

RUHR-UNIVERSITY BOCHUM

FACULTY OF COMPUTER SCIENCE

RUB

Module Guide

For the Master Study Program

Computer Science

<https://informatik.rub.de/en/studies/cs/msc/>



Study Plan Master Computer Science [PO 24] Ruhr-University Bochum

Nr.	Module	Size (CP)	Recommended semester	Grading
Compulsory elective area				
1	Compulsory elective modules**	a*	1-3	graded
2	Specialisation modules***	b*	1-3	graded
3	Specialised practical course	c*	1-3	ungraded
4	Specialised seminar	3	1-3	graded
5	Project	10	2-3	graded
Free Elective area				
6	Free elective modules****	20	1-3	ungraded
Final Thesis				
7	Master thesis and colloquium	27+3	4	graded
Total:		120		

* $a \geq 15, b \geq 35, 3 \leq c \leq 5; a+b+c \geq 57$

** Here, modules from the compulsory elective catalogue "Basic/Foundation" must be taken. The modules that can be selected are listed in the current module handbook.

*** Here, modules are to be taken, which are thematically assigned to different specialisation areas. The modules that can be selected are listed in the current module handbook.

**** Here, (almost) all courses of the RUB course catalogue, as well as courses within the framework of the University Alliance Ruhr can be selected, taking into account §4 (2).

Important Rules for planning your studies:

Specialisation modules amounting to at least 35 CP must be completed, whereby at least 2 specialisation areas with at least 10 CP each must be covered. Modules that are assigned to more than one area of specialisation are only allocated to one area.

It is optional to set a thematic specialisation during your studies. A specialisation is formed when at least 50 CP have been collected from modules that are assigned to the same specialisation area. This may include the Master's thesis, compulsory elective modules, specialisation modules, specialised seminar, specialised practical course. A successfully completed specialisation is shown on the degree certificate.

Offered Basic Modules and Specialisation Modules

Course	Faculty	Size (CP)	Semester	Grading
Compulsory elective modules (Basic/Foundation modules)				
Advanced Algorithms	CS	9	WS	graded
Cryptography	CS	8	WS	graded
Quantum Information and Computation	CS	5	WS	graded
Highlights of Theoretical Computer Science	CS	9	SS	graded
Mathematics for Modelling and Data Analysis	CS	5	SS	graded
Software Languages	CS	5	SS	graded
Theory of Machine Learning	CS	9	SS	graded
Specialisation modules				
Algorithms, Complexity, Data				
Advanced Algorithms	CS	9	WS	graded
Formal Verification and Model Checking	CS	5	WS	graded
Computational complexity theory	CS	9	WS	graded
Proofs are Programs	CS	5	WS (no offer in WS 25/26)	graded
Quantum Information and Computation	CS	5	WS	graded
Advanced Quantum Information and Computation	CS	5	SS	graded
Finite Fields: Theory and Algorithms	CS	5	SS	graded
Foundations of Programming Languages, Verification, and Security	CS	5	SS	graded
Computational Geometry	CS	5	SS	graded
Highlights of Theoretical Computer Science	CS	9	SS	graded
Artificial Intelligence				
Autonomous Vehicles and Artificial Intelligence lab	CS	5	WS	graded
Deep Learning	CS	5	WS	graded
Numerical Optimization	CS	6	WS	graded
Machine Learning: Unsupervised Methods	CS	10	WS	graded
Natural Language Processing with Deep Learning	CS	5	WS	graded
AI Ethics and Society	CS	5	WS	graded
Autonomous Robotics: Action, Perception and Cognition	CS	6	SS (last offer in SS 25)	graded
Autonomous Vehicles and Artificial Intelligence	CS	5	SS	graded
Computational Neuroscience: Single Neuron Models	CS	6	SS	graded
Computational Neuroscience: Vision and Memory	CS	5	SS	graded
Engineering for Large Language Models	CS	6	SS	graded
Machine Learning: Supervised Methods	CS	6	SS (no offer in SS 25)	graded
Privacy-Preserving Natural Language Processing	CS	6	SS	graded
Responsible AI	CS	6	SS	graded
Theory of Machine Learning	CS	9	SS	graded
Machine Learning: Evolutionary Algorithms	CS	6	WS (last offer in WS 24/25)	graded
Computational Neuroscience: Neural Dynamics	CS	6	WS (last offer in WS 24/25)	graded
Computer Security				
Cryptography	CS	8	WS	graded
Software Security 1	CS	5	WS	graded
Microarchitectural Attacks and Defenses	CS	5	WS	graded
Blockchain and Decentralized Security (ehemals Blockchain Security and Privacy)	CS	5	WS	graded
Cryptography on hardware-based platforms	CS	5	WS	graded
Quantum Cryptography	CS	5	WS (no offer in WS 25/26)	graded
Privacy Engineering, Data Governance and Usability	CS	5	WS	graded
Advanced Automatic Testing	CS	5	SS	graded
Cryptographic protocols	CS	5	SS	graded
Digital Sovereignty	CS	6	SS	graded
Mobile Network Security	CS	5	SS	graded
Physical Attacks and Countermeasures	CS	5	SS	graded
Program Analysis	CS	5	SS	graded
Public Key Cryptoanalysis 1	CS	5	SS (no offer in SS 25)	graded

Software Security 2	CS	5	SS	graded
Design, Implementation and Analysis of Computer Systems				
Deterministic Network Calculus	CS	9	WS	graded
Energy-Aware Computing Systems	CS	6	WS	graded
Formal Verification and Model Checking	CS	5	WS	graded
High-Performance Computing on Clusters	Bauing	6	WS	graded
High-Performance Computing on Multicore Processors (formally High-Performance Computing on Multi- and Manycore Processors)	Bauing	6	SS	graded
Scheduling Theory (formally Real-time Networks and Systems)	CS	5	SS	graded
Software Engineering and Programming Languages				
Autonomous Vehicles and Artificial Intelligence lab	CS	5	WS	graded
Foundations of Programming Languages, Verification, and Security	CS	5	SS	graded
High-Performance Computing on Clusters	Bauing	6	WS	graded
Proofs are Programs	CS	5	WS (no offer in WS 25/26)	graded
Advanced Automatic Testing	CS	5	SS	graded
Autonomous Vehicles and Artificial Intelligence	CS	5	SS	graded
High-Performance Computing on Multicore Processors (formally High-Performance Computing on Multi- and Manycore Processors)	Bauing	6	SS	graded
Software Languages	CS	5	SS	graded

Offered Specialised Seminars and Practical Courses

Course	Faculty	Size (CP)	Semester	Grading
Seminars				
Algorithms, Complexity, Data				
Seminar Mathematics and Computation	CS	3	SS	graded
Seminar Randomized Algorithms	CS	3	SS (no offer in SS 25)	graded
Seminar Gems of Logic	CS	3	SS	graded
Seminar on Algorithms	CS	3	WS	graded
Seminar Quantum Information and Computation	CS	3	WS	graded
Learning meets Theoretical Computer Science	CS	3	WS	graded
Artificial Intelligence				
Master Seminar: Building Trust in Large Language Models	CS	3	SS	graded
Seminar Advanced topics in Deep Learning	CS	3	SS	graded
Seminar Safety and Reliability in Artificial Intelligence	CS	3	WS/SS	graded
Algorithms for Decision Making	CS	3	WS	graded
Recent Trends in Interpretability of Artificial Intelligence Model	CS	3	WS (no offer in WS 25/26)	graded
Seminar: Introduction to Bayesian Modeling	CS	3	WS	graded
Seminar: Search-based Code Generation	CS	3	WS	graded
Computer Security				
Master-Seminar "Digital Sovereignty"	CS	3	WS/SS	graded
Seminar Software Security	CS	3	WS/SS	graded
Seminar Internet Security	CS	3	WS/SS	graded
Seminar Safety and Reliability in Artificial Intelligence	CS	3	WS/SS	graded
Seminar Security Engineering	CS	3	WS/SS	graded
Seminar Mobile Network Security	CS	3	WS	graded
Current topics in microarchitectural security	CS	3	SS	graded
Seminar on Applied Privacy and Anonymity	CS	3	SS	graded
Seminar Software and Internet Security	CS	3	WS/SS (last offer in SS 25)	graded
Design, Implementation and Analysis of Computer Systems				
Seminar Distributed Systems	CS	3	WS	graded
Resource-efficient system software concepts	CS	3	WS/SS	graded

Seminar Networked Systemes	CS	3	SS	graded
Software Engineering and Programming Languages				
Seminar Automated Software Engineering	CS	3	WS/SS	graded
Seminar: Search-based Code Generation	CS	3	WS	graded
Seminars without Specialisations				
Ethics in Computer Science Research	CS	3	WS	graded
Seminar Computer science perspectives on Mis- and disinformation	CS	3	WS	graded

Practical Courses					
Algorithms, Complexity, Data					
	Advanced Python Programming	CS	3	WS	ungraded
Artificial Intelligence					
	Practical Course on Machine learning Security	CS	4	WS	ungraded
	Lab Course: Challenging Problems in Reinforcement Learning	CS	3	WS	ungraded
	Lab course: Introduction to Bayesian Modeling	CS	3	WS	ungraded
Computer Security					
	Research in Internet Security	CS	4	WS/SS	ungraded
	Research in Software Security	CS	4	WS/SS	ungraded
	Software Testing via Fuzzing	CS	4	WS/SS	ungraded
	Practical Course Traffic Analysis Attacks	CS	4	WS	ungraded
	Practical Course on Machine learning Security	CS	4	WS	ungraded
	Advanced Research in Microarchitectural Security	CS	4	WS (no offer in WS 25/26)	ungraded
	Introductory project in microarchitectural security	CS	4	SS	ungraded
	Practical Course on Mobile Network Security	CS	4	SS	ungraded
	Embedded Firmware Fuzzing	CS	4	SS	ungraded
Design, Implementation and Analysis of Computer Systems					
	Open-Source Chip Design	CS	4	WS	ungraded
	System software engineering lab course	CS	4	WS	ungraded
Software Engineering and Programming Languages					

Abbreviations:

CS: Faculty of Computer Science
 Baulng: Faculty of Civil and Environmental Engineering

SS: Summer Semester
 WS: Winter Semester

CP: Creditpoints

MODULE GUIDE

Overview of the modules

Computer Science - Master (1-Fach, PO 2024)

Basic/Foundation Modules

Advanced Algorithms
Highlights of Theoretical Computer Science
Cryptography
Mathematics for Modeling and Data Analysis
Quantum Information and Computation [M.Sc.]
Software Languages
Theory of machine learning

Design, Implementation and Analysis of Computer Systems

Deterministic Network Calculus
Energy-Aware Computing Systems
Formal Verification and Model Checking
High-Performance Computing on Clusters
High-Performance Computing on Multicore Processors
Scheduling Theory

Algorithms, Complexity, Data

Advanced Algorithms
Advanced Quantum Information and Computation
Computational Geometry
Finite Fields: Theory and Algorithms
Formal Verification and Model Checking
Highlights of Theoretical Computer Science
Computational complexity theory
Proofs are programs [M.Sc.] (no offer in WS 25/26)
Quantum Information and Computation [M.Sc.]

Computer Security

Advanced Automatic Testing
Blockchain and Decentralized Security
Digital Sovereignty
Cryptography

Cryptography on hardware-based platforms
Cryptographic protocols
Microarchitectural Attacks and Defenses
Mobile Network Security
Physical Attacks and Countermeasures
Privacy Engineering, Data Governance and Usability
Program Analysis
Public Key Cryptanalysis 1
Quantum Cryptography (no offer in WS 25/26)
Software Security 1 [M.Sc.]
Software Security 2

Software Engineering and Programming Languages

Advanced Automatic Testing
Autonomous Vehicles and Artificial Intelligence
Autonomous Vehicles and Artificial Intelligence Lab
Foundations of Programming Languages, Verification, and Security
High-Performance Computing on Clusters
High-Performance Computing on Multicore Processors
Proofs are programs [M.Sc.] (no offer in WS 25/26)
Software Languages

Artificial Intelligence

AI Ethics And Society
Autonomous Robotics: Action, Perception, Cognition
Autonomous Vehicles and Artificial Intelligence
Autonomous Vehicles and Artificial Intelligence Lab
Computational Neuroscience: Single-Neuron Models
Computational Neuroscience: Vision and Memory
Deep Learning
Engineering for Large Language Models
Machine Learning: Supervised Methods (no offer in SS 25)
Machine Learning: Unsupervised Methods
Natural language processing with deep learning [M.Sc.]
Numerical Optimization
Privacy-Preserving Natural Language Processing
Responsible AI
Statistical learning and data mining
Theory of machine learning

Practical Modules

Practical Labs

Project

Seminars

Free Elective Moduls

Free Elective Moduls

Masterthesis

Master's thesis and colloquium

Module title: Advanced Algorithms

Modul code	Credit points 9 CP	Workload 270 h	Term siehe Prüfungsordnung / see Examination regulations	Frequency Winter semester	Duration 1 semester
Courses Advanced Algorithms (212029)			Contact time 6 SWS (90 h)	Self-study 180 h	Group size 40 participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Maike Buchin Lecturers: Prof. Maike Buchin					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Mathematik M.Sc. IT-Sicherheit / Informationstechnik					
Precognitions Recommended: Basic knowledge of algorithm design and analysis as known from the Bachelor's programme is expected.					
Learning goals <ul style="list-style-type: none">• Advanced design methods for algorithms• Advanced analysis methods for algorithms• Knowledge of further data structures and methods for designing data structures• Application of learned methods to new problems					
Contents In this lecture we look at advanced topics in algorithmics. After a short repetition of known contents, we mainly look at graph algorithms, approximation algorithms and FPT algorithms as well as exact algorithms for NP-hard problems. We also look at some new and well-known data structures and their analysis. The problems considered are combinatorial, graph-theoretical as well as geometric.					
Teaching methods Lecture (as slide and blackboard lecture) and exercises in which the presented contents are deepened.					
Examination forms Oral (15-45 minutes) or written exam (120 minutes) (to be announced at the beginning of the semester).					
Requirements for the award of credits Passed final module examination and successful participation in exercises.					
Significance of the grade for the final grade (for a total of 120 ECTS) 9/97: M.Sc. Computer Science 9/105: M.Sc. Angewandte Informatik 9/91: M.Sc. IT-Sicherheit / Informationstechnik [PO 22]					

Module title: Highlights of Theoretical Computer Science					
Modul code	Credit points 9 CP	Workload 270 h	Term see examination regulations/ siehe Prüfungsordnung	Frequency summer semester	Duration 1 semester
Courses Highlights of Theoretical Computer Science (211057)			Contact time 6 SWS (90 h)	Self-study	Group size 30 participants
Teaching language English			Requirements for participation Successful completion of an introductory course on theoretical computer science (covering formal languages, basics of complexity theory including NP-completeness and reductions, basics of computability theory). Interest and motivation to learn about theoretical concepts.		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Michael Walter Prof. Dr. Thomas Zeume Lecturers: Prof. Dr. Michael Walter Prof. Dr. Thomas Zeume Dr. Vladimir Lysikov					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit / Informationstechnik M.Sc. IT-Sicherheit / Netze und Systeme					
Precognitions					
Learning goals You will know some of the most important results and insights of modern theoretical computer science. You will learn approaches and techniques that go well beyond a first course. You will be able to recognize when these can be used and how to adapt them to new situations. You will be able to independently acquire new knowledge in this area.					
Contents The insights and techniques of modern theoretical computer science have been key for advances in all areas of computer science. In this course, we will discuss some highlights and the techniques that underpin them. Possible topics that we might cover: <ul style="list-style-type: none"> • Computational models (is there life beyond Turing machines?) • Kolmogorov complexity (what is the shortest program that produces some output?) • Communication complexity (how many bits must Alice and Bob exchange to jointly solve a problem?) • Fine-grained complexity (are some easy problems easier than others? and why?) • Fast multiplication of numbers and matrices (can you beat the high-school method?) • Randomness (does it really help to compute faster?) • Circuit lower bounds (why is it so hard to prove that problems are hard?) • Convex optimization (how to maximize profit if all you can ask are yes/no questions) • Hardness of approximation (how easy is it to find near-optimal solutions?) 					

- Cryptography and computation

If you enjoyed your first course in theoretical computer science in the Bachelor's and would like to deepen your knowledge by getting an overview of the fascinating theory of computing, then this course will be exactly right for you.

Teaching methods

Lecture with Exercises

Examination forms

Final module examination. Format will depend on number of participants.

Requirements for the award of credits

Passed module final exam (written exam 180 minutes or oral exam 15-45 minutes)

Significance of the grade for the final grade (for a total of 120 ECTS)

9/97: M.Sc. Computer Science

9/105: M.Sc. Angewandte Informatik

9/91: M.Sc. IT-Sicherheit / Informationstechnik [PO 22]

9/84: M.Sc. IT-Sicherheit / Informationstechnik [PO 20]

9/99: M.Sc. IT-Sicherheit / Netze und Systeme [PO 22]

9/96: M.Sc. IT-Sicherheit / Netze und Systeme [PO 20]

Module title: Cryptography					
Modul code	Credit points 8 CP	Workload 240 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Kryptographie (212017)			Contact time 6 SWS (90 h)	Self-study 150 h	Group size 100 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Alex May Lecturers: Prof. Dr. Alex May					
Module application B.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science M.Sc. Angewandte Informatik					
Precognitions Contents of the lectures Introduction to Cryptography 1 and 2.					
Learning goals The students have an understanding of the essential mathematical methods and procedures on which modern cryptographic procedures are based. The depth of the treatment of the methods goes well beyond that taught in the previous courses. Students will be able to analyse and design current and future cryptographic methods. They also have an awareness of the methodology and power of various attack scenarios.					
Contents An introduction to modern methods of symmetric and asymmetric cryptography is provided. For this purpose, an attacker model is defined and the security of the presented encryption, hash and signature methods is proven under well-defined complexity measures in this attacker model. Topic Overview: <ul style="list-style-type: none"> • Secure encryption against KPA, CPA and CCA attackers • Pseudo-random functions and permutations • Message Authentication Codes • Collision-resistant hash functions • Block ciphers • Construction of random number generators • Diffie-Hellman key exchange • Trapdoor one-way permutations • Public key encryption: RSA, ElGamal, Goldwasser-Micali, Rabin, Paillier • One-way signatures • Signatures from collision-resistant hash functions • Random Oracle Model 					

Teaching methods

Lecture and exercises

Examination forms

Written exam (120 minutes)

Requirements for the award of credits

Passed written final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

8/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

8/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

8/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

8/96: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 20]

8/97: M.Sc. Computer Science

8/105: M.Sc. Angewandte Informatik

Module title: Mathematics for Modeling and Data Analysis					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer Semester	Duration 1 semester
Courses Mathematics for Modeling and Data Analysis (211047)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 30 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Laurenz Wiskott Lecturers: Prof. Dr. Laurenz Wiskott					
Module application B.Sc. Angewandte Informatik M.Sc. Computer Science					
Precognitions Basic knowledge of calculus and linear algebra					
Learning goals After the successful completion of this course the students <ul style="list-style-type: none"> • know the material covered in this course, see Content, • do have an intuitive understanding of the basic concepts and can work with that, • can communicate about all this in English 					
Contents This course covers mathematical methods that are relevant for modeling and data analysis. Particular emphasis is put on an intuitive understanding as is required for a creative command of mathematics. The following topics are covered: <ul style="list-style-type: none"> • Functions and how to visualize them • Vector spaces • Matrices as transformations • Systems of linear differential equations • Qualitative analysis of nonlinear differential equations • Bayesian theory • Markov chains 					
Teaching methods This course is given with the flipped/inverted classroom concept. First, the students work through online material by themselves. In the lecture time slot we then discuss the material, find connections to other topics, ask questions and try to answer them. In the tutorial time slot the newly acquired knowledge is applied to analytical exercises and thereby deepened. I encourage all students to work in teams during self-study time as well as in the tutorial.					
Examination forms Oral final module exam 15-45 minutes.					
Requirements for the award of credits Passed oral Exam					

Significance of the grade for the final grade (for a total of 120 ECTS)

5/168: B.Sc. Angewandte Informatik

5/97: M.Sc. Computer Science

Module title: Quantum Information and Computation [M.Sc.]

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Quantum Information and Computation (212011)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size participants
Teaching language German or English (depends on audience)			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Michael Walter Lecturers: Prof. Dr. Michael Walter					
Module application M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science					
Precognitions Familiarity with linear algebra (in finite dimensions) and probability (with finitely many outcomes) at the level of a first Bachelor's course; we will briefly remind you of the more difficult bits in class. In addition, some mathematical maturity, since we will discuss precise mathematical statements and rigorous proofs. No background in physics is required.					
Learning goals You will learn fundamental concepts, algorithms, and results in quantum information and computation. After successful completion of this course, you will know the theoretical model of quantum information and computation, how to generalize computer science concepts to the quantum setting, how to design and analyze quantum algorithms and protocols for a variety of computational problems, and how to prove complexity theoretic lower bounds. You will be prepared for an advanced course or a research or thesis project in this area. Master's students will be expected to understand the material in a deeper way, which will reflect itself in homework and examination.					
Contents This course will give an introduction to quantum information and quantum computation from the perspective of theoretical computer science. Topics to be covered will likely include: <ul style="list-style-type: none">• Fundamentals of quantum computing: quantum bits, states and operations• The power of quantum entanglement: nonlocal games• Entanglement as a resource: superdense coding and teleportation• Quantum circuit model of computation• Quantum computing with oracles: Deutsch-Jozsa, Bernstein-Vazirani, Simon• Quantum Fourier transform and phase estimation• Shor's factoring algorithm• Grover's search algorithm and beyond: how to solve SAT on a quantum computer?• From no cloning to quantum money: a peek at quantum cryptography					

The course should be of interest to students of computer science, mathematics, physics, and related disciplines. Students interested in a BSc or MSc project in quantum information, computing, cryptography, etc. are particularly encouraged to participate.

Teaching methods

Lecture with Exercise

Examination forms

Final written module exam (180 minutes)

Requirements for the award of credits

Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/105: M.Sc. Angewandte Informatik

5 /91: M.Sc. IT-Sicherheit/ Informationstechnik

5/ 99: M.Sc. IT-Sicherheit/ Netze und Systeme

5/97: M.Sc. Computer Science

Module title: Software Languages					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Software Languages (212034)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 25 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Thorsten Berger Lecturers: Prof. Dr. Thorsten Berger					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik					
Precognitions <ul style="list-style-type: none"> • Object-Oriented Programming • Theoretical Computer Science (especially formal languages, grammars, set notation) • Software Engineering • Functional Programming 					
Learning goals					
Knowledge and understanding <ul style="list-style-type: none"> • explain core concepts for engineering software languages, e.g., meta-modeling, abstract syntax, concrete syntax, static semantics, dynamic semantics, model/program transformation • explain linguistic architectures for software languages (e.g., meta-modeling hierarchy) • explain how domain-specific languages can be realized within a contemporary language workbench 					
Competence and skills <ul style="list-style-type: none"> • engineer domain-specific languages for a specific engineering problem • define abstract syntax by modeling domains and creating meta-models • define static semantics with syntactic constraints or type systems • define concrete syntax with Xtext, Sirius, or a comparable technology • define dynamic semantics with model transformations (compilation) or interpretation • reason about configuration spaces expressed in variability models 					
Judgement and approach <ul style="list-style-type: none"> • identify use-cases and the potential of using DSLs for a given domain/problem <p>select and justify appropriate language technology for a given domain/problem</p>					
Contents The course teaches the theory and the pragmatics of using and developing high-level software languages (Domain-Specific Languages, or DSLs) for the effective production of quality software. This includes methods, design patterns, guidelines, and testing practices for defining concrete syntax, abstract syntax, and semantics of languages. The course attempts to be close to technology, while covering multiple paradigms and solutions.					

We cover:

- Domain analysis of a problem domain and designing meta-models
- Engineering external and internal DSLs (we focus mostly on external DSLs)
- Designing the concrete syntax of languages
- Implementation of language semantics using declarative and imperative transformations, code generators and interpreters, in various scenarios, e.g., from text to models, from models to text, involving XML, database, etc.
- Implement declarative constraints and type rules for DSLs
- Testing and quality assurance of language implementations
- Model-driven engineering, especially engineering of highly configurable systems or software product lines (we focus on feature modeling syntax and semantics)

Teaching methods

The teaching of this course consists of different forms: lectures, interactive quizzes, group work, group supervision, and practical assignments.

Examination forms

The examination for the Software Languages module consists of several parts. Submissions are to be made during the semester and must be passed. The submissions themselves are not graded. A written or oral final examination takes place at the end of the semester. In order to be able to take the final examination, students must pass the semester-long assignments.

Requirements for the award of credits

Passed semester-accompanying submissions and passed final module examination (written (120 minutes) or oral (15-45 minutes)).

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

Module title: Theory of machine learning					
Modul code	Credit points 9 CP	Workload 270 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Theorie des maschinellen Lernens (211052)			Contact time 6 SWS (90 h)	Self-study 180 h	Group size 20 participants
Teaching language German			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Asja Fischer Lecturers: Prof. Dr. Asja Fischer					
Module application M.Sc. Angewandte Informatik M.Sc. Computer Science					
Precognitions					
Learning goals Students are familiarised with mathematical models for machine learning. They acquire the ability to evaluate and compare learning algorithms according to the degree to which they achieve (precisely described) success criteria. They acquire techniques both for designing efficient learning algorithms and for proving the inherent hardness of a problem. After the successful completion of the module <ul style="list-style-type: none"> • students know the most important learning machines (such as Support Vector Machines and related models), • students understand the difference between empirical and real error rate and know techniques to deal with the problem of overfitting the data (with a model that is too complex), • can differentiate between uniform and non-uniform learnability of a Hypothesis class and know the appropriate theories and learning rules. 					
Contents The subject of the lecture is the statistics-based theory of machine learning. In particular, the method of structured risk minimisation is taught as well as the statistical theorems on which it is based. Techniques for designing efficient learning algorithms are discussed as well as information- or computational-theoretic barriers that make certain learning problems appear to be inefficiently solvable.					
Teaching methods Lecture with exercise					
Examination forms Final oral module exam (15-45 minutes)					
Requirements for the award of credits Passed final oral module exam.					
Significance of the grade for the final grade (for a total of 120 ECTS) 9/105: M.Sc. Angewandte Informatik					

Module title: Deterministic Network Calculus					
Modul code	Credit points 9 CP	Workload 270 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Deterministic Network Calculus (211054)			Contact time 6 SWS (90 h)	Self-study 180 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Steffen Bondorf Lecturers: Prof. Dr. Steffen Bondorf					
Module application M.Sc. Computer Science M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. Angewandte Informatik					
Precognitions Mathematics (functional analysis), Scheduling Theory, computer networks					
Learning goals Upon successful completion of the module, students will be able to, <ul style="list-style-type: none"> • model complex, networked systems as deterministic queueing systems, • perform worst-case performance analyses of existing systems or models, • understand the challenges of performance dimensioning of planned systems, and explain how central mechanisms in computer networks work using network calculus • differentiate the presented methods from each other and apply them to scientific questions. 					
Contents Distributed systems are ubiquitous nowadays and their interconnectedness is fundamental for the continuous distribution and thus availability of data. The provision of data in real time is one of the most important non-functional aspects that safety-critical networks must ensure. The formal verification of data communication with respect to worst-case deadlines is fundamental for the certification of newly developed x-by-wire systems. This verification allows the take-off of aeroplanes, the steering of cars without mechanical connection and the operation of safety-critical industrial plants. Therefore, several methods for worst-case modelling and analysis of real-time systems have been developed. One of them is Deterministic Network Calculus (DNC), a versatile technique that can be used in various areas such as packet switching, task scheduling, system on chip, software defined networks, data centre networks and network virtualisation. DNC is a method for deriving deterministic bounds on two of the most prioritised performance metrics in communication systems: <ul style="list-style-type: none"> • the end-to-end delay of data flows and • the amount of memory a server needs to buffer all in 					
Teaching methods Lecture with Exercise					
Examination forms Final written exam (150 minutes) or oral exam (15-45 minutes)					

Requirements for the award of credits

Passed final written or oral module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

9/97: M.Sc. Computer Science

9/91: M.Sc. IT-Sicherheit/ Informationstechnik

9/105: M.Sc. Angewandte Informatik

Module title: Energy-Aware Computing Systems

Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Energy-Aware Computing Systems (212030)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size 20 participants
Teaching language English			Requirements for participation none		
Module coordinators and Lecturers Module coordinators: Prof. Dr.-Ing. Timo Hönig Lecturers: Prof. Dr.-Ing. Timo Hönig					
Module application M.Sc. Computer Science M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. Angewandte Informatik					
Precognitions					
Learning goals Students who have successfully attended the lecture and exercises have followed the learning objectives and have acquired the competence listed below. Students can <ul style="list-style-type: none">• understand the importance of electrical energy as an operating resource to computing systems• make trade-off decisions in regard to efficient system design (i.e., power demand vs. performance), in particular of operating systems• model the energy demand for individual synchronous and asynchronous operations• apply strategies to reduce the energy demand for software activities based on specific hardware properties (i.e., sleep states)• analyse software for critical sections that cause high energy demand					
Contents Electrical energy is the single most important operating resource for computer systems. Although the energy demand of computers is an invisible system property by itself, the impact of energy demand is omnipresent and obvious in various forms of appearance. Sudden system failures (i.e., system breakdowns) and recurrent standard system operations (i.e., power management) serve as practical examples. The lecture discusses the design of energy-aware computing systems and focuses on the following topics: <ul style="list-style-type: none">• power and energy management• energy accounting• energy demand analysis• energy-aware operating-system architecture• hardware power management (i.e., DVFS, throttling, sleep states)• thermal management• storage and file systems• memory management• network, wireless, and protocols• energy-aware server/cluster					

- compiler optimisations and code transformation
- display technology
- electricity grid

The lecture is linked to the exercises through research papers. The students read the papers in preparation for lectures. From there, the research papers are the basis for discussion and build the starting point for the assignments of the exercises. As part of the exercises, the students apply concepts and strategies from the research papers to systems and evaluate the impact on the system's energy efficiency.

Teaching methods

The lecture is held with a seminar character. Research papers on energy-aware computing and system design are prepared by the students and discussed and analysed during the sessions. Additionally, the lecture imparts theoretical knowledge of fundamental concepts for the individual topics.

As part of the exercises, the students apply their acquired knowledge by adapting system software and system configurations to improve energy efficiency. They analyse the results by conducting performance and energy-demand evaluations.

Examination forms

Final oral module exam (15-45 minutes)

Requirements for the award of credits

Passed final module exam and successful participation in the exercises.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Informatik

6/105: M.Sc. Angewandte Informatik

6/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

6/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

Module title: Formal Verification and Model Checking

Modul code	Credit points 5 CP	Workload 150h	Term siehe Prüfungsordnung / see Examination Regulations	Frequency winter semester	Duration 1 semester
Courses Formal Verification and Model Checking (212041)			Contact time 60h (4 SWS)	Self-study 90	Group size 40 participants
Teaching language English			Requirements for participation If the previous module “Model Checking” has already been completed in a previous semester, it is not possible to take the examination for the module “Formal Verification and Model Checking”.		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Nils Jansen Lecturers: Prof. Dr. Nils Jansen					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit/Informationstechnik					
Precognitions Basic knowledge of automata theory.					
Learning goals After completing this course, the participants will be able to <ul style="list-style-type: none">• Solve practical verification problems using modern tools such as SAT/SMT solvers and model checkers like PRISM or Storm.• Understand algorithms underlying modern SAT and SMT solvers, such as resolution, CDCL, and CDCL(T).• Understand formal logics such as LTL or CTL, and understand the algorithmic implications of model checking against such properties• Connect SAT and SMT methods to problems from classical planning, program verification, model checking, and probabilistic inference.• Recognize situations in which the applications of model checking and verification techniques for specification and analysis may be useful.• Work on cutting-edge research that combines machine learning and artificial intelligence with the rigorous techniques of formal verification.					
Contents Complex digital systems have an increasing presence and impact on our lives, making the formal verification of the correctness of these systems crucial. For instance, a spacecraft's program should not crash, and a networking system should stay operational even if one server goes down. To ensure this, one approach is deductive verification, involving the generation of a collection of logical constraints that the system must satisfy. These constraints can be algorithmically verified through automatic theorem provers such as SAT and SMT solvers. Another popular approach is model checking, which consists of representing the system via mathematical models and exhaustively checking certain properties, often described in temporal logics. This course covers various aspects of formal verification and model checking, including: <ul style="list-style-type: none">• Satisfiability of propositions: resolution and the main ingredients of modern satisfiability provers.• Satisfiability modulo theories, in particular using linear inequalities, and the underlying simplex algorithm.					

- Explicit-state and symbolic algorithms for model checking linear-time (LTL) and branching-time (CTL) temporal logics for finite machines.
- Symbolic model checking using BDDs.
- Model checking under probabilities and uncertainty, using Markov chains and Markov decision processes, with the logic PCTL and expected reward properties
- The machine learning technique reinforcement learning, and model checking and formal verification techniques to render it safe to use on critical systems

Teaching methods

- 90 minute weekly lecture, in person.
- 90 minute exercise class, in person.
- Weekly exercises, both practical and theoretical

Examination forms

50% written exam (90 minutes) + 50% project work

Requirements for the award of credits

Passed project and final module examination.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik

Module title: High-Performance Computing on Clusters					
Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses High-Performance Computing on Clusters (127511)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size 50 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Andreas Vogel Lecturers: Prof. Dr. Andreas Vogel					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Bauingenieurwesen M.Sc. Subsurface Engineering					
Precognitions					
Learning goals After successfully completing the module, the students <ul style="list-style-type: none"> • are enabled to design and create programs for parallel computing clusters, • can critically evaluate distributed-memory systems and programming patterns, • can assess the mathematical properties of iterative solvers and their scalability. 					
Contents The lecture deals with the parallelization on cluster computers. Distributed-memory programming concepts (MPI) are introduced and best-practice implementation is presented based on applications from scientific computing including the finite element method and machine learning. Special attention is paid to scalable solvers for systems of equations on distributed-memory systems, focusing on iterative schemes such as simple splitting methods (Richardson, Jacobi, Gauß-Seidel, SOR), Krylov-methods (Gradient descent, CG, BiCGStab) and, in particular, the multigrid method. The mathematical foundations for iterative solvers are reviewed, suitable object-oriented interface structures are developed and an implementation of these solvers for modern parallel computer architectures is developed. Numerical experiments and self-developed software implementations are used to discuss and illustrate the theoretical results.					
Teaching methods Beamer, computer lab, numerical experiments					
Examination forms Written exam (120 minutes)					

Requirements for the award of credits

Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: High-Performance Computing on Multicore Processors

Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses High-Performance Computing on Multicore Processors (126509)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Andreas Vogel Lecturers: Prof. Dr. Andreas Vogel					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Bauingenieurwesen M.Sc. Subsurface Engineering					
Precognitions None					
Learning goals After successfully completing the module, the students <ul style="list-style-type: none">• are enabled to design and create programs for multicore processors,• can critically evaluate multi-threaded programs and shared-memory access patterns, can assess the benefits and challenges of multicore programming techniques.					
Contents The lecture addresses parallelization on multicore processors. Thread-based programming concepts and techniques, including pthreads, C++11 threads, OpenMP and SYCL, are introduced and best practices are highlighted using applications from scientific computing. An overview of the relevant hardware aspects including multicore architectures and memory hierarchies is provided. An in-depth introduction to multi-threaded programming on multicore systems with special emphasis on shared-memory parallelization is given and parallelization patterns, thread management and memory access strategies are discussed. In hands-on sessions, programming exercises are used to discuss and illustrate the presented content.					
Teaching methods Lecture with Exercise					
Examination forms Written exam (120 minutes).					

Requirements for the award of credits

Passed written module final exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Scheduling Theory					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Scheduling Theory (212032)			Contact time 3 SWS (45 h)	Self-study 105 h	Group size participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr.-Ing. Steffen Bondorf Lecturers: Prof. Dr.-Ing. Steffen Bondorf					
Module application M.Sc. Computer Science					
Precognitions					
Learning goals Students will gain a deep knowledge of modelling and analysis of (hard) real-time systems, in particular deterministic queuing theory (networking) as well as scheduling (networks, systems).					
Contents The scheduling of shared resources (e.g. forwarding in networks, CPU time of local systems) plays a central role in the performance of modern systems. In real-time systems, requirements for the duration of data transmissions or calculations are set in the form of (hard) deadlines. Compliance with these deadlines must be proven or (constructively) guaranteed. For this purpose, the lecture presents modelling and analysis methods for (distributed) real-time systems.					
Teaching methods The lecture is held as a seminar-style class, the practical exercises on the computer may include other forms of teaching such as group and project work.					
Examination forms Oral (15-45 minutes) or written (150 minutes) final module exam.					
Requirements for the award of credits Passed final module exam.					
Significance of the grade for the final grade (for a total of 120 ECTS) 5/97: M.Sc. Computer Science					

Module title: Advanced Algorithms					
Modul code	Credit points 9 CP	Workload 270 h	Term siehe Prüfungsordnung / see Examination regulations	Frequency Winter semester	Duration 1 semester
Courses Advanced Algorithms (212029)			Contact time 6 SWS (90 h)	Self-study 180 h	Group size 40 participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Maike Buchin Lecturers: Prof. Maike Buchin					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Mathematik M.Sc. IT-Sicherheit / Informationstechnik					
Precognitions Recommended: Basic knowledge of algorithm design and analysis as known from the Bachelor's programme is expected.					
Learning goals <ul style="list-style-type: none"> • Advanced design methods for algorithms • Advanced analysis methods for algorithms • Knowledge of further data structures and methods for designing data structures • Application of learned methods to new problems 					
Contents In this lecture we look at advanced topics in algorithmics. After a short repetition of known contents, we mainly look at graph algorithms, approximation algorithms and FPT algorithms as well as exact algorithms for NP-hard problems. We also look at some new and well-known data structures and their analysis. The problems considered are combinatorial, graph-theoretical as well as geometric.					
Teaching methods Lecture (as slide and blackboard lecture) and exercises in which the presented contents are deepened.					
Examination forms Oral (15-45 minutes) or written exam (120 minutes) (to be announced at the beginning of the semester).					
Requirements for the award of credits Passed final module examination and successful participation in exercises.					
Significance of the grade for the final grade (for a total of 120 ECTS) 9/97: M.Sc. Computer Science 9/105: M.Sc. Angewandte Informatik 9/91: M.Sc. IT-Sicherheit / Informationstechnik [PO 22]					

Module title: Advanced Quantum Information and Computation					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Tentatively every summer semester	Duration 1 semester
Courses Advanced Quantum Information and Computation (211003)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 30 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Michael Walter, Dr. Simon Schmidt Lecturers: Prof. Dr. Michael Walter					
Module application M.Sc. Computer Science M.Sc. IT-Sicherheit / Informationstechnik M.Sc. IT-Sicherheit / Netze und Systeme M.Sc. Angewandte Informatik M.Sc. Mathematik M.Sc. Physik					
Precognitions Successful participation of Quantum Information and Computation (or an equivalent course). No background in physics is required.					
Learning goals You will learn fundamental concepts, algorithms, and results in quantum information and computation that go beyond a first course. You will be prepared for a research or thesis project in this area.					
Contents This topical course is meant as a follow-up to our introductory course Quantum Information and Computation and is aimed at students interested in deepening their knowledge in this area. We plan to cover selected topics in quantum information and computation, e.g. how to model quantum channels, analyze nonlocal games, design quantum algorithms and cryptographic protocols, and obtain insights into which problems are easy and which are likely hard even for quantum computers. Students interested in a Bachelor's or Master's project in quantum information, computing, cryptography, etc. are particularly encouraged to participate.					
Teaching methods Lecture with exercise					
Examination forms Final exam (the format will depend on the number of participants).					
Requirements for the award of credits Passed final exam (written exam 180 min / oral exam 15 - 45 min)					
Significance of the grade for the final grade (for a total of 120 ECTS)					

5/97: M.Sc. Computer Science

5/91: M.Sc. IT-Sicherheit / Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit / Informationstechnik [PO 20]

5/99: M.Sc. IT-Sicherheit / Netze und Systeme [PO 22]

5/96: M.Sc. IT-Sicherheit / Netze und Systeme [PO 20]

5/105: M.Sc. Angewandte Informatik

Module title: Computational Geometry					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Computational Geometry (211056)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 20 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Maïke Buchin Lecturers: Prof. Dr. Maïke Buchin					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik					
Precognitions Mandatory: Basic knowledge of algorithms and data structures. Recommended: Fundamentals of stochastics.					
Learning goals After successful completion of the module <ul style="list-style-type: none"> • students know basic geometric algorithms and data structures • can analyse and design algorithms according to the sweep paradigm • can design and analyse incremental algorithms, especially randomised incremental algorithms • can analyse and design geometric algorithms according to the divide-and-conquer principle • students can select suitable data structures for range queries. 					
Contents Algorithmic geometry deals with the design and analysis of algorithms and data structures for geometric problems. For this purpose, we first look at some basic problems, such as calculating the convex hull of a set of points, the intersections of a set of distances or a triangulation of a simple polygon. We then look at algorithms for computing well-known geometric structures, such as the Voronoi diagram, the Delaunay triangulation and arrangements. We also look at data structures for efficient queries on geometric data, such as rangetrees, kd-trees and quadtrees. Three main types of algorithms are used: incremental, divide-and-conquer, and sweep. Some of these occur as randomised algorithms.					
Teaching methods Lecture as combined slide and blackboard lecture and associated exercises.					
Examination forms Final oral module exam (15-45 minutes)					
Requirements for the award of credits Passed final oral module exam.					
Significance of the grade for the final grade (for a total of 120 ECTS) 5/105: M.Sc. Angewandte Informatik 5/97: M.Sc. Computer Science					

Module title: Finite Fields: Theory and Algorithms					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Finite Fields: Theory and Algorithms (211058)			Contact time 4SWS (60h)	Self-study 90 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Dr. Christof Beierle Lecturers: Dr. Christof Beierle					
Module application M.Sc. Computer Science M.Sc. IT-Sicherheit / Informationstechnik					
Precognitions Linear algebra (e.g., from "Mathematik 1" from the Bachelor programme), Basic knowledge in programming with python or sage, interest in mathematics					
Learning goals The students obtain a basic understanding of the mathematical foundations behind finite fields, their constructions, their algorithmic aspects, and be able to work with finite fields using a computer algebra system like sage.					
Contents Finite fields have numerous applications in computer science, especially in cryptography and coding theory. This course is an introduction to the theory of finite fields and their algorithmic aspects, starting with the algebraic basics. Topics: algebraic basics, polynomial rings and ideals, Euclidean algorithm for polynomials, irreducible polynomials, extension fields, minimal polynomials, splitting field, existence and uniqueness of finite fields, finding primitive elements of the multiplicative group, Frobenius automorphism, trace and norm, quadratic equations over finite fields, Berlekamp's algorithm for factoring polynomials over small finite fields, permutation polynomials					
Teaching methods Lecture with exercise					
Examination forms written exam (120 minutes)					
Requirements for the award of credits Passed written final exam					
Significance of the grade for the final grade (for a total of 120 ECTS) 5/97: M.Sc. Computer Science 5/91: M.Sc. IT-Sicherheit / Informationstechnik [PO 22]					

Module title: Formal Verification and Model Checking

Modul code	Credit points 5 CP	Workload 150h	Term siehe Prüfungsordnung / see Examination Regulations	Frequency winter semester	Duration 1 semester
Courses Formal Verification and Model Checking (212041)			Contact time 60h (4 SWS)	Self-study 90	Group size 40 participants
Teaching language English			Requirements for participation If the previous module “Model Checking” has already been completed in a previous semester, it is not possible to take the examination for the module “Formal Verification and Model Checking”.		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Nils Jansen Lecturers: Prof. Dr. Nils Jansen					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit/Informationstechnik					
Precognitions Basic knowledge of automata theory.					
Learning goals After completing this course, the participants will be able to <ul style="list-style-type: none">• Solve practical verification problems using modern tools such as SAT/SMT solvers and model checkers like PRISM or Storm.• Understand algorithms underlying modern SAT and SMT solvers, such as resolution, CDCL, and CDCL(T).• Understand formal logics such as LTL or CTL, and understand the algorithmic implications of model checking against such properties• Connect SAT and SMT methods to problems from classical planning, program verification, model checking, and probabilistic inference.• Recognize situations in which the applications of model checking and verification techniques for specification and analysis may be useful.• Work on cutting-edge research that combines machine learning and artificial intelligence with the rigorous techniques of formal verification.					
Contents Complex digital systems have an increasing presence and impact on our lives, making the formal verification of the correctness of these systems crucial. For instance, a spacecraft's program should not crash, and a networking system should stay operational even if one server goes down. To ensure this, one approach is deductive verification, involving the generation of a collection of logical constraints that the system must satisfy. These constraints can be algorithmically verified through automatic theorem provers such as SAT and SMT solvers. Another popular approach is model checking, which consists of representing the system via mathematical models and exhaustively checking certain properties, often described in temporal logics. This course covers various aspects of formal verification and model checking, including: <ul style="list-style-type: none">• Satisfiability of propositions: resolution and the main ingredients of modern satisfiability provers.• Satisfiability modulo theories, in particular using linear inequalities, and the underlying simplex algorithm.					

- Explicit-state and symbolic algorithms for model checking linear-time (LTL) and branching-time (CTL) temporal logics for finite machines.
- Symbolic model checking using BDDs.
- Model checking under probabilities and uncertainty, using Markov chains and Markov decision processes, with the logic PCTL and expected reward properties
- The machine learning technique reinforcement learning, and model checking and formal verification techniques to render it safe to use on critical systems

Teaching methods

- 90 minute weekly lecture, in person.
- 90 minute exercise class, in person.
- Weekly exercises, both practical and theoretical

Examination forms

50% written exam (90 minutes) + 50% project work

Requirements for the award of credits

Passed project and final module examination.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik

Module title: Highlights of Theoretical Computer Science					
Modul code	Credit points 9 CP	Workload 270 h	Term see examination regulations/ siehe Prüfungsordnung	Frequency summer semester	Duration 1 semester
Courses Highlights of Theoretical Computer Science (211057)			Contact time 6 SWS (90 h)	Self-study	Group size 30 participants
Teaching language English			Requirements for participation Successful completion of an introductory course on theoretical computer science (covering formal languages, basics of complexity theory including NP-completeness and reductions, basics of computability theory). Interest and motivation to learn about theoretical concepts.		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Michael Walter Prof. Dr. Thomas Zeume Lecturers: Prof. Dr. Michael Walter Prof. Dr. Thomas Zeume Dr. Vladimir Lysikov					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit / Informationstechnik M.Sc. IT-Sicherheit / Netze und Systeme					
Precognitions					
Learning goals You will know some of the most important results and insights of modern theoretical computer science. You will learn approaches and techniques that go well beyond a first course. You will be able to recognize when these can be used and how to adapt them to new situations. You will be able to independently acquire new knowledge in this area.					
Contents The insights and techniques of modern theoretical computer science have been key for advances in all areas of computer science. In this course, we will discuss some highlights and the techniques that underpin them. Possible topics that we might cover: <ul style="list-style-type: none"> • Computational models (is there life beyond Turing machines?) • Kolmogorov complexity (what is the shortest program that produces some output?) • Communication complexity (how many bits must Alice and Bob exchange to jointly solve a problem?) • Fine-grained complexity (are some easy problems easier than others? and why?) • Fast multiplication of numbers and matrices (can you beat the high-school method?) • Randomness (does it really help to compute faster?) • Circuit lower bounds (why is it so hard to prove that problems are hard?) • Convex optimization (how to maximize profit if all you can ask are yes/no questions) • Hardness of approximation (how easy is it to find near-optimal solutions?) 					

- Cryptography and computation

If you enjoyed your first course in theoretical computer science in the Bachelor's and would like to deepen your knowledge by getting an overview of the fascinating theory of computing, then this course will be exactly right for you.

Teaching methods

Lecture with Exercises

Examination forms

Final module examination. Format will depend on number of participants.

Requirements for the award of credits

Passed module final exam (written exam 180 minutes or oral exam 15-45 minutes)

Significance of the grade for the final grade (for a total of 120 ECTS)

9/97: M.Sc. Computer Science

9/105: M.Sc. Angewandte Informatik

9/91: M.Sc. IT-Sicherheit / Informationstechnik [PO 22]

9/84: M.Sc. IT-Sicherheit / Informationstechnik [PO 20]

9/99: M.Sc. IT-Sicherheit / Netze und Systeme [PO 22]

9/96: M.Sc. IT-Sicherheit / Netze und Systeme [PO 20]

Module title: Computational complexity theory

Modul code	Credit points 9 CP	Workload 270 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Computational complexity theory (211028)			Contact time 6 SWS (90 h)	Self-study 180 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Thomas Zeume Lecturers: Prof. Thomas Zeume					
Module application M.Sc. Computer Science M.Sc. IT-Sicherheit/Informationstechnik M.Sc. IT-Sicherheit/Netze und Systeme M.Sc. Angewandte Informatik (until SS 23)					
Precognitions Knowledge from a basic course in theoretical computer science (basics of complexity theory including NP-completeness and reductions) is expected.					
Learning goals Students learn to classify algorithmic problems in terms of their complexity and thus identify suitable algorithmic techniques for their solution. In particular, they can apply algorithmic methods for NP-complete problems. They can deal with different computational models and are able to prove simple statements about them. They learn to evaluate their own and other people's approaches to solutions in discourse.					
Contents Complexity theory examines and classifies computational problems in terms of their algorithmic difficulty. The aim is to determine the inherent resource consumption in terms of various resources such as computing time or memory space, and to group problems with similar resource consumption into complexity classes. The best-known complexity classes are certainly P and NP, which comprise the problems that can be solved or verified in polynomial time. The question of whether P and NP are different is considered one of the most important open questions in theoretical computer science, and even in mathematics. However, P and NP are only two examples of complexity classes. Other classes arise, among others, in the investigation of the required memory space, the efficient parallelisability of problems, the solvability by random algorithms, and the approximate solvability of problems. The lecture aims to give a broad overview of the basic concepts and results of complexity theory: <ul style="list-style-type: none">• Classical results for space and time complexity classes: e.g., the correspondence between games and memory constraints, the proof that with more space or time, more problems can be solved, other basic relations between time and space-based classes, and the complexity world between NP and PSPACE.• Basic features of complexity theory of parallel, random and approximate algorithms.• Introduction to selected recent topics: Complexity theory of interactive computing, probabilistic reasoning and fine-grained complexity.					

Teaching methods

Lecture with exercise

Examination forms

Oral module final exam (15-45 minutes)

Requirements for the award of credits

Passed final oral module exam

Significance of the grade for the final grade (for a total of 120 ECTS)

9/97: M.Sc. Computer Science

9/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

9/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

9/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO22]

9/96: M.Sc. IT-Sicherheit/ Netze und Systeme [PO20]

Module title: Proofs are programs [M.Sc.] (no offer in WS 25/26)

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency winter semester	Duration 1 semester
Courses Proofs are Programms (211003)			Contact time 60 h (4 SWS)	Self-study 90 h	Group size 40 participants
Teaching language English			Requirements for participation None		

Module coordinators and Lecturers

Module coordinators: Dr. Catalin Hritcu
Lecturers: Dr. Catalin Hritcu
Dr. Clara Schneidewind

Module application

M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. IT-Sicherheit/ Netze und Systeme

M.Sc. Computer Science

Precognitions

The lecture is intended for a broad range of students, from motivated BSc students to MSc and PhD students. No specific prior knowledge in logic, programming, functional programming, or programming languages is assumed, though a degree of mathematical maturity is helpful.

Learning goals

After successful completion of this course, students will be able to

- develop purely functional programs using recursive functions on numbers, lists, maps, and various kinds of trees, including the abstract syntax trees of programs;
- use functional programming concepts such as type polymorphism and higher-order functions, which are increasingly becoming mainstream;
- formally state and prove theorems in the Coq proof assistant;
- apply different proof techniques in Coq (e.g. equational reasoning, contradiction, case analysis, induction on natural numbers, lists, and trees, induction on rule derivations, proof automation);
- define new inductive types and relations in Coq and prove statements about them;
- write simple proof terms and understand the connection between constructive logics and typed functional programming that is at the heart of Coq, in which propositions are types and proofs are programs;
- comprehend how the syntax and semantics of simple imperative programs can be formally defined in Coq and how to prove theorems about such programs and languages;
- understand how the absence of information leaks can be formalized as a security property called noninterference and enforced using secure-multi execution or simple type systems.

Contents

Complex proofs on paper are difficult to construct, check, and maintain. This holds not only for interesting proofs in mathematics, but also for complex formal proofs about interesting programs. For this reason, machine-checked proofs created with the help of interactive tools called proof assistants are gaining increased traction in academia and industry. Proof assistants have been used to prove the correctness and security of realistic compilers, operating systems, cryptographic libraries, or smart contracts, and also to construct machine-checked proofs for challenging theorems in mathematics.

This course introduces the Coq proof assistant [3] and explains how to use it to prove properties about functional programs and inductive relations, how to formally define a simple imperative programming language, and how to securely enforce information-flow control for functional and imperative programs. The Coq proof assistant enables us to program formal proofs interactively and it machine-checks the correctness of the proofs along the way. The design of the Coq proof assistant itself exploits a beautiful connection between programs in typed functional programming languages and proofs in constructive logics, which is known as the Curry-Howard Correspondence [4]. This deep connection between programs and proofs should make this course interesting to not only to computer scientists, but also to mathematicians and other scientists. The goal is to demystify proofs as just programs in an elegant programming language, for which the course provides a gentle introduction. The course also shows that proofs are not only a way to convince a human reader, but they can actually be fully formalized in a proof assistant like Coq and automatically checked by a computer.

This hands-on course is based on the Logical Foundations [1] and Security Foundations [2] online textbooks, which are themselves formalized and machine-checked in the Coq proof assistant. The many exercises in each book chapter are to be solved weekly mostly in Coq, from easy exercises allowing the students to practice concepts from the lecture, building incrementally to slightly more interesting programs and proofs and also to various optional challenges. Finally, this course serves as the base for a more advanced course on “Foundations of Programming Languages, Verification, and Security”.

Teaching methods

This course consists of lectures and weekly exercises, in which the students will solve problems using the Coq proof assistant for which they can get help from a tutor.

Examination forms

Written final exam (120 minutes).

Requirements for the award of credits

Passing the final written exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/97: M.Sc. Computer Science

Module title: Quantum Information and Computation [M.Sc.]

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Quantum Information and Computation (212011)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size participants
Teaching language German or English (depends on audience)			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Michael Walter Lecturers: Prof. Dr. Michael Walter					
Module application M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science					
Precognitions Familiarity with linear algebra (in finite dimensions) and probability (with finitely many outcomes) at the level of a first Bachelor's course; we will briefly remind you of the more difficult bits in class. In addition, some mathematical maturity, since we will discuss precise mathematical statements and rigorous proofs. No background in physics is required.					
Learning goals You will learn fundamental concepts, algorithms, and results in quantum information and computation. After successful completion of this course, you will know the theoretical model of quantum information and computation, how to generalize computer science concepts to the quantum setting, how to design and analyze quantum algorithms and protocols for a variety of computational problems, and how to prove complexity theoretic lower bounds. You will be prepared for an advanced course or a research or thesis project in this area. Master's students will be expected to understand the material in a deeper way, which will reflect itself in homework and examination.					
Contents This course will give an introduction to quantum information and quantum computation from the perspective of theoretical computer science. Topics to be covered will likely include: <ul style="list-style-type: none">• Fundamentals of quantum computing: quantum bits, states and operations• The power of quantum entanglement: nonlocal games• Entanglement as a resource: superdense coding and teleportation• Quantum circuit model of computation• Quantum computing with oracles: Deutsch-Jozsa, Bernstein-Vazirani, Simon• Quantum Fourier transform and phase estimation• Shor's factoring algorithm• Grover's search algorithm and beyond: how to solve SAT on a quantum computer?• From no cloning to quantum money: a peek at quantum cryptography					

The course should be of interest to students of computer science, mathematics, physics, and related disciplines. Students interested in a BSc or MSc project in quantum information, computing, cryptography, etc. are particularly encouraged to participate.

Teaching methods

Lecture with Exercise

Examination forms

Final written module exam (180 minutes)

Requirements for the award of credits

Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/105: M.Sc. Angewandte Informatik

5 /91: M.Sc. IT-Sicherheit/ Informationstechnik

5/ 99: M.Sc. IT-Sicherheit/ Netze und Systeme

5/97: M.Sc. Computer Science

Module title: Advanced Automatic Testing					
Modul code	Credit points 5 CP	Workload 150h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Advanced Automatic Testing (211067)			Contact time 60 h (4 SWS)	Self-study 90h	Group size 20 participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Flavio Toffalini Prof. Dr. Yannic Noller Lecturers: Prof. Dr. Flavio Toffalini Prof. Dr. Yannic Noller					
Module application					
Precognitions The students should have prior knowledge of system and software security. Previous classes in system programming languages (e.g., C and C++) and a minimum of adversary security.					
Learning goals Students will gain knowledge of automatic testing techniques with a focus on practical aspects in security problems. Additionally, the course will explore cutting-edge advancements in the field.					
Contents This course delves into the foundations of automatic testing (fuzzing) in software security, providing Master's students with an opportunity to deepen their expertise in the field. The course offers a comprehensive overview of automatic testing, covering fundamental concepts such as White-box, Grey-box, and Black-box testing, standard and modern code exploration techniques, and advanced bug detection using logic-based oracles. Each lesson focuses on a specific aspect of the discipline, progressively building a complete understanding of the subject and equipping students with the skills to independently explore new concepts in this domain. The course will cover the following topics, with adjustments made as needed to suit the class's requirements: <ul style="list-style-type: none"> • Introduction to the fuzzing paradigm: Black-box, Grey-box, and White-box approaches • Types of coverage feedback • Heuristics for code exploration (e.g., mutators, meta-strategies, seed selection, grammars) • Basic concepts of symbolic execution and concolic testing • Bug detection and replication • Bug oracles (e.g., sanitizers, differential testing) • Managing fuzzing campaigns 					

- Domain-specific testing (e.g., kernel, virtual devices, libraries, IoT devices)

Laboratory sessions complement the in-class lectures by offering hands-on experience with the principles taught. These exercises are essential for developing the practical knowledge and problem-solving skills required for the final exam and for thoroughly understanding the material presented in the course.

Teaching methods

The course combines weekly lectures and laboratory sessions. During the lectures, students are encouraged to engage with the instructor and participate in solving simple exercises designed to enhance their learning experience.

Examination forms

Written exam (120 minutes). Bonus points can be earned in the exercises.

Requirements for the award of credits

The students have to pass the final written exam.

They can gain extra points for solving the exercises presented in the laboratory

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit / Informationstechnik

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme

Module title: Blockchain and Decentralized Security

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Blockchain and Decentralized Security (212007)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 30 participants
Teaching language English			Requirements for participation Pre-requisites: Intro to Crypto 1 and 2, System Security, Network Security 1.		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Ghassan Karame Lecturers: Prof. Dr. Ghassan Karame					
Module application M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science					
Precognitions Background in system security, network security, cryptographic primitives (encryption methods, signatures, MACs, hash functions), and principles of communication networks is required.					
Learning goals Upon completion of this course, students are expected to be able to: <ol style="list-style-type: none">1. Reason about the security and privacy definitions of decentralized systems.2. Explain the security of blockchains in light of the state of the art reported attacks.3. Reason about possible network security and cryptographic countermeasures to deter attacks on decentralized platforms (blockchains and distributed platforms).4. Explain best security/privacy practices to strengthen the security of existing blockchains and existing distributed learning platforms, and extract relevant lessons for the design of next-generation decentralized technologies.					
Contents The main objective of the course is to provide a comprehensive overview of the security and privacy of decentralized technologies. Course participants will be also introduced to the basic security and privacy provisions of existing popular blockchains, and will be exposed to the state-of-the-art attacks and threats reported against existing systems/deployments. The participants will also reason on the effectiveness of combining network-level security primitives, with novel cryptographic primitives to deter attacks on payment systems and on the security of federated learning and distributed learning.					
Teaching methods Lecture with Exercise					
Examination forms Written exam (120 minutes)					
Requirements for the award of credits Passed written exam.					

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: Master IT-Sicherheit | Informationstechnik [PO 22]

5/84: Master IT-Sicherheit | Informationstechnik [PO 20]

5/99: Master IT-Sicherheit | Netze und Systeme [PO 22]

5/96: Master IT-Sicherheit | Netze und Systeme [PO 20]

5/97: Master Computer Science

Module title: Digital Sovereignty					
Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Digital Sovereignty (211059)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size 25 participants
Teaching language German or English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Karola Marky Lecturers: Prof. Dr. Karola Marky					
Module application M.Sc. IT-Sicherheit / Informationstechnik M.Sc. IT-Sicherheit / Netze und Systeme M.Sc. Computer Science					
Precognitions recommended: Introduction to Usable Security and Privacy					
Learning goals Students know various definitions, contexts and use cases for digital sovereignty as well as mechanisms of influence of modern digital products. Students can independently analyze and evaluate new use cases.					
Contents In this lecture, students gain an understanding of digital sovereignty in today's world. Various topics will be covered. First, the lecture provides a basic overview of the scope of digital sovereignty and interactions within society. Then design principles and use cases are explained in the context of the general population, organizations and states, including the sharing of data in the digital space, IT security in organizations, and e-demography. The lecture is accompanied by a project, which students work on in groups and thus learn to evaluate certain scenarios independently through "hands-on".					
Teaching methods Lecture with exercise (project)					
Examination forms oral exam (15-45 minutes)					
Requirements for the award of credits Passing final oral exam; Bonus points are awarded for the quality of the implementation of the project to be worked on.					
Significance of the grade for the final grade (for a total of 120 ECTS) 6/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22] 6/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20] 6/99: M.Sc. IT-Sicherheit/Netze und Systeme [PO 22] 6/96: M.Sc. IT-Sicherheit/Netze und Systeme [PO 20]					

Module title: Cryptography					
Modul code	Credit points 8 CP	Workload 240 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Kryptographie (212017)			Contact time 6 SWS (90 h)	Self-study 150 h	Group size 100 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Alex May Lecturers: Prof. Dr. Alex May					
Module application B.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science M.Sc. Angewandte Informatik					
Precognitions Contents of the lectures Introduction to Cryptography 1 and 2.					
Learning goals The students have an understanding of the essential mathematical methods and procedures on which modern cryptographic procedures are based. The depth of the treatment of the methods goes well beyond that taught in the previous courses. Students will be able to analyse and design current and future cryptographic methods. They also have an awareness of the methodology and power of various attack scenarios.					
Contents An introduction to modern methods of symmetric and asymmetric cryptography is provided. For this purpose, an attacker model is defined and the security of the presented encryption, hash and signature methods is proven under well-defined complexity measures in this attacker model. Topic Overview: <ul style="list-style-type: none"> • Secure encryption against KPA, CPA and CCA attackers • Pseudo-random functions and permutations • Message Authentication Codes • Collision-resistant hash functions • Block ciphers • Construction of random number generators • Diffie-Hellman key exchange • Trapdoor one-way permutations • Public key encryption: RSA, ElGamal, Goldwasser-Micali, Rabin, Paillier • One-way signatures • Signatures from collision-resistant hash functions • Random Oracle Model 					

Teaching methods

Lecture and exercises

Examination forms

Written exam (120 minutes)

Requirements for the award of credits

Passed written final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

8/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

8/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

8/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

8/96: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 20]

8/97: M.Sc. Computer Science

8/105: M.Sc. Angewandte Informatik

Module title: Cryptography on hardware-based platforms					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Cryptography on hardware-based platforms (212019)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 50 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr.-Ing. Tim Güneysu Lecturers: Prof. Dr.-Ing. Tim Güneysu					
Module application B.Sc. IT-Sicherheit/ Informationstechnik B.Sc. Angewandte Informatik [until WS 22/23] M.Sc. IT-Sicherheit/ Netze und Systeme [until WS 22/23] M.Sc. Computer Science					
Precognitions					
Learning goals The students know the concepts of practical hardware development with abstract hardware description languages (VHDL) and the simulation of hardware circuits on FPGAs. They master standard techniques of hardware-related processor development and are able to implement symmetric and asymmetric crypto systems on modern FPGA systems.					
Contents Due to their complexity, cryptographic systems place high demands on small processors and embedded systems in particular. In combination with the demand for high data throughput at lowest hardware costs, fundamental problems arise for the developer, which are to be illuminated in this lecture. The lecture deals with the most interesting aspects of how to implement current cryptographic methods on practical hardware systems. Cryptosystems such as the block cipher AES, the hash functions SHA-1 as well as asymmetric systems RSA and ECC are dealt with. Furthermore, special hardware requirements such as the generation of true randomness (TRNG) and the use of Physically Unclonable Functions (PUF) are discussed. The efficient implementation of these cryptosystems, especially with regard to optimisation for high speed, is discussed on modern FPGAs and implemented in practical exercises using the hardware description language VHDL. A Moodle course is offered to accompany the lecture, which provides additional content as well as the practical exercises.					
Teaching methods Lecture with exercise					
Examination forms Written exam (120 minutes)					
Requirements for the award of credits Passed final written module exam; successful participation in the exercise can earn up to 10 percent bonus points, which can be credited to the result of the module exam.					

Significance of the grade for the final grade (for a total of 120 ECTS)

5/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/97: M.Sc. Computer Science

Module title: Cryptographic protocols					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Kryptographische Protokolle (211031)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Eike Kiltz Lecturers: Prof. Dr. Eike Kiltz					
Module application M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Angewandte Informatik [until SS 23] M.Sc. Computer Science					
Precognitions Content of the module Cryptography					
Learning goals <ul style="list-style-type: none"> • Deepening understanding in provable security • Writing error-free security reductions • New techniques for security proofs • Learning advanced cryptographic constructions 					
Contents The lecture deals with advanced cryptographic protocols and their applications. Topic overview: <ul style="list-style-type: none"> • Game-based security definitions and proofs • Bilinear maps • Digital Signatures • Identification Protocols • Zero-Knowledge Proofs • Identity-based Encryption • CCA-secure encryption 					
Teaching methods Lecture with exercise					
Examination forms Oral (15-45 minutes) or written exam (120 minutes), depending on the number of participants. Will be communicated at the beginning of the course.					
Requirements for the award of credits Passed oral or written final module exam.					

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/96: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 20]

5/97: M.Sc. Computer Science

Module title: Microarchitectural Attacks and Defenses					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Microarchitectural Attacks and Defenses (212064)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 30 participants
Teaching language English			Requirements for participation none		
Module coordinators and Lecturers Module coordinators: Prof. Yuval Yarom Lecturers: Prof. Yuval Yarom					
Module application M.Sc. ITS - Informationstechnik M.Sc. ITS - Netze und Systeme M.Sc. Computer Science					
Precognitions The course assumes that students can program in C or learn the language as they go. They need enough experience to be able to program on remote machines, using SSH. Basic understanding of how computers work, assembly language, and the role of the operating system is required. Understanding of basic concepts in computer security (security domains, vulnerabilities, etc.) and familiarity with basic cryptography (AES, RSA, ECC) is helpful.					
Learning goals <ul style="list-style-type: none"> • Diagnose microarchitectural vulnerabilities • Assess software for resilience against microarchitectural vulnerabilities • Design and program proof-of-concept exploits of vulnerable software and hardware • Design and implement countermeasures for software executing on vulnerable hardware 					
Contents The course covers the area of microarchitectural attacks and defences. It starts with cache attacks, covering the main techniques (Prime+Probe, Evict+Time, and Flush+Reload). Building on this basis it explores variants of the attacks targeting other storage elements as well as attacks that exploits bandwidth limitations. In parallel with exploring these attacks, the course will describe various countermeasures, with special focus on constant-time programming. The course then switches to speculative execution attacks, identifying and classifying the various attacks, defences, and counter-attacks. The course further covers several related attacks, including Rowhammer and voltage- and frequency-based attacks. Additionally, the course pays special attention to attack scenarios exploring, in particular, attacks on the operating system kernel, web-based and other remote attacks, and attacks on trusted execution environments. The course puts special focus on practical implementation of both attack and defence techniques.					
Teaching methods Lecture with exercise					
Examination forms Project work with assignments.					

Requirements for the award of credits

Passed project work with assignments.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. ITS - Informationstechnik [PO 22]

5/84: M.Sc. ITS - Informationstechnik [PO 20]

5/99: M.Sc. ITS - Netze und Systeme [PO 22]

5/96: M.Sc. ITS - Netze und Systeme [PO 20]

5/97: M.Sc. Computer Science

Module title: Mobile Network Security					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Mobile Network Security (211012)			Contact time 60h (4 SWS)	Self-study 90 h	Group size participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Katharina Kohls Lecturers: Prof. Dr. Katharina Kohls					
Module application					
Precognitions The lecture focuses on mobile network security. Prior knowledge in the context of computer networks and their security mechanisms helps to understand the technical concepts that will be addressed in the lecture.					
Learning goals <ul style="list-style-type: none"> • Knowing mobile network architectures and their components in 4G and 5G networks • Understanding existing attacks and their attack vectors, as well as security mechanisms that avoid known attacks • Experience with scientific work in mobile network security 					
Contents Mobile networks are an integral part of our everyday lives. Their use cases range from casual web browsing over campus networks in industrial environments to first responder communication. In this course, we cover the technical aspects of mobile networks and address their security capabilities. After an introduction to the technical foundations of mobile network deployments, we will go into detail with scientific work on existing attacks against 4G and 5G networks. To this end, we analyze open attack vectors and discuss the consequences of attacks if being conducted in real-world infrastructures.					
Teaching methods The course consists of lectures that provide theoretical knowledge and practical exercises that help to apply the contents of the lectures.					
Examination forms Written exam with a duration of 120 minutes.					
Requirements for the award of credits Passing the exam					
Significance of the grade for the final grade (for a total of 120 ECTS) 5/97: M.Sc. Computer Science 5/105: M.Sc. Angewandte Informatik 5/91: M.Sc. IT-Sicherheit / Informationstechnik 5/99: M.Sc. IT-Sicherheit/ Netze und Systeme					

Module title: Physical Attacks and Countermeasures

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses			Contact time 4SWS (60 h)	Self-study 90 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Dr. Jan Richter-Brockmann Lecturers: Dr. Jan Richter-Brockmann					
Module application M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science					
Precognitions Understanding the english language, basic knowledge of digital technology, basic knowledge of data security and cryptography, solid programming ability in at least one programming language (e.g. C++), basic knowledge of computer architecture, basic knowledge of signal processing.					
Learning goals The students <ul style="list-style-type: none">• understand the kinds of physical attacks, their prerequisites, and their required conditions to work.• are capable of evaluating measurement data based on the learned methods in order to assess the security level of an implementation.• are aware of the dangers that physical attacks pose for implementations of cryptographic algorithms.• know the countermeasure schemes and how to apply them in order to protect a cryptographic design against physical attacks.					
Contents Modern cryptographic algorithms provide a reasonable level of security against known mathematical and cryptanalytic attacks. These cryptographic primitives are implemented on different platforms to be used in a security-enabled applications. Such a realization is done by implementing the desired cryptographic algorithm using some program code (in software) or using logic elements/circuits (in hardware). Physical access of the users to the cryptographic devices (e.g., a smartcard used for payment, a contactless card used for authentication, or a smartphone) where a secret key is embedded, led to a new form of attacks called physical attacks. This kind of attacks aims at extracting the secret key used by the cryptographic algorithm from the target implementation. Breaking a system by means of a physical attack does not infer to the weakness of the algorithm, but of the implementation. Therefore, considering such kinds of attacks as a potential risk for the security is a must when designing a cryptographic device and weaknesses in that regard need to be avoided from the start. The goal of this lecture is to give an overview about the known physical attacks and most considerably the schemes developed to counter such kinds of attacks. In the first part of the lecture different kinds of physical attacks are introduced, while in the second part we focus on countermeasures and the methods to make implementations resistant against known physical attacks.					
Teaching methods Lecture with exercise					

Examination forms

Written exam (120 minutes) and project work (during the semester)

Requirements for the award of credits

Project-based work is a large part of the course. In addition to a written examination, there is weekly project work (homework). Both parts must be completed individually, are assessed and are included in the final grade. The two parts are assessed as follows:

Weekly project work (homework): 30

Written exam: 70

It is possible to acquire bonus points for the exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/97: M.Sc. Computer Science

Module title: Privacy Engineering, Data Governance and Usability					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung	Frequency	Duration 1 semester
Courses Privacy Engineering, data governance and usability (212037)			Contact time 45 h	Self-study 105 h	Group size 20 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Dr. Veelasha Moonsamy Lecturers: Dr. Veelasha Moonsamy, Dr. Asia Biega Dr. Yixin Zou					
Module application M.Sc. IT-Sicherheit/Informationstechnik M.Sc. IT-Sicherheit/Netze und Systeme					
Precognitions Recommended but not mandatory: Einführung in die Usable Security and Privacy (211036) Datenschutz (260081) Basic knowledge of threat modeling General understanding of machine learning and data science					
Learning goals By the end of the course, the student will be able to: Reason about privacy concerns and perform threat modelling Apply privacy-by-design techniques for systems implementation Develop privacy technologies Understand concepts related to data governance, including data minimization Design privacy-friendly, usable systems Understand concept related to UX design & usable privacy					
Contents This course will provide students with the knowledge and applied skills to tackle the design and implementation of privacy-preserving systems. Students will gain a critical understanding of privacy's role in society and tensions between privacy, technology and security. Students will learn to analyze privacy issues and develop privacy-friendly solutions by considering social, technical, legal and public policy aspects. The course includes mandatory lecture attendance, readings and group project. The course will cover the following topics: Privacy definitions and concepts Privacy by design Privacy engineering: design and evaluation Data governance Notion of "Right to be forgotten" Usable privacy, including UX design Inclusive privacy					

Teaching methods

The course includes mandatory lecture attendance, readings and group project.

Examination forms

There will be one semester-long individual project and a written exam (60 minutes).

Requirements for the award of credits

Final grade: 50% project + 50% exam. You need to pass the exam (i.e. achieve more than 50 points) in order to pass the course.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

5/99: M.Sc. IT-Sicherheit/Netze und Systeme [PO 22]

5/96: M.Sc. IT-Sicherheit/Netze und Systeme [PO 20]

Module title: Program Analysis					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Program Analysis (211015)			Contact time 4 SWS (60 h)	Self-study 90h	Group size 40 participants
Teaching language English			Requirements for participation none		
Module coordinators and Lecturers Module coordinators: Prof. Kevin Borgolte Lecturers: Prof. Kevin Borgolte					
Module application M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science					
Precognitions Experience in system-oriented programming, assembler and programming in C are helpful for understanding the topics taught. Previous knowledge from the lectures on system security/operating system security is helpful but not necessary for understanding the topics.					
Learning goals Students are familiar with various concepts, techniques and tools from the field of program analysis. This includes an overview of various concepts from the field of reverse engineering and binary analysis. Students have a basic understanding of both static and dynamic methods for analyzing a given program. They are able to describe various aspects of program analysis and apply them to new problems.					
Contents The following topics and techniques from the field of program analysis are covered in the lecture: <ul style="list-style-type: none"> • Static and dynamic analysis of programs • Analysis of control and data flow • Symbolic execution • Taint tracking • Binary instrumentation • Program Slicing • Overview of existing analysis tools <p>In addition, the first part of the lecture provides an introduction to x86/x64 assembler and presents the basic techniques from the field of reverse engineering. The lecture is accompanied by exercises in which the concepts and techniques presented are practiced.</p>					
Teaching methods Lecture with exercise					
Examination forms Oral (15-45 minutes) or written exam (120 minutes) (to be announced at the beginning of the semester), registration: FlexNow					

Requirements for the award of credits

Passed final module examination

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/96: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 20]

5/97: M.Sc. Computer Science

Module title: Public Key Cryptanalysis 1					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer Semester	Duration 1 semester
Courses Public Key Kryptanalyse 1 (211055)			Contact time 3 SWS (45 h)	Self-study 105 h	Group size 20 participants
Teaching language German			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Alex May Lecturers: Prof. Alex May					
Module application M.Sc. IT-Sicherheit / Informationstechnik M.Sc. IT-Sicherheit / Netze und Systeme M.Sc. Computer Science					
Precognitions Prerequisites are elementary knowledge of linear algebra (Mathematics 1 for Computer Scientists) and an interest in algorithmic techniques and cryptography, in theory and practice (implemented using the computer algebra system Sage).					
Learning goals Students should acquire a broad knowledge of algorithmic techniques of asymmetric cryptanalysis, in particular for coding-based cryptography. After successfully completing the module students <ul style="list-style-type: none"> • know basic key finding algorithms such as brute force and meet-in-the-middle and can apply them to new cryptographic systems, • master the basics of linear codes and their dual codes, in particular the McEliece cryptosystem as a cryptographic application, • know time-memory techniques such as Pollard Rho and Parallel Collision Search and can apply them to new problems, • have an overview of all current decoding algorithms in the field of information set decoding that are relevant for the security evaluation of modern coding-based cryptosystems, • have learned advanced techniques for speedups using quantum computers, • are able to implement cryptanalysis techniques using the computer algebra Sage. 					
Contents Cryptanalysis is used to instantiate cryptographic systems in such a way that they offer a predefined level of security on the one hand, but are as performant as possible on the other. Cryptanalysis offers a whole toolbox of algorithmic techniques to realize the evaluation of new cryptographic systems. This includes both classical algorithms and algorithms for quantum computers, so that the cryptography used remains secure even in an era of quantum computers.					
Teaching methods The lecture will be held in the form of a seminar, and the practical exercises on the computer with Sage computer algebra will also include other forms of teaching such as group and project work.					
Examination forms Written exam of 120 minutes					

Requirements for the award of credits

Passed written final module examination

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91 M.Sc IT-Sicherheit/ Informationstechnik [PO22]

5/84 M.Sc. IT-Sicherheit/ Informationstechnik [PO20]

5/99 M.Sc IT-Sicherheit/ Netze und Systeme [PO22]

5/96 M.Sc IT-Sicherheit/ Netze und Systeme [PO20]

5/97: M.Sc. Computer Science

Module title: Quantum Cryptography (no offer in WS 25/26)

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Quantum Cryptography (212016)			Contact time 4 SWS (60 h)	Self-study 90	Group size participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Michael Walter Lecturers: Prof. Michael Walter Dr. Giulio Malavolta					
Module application M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Computer Science					
Precognitions None					
Learning goals You will learn fundamental concepts, algorithms, protocols, and results in quantum (and quantum-resistant) cryptography. After successful completion of this course, you will know how to generalize cryptographic concepts to the quantum setting, how quantum algorithms can attack well-known cryptographic protocols, and how to design and analyze classical and quantum protocols for protecting classical and quantum data against quantum adversaries. You will be prepared for a research or thesis project in this area.					
Contents This course will give an introduction to the interplay of quantum information and cryptography, which has recently led to much excitement and insights – including by researchers at CASA right here on our very own campus. We will begin with a brief introduction to both fields and discuss in the first half of the course how quantum computers can attack classical cryptography and how to overcome this challenge – either by protecting against the power of quantum computers or by leveraging the power of quantum information. In the second half of the course, we will discuss how to generalize cryptography to protect quantum data and computation. Topics to be covered will likely include: * Basic quantum computing * Basic cryptography * Quantum attacks on classical cryptography * Quantum random oracles and compressed oracle technique * Quantum-resistant cryptography in light of the NIST competition * Classical vs quantum information					

- * Quantum money
- * Quantum key distribution
- * Quantum complexity theory
- * Quantum pseudorandomness
- * From classical to quantum fully homomorphic encryption
- * Classical verification of quantum computation
- * Quantum rewinding

This course should be of interest to students of computer science, mathematics, physics, and related disciplines. Students interested in a Master's project in quantum or quantum-resistant cryptography, quantum information, quantum computing, and similar are particularly encouraged to participate.

Teaching methods

Lecture with exercise

Examination forms

Oral (30 minutes) or written final module examination (120 minutes), depending on the number of participants. Will be communicated at the beginning of the course.

Requirements for the award of credits

Passed oral or written final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91 M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/84 M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/99 :M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/96 :M.Sc. IT-Sicherheit/ Netze und Systeme [PO 20]

5/97: M.Sc. Computer Science

Module title: Software Security 1 [M.Sc.]

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency winter semester	Duration 1 semester
Courses Software Security 1 (212026)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Kevin Borgolte Lecturers: Prof. Kevin Borgolte					
Module application M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme M.Sc. Angewandte Informatik M.Sc. Computer Science					
Precognitions Prior knowledge about programming in Python, C, and assembler is recommended. The following courses (or equivalent) are required: <ul style="list-style-type: none">• System Security (211011)• Operating Systems (211005)					
Learning goals At the end of this course, students will be able to: <ul style="list-style-type: none">• analyze user-space software vulnerability types and protection mechanisms• understand how to write code to reduce the risk of vulnerabilities and apply defensive programming techniques• identify new software vulnerabilities and evaluate their impact• show the existence of a vulnerability, for example, by developing proof of concept verifications					
Contents The course covers the area of introductory software security, vulnerability discovery, and vulnerability verification, focusing on: <ul style="list-style-type: none">• Assembly and Disassembly, Shellcode• Binary Reverse Engineering and Debugging• Memory and Type Safety/Errors• Stack-based Buffer Overflows• Heap Attacks• Information Leakage• Format String Vulnerabilities• Code Re-use Attacks• Types and Type Safety• Race Conditions					

Teaching methods

Lecture with exercises

Examination forms

Combinated exam: written exam (120 minutes) 40% + practical exercises 60% (both parts need to be passed)

Requirements for the award of credits

Passed combinated exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. Computer Science

Module title: Software Security 2

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency summer semester	Duration 1 semester
Courses Software Security 2 (211063)			Contact time 60h (4 SWS)	Self-study 90 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Kevin Borgolte Lecturers: Prof. Kevin Borgolte					
Module application M.Sc. IT-Sicherheit/Informationstechnik M.Sc. IT-Sicherheit / Netze und Systeme M.Sc. Computer Science					
Precognitions Prior knowledge about programming in Python, C, and assembler is highly recommended. The following courses (or equivalent) are required: <ul style="list-style-type: none">• System Security (211011)• Operating Systems (211005)• Software Security 1 (212026)					
Learning goals At the end of this course, students will be able to: <ul style="list-style-type: none">• classify and describe complex vulnerabilities and advanced protection mechanisms of a diverse set of software systems• analyze and reason about protection mechanisms for modern software systems across its layers from userspace to kernel to hypervisor• identify end-to-end vulnerabilities in software systems• develop proofs of concept exploits/verifications to show the existence of an end-to-end vulnerability in a modern software system with modern defenses• understand how to write code defensively to reduce the risk of vulnerabilities					
Contents The course covers the area of advanced topics in software security, vulnerability discovery, and vulnerability verification, focusing on: <ul style="list-style-type: none">• Attacks on Just-in-time Compilers• Sandboxing Techniques• Browser Vulnerabilities• Kernel and Hypervisor Vulnerabilities• Non-x86 Architectures• Non-Linux Operating Systems• Automated Exploit/Verification Synthesis					
Teaching methods lecture with exercises					

Examination forms

Practical exam.

Requirements for the award of credits

Passed practical exam. The exact schedule and details will be communicated during the first lecture.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/Informationstechnik

5/99: M.Sc. IT-Sicherheit / Netze und Systeme

5/97: M.Sc. Computer Science

Module title: Advanced Automatic Testing					
Modul code	Credit points 5 CP	Workload 150h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Advanced Automatic Testing (211067)			Contact time 60 h (4 SWS)	Self-study 90h	Group size 20 participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Flavio Toffalini Prof. Dr. Yannic Noller Lecturers: Prof. Dr. Flavio Toffalini Prof. Dr. Yannic Noller					
Module application					
Precognitions The students should have prior knowledge of system and software security. Previous classes in system programming languages (e.g., C and C++) and a minimum of adversary security.					
Learning goals Students will gain knowledge of automatic testing techniques with a focus on practical aspects in security problems. Additionally, the course will explore cutting-edge advancements in the field.					
Contents This course delves into the foundations of automatic testing (fuzzing) in software security, providing Master's students with an opportunity to deepen their expertise in the field. The course offers a comprehensive overview of automatic testing, covering fundamental concepts such as White-box, Grey-box, and Black-box testing, standard and modern code exploration techniques, and advanced bug detection using logic-based oracles. Each lesson focuses on a specific aspect of the discipline, progressively building a complete understanding of the subject and equipping students with the skills to independently explore new concepts in this domain. The course will cover the following topics, with adjustments made as needed to suit the class's requirements: <ul style="list-style-type: none"> • Introduction to the fuzzing paradigm: Black-box, Grey-box, and White-box approaches • Types of coverage feedback • Heuristics for code exploration (e.g., mutators, meta-strategies, seed selection, grammars) • Basic concepts of symbolic execution and concolic testing • Bug detection and replication • Bug oracles (e.g., sanitizers, differential testing) • Managing fuzzing campaigns 					

- Domain-specific testing (e.g., kernel, virtual devices, libraries, IoT devices)

Laboratory sessions complement the in-class lectures by offering hands-on experience with the principles taught. These exercises are essential for developing the practical knowledge and problem-solving skills required for the final exam and for thoroughly understanding the material presented in the course.

Teaching methods

The course combines weekly lectures and laboratory sessions. During the lectures, students are encouraged to engage with the instructor and participate in solving simple exercises designed to enhance their learning experience.

Examination forms

Written exam (120 minutes). Bonus points can be earned in the exercises.

Requirements for the award of credits

The students have to pass the final written exam.

They can gain extra points for solving the exercises presented in the laboratory

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit / Informationstechnik

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme

Module title: Autonomous Vehicles and Artificial Intelligence

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Autonomous Vehicles and Artificial Intelligence (211044)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 25 participants
Teaching language English			Requirements for participation none		

Module coordinators and Lecturers

Module coordinators: Prof. Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger, Dr. Sven Peldszus

Module application

B.Sc. Informatik [until SS 23]
B.Sc. IT-Sicherheit [until SS 23]
B.Sc. Angewandte Informatik [until SS 23]
M.Sc. Computer Science
M.Sc. Angewandte Informatik
M.Sc. IT-Sicherheit/Informationstechnik
M.Sc. IT-Sicherheit/Netze und Systeme [until SS 23]

Precognitions

The Software Engineering lecture or a comparable course, Programming experiences e.g. as part of other courses.

Learning goals

- Understanding requirements on autonomous vehicles
- Understanding the architecture of autonomous vehicles
- Ability to build a self-driving car with ROS2
- Understanding and applying quality assurance for autonomous vehicles

Contents

Autonomous driving is the future of individual mobility and all major manufacturers are working on fully autonomous vehicles. While there are robust and good solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle's software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right. In this course, we will investigate the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain. The course will primarily focus on the following topics:

- Requirements on autonomous vehicles
- Architecture of autonomous vehicles
- Operation systems and frameworks for robotic systems
- Specification and Implementation of autonomous vehicles based on ROS2
- Artificial intelligence for autonomous vehicles
- Simulation of autonomous vehicles
- Localization and perception
- Mission planning

- Quality assurance for autonomous vehicles In the course's lecture, we provide the required theoretical background and practically apply the course's content in exercises by building a self-driving robot.

Teaching methods

Lecture with exercises

Examination forms

Final oral module exam (15-45 minutes)

Requirements for the award of credits

Passed final module exam and successful participation in the exercises

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Autonomous Vehicles and Artificial Intelligence Lab

Modul code	Credit points 5 CP	Workload 150h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter Semester	Duration 1 semester
Courses Autonomous Vehicles and Artificial Intelligence Lab (212035)			Contact time 4 SWS (60h)	Self-study 90h	Group size participants
Teaching language English			Requirements for participation -		
Module coordinators and Lecturers Module coordinators: Prof. Dr Thorsten Berger Lecturers: Prof. Dr. Thorsten Berger Dr. Sven Peldszus					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit/Informationstechnik					
Precognitions Recommended: <ul style="list-style-type: none">• Autonomous Vehicles and Artificial Intelligence (Lecture, 211044)• Programming experience in Python or C++ (e.g., as part of other courses)• Software Engineering (Lecture, 212000)					
Learning goals Knowledge - being able to explain relevant theoretical knowledge of artificial intelligence and autonomous vehicles Skill and abilities - define and validate requirements on autonomous vehicles - instantiate an architecture for autonomous vehicles - build a self-driving car with ROS2 - manage and integrate artificial intelligence in complex software-intensive systems - organize a team and its development process for a complex software-intensive system - perform quality assurance for autonomous vehicles - document the development process and create the artifacts that would be needed for a certification according to the ISO Road vehicles standards - communicate with group members and stakeholders professionally (speech and writing)					

Contents

Autonomous driving is the future of individual mobility, and all major manufacturers are working on fully autonomous vehicles. While there are robust and well-researched solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle's software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right.

In this course, we will practically explore the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain by developing a self-driving race car. To this end, the participants will work with ROS2-based model cars. This aims to provide the students with practical experience in developing an autonomous race car and organizing the development process.

Teaching methods

The main learning sequence in the course is a major practical project. The project is carried out in groups that iteratively develop an autonomous race car, applying and consolidating theoretical knowledge about autonomous driving and software development. To support learning, the Autonomous Vehicles and Artificial Intelligence Lab is based on seminar-style lectures that provide a platform for seeking feedback and more information. Based on the information gathered, students update and refine their solutions for an autonomous race car. Continuous reflection on practice and theory is facilitated by the ongoing production of a final project report in parallel with the development of the race car, in which students reflect on their own learning in the course, the way they and their team approach their development process, and their technical solutions. Students receive regular feedback and guidance to support their learning.

Examination forms

The final grade is determined based on the participation in the development of the self-driving race car, written project reports, the developed autonomous race car, and an oral presentation of the group results. Individual grades will be given by compiling the assessment of individual and group-based outcomes.

Requirements for the award of credits

Actively participate in the development of an autonomous race car, regularly attend seminar-style lectures, and successfully complete all deliverables.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Foundations of Programming Languages, Verification, and Security

Modul code	Credit points 5 CP	Workload 150 h	Term see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Foundations of Programming Languages, Verification, and Security (211062)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 20 participants
Teaching language English			Requirements for participation		

Module coordinators and Lecturers
Module coordinators: Apl.-Prof. Dr. Catalin Hritcu
Lecturers: Apl.-Prof. Dr. Catalin Hritcu

Module application
M.Sc. Computer Science

M.Sc IT-Sicherheit/Netze und Systeme

M.Sc. IT-Sicherheit/Informationstechnik

M.Sc. Mathematik

Precognitions
This advanced course for MSc and PhD students requires having attended the Proofs are Programs course or at least having a working knowledge of the contents of the Logical Foundations book, including familiarity with logic, mechanized proofs, and functional programming in the Coq proof assistant.

Learning goals
After successful completion of this course, students will be able to

- understand how to define in Coq the syntax of simple programming languages, in particular variants of a simple imperative language and of the simply-typed lambda calculus;
- define the big-step and small-step operational semantics of such simple languages;
- formally define type systems for such languages as inductive relations;
- work out the metatheory of such languages, by proving results such as type soundness;
- understand the semantic foundations of Hoare Logic and Relational Hoare Logic;
- use Hoare Logic for verifying the correctness of simple imperative programs, both formally in Coq and informally on paper;
- understand the semantic foundations of Secure Information Flow Control and Noninterference.
- use Relational Hoare Logic for proving program equivalence as well as noninterference of simple imperative programs;
- be familiar with static and dynamic enforcement mechanisms for Secure Information Flow Control as well as their formal noninterference guarantees (e.g. security type systems, secure multi-execution, etc.);
- understand how to formalize Cryptographic Constant Time, Speculative Constant Time, and Speculative Load Hardening (SLH) in Coq;
- apply various proof techniques both in Coq and in informal paper proofs (e.g. induction on rule derivations) or just in Coq (e.g. proof automation).

Contents
Complex proofs on paper are difficult to construct, check, and maintain. This holds not only for interesting proofs in mathematics, but also for complex formal proofs about interesting programs. For this reason, machine-checked proofs created with the help of interactive tools called proof assistants are gaining increased traction in academia and industry. Proof assistants have been used to prove the correctness and security of realistic compilers, operating systems, cryptographic libraries, or smart contracts, and also to construct machine-checked proofs for challenging theorems in mathematics.

This course will use the Coq proof assistant [2] to lay down the foundations of Programming Languages, Verification, and Security. The Coq proof assistant enables us to program formal proofs interactively and it machine-checks the correctness of the proofs along the way. We will use Coq to define the syntax and semantics of imperative and functional programming languages, to define type systems, and to prove theorems such as type soundness. We will also formalize Hoare Logic and Relational Hoare Logic in Coq and use them to prove the correctness and security of simple imperative programs. Finally, the course will introduce static and dynamic enforcement mechanisms for Secure Information Flow Control as well as their formal noninterference guarantees. Finally we will formalize Cryptographic Constant Time and Speculative Constant Time and prove in Coq that a software defense called Speculative Load Hardening (SLH) achieves Speculative Constant Time.

This hands-on course is based on the Programming Languages Foundations online textbook [1], which is itself formalized and machine-checked in the Coq proof assistant. The many exercises in each book chapter are to be solved weekly mostly in Coq, from easy exercises allowing the students to practice concepts from the lecture, building incrementally to slightly more interesting programs and proofs and also to various optional challenges.

Teaching methods

This course consists of lectures and weekly exercises, in which the students will solve problems using the Coq proof assistant for which they can get help from a tutor.

Examination forms

Written final exam (120 minutes) and exercise sheets.

Requirements for the award of credits

There will be a mandatory written final exam (120 minutes) that counts for 60% of the grade and weekly exercise sheets that have to be submitted on time and that count for 40% of the grade. We will also have an optional midterm exam that helps students practice for the final exam, but only counts for bonus points, up to 10% of the final grade. One can additionally get bonus points up to 5% of the final grade by solving all exercise sheets.

To pass the course and receive credit points one has to attend the final exam and the weighed sum of your scores including bonus points (which can add up to a maximum of 115%) has to be at least 50%.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO22]

5/99: M.Sc. IT-Sicherheit/Netze und Systeme [PO22]

Module title: High-Performance Computing on Clusters					
Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses High-Performance Computing on Clusters (127511)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size 50 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Andreas Vogel Lecturers: Prof. Dr. Andreas Vogel					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Bauingenieurwesen M.Sc. Subsurface Engineering					
Precognitions					
Learning goals After successfully completing the module, the students <ul style="list-style-type: none"> • are enabled to design and create programs for parallel computing clusters, • can critically evaluate distributed-memory systems and programming patterns, • can assess the mathematical properties of iterative solvers and their scalability. 					
Contents The lecture deals with the parallelization on cluster computers. Distributed-memory programming concepts (MPI) are introduced and best-practice implementation is presented based on applications from scientific computing including the finite element method and machine learning. Special attention is paid to scalable solvers for systems of equations on distributed-memory systems, focusing on iterative schemes such as simple splitting methods (Richardson, Jacobi, Gauß-Seidel, SOR), Krylov-methods (Gradient descent, CG, BiCGStab) and, in particular, the multigrid method. The mathematical foundations for iterative solvers are reviewed, suitable object-oriented interface structures are developed and an implementation of these solvers for modern parallel computer architectures is developed. Numerical experiments and self-developed software implementations are used to discuss and illustrate the theoretical results.					
Teaching methods Beamer, computer lab, numerical experiments					
Examination forms Written exam (120 minutes)					

Requirements for the award of credits

Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: High-Performance Computing on Multicore Processors

Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses High-Performance Computing on Multicore Processors (126509)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Andreas Vogel Lecturers: Prof. Dr. Andreas Vogel					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Bauingenieurwesen M.Sc. Subsurface Engineering					
Precognitions None					
Learning goals After successfully completing the module, the students <ul style="list-style-type: none">• are enabled to design and create programs for multicore processors,• can critically evaluate multi-threaded programs and shared-memory access patterns, can assess the benefits and challenges of multicore programming techniques.					
Contents The lecture addresses parallelization on multicore processors. Thread-based programming concepts and techniques, including pthreads, C++11 threads, OpenMP and SYCL, are introduced and best practices are highlighted using applications from scientific computing. An overview of the relevant hardware aspects including multicore architectures and memory hierarchies is provided. An in-depth introduction to multi-threaded programming on multicore systems with special emphasis on shared-memory parallelization is given and parallelization patterns, thread management and memory access strategies are discussed. In hands-on sessions, programming exercises are used to discuss and illustrate the presented content.					
Teaching methods Lecture with Exercise					
Examination forms Written exam (120 minutes).					

Requirements for the award of credits

Passed written module final exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Proofs are programs [M.Sc.] (no offer in WS 25/26)

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency winter semester	Duration 1 semester
Courses Proofs are Programms (211003)			Contact time 60 h (4 SWS)	Self-study 90 h	Group size 40 participants
Teaching language English			Requirements for participation None		

Module coordinators and Lecturers

Module coordinators: Dr. Catalin Hritcu
Lecturers: Dr. Catalin Hritcu
Dr. Clara Schneidewind

Module application

M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. IT-Sicherheit/ Netze und Systeme

M.Sc. Computer Science

Precognitions

The lecture is intended for a broad range of students, from motivated BSc students to MSc and PhD students. No specific prior knowledge in logic, programming, functional programming, or programming languages is assumed, though a degree of mathematical maturity is helpful.

Learning goals

After successful completion of this course, students will be able to

- develop purely functional programs using recursive functions on numbers, lists, maps, and various kinds of trees, including the abstract syntax trees of programs;
- use functional programming concepts such as type polymorphism and higher-order functions, which are increasingly becoming mainstream;
- formally state and prove theorems in the Coq proof assistant;
- apply different proof techniques in Coq (e.g. equational reasoning, contradiction, case analysis, induction on natural numbers, lists, and trees, induction on rule derivations, proof automation);
- define new inductive types and relations in Coq and prove statements about them;
- write simple proof terms and understand the connection between constructive logics and typed functional programming that is at the heart of Coq, in which propositions are types and proofs are programs;
- comprehend how the syntax and semantics of simple imperative programs can be formally defined in Coq and how to prove theorems about such programs and languages;
- understand how the absence of information leaks can be formalized as a security property called noninterference and enforced using secure-multi execution or simple type systems.

Contents

Complex proofs on paper are difficult to construct, check, and maintain. This holds not only for interesting proofs in mathematics, but also for complex formal proofs about interesting programs. For this reason, machine-checked proofs created with the help of interactive tools called proof assistants are gaining increased traction in academia and industry. Proof assistants have been used to prove the correctness and security of realistic compilers, operating systems, cryptographic libraries, or smart contracts, and also to construct machine-checked proofs for challenging theorems in mathematics.

This course introduces the Coq proof assistant [3] and explains how to use it to prove properties about functional programs and inductive relations, how to formally define a simple imperative programming language, and how to securely enforce information-flow control for functional and imperative programs. The Coq proof assistant enables us to program formal proofs interactively and it machine-checks the correctness of the proofs along the way. The design of the Coq proof assistant itself exploits a beautiful connection between programs in typed functional programming languages and proofs in constructive logics, which is known as the Curry-Howard Correspondence [4]. This deep connection between programs and proofs should make this course interesting to not only to computer scientists, but also to mathematicians and other scientists. The goal is to demystify proofs as just programs in an elegant programming language, for which the course provides a gentle introduction. The course also shows that proofs are not only a way to convince a human reader, but they can actually be fully formalized in a proof assistant like Coq and automatically checked by a computer.

This hands-on course is based on the Logical Foundations [1] and Security Foundations [2] online textbooks, which are themselves formalized and machine-checked in the Coq proof assistant. The many exercises in each book chapter are to be solved weekly mostly in Coq, from easy exercises allowing the students to practice concepts from the lecture, building incrementally to slightly more interesting programs and proofs and also to various optional challenges. Finally, this course serves as the base for a more advanced course on “Foundations of Programming Languages, Verification, and Security”.

Teaching methods

This course consists of lectures and weekly exercises, in which the students will solve problems using the Coq proof assistant for which they can get help from a tutor.

Examination forms

Written final exam (120 minutes).

Requirements for the award of credits

Passing the final written exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/97: M.Sc. Computer Science

Module title: Software Languages					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Software Languages (212034)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 25 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Thorsten Berger Lecturers: Prof. Dr. Thorsten Berger					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik					
Precognitions <ul style="list-style-type: none"> • Object-Oriented Programming • Theoretical Computer Science (especially formal languages, grammars, set notation) • Software Engineering • Functional Programming 					
Learning goals					
Knowledge and understanding <ul style="list-style-type: none"> • explain core concepts for engineering software languages, e.g., meta-modeling, abstract syntax, concrete syntax, static semantics, dynamic semantics, model/program transformation • explain linguistic architectures for software languages (e.g., meta-modeling hierarchy) • explain how domain-specific languages can be realized within a contemporary language workbench 					
Competence and skills <ul style="list-style-type: none"> • engineer domain-specific languages for a specific engineering problem • define abstract syntax by modeling domains and creating meta-models • define static semantics with syntactic constraints or type systems • define concrete syntax with Xtext, Sirius, or a comparable technology • define dynamic semantics with model transformations (compilation) or interpretation • reason about configuration spaces expressed in variability models 					
Judgement and approach <ul style="list-style-type: none"> • identify use-cases and the potential of using DSLs for a given domain/problem <p>select and justify appropriate language technology for a given domain/problem</p>					
Contents The course teaches the theory and the pragmatics of using and developing high-level software languages (Domain-Specific Languages, or DSLs) for the effective production of quality software. This includes methods, design patterns, guidelines, and testing practices for defining concrete syntax, abstract syntax, and semantics of languages. The course attempts to be close to technology, while covering multiple paradigms and solutions.					

We cover:

- Domain analysis of a problem domain and designing meta-models
- Engineering external and internal DSLs (we focus mostly on external DSLs)
- Designing the concrete syntax of languages
- Implementation of language semantics using declarative and imperative transformations, code generators and interpreters, in various scenarios, e.g., from text to models, from models to text, involving XML, database, etc.
- Implement declarative constraints and type rules for DSLs
- Testing and quality assurance of language implementations
- Model-driven engineering, especially engineering of highly configurable systems or software product lines (we focus on feature modeling syntax and semantics)

Teaching methods

The teaching of this course consists of different forms: lectures, interactive quizzes, group work, group supervision, and practical assignments.

Examination forms

The examination for the Software Languages module consists of several parts. Submissions are to be made during the semester and must be passed. The submissions themselves are not graded. A written or oral final examination takes place at the end of the semester. In order to be able to take the final examination, students must pass the semester-long assignments.

Requirements for the award of credits

Passed semester-accompanying submissions and passed final module examination (written (120 minutes) or oral (15-45 minutes)).

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

Module title: AI Ethics And Society

Modul code	Credit points 5 CP	Workload 150h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses AI Ethics And Society (212044)			Contact time	Self-study 105h	Group size 20 participants
Teaching language English			Requirements for participation The participant should have a broad interest in social sciences and humanities.		

Module coordinators and Lecturers

Module coordinators: Dr. Yixin Zou, <https://yixinzou.github.io/>
Gabriel Lima, MSc, <https://thecamilo.github.io/>
Lecturers: Dr. Yixin Zou, <https://yixinzou.github.io/>
Gabriel Lima, MSc, <https://thecamilo.github.io/>

Module application**Precognitions**

Recommended but not mandatory:

- General understanding of machine learning and data science (e.g., from “Machine Learning: Supervised Methods” or “Introduction to Data Science”)
- Responsible AI (211064)
- Privacy Engineering, Data Governance and Usability (212037)

Learning goals

After completing the course, students will be able to:

- Critically analyze the ethical, social, and political implications of artificial intelligence (AI) through lenses other than computer science;
- Apply theories and concepts from the social sciences and humanities to examine sociotechnical systems;
- Reflect on and evaluate proposals on how to develop and deploy AI in high-risk environments.

Contents

Artificial intelligence (AI) systems are now deployed in several high-risk domains, directly impacting people’s lives. Numerous reports and extensive research have identified how AI systems can harm individuals and society not only after deployment, but also during their development. In this course, students will learn how AI-caused harms emerge and examine existing and proposed interventions to mitigate such harms. Rather than focusing on technical solutions to AI ethics, this course will center on perspectives from the social sciences and humanities. The course will cover the following topics:

- Discrimination, bias, and injustice: e.g., how AI disproportionately harms minoritized groups and how to develop AI equitably
- Responsibility: e.g., who is and should be responsible when AI causes harm
- Explainable AI (XAI): e.g., the promises and limitations of AI systems explaining their reasoning and decisions
- The politics of AI datasets: e.g., who has a say in how datasets are developed and used to train AI systems
- Environments costs of AI: e.g., how AI can harm the environment during its development and after deployment
- Surveillance vs. inclusion: e.g., how AI systems are used to surveil marginalized groups under the promise of inclusion
- Auditing: e.g., how AI systems can be audited by those being harmed by them
- Regulation: e.g., how countries around the world propose to regulate AI

- Vision: e.g., different perspectives of how AI can be develop in the future

Teaching methods

lecture with exercise

Examination forms

Combined exam: written exam 90 min (50%) + assignments/projects (50%)

Requirements for the award of credits

Passed combined exam: written exam 90 min (50%) + assignments/projects (50%)

Significance of the grade for the final grade (for a total of 120 ECTS)

Module title: Autonomous Robotics: Action, Perception, Cognition					
Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Autonomous Robotics: Action, Perception, Cognition (211048)			Contact time 3 SWS (45 h)	Self-study 135 h	Group size 30 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Gregor Schöner Lecturers: Prof. Dr. Gregor Schöner					
Module application M.Sc. Angewandte Informatik M.Sc. Computer Science					
Precognitions The emphasis of the course is on learning concepts, practicing interdisciplinary scholarship including reading and writing at a scientific and technical level. Mathematical concepts are used throughout, so understanding these concepts is important. Mathematical skills are not critical to mastering the material, but helpful. The mathematics is mostly from the qualitative theory of dynamical systems, attractors and their instabilities. Short tutorials on some of these concepts will be provided.					
Learning goals After the successful completion of this course the students: <ul style="list-style-type: none"> • are familiar with the concepts in dynamical systems theory and can use them practically, • know the mathematical models of movement generation • have practice in reading and writing academic research papers. 					
Contents Autonomous robotics is an interdisciplinary research field in which embodied systems equipped with their own sensors and with actuators generate behavior that is not completely preprogrammed. Autonomous robotics thus entails perception, movement generation, as well as core elements of cognition such as making decisions, planning, and integrating multiple constraints. This course touches on various approaches to this interdisciplinary problem. In the first half of the course, the main emphasis will be on dynamical systems methods for generating movement in vehicles. The main focus of the course is, however, on solutions to autonomous movement generation that are inspired by analogies with how nervous systems generate movement. In fact, the second half of the course will review core problems in human movement science, including the degree of freedom problem, coordination, motor control, and the reflex control of muscles.					
Teaching methods Lecture with Exercise					
Examination forms Oral Examination (30 Minutes); Exercises for bonus points					
Requirements for the award of credits Passed oral examination					

Significance of the grade for the final grade (for a total of 120 ECTS)

6/105: M.Sc. Angewandte Informatik

6/97: M.Sc. Computer Science

Module title: Autonomous Vehicles and Artificial Intelligence

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Autonomous Vehicles and Artificial Intelligence (211044)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 25 participants
Teaching language English			Requirements for participation none		

Module coordinators and Lecturers

Module coordinators: Prof. Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger, Dr. Sven Peldszus

Module application

B.Sc. Informatik [until SS 23]
B.Sc. IT-Sicherheit [until SS 23]
B.Sc. Angewandte Informatik [until SS 23]
M.Sc. Computer Science
M.Sc. Angewandte Informatik
M.Sc. IT-Sicherheit/Informationstechnik
M.Sc. IT-Sicherheit/Netze und Systeme [until SS 23]

Precognitions

The Software Engineering lecture or a comparable course, Programming experiences e.g. as part of other courses.

Learning goals

- Understanding requirements on autonomous vehicles
- Understanding the architecture of autonomous vehicles
- Ability to build a self-driving car with ROS2
- Understanding and applying quality assurance for autonomous vehicles

Contents

Autonomous driving is the future of individual mobility and all major manufacturers are working on fully autonomous vehicles. While there are robust and good solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle's software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right. In this course, we will investigate the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain. The course will primarily focus on the following topics:

- Requirements on autonomous vehicles
- Architecture of autonomous vehicles
- Operation systems and frameworks for robotic systems
- Specification and Implementation of autonomous vehicles based on ROS2
- Artificial intelligence for autonomous vehicles
- Simulation of autonomous vehicles
- Localization and perception
- Mission planning

- Quality assurance for autonomous vehicles In the course's lecture, we provide the required theoretical background and practically apply the course's content in exercises by building a self-driving robot.

Teaching methods

Lecture with exercises

Examination forms

Final oral module exam (15-45 minutes)

Requirements for the award of credits

Passed final module exam and successful participation in the exercises

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Autonomous Vehicles and Artificial Intelligence Lab

Modul code	Credit points 5 CP	Workload 150h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter Semester	Duration 1 semester
Courses Autonomous Vehicles and Artificial Intelligence Lab (212035)			Contact time 4 SWS (60h)	Self-study 90h	Group size participants
Teaching language English			Requirements for participation -		
Module coordinators and Lecturers Module coordinators: Prof. Dr Thorsten Berger Lecturers: Prof. Dr. Thorsten Berger Dr. Sven Peldszus					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit/Informationstechnik					
Precognitions Recommended: <ul style="list-style-type: none">• Autonomous Vehicles and Artificial Intelligence (Lecture, 211044)• Programming experience in Python or C++ (e.g., as part of other courses)• Software Engineering (Lecture, 212000)					
Learning goals Knowledge - being able to explain relevant theoretical knowledge of artificial intelligence and autonomous vehicles Skill and abilities - define and validate requirements on autonomous vehicles - instantiate an architecture for autonomous vehicles - build a self-driving car with ROS2 - manage and integrate artificial intelligence in complex software-intensive systems - organize a team and its development process for a complex software-intensive system - perform quality assurance for autonomous vehicles - document the development process and create the artifacts that would be needed for a certification according to the ISO Road vehicles standards - communicate with group members and stakeholders professionally (speech and writing)					

Contents

Autonomous driving is the future of individual mobility, and all major manufacturers are working on fully autonomous vehicles. While there are robust and well-researched solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle's software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right.

In this course, we will practically explore the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain by developing a self-driving race car. To this end, the participants will work with ROS2-based model cars. This aims to provide the students with practical experience in developing an autonomous race car and organizing the development process.

Teaching methods

The main learning sequence in the course is a major practical project. The project is carried out in groups that iteratively develop an autonomous race car, applying and consolidating theoretical knowledge about autonomous driving and software development. To support learning, the Autonomous Vehicles and Artificial Intelligence Lab is based on seminar-style lectures that provide a platform for seeking feedback and more information. Based on the information gathered, students update and refine their solutions for an autonomous race car. Continuous reflection on practice and theory is facilitated by the ongoing production of a final project report in parallel with the development of the race car, in which students reflect on their own learning in the course, the way they and their team approach their development process, and their technical solutions. Students receive regular feedback and guidance to support their learning.

Examination forms

The final grade is determined based on the participation in the development of the self-driving race car, written project reports, the developed autonomous race car, and an oral presentation of the group results. Individual grades will be given by compiling the assessment of individual and group-based outcomes.

Requirements for the award of credits

Actively participate in the development of an autonomous race car, regularly attend seminar-style lectures, and successfully complete all deliverables.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Computational Neuroscience: Single-Neuron Models

Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Computational Neuroscience: Single-Neuron Models (211039)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Robert Schmidt Lecturers: Prof. Dr. Robert Schmidt					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik					
Precognitions Programming in Python, mathematical knowledge (linear algebra and calculus) and an interest in neurobiology.					
Learning goals <ul style="list-style-type: none">• apply techniques from computational neuroscience to simulate neural activity• become familiar with different types of single neuron models, their mathematical description, and their different levels of biological abstraction• acquire skills in modelling neurons, synapses and circuits and connect these models to biology and computation• understanding of the biological basis for computation in neurons					
Contents <p>This module starts with a primer on neuroscience and the role of computational neuroscience. The next part of the module covers biologically-grounded models of single neurons, including leaky-integrate-and-fire and conductance-based neurons, but also more abstract models of neural activity and spike trains. You will learn how these different computational models describe and simplify the underlying biological processes to a different degree. We will examine in detail how these different neuron models can be used in numerical simulations to address research questions on computation in single neurons and circuits. In the exercises accompanying the lectures you will gain hands-on experience in implementing the different neuron models in Python, running numerical simulations, and performing calculations related to analytical solutions of the model equations and biophysics. The focus is on single neuron models, but we will also make use of available software (e.g. NEST Desktop) to examine how single neuron models can be integrated into simulations of neural networks. While the emphasis throughout the module is on methodological issues, how models can be built, tested and validated at each level, we will also draw connections to specific brain regions to motivate and illustrate the models.</p>					
Teaching methods Lecture with exercise					
Examination forms Written exam (120 minutes)					
Requirements for the award of credits Passed the written exam.					

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Computational Neuroscience: Vision and Memory

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Computational Neuroscience: Vision and Memory (211049)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 20 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Laurenz Wiskott Lecturers: Prof. Dr. Laurenz Wiskott					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Cognitive Science					
Precognitions The mathematical level of the course is mixed but generally high. The tutorial is almost entirely mathematical. Mathematics required include calculus (functions, derivatives, integrals, differential equations, ...), linear algebra (vectors, matrices, inner product, orthogonal vectors, basis systems, ...), and a bit of probability theory (probabilities, probability densities, Bayes' theorem, ...).					
Learning goals After the successful completion of this course the students: <ul style="list-style-type: none">• know basic neurobiological facts about the visual system and the hippocampus,• know a number of related models and methods in computational neuroscience,• understand the mathematics of these methods,• can communicate about all this in English.					
Contents This lecture covers basic neurobiology and models of selforganization in neural systems, in particular addressing Learning and self-organization <ul style="list-style-type: none">• Hebbian Learning• Neural learning dynamics and constrained optimization• Dynamic field theory Vision <ul style="list-style-type: none">• Receptive fields• Neural maps• Hippocampus• Navigation• Episodic memory• Hopfield Network					

Teaching methods

This course is given with the flipped/inverted classroom concept. First, the students work through online material by themselves. In the lecture time slot we then discuss the material, find connections to other topics, ask questions and try to answer them. In the tutorial time slot the newly acquired knowledge is applied to analytical exercises and thereby deepened. I encourage all students to work in teams during self-study time as well as in the tutorial.

Examination forms

Oral final module exam (15-45 minutes)

Requirements for the award of credits

Passed final oral module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

Module title: Deep Learning					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Deep Learning (212018)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size 50 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Asja Fischer Lecturers: Prof. Dr. Asja Fischer					
Module application M.Sc. IT-Sicherheit/ Informationstechnik M.Sc. IT-Sicherheit/ Netze und Systeme [Until WS 22/23] M.Sc. Angewandte Informatik M.Sc. Computer Science					
Precognitions Basic knowledge of linear algebra and probability theory is an advantage.					
Learning goals The lecture aims to provide an insight into this field. At the beginning, the basic terms and concepts of machine learning are introduced. In the further course, different neural networks, gradient-based optimization methods and generative models will be discussed.					
Contents Deep Learning is a subfield of machine learning that has led to breakthroughs in numerous application areas (such as object and speech recognition and machine translation) in recent years. Deep Learning methods find application in the field of IT Security, among others.					
Teaching methods Lecture with exercise					
Examination forms Written exam (120 minutes)					
Requirements for the award of credits Passed final written exam.					
Significance of the grade for the final grade (for a total of 120 ECTS) 5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22] 5/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20] 5/105: M.Sc. Angewandte Informatik					

Module title: Engineering for Large Language Models

Modul code	Credit points 6 CP	Workload 180h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer Semester	Duration 1 semester
Courses Engineering for Large Language Models (211065)			Contact time 60h (4 SWS)	Self-study 120h	Group size 30 participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Bilal Zafar Lecturers: Prof. Dr. Bilal Zafar					
Module application					
Precognitions This advanced course assumes deep familiarity with fundamental concepts of Machine Learning. The participants should have taken one of the following courses at RUB (or another similar course): (i) Machine Learning, (ii) Deep Learning, (iii) Natural Language Processing with Deep Learning, (iv) Machine Learning: Supervised Methods. Participants should also have prior experience with PyTorch.					
Learning goals Ability to finetune and perform inference with SoTA LLMs even with limited resources. Working with LLMs in a local (laptop or PC), compute cloud, and API setting.					
Contents Modern LLMs like ChatGPT and LLaMA show impressive performance in a range of real-world applications. Much of this performance is enabled by scaling up the model sizes. State-of-the-art models consist of hundreds of billions of parameters. This massive scale makes training, finetuning and inference with LLMs much more challenging as compared to traditional deep learning models like CNNs and LSTMs. In this course, you will learn technical advances that power modern LLMs. You will learn how to operate massive LLMs that do not fit on a single GPU. You will also learn about techniques like caching, quantization, pruning and parameter efficient finetuning that enable efficient fine-tuning and inference. You will also learn how to interact with API-based models like GPT-4 and Claude.					
Teaching methods The course will be split into two blocks. The first block will consist of lectures where we will introduce you to the content in a classroom setting. The second block will entirely consist of lab sessions where you will work on practical project(s).					
Examination forms Graded project (50% of the grade), final e-exam (50% of the grade) over 120 minutes					
Requirements for the award of credits Passing grade in the project and in the exam.					

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Machine Learning: Supervised Methods (no offer in SS 25)

Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Machine Learning: Supervised Methods (211024)			Contact time 4 SWS (60 h)	Self-study 120 h	Group size 80 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Tobias Glasmachers Lecturers: Prof. Dr. Tobias Glasmachers					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. IT-Sicherheit/ Informationstechnik					
Precognitions recommended: Lecture "Mathematics for Modeling and Data Analysis".					
Learning goals Internationalisation: The event will be held in English. Digitisation: Content is conveyed through videos and reading material. Exercises with programming components are provided in the form of Jupyter notebooks. After successful completion of the module <ul style="list-style-type: none">• students understand the basics of statistical learning theory• know the most important algorithms of supervised statistical learning and can apply them to learning problems,• know the strengths and limitations of different learning methods,• can use standard software for machine learning to solve new problems.					
Contents Basics of statistical learning theory, cross-section of the most important machine learning algorithms, concrete problem solving with standard software.					
Teaching methods Lecture with exercise in flipped classroom format					
Examination forms Written exam (90 minutes).					
Requirements for the award of credits Passing the course is a two-stage process. The first stage is an active contribution during the semester, the details of which are communicated in one of the first sessions. The second stage is a written exam of 90 minutes. The active contribution during the semester					

is not graded, but it is a requirement for taking the exam. The grade is based solely on the final exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/105: M.Sc. Angewandte Informatik

6/97: M.Sc. Computer Science

6/91: M.Sc IT-Sicherheit/ Informationstechnik [PO 22]

6/84: M.Sc IT-Sicherheit/ Informationstechnik [PO 20]

Module title: Machine Learning: Unsupervised Methods

Modul code	Credit points 10 CP	Workload 300 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Machine Learning: Unsupervised Methods (212501)			Contact time 8 SWS (120h)	Self-study 180 h	Group size 40 participants
Teaching language English			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Laurenz Wiskott Lecturers: Prof. Dr. Laurenz Wiskott					
Module application M.Sc. Computer Science M.Sc. Angewandte Informatik M.Sc. Elektro-und Informationstechnik					
Precognitions Linear algebra (vectors, matrices, eigenvectors, eigenvalues, ...), Calculus (functions, derivatives, ...), Probability theory (joint/marginal/conditional probabilities in multiple variables, Bayesian theorem, ...), Programming (fluent in at least one programming language, ideally python).					
Learning goals After the successful completion of this course the students <ul style="list-style-type: none">• know fundamentals of machine learning,• know a number of important unsupervised learning methods,• can discuss and decide which of the methods are appropriate for a given data set,• understand the mathematics of these methods,• know how to implement and apply these methods in python,• have gained experience in organizing and working in a team,• know problem solving strategies like brain storming,• can communicate about all this in English.					
Contents This course first introduces into the field of machine learning and covers some basic concepts, such as learning paradigms, training, testing and generalization, and over/underfitting. It then covers a variety of unsupervised methods from classical machine learning such as principal component analysis, independent component analysis, vector quantization, clustering, Bayesian theory, and graphical models. These are shallow methods, this course does not cover deep learning methods by default, but students may use deep learning in the problems.					

On the practical side the students get simple problems to solve in group work and then present to the whole course. In this context, the course will also cover some soft skills, such as concept mapping, team reflection, and brainstorming.

Teaching methods

Lecture + self-studied exercises + inverted classroom style discussion of lecture and exercises + practical problem solving in groups + peer reviewing + presentation of results.

Examination forms

Pass/fail on group work plus graded written exam of 90 minutes.

Requirements for the award of credits

Passed group work and passed final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

10/105: M.Sc. Angewandte Informatik

10/97: M.Sc. Computer Science

Module title: Natural language processing with deep learning [M.Sc]

Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter Semester	Duration 1 semester
Courses Natural language processing with deep learning (212038)			Contact time 4 SWS (60 h)	Self-study 90 h	Group size participants
Teaching language English			Requirements for participation Bring your own laptop with Python and a suitable IDE for the exercises		

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Ivan Habernal
Lecturers: Prof. Dr. Ivan Habernal

Module application
M.Sc. Angewandte Informatik

M.Sc. Computer Science

Precognitions
Practical: Working knowledge of Python (project structures, OOP, unit tests), knowledge of Numpy is a plus
Theoretical: Basics of calculus (derivatives) and linear algebra (vectors), although we will repeat that. Some working knowledge of linguistics (e.g., syntax) on a high-school level is also good.

Learning goals
The students

- understand the backbone of modern NLP, such as embeddings and transformer-based models
- are able to critically assess the capabilities and weaknesses of various deep learning models in NLP
- understand different tasks in NLP, their evaluation and modeling assumptions
- can implement different models and approaches in Python and gain practical experience
- understand advanced contemporary paradigms in large language models

Contents
Typical natural language processing (NLP) tasks and data sets and their evaluation. Why is NLP difficult?
Refresher of mathematical basics, calculus, gradient-based optimization, Backpropagation for arbitrary functions.
Log-linear models and text classification. Deep neural networks. Language models and word embeddings.
Learning static word embeddings. Recurrent neural networks. Encoder-decoder, text generation, attention and autoregressive models. Transformers, Self-attention and BERT. Pure decoder models and GPT. LLMs: prompting and in-context learning.

Teaching methods
We will have lectures and exercises. The lectures will be quite interactive as we try to engage the students in questions and critical thinking as we go along. Slides will be uploaded for each lecture in advance so that students can use them for their notes. Each lecture will include links to relevant research papers for those who want to delve deeper into the topic. The lectures are mostly theoretical, meaning we cover the main concepts, ideas and mathematical descriptions, but not how to program it in a specific framework. In the practical courses we will do exactly the opposite. Students will experiment with current mainstream frameworks for deep learning in NLP, such as Pytorch or Huggingface. We will cover a wide range of complexities, from programming a simple neural network from scratch to using a pre-trained language model.

Examination forms

Written exam of 90 minutes

Requirements for the award of credits

Passed final module exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/105: M.Sc. Angewandte Informatik

5/97: M.Sc. Computer Science

Module title: Numerical Optimization					
Modul code	Credit points 6 CP	Workload 180h	Term siehe Prüfungsordnung / see examination regulations	Frequency Winter semester	Duration 1 semester
Courses Numerical Optimization (212043)			Contact time 60h (4 SWS)	Self-study 120h	Group size participants
Teaching language English			Requirements for participation This course does not have formal requirements. The target audience includes Master students of all technical subjects, like computer science, mathematics, physics, and engineering.		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Tobias Glasmachers Lecturers: Prof. Dr. Tobias Glasmachers					
Module application M.Sc. Angewandte Informatik M.Sc. Computer Science					
Precognitions This course requires a solid grasp on various mathematical concepts for defining and analyzing optimization algorithms with mathematical tools, including a few proofs. Participants need a solid understanding of linear algebra (matrices, eigen decomposition, positive definiteness), analysis (sequences, convergence, convexity, gradient and Hessian matrix), and probability theory (multi-variate normal distribution). Prior knowledge of numerics is a plus, but not required. The exercise sessions as well as the exam require basic Python programming.					
Learning goals The participants can explain important classes of optimization algorithms, like conjugate gradients (CG), BFGS, CMA-ES, and the simplex algorithm for linear programming. They understand properties of different types of search directions (sampling methods, gradient, conjugate gradient, Newton step and quasi Newton methods), as well step size control mechanisms (line search and trust region methods). They can relate these methods to convergence speed classes like linear and super-linear convergence. They understand Lagrangians and they can derive dual optimization problems. They can model real-world problems as mathematical optimization problems. They can pick suitable algorithms, apply them to actual optimization problems, and solve them efficiently.					
Contents This course offers a contemporary introduction to numerical optimization. Optimization algorithm find applications in many areas of engineering, economics, machine learning, and many more. This course covers the most prominent design principles and algorithm classes: <ul style="list-style-type: none"> · gradient and Newton search directions · line search and trust region methods · conjugate gradients · quasi Newton algorithms · constraint handling, duality · linear programming, including the mixed integer case 					

· direct search (gradient-free) methods

Methods are presented and analyzed in the lecture, and implemented and tested in the exercise sessions.

During the practical sessions, participants work on a mix of conceptual and practical exercises. Many practical exercises involve programming in Python.

Teaching methods

The contact time is split roughly 50/50 into lecture and exercise session. The lecture includes flipped classroom elements. For that purpose, homework assignments include (in some cases extensive) reading assignments.

Examination forms

Written electronic exam (90 minutes)

Requirements for the award of credits

Passed the final exam

Significance of the grade for the final grade (for a total of 120 ECTS)

6/105: M.Sc. Angewandte Informatik

6/97: M.Sc. Computer Science

Module title: Privacy-Preserving Natural Language Processing

Modul code	Credit points 6 CP	Workload 180 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Privacy-Preserving Natural Language Processing (211068)			Contact time 60h (4 SWS)	Self-study 120 h	Group size 50 participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Ivan Habernal Lecturers: Prof. Dr. Ivan Habernal					
Module application					
Precognitions Strongly recommended is a proficiency of contemporary natural language processing (NLP), including transformer models, model training with SGD, etc., see for example “Natural Language Processing with Deep Learning” course lectured by Ivan Habernal in WiSe. A very solid working knowledge of probability theory is also necessary. We won't need measure theory, but will be working with concentration inequalities (Markov ineq., Chernoff bounds, etc.) and similar formal methods that are used in differential privacy.					
Learning goals Students will be able to <ul style="list-style-type: none">• Understand the difficulties of privacy protection when using non-formal ad-hoc methods• Appreciate the formal treatment of privacy through algorithmic randomized approaches such as differential privacy• Work with basic privacy mechanisms formally• Train DP-protected models with standard frameworks					
Contents We will cover the following: <ul style="list-style-type: none">• Why is privacy important in contemporary NLP• Formalization of privacy• Text anonymization, redaction, and PII removal• Probabilistic treatment of privacy with differential privacy• Basic privacy mechanisms (Laplace, Gaussian)• Training models with DP-SGD• Attacks on models (membership inference attacks and similar)• Critical assessment of DP methods in NLP					
Teaching methods I'm planning to do the lectures as interactive as possible, to keep everyone on board - so please expect you will be asked questions and you should remember things from previous lectures too (same approach as we did in NLP with DL in WiSe). We will cover theoretical content and some important formal proofs during the lectures, and take some proofs to the exercise sessions to work them out. In the exercises we will have a mix of programming and discussing the nitty-gritty details of privacy, so please bring your laptop with you with your favorite IDE for Python installed and ready to go. We will also do one or two homeworks (to be worked on on your own, not in groups) dealing with certain aspects of privacy in NLP, we will discuss that during the exercises in detail.					
Examination forms Written exam (90 minutes)					

Requirements for the award of credits

Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Responsible AI					
Modul code	Credit points 6 CP	Workload 180h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer Semester	Duration 1 semester
Courses Responsible AI (211064)			Contact time 60h (4 SWS)	Self-study 120h	Group size 50 participants
Teaching language English			Requirements for participation None		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Bilal Zafar Lecturers: Prof. Dr. Bilal Zafar					
Module application					
Precognitions The participants should have taken one of the following courses at RUB (or another similar course): (i) Machine Learning, (ii) Deep Learning, (iii) Machine Learning: Supervised Methods, (iv) Introduction to Data Science. Participants should be very comfortable with the Python programming language.					
Learning goals <ul style="list-style-type: none"> • Develop and understanding of why issues like biases, toxicity and lack of robustness arise in ML models and learn techniques to tackle these issues. • Learn about the limits of technical interventions in controlling bias. • Learn how to explain the decisions of complex ML models and pitfalls of explainability. 					
Contents ML models are increasingly being used in domains where their decisions directly impact people. For instance, ML models are used to diagnose medical conditions, match users with online advertisements, assess creditworthiness, and in some places, even make decisions about pretrial bail. Since these decisions affect real human beings, ensuring that the models decisions are trustworthy is of utmost importance. However, plenty of investigations have shown that these models suffer from various issues. For instance, the models can: unfairly disadvantage people from certain demographics, produce toxic outputs, change their outputs significantly as a result of unimportant changes in the input and violate privacy of their users. Given the complexity of these models, root-causing and alleviating these issues can be quite challenging. In this course, you will learn where these trustworthiness issues originate from, how to detect them, and what actions to take to alleviate them. You will also learn that these issues cannot be solved by technical interventions alone, and that one often needs an interdisciplinary approach to tackle them.					
Teaching methods Lectures followed by assignments. Before each lecture, you will be assigned a required reading which will often be one or more research papers.					
Examination forms Graded project (50% of the grade), final e-exam (50% of the grade) over 120 minutes					

Requirements for the award of credits

Passing grade in the project and in the exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Statistical learning and data mining					
Modul code	Credit points 5 CP	Workload 150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Statistisches Lernen und Data Mining (150331)			Contact time 60 h	Self-study 90	Group size 20 participants
Teaching language English			Requirements for participation none		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Johannes Lederer Lecturers: Prof. Dr. Johannes Lederer					
Module application B.Sc. Informatik M.Sc. Computer Science					
Precognitions					
Learning goals After successful completion of the module <ul style="list-style-type: none"> • students know standard methods of data analysis • they understand when which methods are appropriate • they are able to apply the methods • they are able to interpret the results 					
Contents This course introduces the basic methods of data analysis. Different types of data are considered, especially regression data and classification data. The underlying statistical models are always discussed as well. Likewise, possible applications are presented both in class and in computer exercises. The aim is to teach the whole process of simple data analysis: Data preparation, statistical modelling, selection of a method, implementation of the method, visualisation of the results and interpretation.					
Teaching methods Lecture with media support, in particular data analysis with the computer, tutorials as seminar-based teaching, additional self-study with supplementary materials and tasks provided.					
Examination forms Written module final exam (90 minutes).					
Requirements for the award of credits Passed written final module exam					
Significance of the grade for the final grade (for a total of 120 ECTS) 5/158: B.Sc. Informatik [PO 22] 5/165: B.Sc. Informatik [PO 20] 5/97: M.Sc. Computer Science					

Module title: Theory of machine learning					
Modul code	Credit points 9 CP	Workload 270 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Summer semester	Duration 1 semester
Courses Theorie des maschinellen Lernens (211052)			Contact time 6 SWS (90 h)	Self-study 180 h	Group size 20 participants
Teaching language German			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Prof. Dr. Asja Fischer Lecturers: Prof. Dr. Asja Fischer					
Module application M.Sc. Angewandte Informatik M.Sc. Computer Science					
Precognitions					
Learning goals Students are familiarised with mathematical models for machine learning. They acquire the ability to evaluate and compare learning algorithms according to the degree to which they achieve (precisely described) success criteria. They acquire techniques both for designing efficient learning algorithms and for proving the inherent hardness of a problem. After the successful completion of the module <ul style="list-style-type: none"> • students know the most important learning machines (such as Support Vector Machines and related models), • students understand the difference between empirical and real error rate and know techniques to deal with the problem of overfitting the data (with a model that is too complex), • can differentiate between uniform and non-uniform learnability of a Hypothesis class and know the appropriate theories and learning rules. 					
Contents The subject of the lecture is the statistics-based theory of machine learning. In particular, the method of structured risk minimisation is taught as well as the statistical theorems on which it is based. Techniques for designing efficient learning algorithms are discussed as well as information- or computational-theoretic barriers that make certain learning problems appear to be inefficiently solvable.					
Teaching methods Lecture with exercise					
Examination forms Final oral module exam (15-45 minutes)					
Requirements for the award of credits Passed final oral module exam.					
Significance of the grade for the final grade (for a total of 120 ECTS) 9/105: M.Sc. Angewandte Informatik					

Module title: Practical Labs

Modul code	Credit points 3 CP	Workload 90-150 h	Term siehe Prüfungsordnung / see examination regulations	Frequency Every winter and summer semester	Duration 1 semester
<p>Courses The current practical courses are highlighted:</p> <p>Specialisation Design, Implementation and Analysis of Computer Systems:</p> <ul style="list-style-type: none"> • Open-Source Chip Design (212429) (4 CP) • System software engineering lab course (212405) (4 CP) <p>Specialisation Computer Security:</p> <ul style="list-style-type: none"> • Research in Internet Security (211431) (4 CP) • Research in Software Security (211433) (4 CP) • Introductory project in microarchitectural security (211427) (4 CP) • Advanced Research in Microarchitectural Security (212424) (4 CP) • Practical Course Traffic Analysis Attacks (211436) (4 CP) • Practical Course on Machine learning Security (212427) (4 CP) • Practical Course on Mobile Network Security (211404) (4 CP) • Software Testing via Fuzzing (211418) (4 CP) • Embedded Firmware Fuzzing (211419) (4 CP) <p>Specialisation Artificial Intelligence:</p> <ul style="list-style-type: none"> • Lab Course: Challenging Problems in Reinforcement Learning (212428) (3 CP) • Practical Course on Machine learning Security (212427) (4 CP) • Lab course: Introduction to Bayesian Modeling (212417) (3 CP) <p>Specialisation Algorithms, Complexity, Data:</p> <ul style="list-style-type: none"> • Advanced Python Programming (212426) (3 CP) 			<p>Contact time 2 SWS -4 SWS (30-60 h)</p>	<p>Self-study 60-120 h</p>	<p>Group size participants</p>
<p>Teaching language English</p>			<p>Requirements for participation</p>		

Module coordinators and Lecturers

Module coordinators: Studiendekan der Fakultät für Informatik / Dean of Studies of the Faculty of Computer Science

Lecturers: Dozierende im Studiengang M.Sc. Computer Science / Lecturers in the study program M.Sc. Computer Science

Module application

M.Sc. Computer Science

Precognitions

Depending on the chosen practical lab.

Learning goals

After successfully completing the module

- students will have deepened and expanded their programming skills in a research or application area
- students can use software libraries to solve research-related problems
- depending on the chosen lab course, further learning objectives may be added

Contents

Within the scope of this area, a laboratory practical course of 3 LP - 5 LP is to be completed. This is not a pure programming practical course, but a topic-related course with scientific demands. The practical courses are offered on advanced topics in computer science.

Teaching methods

Practical lab at a chair of the Faculty of Computer Science.

Examination forms

Practical exam and documentation

Requirements for the award of credits

Passed practical exam and documentation

Significance of the grade for the final grade (for a total of 120 ECTS)

unbenotet / ungraded

Module title: Project					
Modul code	Credit points 10 CP	Workload 300 h	Term siehe Prüfungsordnung / see Examination regulations	Frequency Every winter and summer semester	Duration 1 semester
Courses			Contact time 1 SWS (15 h)	Self-study 285 h	Group size participants
Teaching language German or English			Requirements for participation none		
Module coordinators and Lecturers Module coordinators: Studiendekan der Fakultät für Informatik / Dean of Studies of the Faculty of Computer Science Lecturers: Dozierende im Studiengang M.Sc. Computer Science / Lecturers in the study program M.Sc. Computer Science					
Module application M.Sc. Computer Science					
Precognitions Prior knowledge: Good programming skills are required. For most projects, prior technical knowledge is required that is specific to the project.					
Learning goals <ul style="list-style-type: none"> • Application of the acquired technical knowledge at master level • Acquisition of technical competence according to the respective project-specific tasks • Application of project management techniques from the field of software engineering • Development of own solution strategies • Documentation and presentation of results 					
Contents Within the framework of the project work, a task from areas of computer science is to be solved under the guidance of a supervisor. The project is usually carried out at a department of the Ruhr-Universität, although projects in industrial co-operation are possible. The topic of the project covers any aspect of computer science. It can be carried out in group work.					
Teaching methods Project work					
Examination forms Project work with final report					
Requirements for the award of credits Successfully completed project work and final report graded at least satisfactory.					
Significance of the grade for the final grade (for a total of 120 ECTS) 10/97: M.Sc. Computer Science					

Module title: Seminars

Modul code	Credit points 3 CP	Workload 90 h	Term siehe Prüfungsordnung / see Examination regulations	Frequency Every winter and summer semester	Duration 1 semester
Courses The current seminars are highlighted: Specialization Algorithms, Complexity, and Data: <ul style="list-style-type: none">• Perlen der Logik (211117)• Seminar Randomisierte Algorithmen (211139)• Seminar Mathematics and Computation (211136)• Seminar zu Algorithmen (212129)• Seminar Quantum Information and Computation (212123)• Learning meets Theoretical Computer Science (212140) Specialization Computer Security: <ul style="list-style-type: none">• Current topics in microarchitectural security (211134)• Master-Seminar "Digitale Souveränität" (211132)• Seminar Software Security (212126)• Seminar Internet Security (212125)• Seminar Safety and Reliability in Artificial Intelligence (211138)• Seminar Security Engineering (212112)• Seminar Mobile Network Security (212134) • Seminar on Applied Privacy and Anonymity (211141) Specialization Artificial Intelligence: <ul style="list-style-type: none">• Advanced Topics in Deep Learning (212120)• Seminar Safety and Reliability Artificial Intelligence (211138)• Master Seminar: Building Trust in Large Language Models (211140)• Algorithms for Decision Making (212130)• Recent Trends in Interpretability of Artificial Intelligence Model (212137)• Seminar: Introduction to Bayesian Modeling (211144) Specialization Design, Implementation, and Analysis of Computer Systems: <ul style="list-style-type: none">• Seminar Networked Systems (211137)• Seminar Distributed Systems (212114)			Contact time 2 SWS (30 h)	Self-study 60 h	Group size participants

<ul style="list-style-type: none"> • Seminar Ressourceneffiziente Systemsoftwarekonzepte (212111) <p>Specialization Software Engineering and Programming Languages:</p> <ul style="list-style-type: none"> • Automated Software Engineering (212138) • Seminar: Search-based Code Generation (212142) <p>Seminars without specialization:</p> <ul style="list-style-type: none"> • Ethics in Computer Science Research (212143) • Seminar Computer science perspectives on Mis- and disinformation (212108) 			
Teaching language German or English	Requirements for participation		
Module coordinators and Lecturers Module coordinators: Studiendekan der Fakultät für Informatik / Dean of Studies of the Faculty of Computer Science Lecturers: Dozierende im Studiengang M.Sc. Computer Science / Lecturers in the study program M.Sc. Computer Science			
Module application M.Sc. Computer Science			
Precognitions depending on choice of seminar			
Learning goals After successful completion of the module <ul style="list-style-type: none"> • have in-depth scientific knowledge of the selected seminar topic • have practiced giving a scientific presentation and can communicate research results independently in a didactically well-prepared presentation • are able to formulate and receive constructive feedback • students can write a paper on their seminar presentation 			
Contents Important skills such as literature work and presentation techniques are practiced in the seminar. Students can choose from the wide range of seminars in different thematic areas, such as Resource Efficient System Software Concepts, Advanced Topics in Deep Learning, Distributed and Networked Systems .			
Teaching methods seminar			
Examination forms Seminar presentation and, if applicable, written paper			
Requirements for the award of credits Successful seminar presentation, written paper if applicable, and attendance at 9 of 10 individual appointments.			
Significance of the grade for the final grade (for a total of 120 ECTS) 3/97: M.Sc. Computer Science			

Module title: Free Elective Moduls					
Modul code	Credit points 20 CP	Workload 600 h	Term see examination regulations	Frequency Each winter and summer semester	Duration 1 semester
Courses any			Contact time 12-13 SWS (ca. 195 h)	Self-study 405 h	Group size participants
Teaching language depending on choice of event			Requirements for participation		
Module coordinators and Lecturers Module coordinators: Studienfachberatung M.Sc. Computer Science / Study advisory service M.Sc. Computer Science Lecturers: Diverse / Various					
Module application M.Sc. Computer Science					
Precognitions Depending on choice of event					
Learning goals Participants acquire so-called key skills in the free electives.					
Contents Within the scope of the free elective area, courses with a volume of at least 20 LP must be completed, which can be freely selected from the courses offered by the university and the UA-Ruhr. Non-technical courses (e.g. foreign languages, law, economics or social sciences) as well as technical courses (e.g. engineering sciences, computer science) can be chosen.					
Teaching methods depending on choice of event					
Examination forms depending on choice of event					
Requirements for the award of credits depending on choice of event					
Significance of the grade for the final grade (for a total of 120 ECTS) unbenotet / ungraded					

Module title: Master's thesis and colloquium					
Modul code	Credit points 30 CP	Workload 900 h	Term 4	Frequency each semester	Duration 1 semester
Courses			Contact time 1 SWS (15 h)	Self-study 885 h	Group size participants
Teaching language English			Requirements for participation Successfully completed modules amounting to 70 CP.		
Module coordinators and Lecturers Module coordinators: Studiendekan Informatik / Dean of Studies Computer Science Lecturers: Lehrende im Studiengang Informatik / Lecturers in the Computer Science programme					
Module application M.Sc. Computer Science					
Precognitions Depending on the choice of topic.					
Learning goals After successful completion of the module: <ul style="list-style-type: none"> • students can independently work on a scientific topic in a timely manner from research to documentation of the results. • students can select, apply and further develop suitable scientific procedures and methods, which they have become acquainted with during their studies, in order to solve a concrete problem • can critically compare and evaluate their results with the state of the art in research • students can present their own results appropriately in written and spoken form. 					
Contents The Master's thesis is a research-oriented, six-month thesis on a specific topic and is written in the last semester of the degree programme. This has a volume of 30 credit points. The Master's thesis is written in English.					
Teaching methods Thesis					
Examination forms Master's thesis and colloquium presentation					
Requirements for the award of credits Both the written Master's thesis and the colloquium presentation must be passed. The share of the colloquium grade in the overall grade is 10%.					
Significance of the grade for the final grade (for a total of 120 ECTS) 30/97: M.Sc. Computer Science					