PROBLEM STATEMENT 15:

Protecting User Password Keys at Rest (on the Disk)

Category:	System Software, Security
Participants:	5th-8th Semester Students
Team Size:	2 or 3 Member Team
Scope:	Developing an application for file encryption which is in turn protected by user pass phrase
Pre-requisite:	Linux File System Operations, Crypto Algorithms. Programming in any Language suited for System Software like C, C++, Python, etc.

Infrastructure Requirements:

Hardware:

- Any x86 based Desktop or Server with Linux

Description:

Develop an authorization application, which in turn protects the password keys. Following are the high level features:

1. Encrypt [AES-256] a user chosen file or directory using a random key a.k.a File Encryption Key.
2. Store the random key in a file, which has to be protected via user pass phrase.
3. The user pass phrase as well as the random key cannot be stored in plain form in the text file.
4. If the user pass phrase authentication is successful retrieve i.e., decrypt the file using File Encryption Key.

   Hint: You can use user pass phrase as a seed to generate deterministic keys using standard KDF (Key Derivation Function).

Project Outputs:

1. Application workflow.
2. High-level algorithm.
3. Justification for various Crypto algorithms used.

4. Type of open source and System routines used for various tasks.
5. Test plan for testing various simple and corner cases.
6. Actual Source Code with appropriate comments archived in GitHub.

Learning Outcome:

1. Partitioning the high-level problem statement into workflow and smaller independent tasks.
2. Understand different crypto algorithms and usage models.

[**PS**: All the above are possible only if you don’t use any ready-made ChatGPT/Generative AI tool]