CS1811 - Object Oriented Programming I

Prerequisites: None

Co-ordinator: Prof. Adrian Johnstone

Course value: 15 credits

COURSE SUMMARY

This course teaches programming and object-orientation concepts. Students will learn about program basics (variables, types, scope, lifetimes), control flow (if-constructs, for-loops, while-loops), data structures (strings, sets, lists, trees), objects (classes and inheritance), exceptions (throwing and catching) and file I/O (streams, file reading and writing).

LEARNING OUTCOMES

By the end of this course a student should be able to:

- solve basic programming tasks
- understand and use basic object-oriented concepts
- demonstrate familiarity with core elements of the Java API

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes. Normally 3 hours of lectures and 2 hours of laboratory classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

A zero weighted test for which marks will be returned. Verbal feedback in laboratory sessions. Students must pass the laboratory exercises component to pass the course.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS1812 - Object Oriented Programming II

Prerequisites: CS1811

Co-ordinator: Dr. Matthew Hague

Course value: 15 credits

COURSE SUMMARY

This course teaches programming and object-orientation concepts, building on what is taught in CS1811. Students will reinforce their knowledge about program basics (e.g. variables, types, scope, lifetimes), algorithms (e.g. recursion, searching, sorting), data structures (e.g. sets, lists, trees), objects (e.g. classes, dynamic dispatch, interface, inheritance, generics), exceptions and I/O (e.g. streams, file reading, writing and copying). Furthermore, the course also teaches fundamentals of coding best practices.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- demonstrate an ability to implement basic algorithms and data structures
- understand and use object-oriented concepts
- appreciate the need for program documentation, testing, readability and modifiability

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes. Normally 2 hours of lectures and 3 hours of laboratory classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback in laboratory sessions. Students must pass the laboratory exercises component to pass the course.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS1813 - Software Development

Prerequisites: CS1811

Co-ordinator: Dr. Matthew Hague / Dr. Carlos Matos

Course value: 15 credits

COURSE SUMMARY

This course teaches programming and object-orientation concepts, building on what is taught in CS1811. Students will reinforce their knowledge about program basics (e.g. variables, types, scope, lifetimes), algorithms (e.g. recursion, searching, sorting), data structures (e.g. sets, lists, trees), objects (e.g. classes, dynamic dispatch, interface, inheritance, generics), exceptions and I/O (e.g. streams, file reading, writing and copying). Furthermore, the course also teaches fundamentals of coding best practices.

In addition, students learn software development best practices, how to perform small scale software development from a specification, and details about the testing process and software evaluation.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- demonstrate an ability to implement basic algorithms and data structures
- understand and use object-oriented concepts
- appreciate the need for program documentation, testing, readability and modifiability
- develop a simple application based on a design specification
- create a test plan, execute it, and identify faults in an application

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes. Normally 2 hours of lectures and 2 hours of laboratory classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback in laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS1820 - Computing laboratory (robotics)

Prerequisites: None

Co-ordinator: Prof. Dave Cohen

Course value: 15 credits

COURSE SUMMARY
To provide a practical approach to programming and the building of computer systems.
To introduce elementary robotics concepts
To reinforce the earning of elementary programming

Course Content:
- Lego NXT system: control brick capabilities, motors, sensor classes
- Sensor operation: light sensor, ultrasonic sensors, pushbuttons
- Simple movement: motors, gearing, idler wheels, tracked vehicles
- Actuators: grippers, extensible arms
- Projects: a sequence of scripted exercises, with opportunities for student led expansion

LEARNING OUTCOMES
By the end of this course a student should be able to:
- Programme a mobile robot to execute pre-defined movements
- Understand the operation of basic sensors
- Understand the essentials of real time event-driven programming

TEACHING AND LEARNING METHODS
Teacher led laboratory classes and self-driven, scripted group project work. Up to 4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

FORMATIVE ASSESSMENT AND FEEDBACK
Five formative worksheets. Verbal feedback on progress will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT
Details of coursework submission deadlines will be published on the department website at the start of term.
CS1830 - Computing laboratory (games)

Prerequisites: None

Co-ordinator: Mr. Nuno Barreiro

Course value: 15 credits

COURSE SUMMARY
During this course, students learn some basic concepts of 2D game design and apply them to the development of simple games. To achieve that goal, they learn about 2D graphics (sprites and spritesheets), animations (loops and sequences) and motion (position, velocity, acceleration). All those concepts require vectors, which are also extensively covered throughout the course. Furthermore, the course introduces basic concepts of game physics, such as collisions gravity and ballistics. The development should follow an object-oriented approach, and that topic is an important component of the course.

LEARNING OUTCOMES
By the end of this course a student should be able to:

- Write simple 2D games program
- Understand the role of vectors in game development
- Produce state diagrams and use them in the development of simple games
- Use 2D gaming graphics: sprites, spritesheets and animations
- Understand and program gaming physics: collision detection and handling, gravity and ballistics

TEACHING AND LEARNING METHODS
Teacher led laboratory classes and self-driven project work. Up to 4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY
- “Essential Mathematics for Games and Interactive Applications” by James M. Van Verth and Lars M. Bishop
- Ian Millington – Artificial Intelligence for Games
- Ian Millington – Games Physics Engine Development
- David Eberly – Game Physics
- Christer Ericsson – Real Time Collision Detection
- Extra textbooks are available at the following web sites:
  - http://programarcadegames.com
  - http://inventwithpython.com

FORMATIVE ASSESSMENT AND FEEDBACK
Practical skills assessment – in Lab assessment. Verbal feedback on progress will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT
Details of coursework submission deadlines will be published on the department website at the start of term.
CS1840 - Internet Services

Prerequisites: None

Co-ordinator: Prof Zhaohui Luo

Course value: 15 credits

COURSE SUMMARY

The course introduces the basics of how the Internet works and its key protocols. With that foundation, it then addresses the technologies used for web development, including scripting languages and their potential. Web Services and related technologies are also introduced as well as the basics of network security.

Course Content:

- Internet Basics: protocols, delay and loss.
- Web Technologies: HTML and related technologies
- Scripting languages: Client and server side scripting
- Web Services: introduction and related technologies.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- understand the basics of how the Internet works
- understand key Internet protocols
- carry out web development tasks involving current web technologies
- use scripting languages to add dynamic aspects to web sites/applications
- understand the fundamentals of Web Services and network security

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes, guided independent study. Up to 3 hours of lectures and laboratory classes per week

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback on progress will be given during the weekly laboratory sessions

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS1860 - Mathematical structures

Prerequisites: None

Co-ordinator: Prof. José Fiadeiro

Course value: 15 credits

COURSE SUMMARY

The aim of this module is to provide (1) insights and skills in rigor and formal reasoning in a way that allows reasoning about behaviour, correctness and performance in a programming environment; and (2) basic knowledge of the formal structures for program data representation. The course content includes: structures such as sets, relations, functions and cardinality; recursion and proof by induction; graphs; basic probability and statistics.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- reason about sets, relations, functions and cardinality
- reason about recursive definitions and prove results by induction
- represent problems and reason about them using graphs
- understand basic probability and statistics suitable for use in studying artificial intelligence and information security

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by small group tutorials. Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Individual discussion of non-assessed tutorial exercises and discussion of assessed worksheets in small group tutorials.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS1870 - Machine fundamentals

Prerequisites: None

Co-ordinator: Prof. Elizabeth Scott

Course value: 15 credits

COURSE SUMMARY

To explain the theory and use of logic in the description, specification and behaviour of machine processes.

To provide insights and skills for dealing with large and infinite objects in a way that allows them to be implemented in a programming environment.

Course Content:

- Numbers: binary number systems, two’s complement notation.
- Networks: series and parallel switching circuits, network minimisation.
- Low level languages: assembly language programming.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- use formal logic to design, reason about and minimise switching circuits
- write basic programs in assembly language
- understand binary representations of signed and unsigned integers
- write regular expressions to describe sets and build deterministic automata to recognise these sets
- use automata to design and reason about sequential flow systems

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes and group tutorials. Normally 3 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Individual discussion of non-assessed tutorial exercises and discussion of assessed worksheets in small group tutorials.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS1890 - Software Design

Prerequisites: None

Co-ordinator: Dr. Carlos Matos

Course value: 15 credits

COURSE SUMMARY

The course addresses concepts required for performing software design activities. Core course content aims at interpreting requirements, identifying software components, documenting software design and understanding the various stages of software development, as well as techniques with a focus on software design.

Course Content:

- This course will emphasise problem based learning.
- Students will progress through case studies in critiquing software, acquiring and expressing requirements, designing software and documenting their designs.
- Students will learn to use current industry standard notations such as User Stories and UML.
- Students will see how design is achieved in various current Software Engineering processes, including the waterfall and agile processes.

LEARNING OUTCOMES

- Identify common software requirements and how these map to software components. They will be able to recognise how these requirements have been discharged in existing systems and critique their effectiveness.
- Understand several techniques and notations that make it possible to document software design. They will understand that Software Engineering supports communication of design ideas and this will allow them to see how Software Engineering is a team activity.
- Understand the importance of the several activities of a professional software engineer. This will include techniques from agile software development, but will focus on requirements acquisition and software design.
- Apply several techniques to design software based on user requirements. They will be able to judge the appropriateness of designs produced using these techniques, both formally and informally.
- Analyse and critique the design of existing software. This will include the User Experience of software as a measure of its fitness for purpose.

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes. Normally 3-4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback during lab sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2800 - Software engineering

Prerequisites: CS1811; CS1812 or CS1813

Co-ordinator: Prof. Dave Cohen

Course value: 15 credits

COURSE SUMMARY

Introducing software Engineering tools and techniques through practical experience of design and development that enable each individual programmer to contribute effective, working, documented code, as part of a team, in a timely fashion.

Course Content:

- Software engineering: models of software development, planning.
- Object oriented design: Notation for design, identifying objects, classes, attributes and methods. Class relationships, design patterns
- Programming methodologies: program structure, style, and layout. Coding standards, test driven development
- Testing: Program analysis, black and white box testing, defensive programming, system, integration and acceptance testing.
- Use of a variety of modern software engineering tools: in particular version control, debugger, code style checkers, junit and how these can be integrated into an industry standard IDE.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- understand the software engineering techniques and managerial discipline required to work as part of a team
- understand and use basic object-oriented concepts
- appreciate the need for program documentation, testing, readability and modifiability
- use appropriate tools to support software development: Version control, programming standards, a modern IDE,
- be able to use test driven development to deliver a small scale project.

TEACHING AND LEARNING METHODS

Lecture based delivery with class exercises and worksheets. Use of structured practice sessions. Up to 4 hours of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Feedback will be given based on questions on the Moodle forum and any common mistakes made in coursework.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2810 - Team Project

Prerequisites: CS1811; CS1812 or CS1813; CS2800

Co-ordinator: Dr. Carlos Matos / Dr. Matthew Hague

Course value: 15 credits

COURSE SUMMARY

This course aims to assist students in appreciating the role of the computer professional through the practical experience of developing medium scale software as part of a team.

Course content:

The software lifecycle, including: software development, planning and documentation.

Team development, communication, managing risks and conflicts.

Practical experience of standard industrial software engineering.
Agile project management, use of version control in a team, IDEs, etc.

A medium scale agile software development project, conducted by teams of students.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- work in a team to produce a substantial product using software engineering techniques
- apply managerial discipline and a professional attitude.
- understand why project cost and effort is hard to estimate and why project quality is hard to prescribe
- design software following an agile process

TEACHING AND LEARNING METHODS

Up to 4 hours of sessions per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be provided during the team meetings. Written feedback based on team deliverables.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2815 - Small enterprise team project

Prerequisites: CS1813; CS2800

Co-ordinator: Dr. Carlos Matos / Dr. Matthew Hague

Course value: 15 credits

COURSE SUMMARY

This course aims to assist students in appreciating the role of the computer professional through the practical experience of developing medium scale software as part of a team.

Course content:

The software lifecycle, including: software development, planning and documentation.

Team development, communication, managing risks and conflicts.

Practical experience of standard industrial software engineering.

Agile project management, use of version control in a team, IDEs, etc.

A medium scale agile software development project, conducted by teams of students.

Meetings with an external customer to determine the requirements for a product.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- work in a team to produce a substantial product using software engineering techniques
- apply managerial discipline and a professional attitude.
- understand why project cost and effort is hard to estimate and why project quality is hard to prescribe.
- design software following an agile process
- understand an external customer's requirements for a software product

TEACHING AND LEARNING METHODS

Student led team work and meetings with the team supervisor and the client.

Up to 4 hours of sessions per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be provided during the team meetings. Written feedback based on team deliverables.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2850 - Operating Systems

Prerequisites: CS1811; CS1812 or CS1813

Co-ordinator: Dr Carlos Matos / Prof Johannes Kinder

Course value: 15 credits

COURSE SUMMARY

This course aims to introduce students to the principles of the function and architecture of operating systems, and also to give an understanding of how programs operate at system level. Course content includes:

Introductory topics: role of an operating system, computer architecture

Processes and threads: process management and scheduling, inter-process communication, concurrency

Memory: partitioning, swapping and paging, caching, virtual memory, page replacement algorithms

File systems: implementation and maintenance

UNIX shell: starting programs, input and output streams, pipes, filters, utilities

System-level programming: memory handling, processes, threads, synchronisation, I/O

LEARNING OUTCOMES

By the end of this course a student should be able to:

- demonstrate an understanding of the principles of computer operating systems
- evaluate the theory and practice of existing operating systems
- demonstrate a working understanding of program execution, memory hierarchy, and the implementation of data structures
- understand system-level programming aspects such as memory management, interrupts, sockets and basic threading in C
- write simple shell scripts

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by practical sessions. Normally 4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Written grades will be given for the assessed assignments.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2855 - Databases

Prerequisites: CS1811; CS1812 or CS1813

Co-ordinator: Dr Iddo Tzameret

Course value: 15 credits

COURSE SUMMARY

To provide the basic concepts of database technology.

To describe the need for database integrity and robustness.

To demonstrate the use of a modern database system in a web-based environment.

Course Content:

- Data modelling: views, subschema, data dictionary, data independence, entity relationship model.
- The relational model: relations, attributes, domains, relational algebra.
- Database design: normalisation, normal forms, entities and attributes
- SQL: basic SQL, correspondence between the relational model and SQL commands, simple queries, combination and sub-queries
- Administration and implementation: integrity, recovery from failure, concurrency, deletion and updating, forms, report writing. Programmatic access to SQL databases.

LEARNING OUTCOMES

- explain the issues involved in database design and the theory of the relational view of data
- describe the crucial issues concerning database integrity and recovery from failure
- write SQL queries
- be familiar with the steps for the design and implementation of a database, from the user specifications to the final design
- implement an interface to an SQL database using an API

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes, guided independent study. Up to 4 hours of lectures and laboratory classes per week

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback on progress will be given during the weekly laboratory sessions

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2860 - Algorithms and Complexity

Prerequisites: CS1811; CS1812 or CS1813; CS1860

Co-ordinator: Prof Gregory Gutin

Course value: 15 credits

COURSE SUMMARY

The main aim of the course is to teach the design of algorithms and data structures from the point of view of time and space complexity. This includes covering sorting and search algorithms, and graphs. The course content includes the following:

Complexity: counting, big-O notation, best-case, worst-case and average-case analysis.

Basic algorithms, sorting and searching part: implementation and analysis of linear search, binary search, and basic sorting algorithms, especially insertion sort, selection sort, merge sort, quick sort, heap sort.

Data structures: binary search trees, balanced binary search trees, hash tables, (binary) heaps.

Abstract datatypes: Sets, maps, priority queues.

Basic algorithms, graph algorithms part: adjacency matrix and adjacency list representations; algorithms for connectivity, shortest paths, and spanning trees.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- understand and reason about alternative data structure representations, and their use in programs
- implement and reason about alternative implementations for basic algorithms, including graph algorithms
- calculate the complexity of basic algorithms, including graph algorithms

TEACHING AND LEARNING METHODS

Lecture based delivery.
Normally 3 hours of sessions per week.

KEY BIBLIOGRAPHY

  Available at http://www.mpi-inf.mpg.de/~mehlhorn/Toolbox.html

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be given during the laboratory sessions and feedback provided during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2900 - Multi-dimensional Data Processing

Prerequisites: CS1830; CS1860; cannot be taken with MT1820

Co-ordinator: Dr Hugh Shanahan

Course value: 15 credits

COURSE SUMMARY

During this course the student will be introduced to a progression of topics in Linear Algebra, starting from an introduction to vectors and matrices, moving to Singular Value Decomposition, the solution of linear equations and least squares, and then to eigenvalues and eigenvectors. The emphasis of this course will be computational, and a strong focus will be the Computer Science applications of the theoretical concepts covered.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- Demonstrate an understanding of how to carry out vector operations such as dot product, length etc. on vectors in arbitrary dimensions. Demonstrate an understanding of the geometrical interpretation of vectors in 2 and 3 dimensions, and its applications in computer graphics.
- Demonstrate an understanding of linear transformations and their representation as matrices, of the application of matrices to vectors, and of matrix operations and their interpretation in 2, 3 and higher dimensions.
- Apply concepts such as matrix rank, transpose, upper and lower diagonal matrices, orthogonality, trace and symmetry.
- Demonstrate an understanding of Singular Value Decomposition, its numerical stability, and its relationship to the invertibility of a matrix.
- Demonstrate an understanding of the properties of eigenvalues and eigenvectors, and their construction for a given matrix.
- Throughout all the topics, demonstrate an understanding of the application of theoretical concepts and results in various areas of Computer Science.

TEACHING AND LEARNING METHODS

Lecture based delivery supported by laboratory sessions. Normally 3 hours of sessions per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS2910 - Artificial Intelligence

Prerequisites: None

Co-ordinator: Prof Kostas Stathis

Course value: 15 credits

COURSE SUMMARY

Artificial Intelligence (AI) is usually defined as the science of making computers do things that require intelligence when done by humans. AI has had some success in limited, or simplified, domains. However, more recently, successes with developments of AI systems such as game playing and robotics have regenerated optimism concerning the attainment of human-level intelligence in a variety of domains despite the profound difficulty of the problem.

The aim of this course is to introduce students to the basic principles, methods and techniques of AI to provide the foundations for more advanced courses in this area. The course will start by providing an overview of the approaches in the field by referring to the wider historical context in which the AI vision was set and will further motivate the content to be taught by presenting existing and potential applications. After the overview, the course will deliver a series of topics from first principles, including the role of first-order logic for knowledge representation, computational reasoning and problem solving systems, the use of search as a capability for exploring alternative solutions, and how AI systems use knowledge to plan and learn from first principles.

Upon completion of the course, students should be able to develop intelligent systems by assembling capabilities to concrete computational problems; understand the role of knowledge representation, problem solving, and learning in intelligent-system engineering; and appreciate the role of problem solving, in wide applications that require a basic understanding of human intelligence from a computational perspective.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- Use computational logic to model domains and reasoning tasks of an intelligent system
- Understand the role of knowledge representation, problem solving and learning in building domain independent and domain dependent AI capabilities
- Explain conceptual and computational trade-offs between the expressiveness of different representations and capabilities
- Demonstrate how to develop and combine AI capabilities in a suitable programming language

TEACHING AND LEARNING METHODS

Lecture based delivery supported by laboratory sessions. Normally 2 hours of lectures per week for eleven weeks and 2 hours of labs for six weeks.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY2760 - Introduction to Information Security

Prerequisites:

Co-ordinator: Dr Elizabeth Quaglia

Course value: 15 credits

COURSE SUMMARY

Course Content:

- **Elements of cryptography:** Ciphers (DES/AES). Message Authentication codes (MACs). Public key ciphers and digital signatures (RSA).
- **Identity verification:** use and storage of conventional passwords. Dynamic password schemes. Biometric techniques. Use of tokens (dumb and intelligent), including the use of secure elements such as smart cards and trusted execution environments (TEEs).
- **Access control:** Access Control Lists, capabilities, security labels (MAC and DAC), and role-based access control.
- **CASE STUDY I:** electronic payments (EMV). Examine the overall security functionality provided by widely utilised Europay-Mastercard-VISA (EMV) standard.
- **Network security concepts:** the concepts of security services and security mechanisms (as in ISO 7498-2) firewalls.
- **Computer security:** viruses, spyware, restricting access.
- **Authentication and key distribution:** The importance and relatedness of the concepts of key management and entity authentication in a network. Objectives of an entity authentication protocol. Some fundamental protocols (e.g. Kerberos). Using authentication protocols for key distribution, and other approaches to key establishment (including public key certificates and X.509).
- **Cyber Physical security:** Examine the security provisions, strengths and weaknesses of existing multi-application smart card platforms and operating systems along with the security of embedded systems and tokens.
- **CASE STUDY II:** Chip migration for financial institutions. Examine the relevant information security, business, design, architectural and other factors that may influence the adoption of chip card technology by financial institutions.

LEARNING OUTCOMES

On successful completion of this course, students will be able to:

- Identify, through the case studies how information security may be influenced by real world design and implementation decisions.
- Appreciate the different cryptographic algorithms, their use, advantages and disadvantages.
- Apply the above identified cryptographic primitives in the review and evaluation of cryptographic protocols.
- Identify, through the case studies how information security may be influenced by real world design and implantation decisions.
- Appreciate the rational decisions in the design of a number tokens and secure elements.

TEACHING AND LEARNING METHODS

33 hours of lectures. The coursework is designed to allow students to apply their knowledge to applications with which they are familiar.

KEY BIBLIOGRAPHY

- The main recommended text for this course is:
• Useful background:
• D. Saloman: Elements of Computer Security, Springer 2010
• K. Martin: Everyday Cryptography, Oxford University Press 2012

FORMATIVE ASSESSMENT AND FEEDBACK

In class verbal feedback.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY2840 - Computer and network security

Prerequisites: CS2850

Co-ordinator: Dr Bertram Poettering

Course value: 15 credits

COURSE SUMMARY

This course is intended to provide a detailed exposition of computer and network security, which will build on knowledge acquired on previous computer science courses.

Course Content:
- Software vulnerabilities and hands on hacking-oriented attacks
- Memory errors
- Web
- Network
- Countermeasures
- Pointers to research papers

LEARNING OUTCOMES

- Identify and exploit the software vulnerabilities that can be introduced into programs through language features and poor programming practice
- Discuss the countermeasures that can mitigate the exploitation of such software vulnerabilities
- Introduce (briefly) malicious software (malware) as a typical consequence of a successful software exploitation
- Provide pointers to/discuss academic and/or industry research-oriented publications on the subject

TEACHING AND LEARNING METHODS

Lecture based delivery, guided independent study. Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY

- Hacking: The art of exploitation, 2nd edition, Jon Erickson

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be provided during laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3003 - IT Project Management

Prerequisites: CS2810 or CS2815

Co-ordinator: Dr Giorgios Koutsoukos

Course value: 15 credits

COURSE SUMMARY

This course aims to assist students in appreciating the complexity of IT projects, and give insight of how these can be managed.

Course content:

Project Management (PM) Fundamentals: fundamental definitions and terminology (e.g. Business Case, Project Constraints/Triangle, Portfolios and Programs Vs Projects, inter-alia); Main phases of IT Projects, with emphasis on the basics of requirements elicitation/analysis methods; Unique characteristics of IT projects/products in comparison to other domains; Agile Software Development vis-a-vis “traditional” IT Project Management.

Budgeting and Estimation aspects: Investment Appraisal methods (TCO, ROI, NPV); CAPEX/OPEX costs and depreciation; Estimation approaches/methods; Models of S/W acquisition/licensing and common cost elements in IT projects; Basics of Procurement and Contracts Management.

Project Organization: Project Governance Structure; Stakeholders Analysis; Role of Project Manager; Best-practices in projects’ organization.

Basic Processes of a Project, from initiation to closing, with references to relevant industry standards and frameworks, such as PMI, ISO 21500 and Prince2.

Project Planning: main activities and deliverables, with emphasis on techniques and tools such as Breakdown Structures, Network Analysis and Gantt charts; Project Management toolset demo/short lab.

Project Monitoring, Control and Implementation Processes: main activities and deliverables, with emphasis on Change Management.

Fundamentals of Team Management: notion of Team and Team Building (e.g. Tuckman Model); Leadership Styles; Fundamental concepts and best-practices in Project and Team Communications, Decision-making, Conflict resolution, Negotiation, Goal-setting, Time and Meetings management; Important principles and rules of thumb in managing software development teams.

Project Risk Management: fundamental concepts and terminology, the risk management process and related artifacts (e.g. risk register); Risk management in IT projects.

Project Quality: main concepts, tasks and deliverables on project and product quality aspects; IT systems quality factors and key-metrics; Fundamental aspects of IT Testing; Introduction to ISO Quality Management Systems and quality improvement.

The IT Project Manager Skills, Profession and Code of Ethics: introduction and outline of relevant topics.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- Apply basic project planning techniques
- Demonstrate an understanding of steps needed to build and maintain effective development teams
- Explain the procedures needed to monitor, control and report upon an IT development project
- Discuss and where appropriate apply the principles of project risk management
- Explain the ways in which appropriate quality attributes of the products of an IT development project can be assessed and assured

TEACHING AND LEARNING METHODS

Lecture based delivery, guided independent study. Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback during lectures. Written feedback from mandatory assessment

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3110 - Bioinformatics

Prerequisites: None

Co-ordinator: Prof. Alberto Paccanaro

Course value: 15 credits

COURSE SUMMARY

To introduce the main approaches currently in use in bioinformatics, with special emphasis on the analysis of DNA and protein sequences emerging from genome sequencing projects and genome-wide experimental assays.

Course Content:

- Basic molecular biology: introduction to the basic components of living cells, their functions and interactions, and to other concepts essential to understanding the use of computers in biology Sequence alignments, substitution matrices.
- Phylogenetic trees Dynamic Programming Systems biology: gene expression analysis, Protein-Protein Interaction analysis, biological networks, clustering

LEARNING OUTCOMES

By the end of this course a student should be able to:

- assess the main approaches currently in use in bioinformatics;
- demonstrate an understanding of the analysis of DNA and protein sequences.

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY

- Understanding Bioinformatics Marketa Zvalebil, Jeremy O. Baum Garland Science, 2007

FORMATIVE ASSESSMENT AND FEEDBACK

Lectures in which students work through exercises getting verbal feedback.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3220 - Digital Audio and Applications

Prerequisites: CS2800

Co-ordinator: Mr Nuno Barreiro

Course value: 15 credits

COURSE SUMMARY

Digital audio has many applications, ranging from music production to audio enhancement, or from noise cancellation in mobile phones to automatic music generations for games. These applications share common features that will be covered in this course. Starting with the fundamentals of sound (nature, transmission, volume), students will then learn how to convert sound to and from the digital domain. Once in the digital domain, audio can be manipulated using algorithms from Digital Signal Processing. The course covers some basic DSP algorithms, engaging the students both in audio analysis (amplitude, frequency, pan, etc.) and audio synthesis (additive, FM, etc.). By the end of this course, students should be able to write simple applications that analyse, manipulate and generate sound.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- explain digital audio and the fundamentals of AD/DA conversion
- use Fourier analysis to synthesize and transform sounds in the frequency domain
- demonstrate an understanding lossy and lossless compression algorithms
- explain the basics of digital signal processing (post-processing and effects)
- develop applications of digital audio (music production, noise cancellation, audio enhancement, automatic music generation)
- acquire a basic knowledge of sound perception and psychoacoustics

TEACHING AND LEARNING METHODS

Up to 4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback during laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3250 - Visualisation and Exploratory Analysis

Prerequisites:

Co-ordinator: Prof Zhiyuan Luo

Course value: 15 credits

COURSE SUMMARY

The course aims to teach the principles and arts of statistical visualisation and exploratory analysis of data.

Course Content:

- Construction of informative bivariate plots.
- Visualisation of multivariate data
- Dimensional reduction. Non-linear methods (t-SNE, isomap, Proxigrams)
- Exploratory cluster analysis
- Standard methods for visualisation of relational and graph data (Gephi)
- Importance of guarding against "snooping"
- Basic principles of colour scale design and glyph choice

LEARNING OUTCOMES

By the end of the course students should be able to:

- Have some of the skills and wisdom that are needed for open-ended exploratory analysis of data, and for the analytical presentation of the results of statistical analyses
- Understand and be able to construct linear projections of multivariate data; they should also have experience in using some non-linear dimension reduction methods
- have some practical experience of using standard graph visualisation methods and evaluation of results
- Be aware of the dangers of data snooping, through practical experience of this
- Be able to make reasoned choices in representational mode, glyph design, and colour design for presentation graphics

TEACHING AND LEARNING METHODS

Lecture based delivery, laboratory classes, guided independent study Normally 3 hours of lectures / laboratory per week

KEY BIBLIOGRAPHY

- Visualising data, by William S. Cleveland
- The elements of graphing data by William s. Cleveland
- The visual display of quantitative data, by Edward Tufte
- Envisioning information by Edward Tufte

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback during laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3470 - Compilers and code generation

Prerequisites: CS1811; CS1812 or CS1813; CS1870

Co-ordinator: Prof Elizabeth Scott

Course value: 15 credits

COURSE SUMMARY

To describe how to construct and implement interpreters and compilers for modern processors

Course Content:

- Lexical analysis: input buffering, regular expressions, finite state automata, hash coded symbol tables, LEX
- Syntax analysis: context free grammars, derivations, recursive descent and table-based LR parsing, YACC
- Semantics: abstract parse trees, syntax directed translation, intermediate forms and three address code, attributes, annotated parse trees, semantic rules and attribute grammars, translation schemes and L-attribute grammars
- Error detection and recovery: classes of error and error recovery in top down parsers
- Intermediate code generation: using attribute grammars, three address code
- Code improvement: basic blocks and code improvement techniques, flow graphs, loop improvement and fusion, directed cyclic graphs for identifying code improvements
- Compiler generation with tool support

LEARNING OUTCOMES

By the end of this course a student should be able to:

- explain the role and structure of a compiler and the standard stages of compilation
- to build a DFA based lexical analyser for a set of tokens specified using regular expressions
- Advanced students should be able to construct regular expressions, which define specified set of strings
- use grammars to define context free languages and to build parsers for them
- describe syntax directed translators and use them to construct intermediate code
- describe various types of error detection and recovery
- generate three address code from source code.

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by small group tutorials. Normally 3 hours of lectures per week

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Return of coursework grades and comments, in-class discussion of coursework solutions

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3480 - Software Language Engineering

Prerequisites: CS1811; CS1812 or CS1813; CS2810 or CS2815

Co-ordinator: Prof Adrian Johnstone

Course value: 15 credits

COURSE SUMMARY

Course Content:
- Domains
- Domain Specific Languages
- Concrete and abstract syntax, generalised vs near deterministic parsing
- Code generation
- The ART toolset
- Case studies

LEARNING OUTCOMES

- understand domain specific language design and implementation
- learn the significance of DSL’s and meta-modelling in advanced software engineering
- design and implement small specialised languages

TEACHING AND LEARNING METHODS

Up to 3 hours of lectures and laboratory sessions per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be provided during laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3490 - Computational Optimisation

Prerequisites: CS2860

Co-ordinator: Prof Gregory Gutin

Course value: 15 credits

COURSE SUMMARY

To introduce the basic models of computational optimisation and the basic algorithms for solving computational optimisation problems. To demonstrate the theoretical and computational methods of analysing computational optimisation algorithms and will discuss available software packages for solving problems.

Course Content:

- Introduction: algorithm efficiency and problem complexity
- Linear programming (LP): LP model, formulating problems as LP problems, graphical solution, simplex method, duality in LP, decomposition of LP problems, LP software
- Integer Programming (IP): IP models, branch-and-bound algorithm
- Computational optimisation problems: greedy-type algorithms, construction heuristics and local search for the TSP
- Heuristics: DMERN problem and signed graphs; heuristics for DMERN, experimental analysis of embedded network and TSP heuristics, theoretical analysis of heuristics, meta-heuristics

LEARNING OUTCOMES

By the end of this course a student should be able to:

- demonstrate an understanding of the basic models of computational optimisation
- apply the basic algorithms for solving computational optimisation problems
- evaluate theoretical and computational methods of analysing computational optimisation algorithms
- use enhanced algorithmic and mathematical skills.

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hours of lectures per week plus independent guided study

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3510 - Functional Programming and Applications

Prerequisites:

Co-ordinator: Prof Zhaohui Luo

Course value: 15 credits

COURSE SUMMARY

This course introduces the principles of functional programming (FP). Functional programming has become more and more popular in recent years because it promotes code that's safe, concise, and elegant, and makes it easier to write parallel code for parallel computing. You will also study how FP may be combined with other programming paradigms (eg, OOP) in languages such as Scala, a language that fuses FP and OOP in a practical package for applications.

Course Content:

- Introduction to the basic principles of functional programming (FP) as compared to the imperative object-oriented programming such as Java
- Some basic features in an FP language: recursion, abstraction and higher-order functions
- Basic typing and data types such as lists and tuples
- High-order functions and pattern matching
- Classes and objects in a hybrid language (such as Scala)
- Functional paradigm as compared with the imperative paradigm
- Applications (eg. In parallel programming)
- Basics in some advanced topics such as genericity, monads and dependent typing

LEARNING OUTCOMES

- Understand the basic principles of functional programming; its basic ideas, foundations, advantages as compared with imperative programming paradigm
- Master the fundamental programming concepts such as recursion, abstraction, higher-order functions and data types
- Grasp skills in developing elegant, efficient, correct, and reusable programs, using the key features in functional programming languages
- Understand the advantages and disadvantages of different programming paradigms as compared to each other and, on the basis of this, the basic ideas of combining different programming styles in a uniform paradigm
- Understand the basic foundational principles of functional programming and some of the advanced issues such as genericity, polymorphism and typing in functional programming

TEACHING AND LEARNING METHODS

Normally 3 hrs per week of lectures and practicals.

KEY BIBLIOGRAPHY

- C.S. Horstmann. Scala for the Impatient. Addison-Wesley 2012

FORMATIVE ASSESSMENT AND FEEDBACK

Written assignment with written feedback

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3846 - Human-Computer Interaction

Prerequisites: CS1840 or CS2841

Co-ordinator: Mr Nuno Barreiro and Dr Carlos Matos

Course value: 15 credits

COURSE SUMMARY

This course introduces human-computer interaction aspects and challenges, and addresses the approaches that can be used to create interfaces matching users' needs and expectations.

Course Content:

- introduction to Human-Computer Interaction (definition and history);
- user experience (UX) vs. user interface (UI);
- heuristic evaluation;
- rapid prototyping;
- HCI studies/experiments.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- demonstrate an understanding of what HCI is, its principles, challenges, and teamwork development aspects
- build prototypes using different media and technologies
- evaluate the usability and suitability of user interfaces
- undertake HCI tests, namely Web experiments
- explain how perception and cognition influence HCI

TEACHING AND LEARNING METHODS

Up to 4 hours of lectures and laboratory classes per week

KEY BIBLIOGRAPHY

- Eric Freeman, Head First HTML5 Programming. 2011.
- Jon Duckett, JavaScript and jQuery: Interactive Front-End Web Development. 2014.

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback on progress will be given during the weekly laboratory sessions

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3870 - Advanced Algorithms and Complexity

Prerequisites: CS1811; CS1812 or CS1813; CS1860; CS2860

Co-ordinator: Dr Magnus Wahlström

Course value: 15 credits

COURSE SUMMARY

The main aim of this course is to build on the knowledge students previously obtained about algorithms and complexity, and introduce them to more advanced topics, with an emphasis on graphs and algorithm design. Content includes:

Graphs and graph algorithms: directed and undirected graphs; directed acyclic graphs and acyclic orderings; connectivity in directed graphs; applications of graph algorithms.

Algorithm design paradigms and their applications: amortised analysis; dynamic programming; divide and conquer; greedy algorithms; backtracking search.

NP-completeness: basic notions and their interpretation; problem reductions and completeness; ways of dealing with NP-complete problems, including heuristic approaches, exact algorithms, and tractable special cases.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- implement and reason about fundamental algorithms for a range of problems
- describe and apply various algorithm design strategies for the construction of efficient algorithms
- understand the significance of NP-complete problems and know ways to deal with them, such as heuristic approaches

TEACHING AND LEARNING METHODS

Lecture based delivery, guided independent study. Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback during lectures. Written feedback from mandatory formative assessment

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3920 - Machine Learning

Prerequisites: A-level Mathematics

Co-ordinator: Prof Volodya Vovk

Course value: 15 credits

COURSE SUMMARY

The aim of the course is to acquaint the students with several key methods and techniques of machine learning. In particular, it will cover the following topics. Nearest neighbours for classification and regression; interesting distances. Conformal prediction and conformalizing nearest neighbours. Ridge regression and Lasso. Inductive conformal predictors and cross-conformal predictors. Support vector machines for classification and regression. Kernel trick and its applications to the algorithms covered so far. Practically useful kernels. Conformalized versions.

LEARNING OUTCOMES

- Demonstrate knowledge of the theoretical background in machine learning methods.
- Have an understanding of the main advantages and limitations of various approaches to machine learning and specific machine-learning algorithms.
- Be able to implement versions of several machine-learning algorithms.
- Have an understanding of some ways to apply the ideas and algorithms of machine learning in industry, medicine and other fields.

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes. Normally 3 hours of lectures/labs per week. Guided independent study

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

In lectures / labs, Immediate help and verbal feedback

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3930 - Computational Finance

Prerequisites: A-level Mathematics

Co-ordinator: Dr Yuri Kalnishkan

Course value: 15 credits

COURSE SUMMARY

To understand the key role played by the advent of derivatives, financial instruments which facilitate managing financial risks. To understand pricing derivatives (and associated strategies of dynamic hedging) using advanced computational models are required.

Course Content:

- Introduction: financial markets; the rules of the game.
- Taxonomy of securities: main kinds of derivative securities and underlying markets.
- Mathematical techniques: Wiener process; diffusion processes as mathematical models of price dynamics; stochastic differential equations; computer simulations.
- Pricing and hedging in the Black-Scholes world: risk-neutral valuation; the Black-Scholes equation and analytic formulae; the “Greeks” and their use.
- Beyond the Black-Scholes world: application issues; computational models; fractals and their use in finance.
- Efficient markets hypothesis: theory vs empirical evidence.
- Risk management: Value at Risk.
- Coursework Project: implementing valuation algorithms for different derivatives (e.g., in MatLab); solving numerical and theoretical problems.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- demonstrate an understanding of mathematical and computational models of underlying and derivative securities;
- master techniques for pricing derivatives and for dynamic hedging;
- apply these models and techniques for creating computer programs.

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by practical classes. Normally 3 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- ISBN-10: 0131977059 (earlier editions starting from the 2nd are acceptable)

FORMATIVE ASSESSMENT AND FEEDBACK

Return of grades and written comments on coursework submission of coursework project

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS3940 - Intelligent agents and multi-agent systems

Prerequisites: None

Co-ordinator: Prof. Kostas Stathis

Course value: 15 credits

COURSE SUMMARY

The aims of this course include introducing the students to the concept and design of an agent and multi-agent system, and the main applications for which they are appropriate. In addition, it presents a contemporary platform for implementing agents and multi-agent systems. Course contents include:

Introduction: agents and objects, expert systems, distributed systems; typical application areas for agent systems.

Intelligent Agents: abstract architectures for agents; tasks for agents, the design of intelligent agents - reasoning agents, agents as reactive systems; hybrid agents (e.g., PRS); layered agents (e.g., Interap).

Multi-Agent Systems: classifying multi-agent interactions - cooperative versus non-cooperative; zero-sum and other interactions; cooperation - the Prisoner’s dilemma and Axelrod’s experiments; interactions between self-interested agents: auctions systems; negotiation; argumentation; interaction languages and protocols: speech acts, KQML/KIF, the FIPA framework, ontologies, coordination languages; interactions between benevolent agents: cooperative distributed problem solving (CDPS), partial global planning; coherence and coordination; applications of intelligent agents and multi-agent systems.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- understand the notion of an agent and understand the characteristics of applications that lend themselves to an agent-oriented solution
- understand the key issues associated with constructing agents capable of intelligent autonomous action
- understand the key issues in designing societies of agents that can effectively cooperate in order to solve problems
- understand the main application areas of agent-based solutions, and be able to develop a meaningful agent-based system

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by tutorial sessions. Normally 3 hours of sessions per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback provided through the tutorial sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY3606 - Smart Cards, RFIDs and Embedded Systems Security

Prerequisites: IY2760 or equivalent

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

Students will have developed a mastery of the topic of smart cards and token security.

Course contents:
- Introduction to Smart Cards/Chips & RFID/NFC; Assets for Cyber Security
- Smart Cards – Trusted Production Environment
- Operating systems, Interoperability and Security
- Applications & Security for Mobile Communications, USIM/SIM, Services and Clouds
- Smart Cards for Secure Banking & Finance
- Smart Cards in eIDs/Passports - & RFIDs/NFC explained
- Advances in Smart Chips/Tokens, and Transport System Case Study
- Common Criteria and Smart Cards
- Security Attacks, Countermeasures and Testing for Smart Cards/RFIDs/NFC
- Application Development Environments for JAVA and SIM Toolkit
- Comparing Alternative Security Tokens/Environments; including TPM and TEE

LEARNING OUTCOMES

- identify constituent components, analyse strengths and weaknesses, identify new applications of smart cards/security tokens and their use as assets in cyber security; identify the steps in the manufacturing/personalisation processes, analyse and evaluate potential risks and compare security safeguards

- identify and compare the systems in use, analyse the strengths and weaknesses and evaluate interoperability and security issues; analyse the range of capabilities of SIM/USIM cards in Smartphones and apply them to new service ideas, evaluate the possible range of services and security measures

- understand the main standards and applications of smart cards for banking and finance, compare with earlier card solutions and analyse strengths and weaknesses of approaches; analyse the key role of the smart card/RFID for passports, IDs and satellite TV, evaluate the security measures that have protected past and current cards,

- identify and describe new technologies, including NFC, TPM, TEE; and apply them to new applications and evaluate the likely suitability/success of approach; explain how common criteria may affect smart card design/development, analyse the different approaches and compare with less formal methods

- identify and describe the classes of attack and notable methods within each class, analyse countermeasures and evaluate practicality of attacks and the effects on cyber security; identify, compare and evaluate different methods of developing applications for smart cards, and understand the development cycle and the use of practical tools

- analyse the issues concerning smart card lifecycle management, and evaluate and compare methods of local and remote card management

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK
Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY3609 - Digital Forensics

Prerequisites: IY2760 or equivalent

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

Introduction to forensic science, steps from collecting data to preserving evidence, and a framework for digital forensic evidence collection and processing.
Fundamentals of host forensics for Microsoft Windows, including kernel architecture, device driver architecture, registry, auditing, and security architecture, file system handling, and reconstruction of file and directory structures on the FAT and NTFS file system families.
Fundamentals of host forensics for Unix derivatives using the Linux operating system as an exemplar, including kernel and device driver architecture, security and audit mechanisms, file systems and pseudo file systems, and the reconstruction of file and directory structures using UFS and Ext2/3fs as exemplars.
Foundations of network forensics from data capturing and collection to network file systems and supplementary protocols as well as selected application-layer protocols and techniques used for identifying and reverse-engineering protocols used on networks.
Introduction to malware including anti-forensics and propagation techniques.
Introduction to steganographic techniques for images, video, textual data, and audio as well as steganalytical techniques for selected media types and approaches to traitor tracing.
A survey of non-standard storage mechanisms from retention characteristics to mobile and smart phones and vehicular systems as well as network-based search and storage mechanisms.

LEARNING OUTCOMES

- understand the concepts of audit and indirect activity records retained by operating systems, particularly in file systems, and how to retrieve such information
- understand selected network protocols and the collection and derivation of evidence leading to the reconstruction of system and user activity based on network trace information
- understand infiltration and anti-forensics techniques used particularly by malicious software
- apply steganographic and particularly steganalytical methods for different types of media
- understand the retention characteristics of storage systems and non-standard devices such as mobile/smart phones, cloud computing, and vehicular systems

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY

- B. Carrier: File System Forensic Analysis. Addison-Wesley, 2005

FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY3612 - Cyber Security

Prerequisites: IY2760 or equivalent

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

Week 1: Introduction
Week 2: Networks and dependencies
Week 3: Critical infrastructures and interdependencies
Week 4: Security of Cyber-Physical Systems
Week 5: Control systems security
Week 6: Advanced persistent threats
Week 7: Attack modelling techniques
Week 8: System assurance
Week 9: Incident response mechanisms
Week 10: Offensive cyber operations
Week 11: Future challenges

LEARNING OUTCOMES

- have an understanding of network robustness and failures, together with key underlying theoretical concepts
- understand critical (information) infrastructures, vulnerabilities, and their dependencies
- appreciate the specific security problems of cyber-physical including SCADA systems and selected infrastructure
- understand complex attacks, analytical models for such attacks, and assurance mechanisms

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY3660 - Applications of Cryptography

Prerequisites: IY2760

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

The aim of this module is to explain the fundamentals behind cryptography and how it is deployed in real systems.

Course Content:

- Fundamentals: security services, security models, basic attacks on cryptosystems
- Cryptographic mechanisms: symmetric and public-key encryption, hash functions, MACs, digital signatures, authentication protocols
- Real world cryptography: key management, implementation issues, cryptographic standards, crypto politics
- Applications: case studies of systems such as 3G, EMV, SSL/TLS

LEARNING OUTCOMES

On successful completion of this course, students will be able to:

- Appreciate the full range of security services that can be provided by cryptography
- Compare a number of different cryptographic mechanisms that can be used to provide these security services
- Explain the full nature of the architecture within which cryptography is deployed in a security system
- Identify appropriate cryptographic standards that should be followed when implementing cryptography
- Explain the rationale for the design decisions taken in a number of widely deployed cryptographic systems

TEACHING AND LEARNING METHODS

Up to 3-4 hours of lectures and laboratory sessions per week.

KEY BIBLIOGRAPHY

- D.R. Stinson, Cryptography: Theory and Practice, CRC Press
- K.M. Martin: Everyday Cryptography, OUP.
- C. Paar and J. Pelzl, Understanding Cryptography, Springer
- J. Katz and Y. Lindell, Introduction to Modern Cryptography, CRC Press
- N. Ferguson, B. Schneier and T. Kohno, Cryptography Engineering, Wiley

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be provided during laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY3840 - Malicious Software

Prerequisites: IY2760 and IY2840

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

Cybercrime has become both more widespread and harder to battle. Researchers and anecdotal experience show that the cybercrime scene is becoming increasingly organized and consolidated, with strong links also to traditional criminal networks. Modern attacks are indeed stealthy and often profit oriented. Malicious software (malware) is the traditional way in which cybercriminals infect user and enterprise hosts to gain access to their private, financial, and intellectual property data. Once stolen, such information can enable more sophisticated attacks, generate illegal revenue, and allow for cyber-espionage.

By mixing a practical, hands-on approach with the theory and techniques behind the scene, the course discusses the current academic and underground research in the field, trying to answer the foremost question about malware and underground economy, namely, “Should we care?”.

Students will learn how traditional and mobile malware work, how they are analyzed and detected, peering through the underground ecosystem that drives this profitable but illegal business. Understanding how malware operates is of paramount importance to form knowledgeable experts, teachers, researchers, and practitioners able to fight back. Besides, it allows us to gather intimate knowledge of the systems and the threats, which is a necessary step to successfully devise novel, effective, and practical mitigation techniques.

LEARNING OUTCOMES

On successful completion of this course, students will be able to:

- Comprehend what malicious software is
- Understand the malware landscape of the early days and what efforts and challenges the AV industry was facing to fight malware threats; (basic) static analysis to analyze and detect malware; (basic) assembly and reverse engineering notions are provided
- Explain dynamic analysis. Packing and algorithmic-agnostic unpacking is introduced as an initial step toward full dynamic analysis; sandboxes and limits of dynamic analysis and sandboxes.
- Describe and explain the underground economy and cybercrime that surrounds malicious software
- Demonstrate a critical appreciation of some of the newer research trends that are likely to influence software security work in the coming years

TEACHING AND LEARNING METHODS

Up to 3-4 hours of lectures and laboratory sessions per week.

KEY BIBLIOGRAPHY

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be provided during laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS4100 - Data Analysis

Prerequisites:

Co-ordinator: Prof. Volodya Vovk

Course value: 15 credits

COURSE SUMMARY

The course teaches fundamental facts and skills in data analysis, including machine learning, data mining and statistics

Course Content:

- Supervised learning: classification, regression, and ensemble methods.
- Algorithm-independent machine learning
- Unsupervised learning and clustering
- Exploratory data analysis
- Bayesian methods. Bayes networks and causality
- Applications, such as information retrieval and natural language processing

LEARNING OUTCOMES

- Develop, validate, evaluate, and use effectively machine learning models and statistical models
- Apply methods and techniques such as clustering, regression, decision trees, and neural networks
- More generally, extract value and insight from data

TEACHING AND LEARNING METHODS

Lecture based delivery support by lab session, guided independent study. Normally 3-5 hours of lectures / practical classes and workshops per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback in lab sessions Written feedback on three programming coursework assignments

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS4200 - On Line-Machine Learning

Prerequisites: Working knowledge of a programming language with good data manipulation and visualisation capabilities, such as MATLAB, R, or Python.

Co-ordinator: Dr Yuri Kalnishkan

Course value: 15 credits

COURSE SUMMARY

The course addresses the on-line framework of machine learning in which the learning system learns and issues predictions or decisions in real time, perhaps in a changing environment. The course teaches protocols, methods and applications of on-line learning.

Course Content:

- Prediction with expert advice: learning protocol, loss function, regret.
- Aggregating algorithm, its optimality properties and applications to general loss functions. Sleeping and switching experts.
- Universal algorithms in on-line learning. Applications to portfolio theory: Cover’s universal rebalanced portfolios.
- Reinforcement learning framework, Markov decision process, Q-learning.

LEARNING OUTCOMES

- understand and evaluate probabilistic and non-probabilistic on-line learning protocols;
- demonstrate advanced understanding of the prediction with expert advice and reinforcement learning frameworks;
- demonstrate advanced knowledge of methods of Markov models, time series, prediction with expert advice and reinforcement learning;
- analyse the properties of on-line learning algorithms;
- apply on-line algorithms to real-world data and evaluate the results.

TEACHING AND LEARNING METHODS

Lecture based delivery support by practical classes, guided independent study. Normally 4-5 hour of lectures / practical classes per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback in practical classes / lab sessions. Written feedback on coursework assignment.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS4234 - Large-Scale Data Storage and Processing

Prerequisites: CS1830
Co-ordinator: Prof Gregory Chockler
Course value: 15 credits

COURSE SUMMARY
The main aims of this course are for students to study the underlying principles of storage and processing massive collections of data, typical of today's Big Data systems, and to gain hands-on experience in using large and unstructured data sets for analysis and prediction. The topics covered include techniques and paradigms for querying and processing massive data sets (Spark, MapReduce, Hadoop, data warehousing, SQL for data analytics, stream processing), fundamentals of scalable data storage (NoSQL data bases such as MongoDB, Cassandra, HBase), working with dynamic web data (data acquisition, data formats), elements of cloud computing, and applications to real world data analytics and data mining problems (sentiment analysis, social network mining).

LEARNING OUTCOMES

- Knowledge and understanding of core concepts, theories and principles of large scale data storage and processing frameworks
- Sound evaluation of opportunities and challenges related to leveraging those frameworks for building massive scale analytics solutions, and an ability to make recommendations to resolve these challenges
- Proficient knowledge and use of at least one large scale data store systems, and at least one massive scale processing framework
- Ability to design, develop, and evaluate an end to end analytics solution combining large scale data storage and processing frameworks
- Knowledge of cloud computing as a platform for Big Data analytics

TEACHING AND LEARNING METHODS
Lecture based delivery and laboratory classes, guided independent study. Normally 4 hours of sessions per week.

KEY BIBLIOGRAPHY

- Miner, Shook, MapReduce Design Patterns, O'Reilly, ISBN-10: 1449327176

FORMATIVE ASSESSMENT AND FEEDBACK
Verbal feedback will be provided during laboratory sessions.

SUMMATIVE ASSESSMENT
Details of coursework submission deadlines will be published on the department website at the start of term.
CS4504 - Business Intelligence Systems, Infrastructures and Technologies

Prerequisites: CS2855

Co-ordinator: Dr Giorgios Koutsoukos

Course value: 15 credits

COURSE SUMMARY

Business Intelligence (BI) refers to the skills, processes, methodologies, technologies, applications, and practices used in order to leverage (gathering, storing, analyzing) an organization’s internal and external information assets to support and improve decision-making. With the advent of Big-Data there is considerably increased demand for skills and knowledge, both conceptual and technological, that can be effectively applied to support this new era of Big-Data based decision-making.

This course aims to provide students with
(a) a broad understanding of the information assets and the conceptual and technical architectures of information and business intelligence systems in modern organizations
(b) the necessary background knowledge of, and skills to design, implement and evaluate business intelligence systems and technologies.

Course Content includes the following topics:

Introduction to Information Systems & Business Intelligence: Overview of Information Systems and BI Systems, Information Systems and BI Technical Architectures (Logical & Physical aspects), Acquisition models and Business cases.

Data Warehousing and Dimensional Modelling (Definitions, Concepts, Architectures, Design Processes, Implementation Aspects)

BI Applications: OLAP, Reports, Dashboards, Data Mining, Visualization and UI design

Practical Sessions (Surgeries, Labs) on Dimensional Modelling, Reports and Dashboards using BI tools.

LEARNING OUTCOMES

By the end of the course students should be able to:

- demonstrate a holistic view of business intelligence systems and their role in the IT environment of modern organizations
- understand and evaluate the concepts, terminology and architectures of Data Warehouses and BI solutions
- understand Data Modelling concepts and provide design solutions using Dimensional Modeling
- know the important elements of business intelligence applications such as Data Analysis, Data Mining and Dashboards; understand and evaluate BI Visualization aspects and the relationship of BI solutions to CRM and ERP systems
- use industrial business intelligence tools

TEACHING AND LEARNING METHODS

The course will be taught in weekly sessions over the course of a term.

KEY BIBLIOGRAPHY

Core reading material:


FORMATIVE ASSESSMENT AND FEEDBACK
The lecturers will provide oral feedback during practical sessions and written feedback to coursework.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS4860 - Advanced Distributed Systems

Prerequisites:

Co-ordinator: Dr Dan O'Keeffe

Course value: 15 credits

COURSE SUMMARY

The course will cover fundamental principles of building modern distributed systems, for example in the context of the Internet of Things. The specific emphasis will be on the two central components of the IoT reference architecture: cloud infrastructure and wireless networking.

The course will discuss major challenges found in these environments (such as massive scales, wide distribution, decentralisation, unreliable communication links, component failures and network partitions) and general approaches for dealing with these challenges.

The topics covered will include:
- abstract models (such as the synchronous and asynchronous distributed computing models, models for wireless networks);
- algorithmic techniques (such as distributed coordination, fault-tolerant design of distributed algorithms, synchronization techniques);
- practical case studies.

The students will also have an opportunity to apply the studied material for implementing various components of a realistic distributed system through a series of formative coursework assignments, lab practicals, and a final project.

LEARNING OUTCOMES

- Display a mastery of core concepts, theories and principles of distributed systems
- Demonstrate knowledge of algorithmic techniques for solving problems in a distributed environment
- Explain the practical aspects of implementing various components of a distributed system
- Implement various components of a realistic distributed system

TEACHING AND LEARNING METHODS

Normally 4 hrs per week of lectures and laboratory sessions.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback in laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
CS4950 - Deep Learning

Prerequisites: CS2900

Co-ordinator: Prof Chris Watkins

Course value: 15 credits

COURSE SUMMARY

The aim of the course is to give students an introduction to deep learning that covers neural network optimisation by gradient descent from first principles, and which also gives a broader introduction to a range of advanced architectures, with hands-on implementation.

The course starts by considering models of artificial neural networks for supervised learning, and introduces notions of activation function, loss function, and computation of loss-gradients using back-propagation with the chain rule. Neural network learning with back-propagation and different gradient descent algorithms will be covered in detail, and visualised in lab-sessions. Next, the 'disappearing gradient' problem in deep architectures will be raised, and methods for resolving this problem will be discussed. A range of deep architectures will be described for discriminative learning, generative learning and learning of representations, and for reinforcement learning. Students will implement a deep architecture using a toolkit in a project assignment at the end of the course.

LEARNING OUTCOMES

By the end of this course a student should be able to:

- Demonstrate an advanced understanding of the notions of an artificial neural network, and of learning by minimising a loss function, using training, validation, and test data sets.
- Master computation of loss-gradients for different neural network architectures, and of a range of algorithms for optimisation by gradient descent, as applied in neural networks.
- Demonstrate a good comparative understanding of a range of deep learning architectures.
- Master the basic techniques necessary for gradient optimisation of deep networks, and of some diagnostics needed for determining whether gradient descent is working correctly.
- Apply deep learning algorithms to real-world data and evaluate the results.
- Implement and run deep learning algorithms using appropriate tool-kits.

TEACHING AND LEARNING METHODS

Lecture based delivery supported by laboratory sessions. Normally 4 hours of sessions per week.

KEY BIBLIOGRAPHY

- Deep Learning, by Ian Goodfellow, Yoshua Bengio, and Aaron Courville, MIT Press Dec 2016 (already available on web)
- The course will make extensive use of notes, handouts, and freely available material on the web, which is abundant and rapidly changing.

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY4501 - Security Management

Prerequisites: Cannot be taken by students who took IY3501

Co-ordinator: Prof Chris Mitchell

Course value: 15 credits

COURSE SUMMARY

This course will explain the need for effective security management, identify the main elements of security management, and consider the ways in which organisations implement security management. The list of topics may vary slightly to reflect developments in the subject but typically will include:

- What is information security management and is it necessary?
- The role of standards in information security management, with emphasis on the ISO/IEC 27000 series.
- Security controls and ISO/IEC 27002.
- Information security risk management.
- Legal and regulatory aspects of information security.
- Internal control, audit and security.
- Incident management and disaster recovery.
- Staff management.
- Procedural issues.
- Failures in information security management.
- Case studies in information security management

LEARNING OUTCOMES

- analyse information security risks, strategies and methods
- evaluate security management requirements
- define security treatments to meet identified requirements
- perform ongoing security functions, including internal and external audit

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY

- ISO/IEC 27000 series standards.

FORMATIVE ASSESSMENT AND FEEDBACK

Feedback on formative coursework.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY4523 - Secure Business Architectures

Prerequisites:

Co-ordinator: Dr Geraint Price

Course value: 15 credits

COURSE SUMMARY

The high level aim of this course is to discuss the appropriate system design and business response to technical/architectural
decisions which impact the security of the organisation's information.
The content of the course will include:
Introduce the concept of a security development lifecycle
An overview of the basics of Risk Assessment
Elaborate on the Governance, Risk and Compliance issues discussed in the Security Management module
Outsourcing/Cloud Computing architectures
Identity Management
The Payment Card Industry – Data Security Standard
Supply Chain Security
Big Data

LEARNING OUTCOMES

- understand, and be able to apply, the concept of a “security lifecycle” in relation to specific security architectures
- understand, and be able to outline the high-level components of a Risk Assessment and apply these components to
candidate security architectures.
- appreciate the Governance, Risk and Compliance issues related to business architectures
- apply these to a number of architectures, such as: Identity Management; Outsourcing; PCI-DSS; Supply Chain

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY

The course will not follow any individual text book, but is likely to make use of: international standards; industry white papers
and research reports; case studies using common industry architectures.

FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY4606 - Smart Cards, RFIDs and Embedded Systems Security

Prerequisites: IY2760 or equivalent; Cannot be taken by students who took IY3606.

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

Students will have developed a mastery of the topic of smart cards and token security.
Course contents:
- Introduction to Smart Cards/Chips & RFID/NFC; Assets for Cyber Security
- Smart Cards – Trusted Production Environment
- Operating systems, Interoperability and Security
- Applications & Security for Mobile Communications, USIM/SIM, Services and Clouds
- Smart Cards for Secure Banking & Finance
- Smart Cards in eIds/Passports - & RFIDs/NFC explained
- Advances in Smart Chips/Tokens, and Transport System Case Study
- Common Criteria and Smart Cards
- Security Attacks, Countermeasures and Testing for Smart Cards/RFIDs/NFC
- Application Development Environments for JAVA and SIM Toolkit
- Comparing Alternative Security Tokens/Environments; including TPM and TEE

LEARNING OUTCOMES

- identify constituent components, analyse strengths and weaknesses, identify new applications of smart cards/security tokens and their use as assets in cyber security; identify the steps in the manufacturing/personalisation processes, analyse and evaluate potential risks and compare security safeguards
- identify and compare the systems in use, analyse the strengths and weaknesses and evaluate interoperability and security issues; analyse the range of capabilities of SIM/USIM cards in Smartphones and apply them to new service ideas, evaluate the possible range of services and security measures
- understand the main standards and applications of smart cards for banking and finance, compare with earlier card solutions and analyse strengths and weaknesses of approaches; analyse the key role of the smart card/RFID for passports, IDs and satellite TV, evaluate the security measures that have protected past and current cards
- identify and describe new technologies, including NFC, TPM, TEE; and apply them to new applications and evaluate the likely suitability/success of approach; explain how common criteria may affect smart card design/development, analyse the different approaches and compare with less formal methods
- identify and describe the classes of attack and notable methods within each class, analyse countermeasures and evaluate practicality of attacks and the effects on cyber security; identify, compare and evaluate different methods of developing applications for smart cards, and understand the development cycle and the use of practical tools
- analyse the issues concerning smart card lifecycle management, and evaluate and compare methods of local and remote card management

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK
Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY4609 - Digital Forensics

Prerequisites: IY2760 or equivalent; Cannot be taken by students who took IY3609.

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

Introduction to forensic science, steps from collecting data to preserving evidence, and a framework for digital forensic evidence collection and processing.

Fundamentals of host forensics for Microsoft Windows, including kernel architecture, device driver architecture, registry, auditing, and security architecture, file system handling, and reconstruction of file and directory structures on the FAT and NTFS file system families.

Fundamentals of host forensics for Unix derivatives using the Linux operating system as an exemplar, including kernel and device driver architecture, security and audit mechanisms, file systems and pseudo file systems, and the reconstruction of file and directory structures using UFS and Ext2/3fs as exemplars.

Foundations of network forensics from data capturing and collection to network file systems and supplementary protocols as well as selected application-layer protocols and techniques used for identifying and reverse-engineering protocols used on networks.

Introduction to malware including anti-forensics and propagation techniques.

Introduction to steganographic techniques for images, video, textual data, and audio as well as steganalytical techniques for selected media types and approaches to traitor tracing.

A survey of non-standard storage mechanisms from retention characteristics to mobile and smart phones and vehicular systems as well as network-based search and storage mechanisms.

LEARNING OUTCOMES

- understand the concepts of audit and indirect activity records retained by operating systems, particularly in file systems, and how to retrieve such information

- understand selected network protocols and the collection and derivation of evidence leading to the reconstruction of system and user activity based on network trace information

- understand infiltration and anti-forensics techniques used particularly by malicious software

- apply steganographic and particularly steganalytical methods for different types of media

- understand the retention characteristics of storage systems and non-standard devices such as mobile/smart phones, cloud computing, and vehicular systems

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY


- B. Carrier: File System Forensic Analysis. Addison-Wesley, 2005


FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY4610 - Security Testing Theory and Practice

Prerequisites: IY2760

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

To give students a mastery of Security testing theory and practice

Course Content:

- Security testing, legal aspects of penetration testing, standards and certification.
- Security testing frameworks and methodologies, and how to prepare, manage and conduct a professional penetration testing.
- Technical aspects of network security covering standards, protocols, routing, firewalls showing the theoretical basis of vulnerabilities and how these may be exploited in practice.
- Technical aspects of computer security covering operating systems, access control in windows and linux/unix, host based intrusion detection, escalation of privileges and how to exploit these vulnerabilities in practice and how to harden systems.
- Technical aspects of Internet based applications, web services, protocols, languages (e.g. SQL) and how these may be exploited using for example SQL injection and cross-site scripting; how to exploit these vulnerabilities in practice, and how to harden the applications.
- A survey of non-standard and emerging technologies and review of potential threats these may lead to.

LEARNING OUTCOMES

- Understand the legal aspects of carrying out a penetration test and an approach to preparing and managing such an audit.
- Understand, at a deep level, network protocols; computer system architectures; and application systems.
- Understand the vulnerabilities in existing protocols, systems, and applications.
- Understand the security technologies designed to mitigate these vulnerabilities.
- Understand how these vulnerabilities may be exploited in practice to penetrate a system.

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK
Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.
IY4612 - Cyber Security

Prerequisites: IY2760 or equivalent; Cannot be taken by students who took IY3612.

Co-ordinator:

Course value: 15 credits

COURSE SUMMARY

Week 1: Introduction
Week 2: Networks and dependencies
Week 3: Critical infrastructures and interdependencies
Week 4: Security of Cyber-Physical Systems
Week 5: Control systems security
Week 6: Advanced persistent threats
Week 7: Attack modelling techniques
Week 8: System assurance
Week 9: Incident response mechanisms
Week 10: Offensive cyber operations
Week 11: Future challenges

LEARNING OUTCOMES

- have an understanding of network robustness and failures, together with key underlying theoretical concepts
- understand critical (information) infrastructures, vulnerabilities, and their dependencies
- appreciate the specific security problems of cyber-physical including SCADA systems and selected infrastructure
- understand complex attacks, analytical models for such attacks, and assurance mechanisms

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hour of lectures per week.

KEY BIBLIOGRAPHY


FORMATIVE ASSESSMENT AND FEEDBACK

Oral feedback during lectures.

SUMMATIVE ASSESSMENT

Details of coursework submission deadlines will be published on the department website at the start of term.