Semester VIII

COMP 8011: Network Security & Cryptography (Theory)

Course Objective:

The core objectives of a Network Security & Cryptography course are to provide a foundational understanding of data protection, enabling students to understand, analyze, and implement cryptographic algorithms, security protocols (SSL/TLS, IPSec), and defense mechanisms against network attacks. The course focuses on achieving confidentiality, integrity, availability, and non-repudiation.

Course Learning Outcomes:

After undergoing the course, Students will be able to:

- To understand basics of Cryptography and Network Security.
- To be able to secure a message over insecure channel by various means.
- To learn about how to maintain the Confidentiality, Integrity and Availability of a data.
- To understand various protocols for network security to protect against the threats in the networks.

Credit: 04 **60 Hours**

Syllabus

Unit I: Introduction to security, brief history of cryptography, understanding attacks, services, mechanisms, security attacks, security services, model for network security, internet standards, Basic principles of good cryptosystems, Modular arithmetic and GF (2), substitution and transposition ciphers, Stream ciphers, Pseudo Random Number generators. [6Hrs]

Unit II: Symmetric block ciphers, DES, Gallois field, polynomial arithmetic, AES, Principles of S Box design, Block cipher design principles, Other Block ciphers. [6Hrs]

Unit III: Introduction to public key cryptography, Number theoretic foundations, public key cryptography principles, RSA encryption system, primality testing, Number theoretic algorithms, Attacks on RSA. [12 Hrs]

Unit IV: Discrete Logarithm and Diffie-Hellman Key exchange, ElGamal system, Digital Signature, RSA and ElGamal based Digital signature, DSA, different attacks on digital signatures. [12 Hrs]

Unit V: Secure hash functions, Understanding Collisions, Secure Hash Algorithms, SHA, HMAC, Key establishment and key management, Digital Certificates. [6Hrs]

Unit VI: Advanced topics on cryptography: Integer factorization, Strong primes, distribution of Primes, Lattice based cryptography, Introduction to Elliptic curves, Introduction to Elliptic curve-based cryptography,

Secret Sharing, Zero Knowledge Proof. [10Hrs]

Unit VII: Authentication, e mail security, Kerberos, x.509 directory authentication services, PGP, S/MIME. Overview of IP security, IP security architecture, authentication header, encapsulating security pay load, combing security associations. [4 Hrs]

Unit VIII: Introduction to web security, web security requirements, SSL & transport layer security, SET network management security. Basics of system security, intruders, viruses-related threats, firewalls, design principles, trusted systems. [4Hrs]

COMP 8011: Network Security & Cryptography (Practical)

Credit: 02 **60 Hours**

1. Implement RSA Algorithm.
2. Implement Elliptic curve-based cryptography
3. Implement SSL/TLS for secure browser communication using OpenSSL
4. Implement Digital Signature Algorithm (DSA) .
5. Implement SSL/TLS for secure browser communication using OpenSSL
6. Setting up Firewall rules using iptables in Linux
7. Implement Secure Hash Algorithm implementation
8. Implement Primality testing using Farmet's theorem.
9. Implement Miller-Rabin Test.
10. Find inverse of an element in a ring using EEA.
11. Implement Modular Arithmetic and Polynomial Arithmetic.

COMP 8012: Computer Graphics (Theory)

Course Objectives

Computer Graphics course aims to equip students with the theoretical foundation and practical skills to create, manipulate, and render 2D and 3D images. Key objectives include mastering graphics pipelines, geometric transformations, rendering techniques, and using libraries like OpenGL to build interactive, animated, and realistic digital environments.

Course Learning Outcomes:

After undergoing the course, Students will be able to:

- Define the basic elements of computer graphics and identify its diverse applications across various industries.
- Compare and contrast the architectures of Raster and Random scan display devices and understand the integration of input/output hardware.
- Implement and analyze algorithms for scan conversion of lines, circles, ellipses, and polygon filling, including the handling of thick primitives.
- Apply mathematical principles to perform 2D and 3D transformations and execute line and polygon clipping techniques.
- Construct Viewing Transformations, including Parallel and Perspective projections, while accounting for vanishing points to create 3D depth.

Credit: 03 **45 Hours**

1. Introduction (5 Hrs)
Basic elements of Computer graphics, Applications of Computer Graphics.

2. Graphics Hardware (8 Hrs)
Architecture of Raster and Random scan display devices, input/output devices.

3. Fundamental Techniques in Graphics (10 Hrs)
Raster scan line, circle and ellipse drawing, thick primitives, Polygon filling, line and polygon clipping algorithms, 2D and 3D Geometric Transformations, 2D and 3D Viewing Transformations (Projections- Parallel and Perspective), Vanishing points.

4. Geometric Modeling (8 Hrs)
Representing curves & Surfaces.

5. Visible Surface determination (8 Hrs)
Hidden surface elimination.

6. Surface rendering (6 Hrs)
Illumination and shading models. Basic color models and Computer Animation

Books Recommended:

1. J.D.Foley, A.Van Dan, Feiner, Hughes Computer Graphics Principles & Practice 2nd edition Publication Addison Wesley 1990.
2. D.Hearn, Baker: Computer Graphics, Prentice Hall of India 2008.
3. D.F.Rogers Procedural Elements for Computer Graphics, McGraw Hill 1997.
4. D.F.Rogers, Adams Mathematical Elements for Computer Graphics, McGraw Hill 2nd edition 1989.

COMP 8012: Computer Graphics (Practical)

Credit: 01 **30 Hours**

All programs should be developed using C / Java / Python

1. Write a program to implement DDA and Bresenham's line drawing algorithm.
2. Write a program to implement mid-point circle drawing algorithm Bresenham's circle drawing algorithm.
3. Write a program to clip a line using Cohen and Sutherland line clipping algorithm.
4. Write a program to clip a polygon using Sutherland Hodgeman algorithm.
5. Write a program to apply various 2D transformations on a 2D object (use homogenous coordinates).
6. Write a program to apply various 3D transformations on a 3D object and then apply parallel and perspective projection on it.
7. Write a program to draw Hermite/Bezier curve.

COMP 8013: Machine Learning

Credit:03 **45 Hrs**

Course Objectives:

Machine learning courses aim to provide a foundational understanding of algorithms that allow computers to learn from data, identify patterns, and make predictions without explicit programming. Key objectives include mastering supervised/unsupervised learning, implementing models using Python and applying these techniques to real-world problems.

Course Learning Outcomes:

(i) Define and explain machine learning concepts, types, and basic metrics.
(ii) Implement and apply supervised learning techniques (e.g., KNN, Linear Regression, Logistic Regression).
(iii) Apply unsupervised learning methods (e.g., K-Means, Hierarchical Clustering, Association Rules).
(iv): Develop and evaluate simple machine learning models (e.g., Perceptron, single-layer,neural networks).
(v) Analyze and apply appropriate machine learning algorithms depending on the problems with some real-world data.

(vi) Implement and evaluate supervised learning techniques, including K-Nearest Neighbours, linear regression, and logistic regression, and measure model performance using accuracy, precision, recall, and F1 score.

Syllabus

UNIT I: Introduction to Machine Learning(20 Hrs)
Introduction: Definition, History and Application of Machine Learning,
Types of Machine Learning: Supervised, Unsupervised, Semi-Supervised, and Reinforcement Learning.
Labelled and Unlabelled Dataset, Over fitting- under fitting, balanced and imbalanced data sets.
Supervised Learning Tasks: Regression vs. Classification,
Learning Framework: Training, Validation and Testing of ML models. *Performance Evaluation Parameters*: Confusion matrix, Accuracy, Precision, Recall, F1 Score, and AUC.

UNIT II: Supervised Learning and Unsupervised Learning(25 Hrs)
Regression: Linear and non-linear Regression, Logistic Regression.
Classification: Naïve Bayes, K-Nearest Neighbors, Decision Trees.
Models: Introduction to Artificial Neural Networks, Perceptron Learning Algorithm, Single Layer Perceptron, Introduction to Support
Vector Machine for linearly separable data.
Clustering: K-Means, Hierarchical Clustering,
DBSCAN, Clustering Validation Measures.
ML Applications: Ethical Considerations in Machine Learning, Case study and Real-world Applications.

Reference Books:
1. Rajiv Chopra (2024), Machine Learning and Machine Intelligence, Khanna Publishing House.
2. Jeeva Jose (2023), Introduction to Machine Learning, Khanna Publishing House.
3. Mitchell T. (1997). Machine Learning, First Edition, McGraw-Hill.
4. Kalita, J. K., Bhattacharyya, D. K., & Roy, S. (2023). Fundamentals of Data Science: Theory and Practice. Elsevier. ISBN9780323917780
5. Flach, P. A. (2012). Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press. ISBN: 9781107422223, 2012.
6.. Duda, R. O., Hart, P. E., Stork, D (2007). Pattern classification (2Ed), John Wiley & Sons, ISBN-13: 978-8126511167.
7. Haykin S. (2009). Neural Networks and Learning Machines, Third Edition, PHI Learning.
8. Chollet, F. (2018). Deep Learning with Python. Manning Publications.

Machine Learning (Practical)

Credit:1 **30 Hrs**

LAB Experiments (Program should be developed using Python)

The lab experiments may be implemented in Python using relevant ML libraries, and datasets from Kaggle, public repositories, or generated datasets.

Suggested list of Experiments:

1. Implement linear regression on a dataset and visualize the regression line.
2. Implement logistic regression on a binary classification dataset and plot the decision boundary.
3. Implement and evaluate the performance of Decision tree ID3/Cart classifier for any given dataset.
4. Implement and evaluate the performance of the Naive Bayes Classifier on a given dataset.
5. Build and evaluate a random forest classifier using a numerical dataset.
6. Implement a support vector machine for linearly separable classes and visualize the margins and decision boundary.
7. Implement K-Means clustering on a point dataset and visualize and evaluate the clusters.
8. Implement hierarchical clustering on a dataset and plot the dendrogram.
9. Implement DBSCAN clustering on a dataset and visualize and evaluate the clusters.
10. Perform Principal Components Analysis (PCA) and apply any one or more classifiers to show the performance variation with or without feature reduction.
11. Build a single layer perceptron model to classify AND, OR, and XOR problems (may use TensorFlow/Keras) and visualize their decision boundaries. Also evaluate its performance.
12. Demonstrate the concept of boosting using the AdaBoost algorithm.

COMP 8014 System Programming using UNIX/Linux

Course Objective:

The primary objective of a System Programming using UNIX/Linux course is to provide students with a deep understanding of the internal mechanisms of UNIX-like operating systems and the skills to develop system-level software that interacts directly with the kernel and hardware using system calls and library functions.

Course Learning Outcome

i) Proficiency in the Linux Environment
ii) Mastery of Shell Scripting
iii) Understanding Operating System Concepts
iv) Understanding Application of System Calls and APIs
v) Implementation of Inter-Process Communication (IPC)

vi) Concurrent Programming and Networking

Credit: 3 **45 Hrs**

Unit I: Introduction to UNIX/Linux (4 Hours)

History and philosophy of UNIX, GNU and Linux evolution, UNIX architecture (kernel, shell, system utilities), user space and kernel space, system call interface, Linux file system hierarchy, basic UNIX commands (ls, cp, mv, rm, cat, grep, wc, awk etc.), I/O redirection and pipes, file permissions and ownership, environment variables, types of shells, shell initialization files.

Unit II: UNIX Shell Programming (10 Hours)

Introduction to shell scripting, structure of shell script, execution permissions, variables and environment variables, command substitution, arithmetic operations (expr, let, bc), conditional statements (if, if-else, case), looping constructs (for, while, until), break and continue, positional parameters, shift command, functions in shell, arrays, string operations, test command and file testing, input/output in scripts, trap and signal handling in shell, writing practical automation scripts.

Unit III: C Programming for System Programming (5 Hours)

Compilation process (preprocessing, compilation, linking), GCC compiler options, structure of C program in Linux, memory layout of C program (text, data, bss, heap, stack), pointers and dynamic memory allocation, file handling in C, low-level vs high-level I/O, static and shared libraries, Makefile basics, debugging using gdb, memory debugging using valgrind.

Unit IV: System Calls and File Management (5 Hours)

Concept of system calls, kernel mode switching, file descriptors, open(), close(), read(), write(), lseek(), stat(), fstat(), dup(), dup2(), directory handling (opendir(), readdir(), closedir()), file and directory permissions, chmod(), chown(), umask().

Unit V: Process Management (5 Hours)

Concept of process and process states, process control block, fork(), exec() family, wait() and waitpid(), exit() and _exit(),process scheduling basics, process identifiers (getpid(), getppid())

Unit VI: Inter-Process Communication (IPC) (5 Hours)

Need for IPC, unnamed pipes, named pipes (FIFO), message queues (System V and POSIX), shared memory, semaphores, synchronization between processes, introduction to sockets, client-server model basics.

Unit VII: Threads and Memory Management (5 Hours)

Processes vs threads, POSIX threads (pthread_create, pthread_join), thread synchronization using mutex, condition variables, race conditions and deadlocks, virtual memory concept, paging and segmentation, dynamic memory allocation (malloc(), free(), brk(), sbrk()), memory leaks and buffer overflow.

Unit VIII: File Systems, Security and Advanced Topics (6 Hours)

File system structure and inodes, hard and symbolic links, ext4 overview, mounting and unmounting file systems, file security and access control.

Reference Books

1. Advanced Programming in the UNIX Environment — W. Richard Stevens & Stephen A. Rago
2. *"Unix system administration" Maxwell – TMH.*
3. Sumitabha, Das, Unix Concepts and Applications, Tata McGraw-Hill Education, 2006
4. Michael Jang RHCSA/ RHCE Red Hat Linux Certification: Exams (Ex200 & Ex300) (Certification Press), 2011
5. Nemeth Synder & Hein, Linux Administration Handbook, Pearson Education, 2nd Edition ,2010
6.W. Richard Stevens, Bill Fenner, Andrew M. Rudoff, Unix Network Programming, The sockets Networking API, Vol. 1, 3rd Edition,2014

System Programming using UNIX/Linux (Practical)

Credit:1 **30 Hrs**

1. Write a shell script to check if the number entered at the command line is prime or not.
2. Write a shell script to modify ―cal‖ command to display calendars of the specified months.
3. Write a shell script to modify ―cal‖ command to display calendars of the specified range of months.
4. Write a shell script to accept a login name. If not a valid login name display message – Entered login name is invalid‖.
5. Write a shell script to display date in the mm/dd/yy format.
6. Write a shell script to display on the screen sorted output of ―who‖ command along with the total number of users.
7. Write a shell script to compare two files and if found equal asks the user to delete the duplicate file.
8. Write a shell script to find the sum of digits of a given number.
9. Write a shell script to merge the contents of three files, sort the contents and then display them page by page.
10.Develop UNIX shell scripts using variables, control structures, and functions for automation tasks.
11.Write C programs using system calls for file handling and low-level I/O operations.
12.Implement process management concepts using fork(), exec(), wait(), and signals.
13.Design concurrent and IPC-based programs using pipes, shared memory, semaphores, and threads.

Minor Course

COMP 8021: Introduction to Machine Learning

Course Objectives:
Machine learning courses aim to provide a foundational understanding of algorithms that allow computers to learn from data, identify patterns, and make predictions without explicit programming. Key objectives include mastering supervised/unsupervised learning, implementing models using Python and applying these techniques to real-world problems.

Course Learning Outcomes:

(i) Define and explain machine learning concepts, types, and basic metrics.
(ii) Implement and apply supervised learning techniques (e.g., KNN, Linear Regression, Logistic Regression).
(iii) Apply unsupervised learning methods (e.g., K-Means, Hierarchical Clustering, Association Rules).
(iv): Develop and evaluate simple machine learning models (e.g., Perceptron, single-layer neural networks).
(v) Analyze and apply appropriate machine learning algorithms depending on the problems with some real-world data.
(vi) Implement and evaluate supervised learning techniques, including K-Nearest Neighbours, linear regression, and logistic regression, and measure model performance using accuracy, precision, recall, and F1 score.

Credit:03 **45 Hrs**

Syllabus

UNIT I: Introduction to Machine Learning(20 Hrs)
Introduction: Definition, History and Application of Machine Learning,
Types of Machine Learning: Supervised, Unsupervised, Semi-Supervised, and Reinforcement Learning. Labelled and Unlabelled Dataset.
Supervised Learning Tasks: Regression vs. Classification,
Learning Framework: Training, Validation and Testing of ML models.
Performance Evaluation Parameters: Confusion matrix, Accuracy, Precision, Recall, F1 Score, and AUC.

UNIT II: Supervised Learning and Unsupervised Learning(25 Hrs)
Regression: Linear and non-linear Regression, Logistic Regression.
Classification: NaïveBayes, K-Nearest Neighbors, Decision Trees.
Models: Introduction to Artificial Neural Networks, Perceptron Learning Algorithm, Single Layer Perceptron, Introduction to Support Vector Machine for linearly separable data.
Clustering: K-Means, Hierarchical Clustering,DBSCAN, Clustering Validation Measures.
ML Applications: Ethical Considerations in Machine Learning, Case study and Real-world Applications.

Reference Books:

1. Rajiv Chopra (2024), Machine Learning and Machine Intelligence, Khanna Publishing House.
2. Jeeva Jose (2023), Introduction to Machine Learning, Khanna Publishing House.
3. Mitchell T. (1997). Machine Learning, First Edition, McGraw-Hill.
4. Kalita, J. K., Bhattacharyya, D. K., & Roy, S. (2023). Fundamentals of Data Science: Theory and Practice. Elsevier. ISBN9780323917780
5. Flach, P. A. (2012). Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press. ISBN: 9781107422223, 2012.
6.. Duda, R. O., Hart, P. E., Stork, D (2007). Pattern classification (2Ed), John Wiley & Sons, ISBN-13: 978-8126511167.
7. Haykin S. (2009). Neural Networks and Learning Machines, Third Edition, PHI Learning.
8. Chollet, F. (2018). Deep Learning with Python. Manning Publications.

Introduction to Machine Learning (Practical)

Credit:1 **30 Hrs**

LAB Experiments(Program should be developed using Python)

The lab experiments may be implemented in Python using relevant ML libraries, and datasets from Kaggle, public repositories, or generated datasets.

Suggested list of Experiments:

1. Implement linear regression on a dataset and visualize the regression line.
2. Implement logistic regression on a binary classification dataset and plot the decision boundary.
3. Implement and evaluate the performance of Decision tree ID3/Cart classifier for any given dataset.
4. Implement and evaluate the performance of the Naive Bayes Classifier on a given dataset.
5. Build and evaluate a random forest classifier using a numerical dataset.
6. Implement a support vector machine for linearly separable classes and visualize the margins and decision boundary.
7. Implement K-Means clustering on a point dataset and visualize and evaluate the clusters.
8. Implement hierarchical clustering on a dataset and plot the dendrogram.
9. Implement DBSCAN clustering on a dataset and visualize and evaluate the clusters.
10. Perform Principal Components Analysis (PCA) and apply any one or more classifiers to show the performance variation with or without feature reduction.
11. Build a single layer perceptron model to classify AND, OR, and XOR problems (may use TensorFlow/Keras) and visualize their decision boundaries. Also evaluate its performance.
12. Demonstrate the concept of boosting using the AdaBoost algorithm.