School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Assignment 1 (14 marks, worth 14%)
Due date: **6 November 2025, 9:00 pm** Singapore time.

Part One: Application and Discussion Questions: (8.0 marks)

Question 1 (3 marks)

Objective:

- To compute password entropy based on character selection and position constraints.
- To understand that cryptographic hashing does **not increase entropy** but protects the password's value.
- To relate theoretical entropy to real-world password security and brute-force feasibility.

A fictional social media platform, "SIMChat", is updating its password storage protocol to be more robust. The current scheme for password creation and storage is as follows:

Password Generation Scheme:

1. Choose **three** lowercase English letters.
2. Choose **two** digits (0–9).
3. Choose **one** symbol from the set {+, -, <, >, =}.
4. Arrange them in the order: letters, digits, symbol to form the final password string.

Password Storage Scheme:

Before storing the password, the system generates a random **12-bit** salt (chosen uniformly from all possible 12-bit binary strings). The final stored entry in the database is a record containing the salt and the hash, which is computed as follows:

hash = SHA3-256(salt || password)

where "||" denotes concatenation.

School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Tasks Specification

(a) **Password-Only Entropy** **(1.0 mark)**

Calculate the total entropy (in bits) of the password before adding the salt. Justify your answer by indicating the number of choices in each part and the total number of possible passwords.

(b) **Total Entropy with Salt** **(1.0 mark)**

Calculate the total entropy of the system after including the salt. Assuming the salt is truly random and independent of the password. Explain your reasoning.

(c) **The Role of the Salt** **(1.0 mark)**

In your own words, explain the following statements that seem contradictory.

i. Adding salt does not improve the strength of a given password against targeted brute force attacks.

ii. Adding salt is essential to protect against large-scale attacks using rainbow tables and precomputed hashes.

Deliverables

Submit a single document (e.g., PDF) containing the following clearly labelled sections:

- **Question 1(a):** Password-only entropy — calculation and justification
- **Question 1(b):** Total entropy with salt — calculation and explanation
- **Question 1(c):** Role of salt — conceptual explanation

School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Question 2 (2 marks)

Objective:
To help students understand how the **Bell–LaPadula (BLP)** security model enforces confidentiality using a lattice-based structure and how it relates to an **Access Control Matrix (ACM)**.

DataSecure Labs is a small research organization that manages sensitive documents and personnel at various security levels. The access control matrix below defines the privileges of each subject $(S_1 - S_5)$ to objects $(O_1 - O_6)$. "R" means read access (reads allowed) and "w" means append or write access.

	O_1	O_2	O_3	O_4	O_5	O_6
S_1	R	R		W		
S_2		R	W		R	
S_3	R		RW			W
S_4	R	R			W	
S_5			R	R		

Task Specification

Task 1 **(1.0 mark)**
Construct a **BLP lattice-structured system** that satisfies the above access control matrix using only the **mandatory access control (MAC) rules** of the Bell–LaPadula model.

- Assign appropriate security levels (e.g., *Top Secret, Secret, Confidential, Unclassified*) or numeric dominance levels to subjects and objects.
- Add intermediate or additional levels as necessary to ensure the structure forms a **valid lattice** (where every pair of elements has a least upper bound and a greatest lower bound).
- Clearly state any assumptions made.

School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Task 2 **(1 Mark)**

Describe the **process** by which you derived your lattice and verified consistency with the BLP *Simple Security* ("no read up") and *Star Property* ("no write down") rules.
Your explanation should:

- Identify how you determined the relative ordering between subjects and objects.

Explain why the final structure satisfies BLP confidentiality requirements.

Deliverables

Students must submit a document (e.g., PDF) containing:

- **Part 1:** The final **lattice diagram** showing the security levels of subjects and objects.
  (A simple textual or hand-drawn diagram is acceptable.)
- **Part 2:** A short-written **explanation** (about 150–200 words) describing the reasoning steps used to form the lattice.

School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Question 3 (3 marks)

Objective:

Demonstrate a deep understanding of access control matrices (ACMs), access control lists (ACLs), and capability lists (CLs) by modelling dynamic systems, applying security policies, and analysing the impact of changes.

Part 1: Initial System State and Policy Analysis (1.6 marks)

Consider the following state of affairs in a digital realm:

- Subjects: Knight Alice, Wizard Bob, Rogue Carol, and Shun
- Objects: Sword, Spellbook, TreasureChest, and CastleDoor

The security policy of the realm is defined as follows:

1. All knights can read the Manifesto and launch an attack against the Shun.
2. Wizard Bob can read and write the Book of Spells. He can also cast magical spells on castle doors.
3. Rogue Carol can read the manifest, take the sword, and steal it from the chest.
4. Shuns can burn treasure chests and castle doors.
5. Only Knights Alice can read the treasure chest inventory list.

Task Specification:

a) Define the sets of Subjects, Objects, and Actions for this scenario. Note that some actions like execute have sub-actions like attack, charm, and burn. Justify how you choose to model them. (0.4 mark)

b) Draw the initial Access Control Matrix (ACM) that enforces the stated security policy. (0.4 mark)

c) List the ACL for each object. Which real-world file system concept does this model resemble? (0.4 mark)

d) List the capabilities held by each subject. With what entity, e.g., user and/or process is each list associated in a real operating system? (0.4 mark)

School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Part 2: Dynamic System and The Principle of Least Privilege (1.4 marks)

A new decree is issued: "The Shun must be defeated. To facilitate this, enchant, on the Sword. Knight Alice must now be able to use the enchanted Sword on the Shun. Note, consider this a new right use on the Shun.

Task Specification:

a) Modify your ACM from Part 1 to reflect this change. Wizard Bob needs a new right, enchant, on the Sword. Knight Alice must now be able to use the enchanted Sword on the Shun (consider this a new right use on the Shun). (0.4 mark)

b) Update the Capability List for Wizard Bob and Knight Alice to implement the change. (0.4 mark)

c) Analysis: Which method (Access Control List or Capability List) made it easier to implement this change from the perspective of a system administrator managing the Sword? Which method made it easier from the perspective of auditing Knight Alice's total privileges? Explain why. (0.6 mark)

Deliverables

Students must submit a single document (e.g., PDF) containing **clearly labelled sections** corresponding to each task.

Part 1: Initial System State and Policy Analysis

Your submission should include:

a) Sets of Subjects, Objects, and Actions

- Define each set clearly (S, O, A).
- Justify how actions with sub-actions (e.g., *attack*, *burn*, *charm*) are modelled.

b) Access Control Matrix (ACM)

- Construct and label the ACM that enforces the initial policy.
- Ensure it reflects all rights granted by the security policy.

c) Access Control Lists (ACLs)

- Provide the ACL for each object.

- Include a short explanation identifying the real-world file system concept (e.g., UNIX file permissions, Windows ACLs) that this model most closely resembles.

d) Capability Lists (CLs)

- List all capabilities held by each subject.
- Explain with what entity (e.g., user, process, or role) each capability list would typically be associated in a real operating system.

Part 2: Dynamic System and the Principle of Least Privilege

Your submission should include:

a) Updated Access Control Matrix (ACM)

- Modify your Part 1 ACM to reflect the new rights:
  - Wizard Bob: add right *enchant* on *Sword*.
  - Knight Alice: add right *use* on *Shun*.

b) Updated Capability Lists (CLs)

- Provide the updated CLs for Wizard Bob and Knight Alice after the decree.

c) Analysis and Discussion

- Compare which method (ACL or CL) made it easier to implement the change, from the perspective of:
  - System administration (managing rights for the Sword), and
  - Auditing (reviewing Knight Alice’s total privileges).

Justify your reasoning clearly.

School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Part Two: Implementing a rainbow table (6.0 Marks)

You are to write a program, in **C/C++**, **Java**, or **Python**, that generates and uses a **Rainbow Table** to find pre-images for given hash values. Rainbow tables are a time–memory trade-off method used to solve pre-image problems for hash functions.

At the simplest level, a rainbow table consists of pairs of passwords and their hash values. However, such tables can be very large. A more efficient approach uses *hash* and *reduction functions* to form chains of password–hash pairs, reducing storage requirements while maintaining lookup efficiency.

Program Execution

Your program should accept one command-line argument, the password file, and run as follows (depending on the language used):

- **C/C++:** ./Rainbow Passwords.txt
- **Python:** python Rainbow.py Passwords.txt
- **Java:** java Rainbow Passwords Passwords.txt

The file Passwords.txt contains a list of possible passwords, one per line, consisting of printable characters. All passwords used by your program must come from this file.

First step: Rainbow Table Generation (3 marks)

Your program will do some initial computations to generate the rainbow table. The process is as follows:

1. Read in the list of possible passwords. Report on the number of words read in. (**0.5 mark**)
2. For each previously unused word W, first mark it as used and then carry out the following process (**1 marks**):
   (a) Apply the hash function H to the word W to produce a hash value $H(W)$, which we refer to as the current hash.
   (b) Apply the reduction function R to the current hash, which will give a different possible password which should be marked as used and then hashed. The resulting hash value is recorded as the current hash.
   (c) Repeat the previous step five times. You can deal with collisions if

you like but are not required to.

(d) Store the original word W and the final current hash as an entry in your rainbow table.

3. To assist with the later identification of the pre–images you should sort the rainbow table based on the hash values (**0.5 mark**).

4. Output the list of words and corresponding "final current hashes" to a text file Rainbow.txt. Report to standard out the number of lines in your rainbow table (**1 mark**).

Second Step: Finding Pre-Images (3 marks)

In this step, your program should allow the user to input a hash value and attempt to find its corresponding pre-image (original password).

1. **User input. (0.5 mark)**
   - Request a hash value from the user.
   - Perform basic validation (e.g., check for correct length and valid hexadecimal characters).
2. **Lookup process. (1.5 marks)**
   - Check if the hash value exists in the rainbow table.
   - If not found, perform the following:
     - Apply the **reduction function R** and then **hash** repeatedly until a hash value from the rainbow table is found.
     - Continue this process for up to the same number of iterations used during table generation (five).
     - If no match is found after all iterations, display a clear message such as:

       "No pre-image found — hash not present in rainbow table."

3. **Reconstruction. (0.5 mark)**
   - Once a matching hash is found in the rainbow table, retrieve the corresponding original password.
   - Apply the **hash** → **reduce** steps again to reconstruct the chain until the hash being searched for is reached.

- When a reduced word hashes to the target hash, that word is your pre-image.

4. **Output. (0.5 mark)**
    - Display the identified pre-image in standard output, e.g.:
    - Pre-image found: password123

Reduction Function (R)

A **reduction function** converts a hash value into a valid password — in this case, one from *Passwords.txt*.
Since the password file length is not fixed, your reduction method must be **dynamic**.

Your reduction function should:

- Deterministically convert a hash value into an index of the password list.
- For example:
    - Convert part of the hash (e.g., the first 8 hex digits) to an integer.
    - Compute index = integer % number_of_passwords.
    - Use the password at that index as the next candidate.

You may design your own reduction function or adapt an example found online. Explain clearly how it works in **readme.txt**.

You should use **MD5** as the hash function. You are not required to implement MD5 yourself – use an available library or open-source implementation and cite your source in **readme.txt**.

School of Computing & Information Technology
CSCI262 System Security
SI-2025-S4

Submission of Assignment 1

Note that you have only one submission. So, make absolutely sure that you submit the correct files with the correct contents. Please submit an Academic Consideration in SOLS if an extension (1 week maximally) is required.

1. Submission is via Moodle.
2. Include the compilation instructions with your submission in a file readme.txt. That file should also contain a description of how you do the reduction. If no such file exists in your submission, then you will get zero for this Part two.
3. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.
4. Submissions more than three days late will not be marked unless an extension has been granted.
5. If you need an extension apply through SOLS, if possible before the assignment deadline.
6. Plagiarism is treated seriously. Students involved will receive zero.

End of specification