Course Name: Cryptography Professional Certification Course

Course Overview

The "Cryptography Professional Certification Course" provides a comprehensive understanding of cryptographic principles, techniques, and applications. It covers the evolution of cryptography, mathematical foundations, classical and modern encryption methods, hashing, digital signatures, and cryptographic protocols. With a hands-on approach, learners will explore how cryptography secures digital communications and prevents cyber threats. By the end of the course, students will implement secure systems and critically analyze real-world cryptographic applications and vulnerabilities. The capstone project consolidates learning by designing a secure application.

Course Type

Entry to Intermediate Level

Course Objectives

1. Understand the foundational principles and goals of cryptography.

2. Develop mathematical skills critical to cryptographic algorithms.

3. Learn about classical, symmetric, and asymmetric encryption techniques and their modern applications.

4. Explore hash functions, digital signatures, and cryptographic protocols in detail.

5. Gain practical experience through coding exercises and mini-projects.

6. Analyze cryptographic applications, vulnerabilities, and mitigation strategies.

7. Design and present a secure system in the capstone project.

What You'll Learn?

- The history and goals of cryptography and its significance in securing digital systems.

- Mathematical concepts such as modular arithmetic, prime numbers, and probability used in cryptographic algorithms.

- Classical and modern encryption techniques, including AES, RSA, and ECC, and their practical applications.

- Cryptographic tools like hashing, MACs, and digital signatures to ensure data integrity and authentication.

- Real-world applications such as blockchain, secure messaging, and cryptographic protocols like SSL/TLS.

- Mitigation strategies against common cryptographic attacks like brute force and side-channel attacks.

Duration

Approximately 50-60 hours of coursework, including hands-on exercises and projects.

Requirements

- A computer with Python (and necessary libraries like PyCryptodome) installed.

- Stable internet connection for accessing cryptographic tools and research materials.

Pre-requisites

- Basic programming knowledge (preferably in Python).

- Familiarity with basic algebra and mathematical concepts.

Target Audience

- Students and professionals new to cryptography or cybersecurity.

- Aspiring cryptographers and security analysts.

- Developers seeking to integrate cryptographic security into applications.

- IT professionals looking to enhance their understanding of secure systems.

Curriculum

Module 1. Introduction to Cryptography

- History and evolution of cryptography.

- Cryptographic goals: confidentiality, integrity, authentication.

- Types of cryptography: symmetric, asymmetric, and hashing.

- Applications in modern systems.

- Understand the basics of cryptography and its role in securing digital communication.

- Research a historical cryptographic system (e.g., Caesar Cipher) and explain its strengths and weaknesses.

Module 2. Mathematical Foundations

- Modular arithmetic.

- Prime numbers and their properties.

- Euclidean algorithm and GCD.

- Groups, rings, and fields in cryptography.

- Introduction to probability in cryptography.

- Build a strong mathematical foundation for understanding cryptographic algorithms.

- Implement modular arithmetic functions and solve related cryptographic problems in Python.

Module 3. Classical Cryptography

- Substitution ciphers (Caesar, Vigenère).

- Transposition ciphers.

- Breaking classical ciphers (frequency analysis).

- Understand classical encryption methods and their vulnerabilities.

- Encrypt and decrypt text using classical ciphers and attempt to break ciphertext using frequency analysis.

Module 4. Symmetric Key Cryptography

- Principles of symmetric encryption.

- Block vs. stream ciphers.

- Data Encryption Standard (DES).

- Advanced Encryption Standard (AES).

- Modes of operation (ECB, CBC, CFB, OFB).

- Understand symmetric encryption, including block cipher techniques and their use cases.

- Encrypt a dataset using AES and compare the results of different modes of operation.

Module 5. Asymmetric Key Cryptography

- Public key cryptosystems: principles and differences from symmetric encryption.

- RSA algorithm: key generation, encryption, and decryption.

- Diffie-Hellman Key Exchange.

- ElGamal encryption.

- Learn the mechanics and applications of asymmetric cryptographic algorithms.

- Implement RSA encryption and decryption in Python. Perform a secure key exchange simulation using Diffie-Hellman.

Module 6. Cryptographic Hash Functions

- Properties: collision resistance, pre-image resistance, and second pre-image resistance.

- Common hash functions: MD5, SHA-1, SHA-256.

- Applications: password storage, integrity checks, and digital signatures.

- Understand how hash functions ensure data integrity and their limitations.

- Create a program to hash text and analyze collisions using MD5 and SHA-256.

Module 7. Message Authentication Codes (MACs)

- Concept and applications.

- HMAC (Hashed Message Authentication Code).

- Comparison with digital signatures.

- Learn methods for ensuring message integrity and authenticity.

- Implement HMAC for verifying the integrity of transmitted messages.

Module 8. Digital Signatures

- Principles of digital signatures.

- RSA and DSA-based digital signatures.

- Applications: certificates, legal documents, and blockchain.

- Understand how digital signatures provide authenticity and integrity.

- Simulate signing and verifying documents using digital signature algorithms in Python.

Module 9. Elliptic Curve Cryptography (ECC)

- Basics of elliptic curves.

- ECC-based key exchange and encryption.

- Advantages over RSA.

- Applications in modern cryptography (blockchain, secure messaging).

- Learn the importance of ECC in resource-constrained environments.

- Implement elliptic curve key generation and encryption using libraries like PyCryptodome.

Module 10. Cryptographic Protocols

- SSL/TLS: principles and applications.

- Secure file sharing and messaging (PGP, S/MIME).

- Zero-Knowledge Proofs.

- Understand how cryptographic protocols secure communication over the internet.

- Set up an HTTPS server or use PGP to encrypt and share a file securely.

Module 11. Applications and Attacks

- Applications: Blockchain, digital currencies, secure voting systems.

- Common attacks: brute force, side-channel attacks, replay attacks.

- Mitigation strategies. Analyze real-world cryptographic applications and threats to their security.

- Research and write a report on a cryptographic attack and its mitigation (e.g., side-channel attack).

Module 12. Capstone Project

- Combine cryptographic concepts to design a secure system.

- Include encryption, hashing, and authentication mechanisms.

- Apply all learned concepts to solve a real-world cryptographic problem.

- Design and present a cryptographic system, e.g., secure messaging or file sharing application.

Key Deliverables:

1.Weekly Mini-Projects:

- Cipher implementations, encryption/decryption scripts, hashing experiments, and signature verification tasks.

2. Capstone Project:

- Build a secure application (e.g., encrypted chat, file-sharing app, or secure voting system) combining learned techniques.

Tools and Resources:

- Programming Languages: Python (PyCryptodome, hashlib), JavaScript (Node.js crypto library).

- Cryptographic Libraries: OpenSSL, GPG, PyCryptodome.

- Mathematics Tools: Wolfram Alpha, SymPy (Python library).

- Books:

- Cryptography and Network Security by William Stallings.

- Applied Cryptography" by Bruce Schneier.

- Introduction to Modern Cryptography" by Jonathan Katz.