrated within the `verify_sign(input_file,signature_file,public_key_file)` function.

Submission Folder Structure: Students must maintain the

following structure with the files in their ZIP file submission (The naming convention for your ZIP file should follow this precise format: `<studentID>_<FirstName_LastName>_assignment02.zip`):

/
├ secure_processor_`StudentID`.py (or .java, etc)
├ confidential_message.txt # original message file provided
├ secure_message.enc # RSA-encrypted file (refer 6.2)
├ digital_signature.sig # digital signature file (refer 6.4)
├ private.pem # RSA private key (refer 6.1)
├ public.pem # RSA public key (refer 6.1)
├ `StudentID_ReportA2.pdf` # Contains *all* the tasks of Assignment 2 in detail. It describes approach, issues, and analysis.
├ `DeclarationOfOriginality_v1.1.pdf`

Note:

- Replace `StudentID` with your actual Curtin Student ID (e.g., 22123456). You may include additional files - such as plaintext files, source code, or any images used or generated as part of the assignment - in the ZIP folder.
- You are expected to build a secure document processor using cryptographic practices. This means using a library like *cryptography* in Python (or similar libraries in other programming language of your choice) is completely valid and even encouraged, as long as you understand what each function does and document it clearly in your report.

END OF ASSIGNMENT