PESU I/O Course Plan

General instructions:

1. The course plan caters to first years, so should start from the **very basics** of the course and go **extensively covering course content in great detail.**
2. The PESU I/O courses are divided into 4 weeks, and by the end of the course, it is expected from the students to come up with a **final project**, related to the course. The product should be **built consecutively as the days progress**. The product they will be coming up, needs to be **specified by the SME at the beginning of the course**. For Example: A dance course can teach you certain steps each week and the final product should be a well choreographed sequence consisting of all the steps taught, or an Algorithms course can teach you certain algorithms each week and the final product can be a software implementing those algorithms.
3. Each day starts with the SME explaining the tasks of the next couple of days. This is followed by the next day at home, where the students, on their own,without the SME, work and discuss with their peers. Next day, we have a SME Meetup, where the students meet and discuss with the SME any doubts, teach any topic if the majority of the class want it to and evaluate students based on their performance in the assignments and SME Meetup. Then followed by the next topic task assignment.
4. The students will have to submit their project by the end of the course as a presentation video, which will be assessed by the SME after the course ends.
5. The Course Plan should be divided into **13 days**. Each day should contain the following mandatorily: 1. Topics to be taught. 2. Assignments/Quizzes/Other weekly tasks of the day. 3.Daily Project Work to be done, which will build towards the final product. You should specifically outline what your students will accomplish with this course and what will be the final product your students will be making.
6. Along with the course material, include a general overview of the content intended for video shoots. This should briefly describe the key concepts, case studies, demonstrations, or discussions that will be featured in the videos. You can either choose a 3 week or 4 week format.

Cracking the Code: Applied Cryptography for Engineers – Course Plan

Instructor Name : Pranav Hemanth & Pranavjeet Naidu
SRN : PES1UG23CS433 & PES1UG23CS586

Branch: CSE

Semester: 5

Course Duration: 30 Hours (20 hours of in-person mentoring + 10 hours of self learning)

Prerequisites for the course:

1. Basic Python programming skills (not mandatory but recommended).
2. Curiosity, logical thinking, and willingness to solve puzzles.

Deliverables from the course:

1. Ability to implement and break classical ciphers in Python.
2. Understand the principles of symmetric and asymmetric cryptography.
3. Gain hands-on exposure to cryptography tools like CrypTool, CyberChef, GPG, and Steghide.
4. Create and solve cryptography-based puzzles, culminating in a final project.

Final Project:

Students will design a **multi-step Cicada-inspired cryptography puzzle** that combines classical ciphers, hashing, steganography, and RSA encryption. The project will be showcased as a challenge for peers, and students will be evaluated based on creativity, accuracy, and application of cryptographic concepts.

Video Shoot Content:

Week 1: Introduction to cryptography through real-world examples and Cicada 3301 case study.
Week 2: Demonstration of cryptanalysis (frequency analysis and cipher breaking).
Week 3: Practical encryption using AES and RSA with tools and code walkthroughs.
Week 4: Student-led project demos and discussion of real-world applications of cryptography.

Day 0 – Student Onboarding

- Meet and greet, introduction to the course structure and expectations.
- Icebreaker activity: Solve a simple Caesar cipher puzzle together as a class.
- Explain how daily tasks and weekly assignments will build into the final project.

Day 1 – The World of Secrets

1. Topics to be taught:

- History of cryptography (Caesar → Enigma → RSA).
- Basics of information security (CIA triad).
- Classical ciphers (Caesar, Vigenère, ROT13).

2. Tasks to be completed:

- Implement Caesar & Vigenère cipher in Python.
- Solve a short quiz on classical ciphers.
- Students encrypt and share short messages with peers using Caesar/Vigenère.

Day 2 – Classical Cipher Applications

1. Topics to be taught:

- Playfair cipher and substitution ciphers.
- Case study: Cicada 3301 intro puzzles.

2. Tasks to be completed:

- Solve puzzles using dCode / CyberChef.
- Students start designing simple puzzle chains using multiple classical ciphers.

3. Weekly Assignment 1

- Decode a set of encrypted texts (Caesar, Vigenère, ROT13, Playfair). Submit solutions and share one self-made puzzle with peers.

Day 3 – Breaking Ciphers

1. Topics to be taught:

- Frequency analysis basics.
- Known plaintext attacks.

2. Tasks to be completed:

- Write a frequency analysis script in Python.
- Attempt to break a substitution cipher.
- Incorporate frequency analysis concepts into their ongoing puzzle project.

Day 4 – Case Study: Enigma & Cryptanalysis

1. Topics to be taught:

- The Enigma machine: history and cryptographic weaknesses.
- Manual vs automated cryptanalysis.

2. Tasks to be completed:

- Use CrypTool to simulate Enigma encryption/decryption.
- Mini-quiz on Enigma's flaws.
- Explore how layering ciphers increases or decreases security; apply this idea in puzzles.

Day 5 – Symmetric Cryptography

1. Topics to be taught:

- Block vs stream ciphers.
- AES basics and how modern encryption works.
- Introduction to hashing (MD5, SHA256).

2. Tasks to be completed:

- Implement AES encryption/decryption in Python.
- Hash text and files using Python libraries.
- Students add a “hashing challenge” step to their puzzle project.

3. Weekly Assignment 2

- Hash a password list, check membership for given inputs, and encrypt/decrypt a short text file with AES.

Day 6 – Passwords and Cracking

1. Topics to be taught:

- Salts and password security.
- Demonstration of cracking with Hashcat/John the Ripper.

2. Tasks to be completed:

- Students attempt a small password cracking challenge with a given wordlist.
- Incorporate a weak password challenge step in the puzzle chain.

Day 7 – Asymmetric Cryptography

1. Topics to be taught:

- RSA basics and math intuition.
- Key generation and the public/private key concept.

2. Tasks to be completed:

- Generate RSA keys in Python.
- Encrypt and decrypt sample messages.
- Add a step in the puzzle project that requires RSA decryption.

Day 8 – Digital Signatures & Certificates

1. Topics to be taught:

- Digital signatures and message integrity.
- SSL/TLS overview.

2. Tasks to be completed:

- Use GPG to sign and verify a message.
- Add a digital signature step in the puzzle project.

3. Weekly Assignment 3

- Exchange encrypted and signed messages with peers, ensuring integrity checks.

Day 9 – Steganography in Cryptography

1. Topics to be taught:

- Basics of steganography (image, audio).
- Cicada-style hidden messages.

2. Tasks to be completed:

- Hide a secret message inside an image using Steghide.
- Extract a hidden message shared by the SME.
- Students add a hidden message clue in an image/audio file as part of their project.

Day 10 – Real-World Crypto Applications

1. Topics to be taught:

- Everyday crypto: WhatsApp/Signal E2EE.
- Bitcoin and blockchain basics.

2. Tasks to be completed:

- Group discussion on strengths and flaws of real-world crypto protocols.
- Reflect on how concepts like blockchain or E2EE could inspire puzzle design.

Day 11- Mini Crypto CTF

1. Topics to be taught:

- Integrating multiple crypto techniques into challenges.
- Practice for the final project.

2. Tasks to be completed:

- Solve SME-created multi-step crypto challenge (cipher → hash → stego → RSA).
- Finalize all components of their puzzle chain project.

Day 12 – Final Project Presentation

Tasks to be completed:

- Students present their Cicada-inspired cryptography puzzles.
- Each group explains the design choices and cryptographic concepts used.
- SME evaluates creativity, technical accuracy, challenge flow, and problem design.
- Alternatively if students are interested they can build a secure chat, steganography tool, cryptanalysis toolkit or password managers to demonstrate their learning from the course also.