University of Technology and Applied Sciences

Course Code:	**Course Name: Practical Cryptography**	**Level: 2**
Prerequisite: Information Security 1	**Course type: MR**	**Passing Grade / Mark: 50**
Credit Hours: 3H	**Theory contact hours: 15H**	**Practical contact hours: 45H**

Course Information

Course Goals/ Description	This course provides basic and practical concepts on cryptography and cryptanalysis. The course covers a detailed description of the building blocks of symmetric ciphers, hash/HMAC algorithms, asymmetric ciphers, key management process with a practical implementation using Python 3.10. Spyder or PyCharm are recommended IDEs for Python programming. Alternatively replit.com (a collaborative browser based IDE) can be used for fast prototyping.

Course Objectives

This course should enable the students to:

1. Have an Extensive, detailed and critical understanding of basic concepts behind most used cryptographic primitives.
2. Develop a familiarity in modern cryptographic algorithms and enrich the knowledge to the students of existing deployed standards.
3. Equip students with practical implementation of symmetric and asymmetric cryptographic Algorithms
4. Implement most of the algorithms using Python and Openssl.

S#	Proposed Outcomes By the end of the course, the students should be able to:	LOTs	HOTs
1.	Understand basic background behind most cryptographic standards	**x**	
2.	Implement cryptographic algorithms defined in cryptographic standards	**x**	**x**
3.	Describe the purpose of cryptography and list ways it is used in data communications	**x**	**x**
4.	Describe the following terms: cipher, cryptanalysis, cryptographic algorithm, and cryptology, and describe the two basic methods (ciphers) for transforming plaintext in ciphertext.	**x**	
5.	Explain how public key infrastructure supports digital signing and encryption and discuss the limitations/vulnerabilities		**x**
6.	Discuss the dangers of inventing one's own cryptographic methods		**x**
7.	Describe which cryptographic protocols, tools and techniques are appropriate for a given situation		**x**

Facilities and resources required

Hardware requirements	Software requirements
PC with a minimum of 2.6 GHz CPU and 16 GB of RAM memory and 64-bit operating system, x64 based processor. **OR** An internet connection to use replit.com :a collaborative browser based IDE	- Ubuntu machine : besides python based labs, there are an intensive use of command-line Linux shell for OpenSSL and tools as well as other Linux specific open-source tools - Python 3.10 through Anaconda distribution. This is to install the right version of python that is needed for the described labs and to properly fetch additional modules with appropriate versions. Jupyter Notebook and Spyder IDE can be installed through Anaconda Navigator. Alternatively the same packages can be installed and imported through the use of replit.com.

Learning Resources

Books	OER	Useful links
- Full Stack Python Security : Cryptography, TLS, and attack resistance. Dennis Byrne. Manning Publications Co. ISBN: 9781617298820. 2021 - Implementing Cryptography Using Python. Shannon W. Bray. John Wiley & Sons, Inc. 2020. ISBN: 978-1-119-61220-9. 2020 - Hands-On Image Processing with Python. Sandipan Dey. Packt publishing. ISBN : 978-1-78934-373-1. 2018		- https://docs.anaconda.com/anaconda/navigator/tutorials/index.html

Course Working Plan:

Week#	Lectures/Contents	Ref.	Practical / Lab	Assessment
Week 1	**INTRODUCTION TO PYTHON** ● Using variables, strings, tuples, lists. Dictionaries, ● Arithmetic operators, comparison operators, logical operators, assignment operators, bitwise operators, membership operators, identity operators, ● Conditionals, loops, Manipulating files: read, create, update ● Working with functions ● json files representation Binary data, hexadecimal ASCII encoding, utf-8 encoding, Base64 encoding and decoding	Text2, Chap1: Introduction to Cryptography and Python	**LAB1** Install Anaconda for python3.8 Create a virtual environment for the labs using Anaconda navigator Use Spyder and Jupyter Notebook to run scripts and interactive code. IInstall python modules “Pycryptodome”, “cryptography” , “hashlib” and “bitvector” parse/serialize json files	

Week 2	**INTRODUCTION TO CRYPTOLOGY** ● Overview of Secret communications using cryptography ● Symmetric Cryptography ● Asymmetric Cryptography ● Understanding the brute force attack Password best practices : hashing, salting, stretching,	Text2, chap 3: Classical Cryptography pp 66-72	**LAB2** Create a reverse encryption algorithm Use bcrypt to create secure passwords Authentication of a user using password	**Project Assignment**
Week 3	**CLASSICAL CRYPTOGRAPHY** ● Caesar Cipher Brute forcing Caesar Vigenere Playfair Affine Cipher	Text2, chap 3 : Classical Cryptography	**LAB3** - Implement Caesar - Implement brute force attack on Casear	
Week 4	**BLOCK CIPHERS AND BLOCK ENCRYPTION MODES** DES Structure Modes of Block encryption ECB, CBC 3DES structure	Text2, Chap 5: Stream Ciphers and Block Ciphers	**LAB4** - Use "Pycryptodome" module to Implement DES to encrypt/decrypt a message using ECB, CBC - Use DES/3DES to encrypt/decrypt a file stored in your disk.	
Week 5	**AES** ● General Design of AES ● Addroundkey, ● SubBytes and InvSubBytes ● Shiftrows and InvShiftRows ● Mixcolumns and InvMixColumns	Text1, Chap 4: Symmetric Encryption	**LAB5** Implement AES using the "Pycryptodome" module with different modes of bloc encryption. Implement AES using the "cryptography.fernet" module	**Practical Exam 1**

Week 6	**ASYMMETRIC CRYPTOGRAPHY** ● Introducing the Key Distribution problem Naive RSA and complexity of operations Encryption, Decryption and Key Generation Weakness of Naive RSA Padded RSA, OAEP	Text1, Chap 5: Asymmetric Encryption. Sections 5.1 and 5.2	**LAB6** Openssl : RSA key generation using openssl, extract RSA pem keys, encryption/decryption Python : use "cryptography.hazmat" to generate RSA keys, public/private key serialization, encrypt/decrypt with OAEP-padding RSA	
Week 7	**HASH FUNCTIONS** ● Defining hash functions ● Cryptographic hash function properties (one-way, weak collision, strong-collision) ● Verifying data integrity with hashing ● Choosing a cryptographic hash function ● Using the hashlib module for cryptographic hashing	Text1, Chap 2: Hashing	**LAB7** Built-in "hash" function of Python Use of Hashlib module to implement SHA256, MD5 to hash a message Use digest(), hexdigest() use update() and base64	
Week 8	**KEYED HASHING AND HMAC** ● Pseudo Random Number Generators Verifying data authentication with keyed hashing The HMAC primitive Using the hmac module for cryptographic hashing of documents in transit	Text1, Chap 3: Keyed Hash	**LAB8** - Random number generation using the modules random (random, randint, uniform,sample, shuffle), os.urandom and secrets - Hash a massage + passphrase - Use the hmac module - Simulate a hmac communication between two users (use a json file to store the message+digest)	**Midterm**

Week 9	**IMAGE MANIPULATION FOR CRYPTOGRAPHY** Images representation Gray Scale Images RGB images Images formats Image basic manipulations	Text3, Chap 1: Getting Started with Image Processing	**LAB9** - Read, write, and display an image using “Matplotlib” cropp, resize, negate, convert to/from RGB/Gray, rotate, affine transformation, change pixel values, add noise, mask, draw in images compute image statistics, histograms, separating RGB channels, compute images difference, image similarity	
Week 10	**IMAGE CRYPTOGRAPHY AND STEGANOGRAPHY** Image Cryptography Data hiding and Steganography Digital Watermarking techniques	Text2, Chap 6: Using Cryptography with Images	**LAB10** - Use "Fernet" to implement image encryption - use "Pycryptodome" to implement AES-ECB, AES-CBC image encryption - use "cryptosteganography" to hide a text file into an image. - use “cryptosteganography" to hide an mp3 file into an image.	
Week 11	**DIGITAL SIGNATURES AND CERTIFICATES** ● Non repudiation Problem ● Digital Signature Generation and Verification scheme ● RSA digital signature ● X.509 Format certificates	Text1, Chap5: Asymmetric Encryption, section 5.3 : Non-Repudiation. + Chap 6: TLS. section 6.3.3 Server authentication : public certificates	**LAB11** <u>Openssl :</u> - decode a digital certificate with openssl. the certificate can be loaded from a secure web site using python or openssl <u>Python :</u> - Use “cryptography.hazmat” to sign/verify a message.	

Week 12	**KEY MANAGEMENT OF SYMMETRIC CRYPTOGRAPHY** ● Diffie-Hellman Key Exchange ● Digital envelope ● KDC	Text2, Chapter 8: Cryptographic applications and PKI pp 242-245	**LAB12** - Use Diffie-Hellman to share a secret key and use it in a symmetric encryption/decryption algorithm - implement a simple PKI and use it.	**Practical Exam 2**
Week 13	**ENCRYPTING DATA IN TRANSIT** ● pure practical work to implement lab13	Text2, Chap 7: Message Integrity. Section :Sending Secure Messages Over IP Networks" pp212-	**LAB13** - Use the "socket" module to create UDP sockets Use "cryptography.fernet" module to incorporate encryption to sockets communications between client and server	
Week 14 & 15	**Project Presentation & Course Review**			

Assessment methods	LO #1	LO #2	LO #3	LO #4	LO #5	LO #6	LO #7
Practical Exam I 10%	x	x					x
Midterm 20%	x		x	x	x		
Project 20%	x	x		x	x	x	x

Practical Exam II 10%	x	x					x
Final 40%	x		x	x	x		x