UNIVERSITÄT
DES
SAARLANDES

Faculty of Mathematics and Computer Science

Module Descriptions

Cybersecurity BSc (English)

17th December 2024

List of module categories and modules

Module Category 1

Fundamentals of Mathematics

Mathematics for Computer Scientists 1 — MfCS1

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
1	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Joachim Weickert

lecturers Prof. Dr. Joachim Weickert
Prof. Dr. Mark Groves
Prof. Dr. Henryk Zähle
Prof. Dr. Christian Bender

entrance requirements none

assessments / exams
- Regular and active participation in tutorials and completion of weakly exercise sheets. An overall score of 50 percent on the tutorial sheets is required to qualify for the examination.
- Examination at the end of the module.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade To be determined from performance in examinations and tutorials. Exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

- Basic mathematical knowledge required in the context of a computer science or bioinformatics degree.
- Ability to formalise and abstract
- Ability to acquire further mathematical knowledge with the help of text books

content

The numbers in parentheses indicate the total number of 2 hour lectures.

```
DISCRETE MATHEMATICS AND ONE-DIMENSIONAL ANALYSIS

A. Fundamentals of discrete mathematics (8)
    1. sets (1)
    2. logic (1)
    3. methods of mathematical proof, including induction (1)
    4. relations (1)
    5. maps (2)
        - injective, surjective, bijective
        - cardinality, countability
        - pigeon-hole principle
    6. prime numbers and divisors (1)
    7. modular arithmetic (1)
```

```
B. One-dimensional analysis (22)

B.1 Numbers, sequences and series (8)
     8. Axiomatics of real numbers, supremum, infimum (1)
     9. complex numbers (1)
    10. sequences (1 1/2)
    11. big O notation (1/2)
    12. series: convergence tests, absolute convergence (2)
    13. power series (1/2)
    14. representations of numbers (1/2)
    15. binomial coefficients and binomial series (1)

B.2 One-dimensional differential calculus (8)
    16. continuity (1)
    17. elementary functions (1)
    18. differentiability (1 1/2)
    19. mean-value theorems and L'Hopital's rule (1/2)
    20. Taylor's theorem (1)
    21. local extrema, convexity, curve sketching (2)
    22. numerical differentiation (1)

B.3 One-dimensional integral calculus (6)
    23. definite integrals (2)
    24. indefinite integrals and the antiderivative (1)
    25. improper integrals (1)
    26. numerical methods for integration (1)
    27. curves and arc length (1)
```

literature & reading

To be announced before the start of the module on the relevant internet page.

additional information

This module is identical in content to the German-language module *Mathematik für Informatiker 1*.

Mathematics for Computer Scientists 2 — MfCS2

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
2	**6**	**every summer semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Joachim Weickert

lecturers Prof. Dr. Joachim Weickert
Prof. Dr. Mark Groves
Prof. Dr. Henryk Zähle
Prof. Dr. Christian Bender

entrance requirements *Mathematics for Computer Scientists 1* is recommended.

assessments / exams
- Regular and active participation in tutorials and completion of weakly exercise sheets. An overall score of 50 percent on the tutorial sheets is required to qualify for the examination.
- Examination at the end of the module.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade To be determined from performance in examinations and tutorials. Exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

- Basic mathematical knowledge required in the context of a computer science or bioinformatics degree.
- Ability to formalise and abstract
- Ability to acquire further mathematical knowledge with the help of text books

content

The numbers in parentheses indicate the total number of 2 hour lectures.

```
LINEAR ALGEBRA

C. Algebraic structures (5)
    29. groups (2)
    30. rings and fields (1)
    31. polynomial rings over fields (1/2)
    32. Boolean algebras (1/2)

D. Linear algebra (21)
    33. vector spaces (2)
        - definition, examples
        - linear maps
        - subspaces
        - linear span, linear dependence, basis, exchange theorem
```

34. linear transformations (image, kernel) (1)
35. matrix representations of linear transformations (1 1/2)
    - interpretation as linear transformations
    - multiplication by composition
    - ring structure
    - inverses
36. rank of a matrix (1/2)
37. Gaussian algorithmn for systems of linear equations (2)
    - Gaussian elimination (1)
    - Back substitution (1)
38. iterative methods for systems of linear equations (1)
39. determinants (1)
40. Euclidean vector spaces, scalar products (1)
41. functional-analytic generalisations (1)
42. orthogonality (2)
43 Fourier series (1)
44. orthogonal matrices (1)
45. eigenvalues and eigenvectors (1)
46. eigenvalues and eigenvectors of symmetric matrices (1)
47. quadratic forms and positive-definite matrices (1)
48. quadrics (1)
50. matrix norms and eigenvalue estimates (1)
51. numerical calculation of eigenvalues and eigenvectors (1)

literature & reading

To be announced before the start of the module on the relevant internet page.

additional information

This module is identical in content to the German-language module *Mathematik für Informatiker 2*.

Module Category 2

Fundamentals of Computer Science

Elements of Machine Learning — EML

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5	**6**	**every winter semester**	**1 semester**	**4**	**6**

responsible Prof. Dr. Jilles Vreeken
Prof. Dr. Isabel Valera

lecturers Prof. Dr. Jilles Vreeken
Prof. Dr. Isabel Valera

entrance requirements The lecture assumes basic knowledge in statistics, linear algebra, and programming. It is advisable to have successfully completed *Mathematics for Computer Scientists 2* and *Statistics Lab*. The exercises use the programming language R. We will give a basic introduction to R in the first tutorial. In addition, for preparation the following materials are useful: *R for Beginners* by Emmanuel Paradis (especially chapters 1, 2, 3 and 6) and *An introduction to R* (Venables/Smith).

assessments / exams Prerequisite for admission to the examination is a cumulative 50% of the points of the theoretical and a cumulative 50% of the points of the practical tasks on the exercise sheets. Depending on the number of participants, the examinations are either written or oral. The final modality will be announced in the first two weeks of the lecture.

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams.

language English

aims / competences to be developed

In this course we will discuss the foundations – the elements – of machine learning. In particular, we will focus on the ability of, given a data set, to choose an appropriate method for analyzing it, to select the appropriate parameters for the model generated by that method and to assess the quality of the resulting model. Both theoretical and practical aspects will be covered. What we cover will be relevant for computer scientists in general as well as for other scientists involved in data analysis and modeling.

content

The lecture covers basic machine learning methods, in particular the following contents:

- Introduction to statistical learning
- Overview over Supervised Learning
- Linear Regression
- Linear Classification
- Splines
- Model selection and estimation of the test errors
- Maximum-Likelihood Methods
- Additive Models

- Decision trees
- Boosting
- Dimensionality reduction
- Unsupervised learning
- Clustering
- Visualization

literature & reading

The course broadly follows the book *An Introduction to Statistical Learning with Applications in R*, Springer (2013). In some cases, the course receives additional material from the book *The Elements of Statistical Learning*, Springer (second edition, 2009). The first book is the introductory text, the second covers more advanced topics. Both books are available as free PDFs. Any change of, or additional material will be announced before the start of the course on the course webpage.

Fundamentals of Data Structures and Algorithms

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	**6**	**every winter semester**	**1 semester**	**4**	**6**

responsible Prof. Dr. Raimund Seidel

lecturers Prof. Dr. Raimund Seidel
Prof. Dr. Markus Bläser
Prof. Dr. Karl Bringmann

entrance requirements *Programming 1* and *2*, and *Mathematics for Computer Scientists 1* and *2* or comparable courses in mathematics are recommended.

assessments / exams Successful completion of the exercise sheets entitles to take part in the exam.

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
  60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Students get to know the most important methods of designing algorithms and data structures:
divide-and-conquer, dynamic programming, incremental construction, ”greedy algorithms”, decimation, forming hierarchies, randomization. They learn to analyze algorithms and data structures for their time and space requirements with respect to the usual RAM machine model and to compare them on this basis. Various kinds of analysis are considered (worst case, amortized, expected case).

Students get acquainted with important efficient data structures and algorithms. They should acquire the ability to apply theoretial analyses and considerations to given methods in order to check their applicability to actually occuring scenarios. Moreover, students should school their skills in developing or adjusting algorithms and data structures with performance guarantees in mind.

content

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Grundzüge von Algorithmen und Datenstrukturen*.

Introduction to Theoretical Computer Science

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Raimund Seidel

lecturers Prof. Dr. Raimund Seidel
Prof. Dr. Bernd Finkbeiner
Prof. Dr. Markus Bläser
Prof. Dr. Karl Bringmann

entrance requirements *Programming 1* and *2* and *Mathematics for Computer Scientists 1* and *2* or comparable courses in mathematics are recommended.

assessments / exams Successful completion of the exercises entitles the student to take the exam.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Students know various models of computation and their relative strengths and abilities.

For selected problems they can show, whether they are solvable in a certain model of computation or not.

They understand the formal notion of computability as well as non-computability.

They can reduce problems to each other.

They are familiar with basics of bounding resources (time, space) for computations and the resulting complexity theory.

content

The language classes of the Chomsky hierarchy and their various definitions via grammars and automata; closure properties; classification of particular languages (”pumping lemmas”);

determinism and non-determinism;

Turing machines and equivalent models of general computability (e.g. µ-recursive function, random acces machins), reducibility, decidability, undecidability;

the complexity measures time and space; the complexity classes P and NP;

the basics of the theory of NP-completeness.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Grundzüge der Theoretischen Informatik*.

Programming 1 — Prog1

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
1	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Gert Smolka

lecturers Prof. Dr. Gert Smolka
Prof. Dr.-Ing. Holger Hermanns
Prof. Bernd Finkbeiner, Ph.D

entrance requirements none

assessments / exams
- Weekly exercises / tests
- Midterm and endterm exam
- Re-examination at end of semester

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Grade combines performance in exams and weekly exercises.

language English

aims / competences to be developed

- functional programming, higher-order and typed
- practical programming skills using an interpreter, debugging, testing
- recursive data structures and recursive algorithms (numbers, lists, trees)
- exceptions
- type abstraction and modularity
- data structures with mutable state, exceptions
- correctness proofs and runtime estimates
- structure of programming languages
- formal description of programming languages (syntax and semantics)
- implementation of programming languages (parsers, interpreters, compilers, stack machines)

content

see above

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Programmierung 1*.

Programming 2 — Prog2

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
2	**6**	**every summer semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Sebastian Hack

lecturers Prof. Dr. Sebastian Hack
Prof. Dr. Jörg Hoffmann

entrance requirements *Programming 1* and *Mathematics for Computer Scientists 1* and mathematics courses in the study semester or comparable knowledge from other mathematics courses (recommended)

assessments / exams Examination performances are given in two parts, which contribute equally to the final grade. To pass the entire course, each part must be passed individually.

In the **practical part**, students must implement a series of programming tasks independently. These programming tasks allow students to practise language concepts and also introduce more complex algorithms and data structures. Automatic tests check the quality of the implementations. The grade of the practical part is largely determined by the test results.

In the **lecture part**, students must complete written examinations and work on exercises. The exercises deepen the material of the lecture. Admission to the written examination depends on the successful completion of the exercises.

In the practical part, a follow-up task can be offered.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

This course teaches the foundations of imperative and object-oriented programming.

In more detail students learn:
* how computers execute programs and how to write programs in assembly language * to implement, debug, and test smaller C programs * to design, implement, debug, and test mid-size Java programs * the basics of object-oriented programming * a basic understanding of formal semantics, type systems, correctness, testing, and verification of imperative languages

content

- Programming at the machine level (assembly)
- Imperative programming
- Object-oriented programming
- Classes and objects
- Inheritance, sub-typing, and dynamic dispatch
- Formal semantics and a type system of a simple imperative language

- Type safety, undefined behavior and their implications
- Foundations of testing and verification

as well as lectures specifically designed for the individual programming tasks.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Programmierung 2*.

Statistics Lab

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4	**6**	**jedes Sommersemester**	**1 Semester**	**4**	**6**

responsible Prof. Dr. Verena Wolf
Prof. Dr. Vera Demberg

lecturers Prof. Dr. Verena Wolf
Prof. Dr. Vera Demberg

entrance requirements none

assessments / exams mündliche oder schriftliche Prüfung

course types / weekly hours
```
  2 SWS Vorlesung
+ 2 SWS Übung
= 4 SWS
```

total workload
```
   60 h Präsenzstudium
+ 120 h Eigenstudum
= 180 h (= 6 ECTS)
```

grade Wird aus Leistungen in der Klausur, sowie den Prüfungsvorleistungen ermittelt. Die genauen Modalitäten werden vom Modulverantwortlichen bekannt gegeben. Alle Modulelemente sind innerhalb eines Prüfungszeitraumes erfolgreich zu absolvieren.

language Deutsch oder Englisch

aims / competences to be developed

- Verständnis der mathematischen Konzepte von Zufallsvariablen und Verteilungen
- Verständnis und Anwendung von Methoden der Punkt-und Intervalschätzung, statistischer Tests
- Verständnis der mathematischen Konzepte von zustandsdiskreten Markovprozessen und Verwendung solcher Prozesse zur Beschreibung von realen Phänomenen

content

Probabilities and Discrete Random Variables

- Probability
- discrete RVs
- expectation, variance and quantiles (also visualization of them)
- higher moments
- important discrete probability distributions
- Generating discrete random variates Continuous Random Variables and Laws of Large Numbers
- σ-algebras (very lightweight)
- Continuous Random Variables
- Important Continuous Distributions
- generating continuous random variates
- Chebyshev's inequality
- Weak/Strong Law of Large Numbers
- Central Limit Theorem

Multidimensional Probability Distributions

- joint probability distribution

- conditional probability distribution
- Bayes' Theorem
- covariance and correlation
- independence
- important multidimensional probability distributions

Point Estimation

- (generalized) method of moments
- maximum likelihood estimation
- Bayesian inference (posterior mean/median, MAP)
- Kernel density estimation
- OLS estimator (this is simple regression but should be mentioned here!)
- (shortly: model selection)

Interval Estimation

- confidence intervals for sample mean/variance
- confidence intervals for MLE
- bootstrap confidence interval
- Bayesian credible interval

Statistical Testing

- Level α tests (Z-Test, T-Test)
- p-value
- chi-squared tests, Fisher test
- multiple testing (Bonferroni correction, Holm-Bonferroni method, Benjamini-Hochberg, etc)

Discrete-time Markov chains (only if time)

- transient distributions
- equilibrium distributions
- Monte-Carlo simulation

HMMs

- Baum-Welch-Algorithmus
- Viterbi-Algorithmus

literature & reading

System Architecture — SysArch

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4	**6**	**every summer semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Jan Reineke

lecturers Prof. Dr. Jan Reineke

entrance requirements *Programming 1*, *Programming 2* (in the same semester), and *Mathematics for Computer Scientists 1* or comparable courses in mathematics are recommended.

assessments / exams The course consists of two parts, which each have to be passed individually in order pass the course as a whole.

In the *projects part*, students have to independently implement a series of projects. These projects deepen the practical comprehension of the lecture material in the areas of computer architecture and operating systems.

In the *lecture part*, students must pass the written exams and work on written assignments and/or quizzes. Successful completion of the written assignments and/or the quizzes is a prerequisite for participation in the written exams.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined based on the performance in exams, exercises, and projects. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Students shall understand the functionality and the most important properties of modern computer architectures and operating systems.

Furthermore students shall understand the design principles underlying their implementations.

content

1. Computer architecture
    a. Boolean algebra and combinatorial circuits
    b. Number representations and arithmetic circuits
    c. Instruction set architectures
    d. Microarchitectures, in particular, the design of a basic reduced instruction set machine, and performance optimizations such as pipelining and caches

2. Operating systems
    a. Virtualization mechanisms
    b. Scheduling algorithms
    c. File systems

literature & reading

Will be announced before the start of the course on the course page on the internet.

additional information

This module is identical in content to the German-language module *Systemarchitektur*.

Module Category 3

Practical Skills Classes

Practical Training: Cybersecurity Lab — CySecLab

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	6	Winter			6

responsible	Dr. Ben Stock
lecturers	
entrance requirements	none
assessments / exams	Successful passing of project goals.
course types / weekly hours	
total workload	
grade	The course is ungraded and only gets a pass/fail.
language	

aims / competences to be developed

The students apply the knowledge they have learned within the Foundations of Cybersecurity 1 & 2 lectures. In particular, they extend their theoretical knowledge through practical exercises. They learn to solve IT security challenges and organize themselves within a team. They are hence knowledgable about the core aspects of IT security, privacy, and usability.

content

See ”Aims / Competences to be developed”

literature & reading

If need be, will be presented in the kick-off meeting.

Software Engineering Lab — SE Lab

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**lecture free time after SS**	**7 weeks**	**BLOCK**	**9**

responsible Prof. Dr. Sven Apel

lecturers Prof. Dr. Sven Apel
Dr. Norman Peitek

entrance requirements Participation in the Software Engineering Lab requires extensive programming skills as taught in the courses *Programming 1* and *Programming 2*. A passing grade in *Programming 2* is required to enroll in this course.

Students are required to bring their own laptops.

assessments / exams The goal of the Software Engineering Lab is to develop a non-trivial software system in a team effort. In this course, a number of documents (design models, documentation, implementation plan, etc.) and artifacts (source code, tests, etc.) need to be developed and submitted. Correctness, completeness, quality, and timely submission of all documents and artifacts are major grading criteria.

The Software Engineering Lab consists of two phases: exercise phase and group phase.

In the *exercise phase*, participants will complete an entry exam, covering current topics from the lecture. Only participants that have passed the exercise phase will be admitted to the group phase.

In the *group phase*, participants will first design and then implement and test a substantial software system in a team effort, and submit both their design and their implementation (including tests) for evaluation. All documents (design models, documentation, implementation plan, etc.) and artifacts (source code, tests, etc.) of the group phase will be evaluated based on the principles and quality criteria conveyed in the lectures. To pass the group phase, students must pass both the design submission and the implementation submission, and prove individually their substantial contribution to the group project.

More details on the exams will be announced at the beginning of the course.

course types / weekly hours Daily exercises and lectures (first few weeks)
Daily project work with tutoring

total workload

```
  35 h of lectures and exercises
+ 235 h project work
= 270 h (= 9 ECTS)
```

grade ungraded

language English

aims / competences to be developed

Participants acquire the ability to solve complex software development problems individually and in teams.

Participants are aware of common problems and pitfalls of software development and know how to address them.

Participants are able to accomplish and coordinate software development tasks based on a set of given requirements. For this purpose, they are able to select proper methods and techniques to minimize risks and maximize software quality.

Participants know about foundations and principles of software design, including cohesion, coupling, modularity, encapsulation, abstraction, and information hiding. They are acquainted with a whole array of design patterns, knowing their aim

and individual strengths and weaknesses. They are able to apply design patterns beneficially and to judge and improve the quality of software designs.

Participants master fundamental techniques and tools for software testing, debugging, and version control.

content

- Software design
- Software testing
- Team work
- Debugging

literature & reading

- Software Engineering. I. Sommerville, Addison-Wesley, 2004.
- Software Engineering: A Practioner's Approach. R. Pressman, McGraw Hill Text, 2001.
- Using UML: Software Engineering with Objects and Components. P. Stevens, et al., Addison-Wesley, 1999.
- UML Distilled. M. Fowler, et al., Addison-Wesley, 2000.
- Objects, Components and Frameworks with UML, D. D'Souza, et al., Addison-Wesley, 1999.
- Designing Object-Oriented Software. R. Wirfs-Brock, et al., Prentice Hall, 1990.
- Design Patterns. Elements of Reusable Object-Oriented Software. E. Gamma, et al., Addison-Wesley, 1995.
- Head First Design Patterns. E. Freeman, et al. O'Reilly, 2004.
- Software Architecture: Perspectives on an Emerging Discipline. M. Shaw, et al., Prentice-Hall, 1996.
- Refactoring: Improving the Design of Existing Code. M. Fowler, et al., Addison-Wesley, 1999.
- Software Testing and Analysis: Process, Principles and Techniques. M. Pezze, Wiley. 2007.

additional information

This module is identical in content to the German-language module *Softwarepraktikum*.

Module Category 4

Specialist Area of Cybersecurity

Cryptography — Crypto

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Dr. Nico Döttling

lecturers Prof. Dr. Cas Cremers
Dr. Nico Döttling
Dr. Antoine Joux
Dr. Lucjan Hanzlik
Dr. Julian Loss

entrance requirements For graduate students: Basic knowledge in theoretical computer science required, background knowledge in number theory and complexity theory helpful

assessments / exams
- Oral / written exam (depending on the number of students)
- A re-exam is normally provided (as written or oral examination).

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The students will acquire a comprehensive knowledge of the basic concepts of cryptography and formal definitions. They will be able to prove the security of basic techniques.

content

- Symmetric and asymmetric encryption
- Digital signatures and message authentication codes
- Information theoretic and complexity theoretic definitions of security, cryptographic reduction proofs
- Cryptographic models, e.g. random oracle model
- Cryptographic primitives, e.g. trapdoor-one-way functions, pseudo random generators, etc.
- Cryptography in practice (standards, products)
- Selected topics from current research

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Foundations of Cybersecurity 1 — CySec1

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
1	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Dr. Ben Stock

lecturers Dr. Ben Stock

entrance requirements none

assessments / exams Students need to solve the exercise during the semester to be allowed to take the exam.

course types / weekly hours

```
  4 SWS Lecture
+ 2 SWS Tutorials
= 6 SWS
```

total workload

```
   90 h of classes
+ 180 h of private study
= 270 h (= 9 ECTS)
```

grade By default, the final grade is calculated through the exam only. This mode can be changed by the lecturer and such changes will be announced at the beginning of the term.

language English

aims / competences to be developed

The students know the legal foundations of information security in Germany. In addition, they know the basic building blocks of modern cryptography, network security as well as privacy. Special emphasis is on network security, such that students know relevant protocols for secure communication and can utilize them.

To apply the learned theoretical knowledge, the students will also learn Python to apply the concepts in practical tasks.

content

- Foundations of the Strafgesetzbuch w.r.t. to information security
- Basic understanding of symmetric and asymmetric cryptographic protocol and their usage scenarios
- Basic understanding of hash functions and important properties of hash functions
- Network foundations of all layer (according to the TCP/IP model)
- Security protocols for each network layer
- Foundations of privacy and anonymity
- Basics of Web security
- Parallel to the security topics, we will also introduce the Python programming language

literature & reading

The literature is english and will be announced at the beginning of the lecture.

additional information

Programming tasks in Python. Pen&paper exercises in groups (and tutorials).

Foundations of Cybersecurity 2 — CySec2

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
2	6	every summer semester	1 semester	4	6

responsible Dr. Michael Schwarz

lecturers Dr. Michael Schwarz

entrance requirements none

assessments / exams Written exam, and possibly mid-term exams and/or graded exercise sheets

course types / weekly hours

```
  2 SWS lectures
+ 2 SWS tutorial
= 4 SWS
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade The module is passed in its entirety if the examination performance has been passed.

language Englisch

aims / competences to be developed

Students know the foundations of security in software, operating systems and IT systems in general.

content

- Basic Introduction to Operating Systems
- Foundations of System Security
- Foundations of Software Security
- Foundations of Attack Detection and Defense

literature & reading

The teaching material will be in English and it will be announced at the beginning of the lecture.

Module Category 5

Seminars in Cybersecurity

Proseminar

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	**6**	**every semester**	**1 semester**	**2**	**5**

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Basic knowledge of the relevant sub-field of the study program.

assessments / exams
- Thematic presentation with subsequent discussion
- Active participation in the discussion
- short written report and/or project possible

course types / weekly hours `2 h proseminar`

total workload
```
  30 h of lectures and exercises
+ 120 h project work
= 150 h (= 5 ECTS)
```

grade Will be determined from the performance in the presentation and the written report and/or the seminar project. The exact modalities will be announced by the respective instructor.

language English or German

aims / competences to be developed

At the end of the proseminar, students have gained a basic understanding of current or fundamental aspects of a specific subfield of computer science.

In particular, they have gained basic competence in independent scientific research, classification, summarization, discussion, criticism and presentation of scientific findings.

Compared to the seminar, the focus of the proseminar is on the acquisition of basic scientific working methods.

content

With guidance, the following will be practiced hands-on:

- Reading and understanding scientific papers
- Discussion of the scientific work in the group
- Analyzing, summarizing and reporting the specific topic
- Presentation techniques

Specific in-depth study related to the individual topic of the seminar.

The typical procedure of a proseminar is usually as follows:

- Preparatory discussions for topic selection
- Regular meetings with discussion of selected contributions
- if applicable, work on a project related to the topic
- Presentation and, if necessary, writing a report on one of the presentations

literature & reading

Material is selected according to the topic.

additional information

The proseminars available will be announced prior to the beginning of the semester and will vary by study programme.

Seminar

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5	**6**	**every semester**	**1 semester**	**2**	**7**

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Basic knowledge of the relevant sub-field of the study program.

assessments / exams
- Thematic presentation with subsequent discussion
- Active participation in the discussion
- short written report and/or project possible

course types / weekly hours `2 h seminar (weekly)`

total workload
```
  30 h of lectures and exercises
+ 180 h project work
= 210 h (= 7 ECTS)
```

grade Will be determined from the performance in the presentation and the written report and/or the seminar project. The exact modalities will be announced by the respective instructor.

language English or German

aims / competences to be developed

At the end of the seminar, students have primarily gained a deep understanding of current or fundamental aspects of a specific subfield of computer science.

They have gained further competence in independent scientific research, classifying, summarizing, discussing, criticizing and presenting scientific findings.

content

Largely independent research of the seminar topic:

- Reading and understanding of scientific papers
- Analysis and evaluation of scientific papers
- Discussion of the scientific work in the group
- Analyzing, summarizing and reporting the specific topic
- Developing common standards for scientific work
- Presentation techniques

Specific in-depth study related to the individual topic of the seminar.

The typical procedure of a seminar is usually as follows:

- Preparatory discussions for topic selection
- Regular meetings with discussion of selected presentations
- if applicable, work on a project related to the topic
- Presentation and, if necessary, writing a report on one of the presentations

literature & reading

Material is selected according to the topic.

additional information

The seminars available will be announced prior to the beginning of the semester and will vary by study programme.

Module Category 6

Core Topics in Cybersecurity

Advanced Public Key Cryptography — APC

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Nico Döttling

lecturers Dr. Nico Döttling

entrance requirements Cryptography

assessments / exams Mündliche Prüfung oder Abschlussklausur

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistung bestanden wurde.

language English

aims / competences to be developed

Students will be obtaining a basic understanding of advanced concepts of modern cryptography, such as how to modeling security of complex systems, advanced encryption schemes like fully homomorphic encryption and functional encryption, as well as zero-knowledge proofs and multiparty computation.

content

- Modelling Security for Encryption Schemes
- Proving Security of Encryption Schemes
- Tools and Paradigms for designing Encryption Schemes
- Advanced notions of encryption such as homomorphic encryption, identity based encryption, attribute-based encryption and functional encryption

literature & reading

The teaching material will be in English and it will be announced at the beginning of the lecture.

Algorithms in Cryptanalysis

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible	Dr. Antoine Joux
lecturers	Dr. Antoine Joux
entrance requirements	Good working knowledge of algebra and algorithms
assessments / exams	Written exam.
course types / weekly hours	
total workload	
grade	Determined by the performance in exams.
language	

aims / competences to be developed

The goal of this course is to familiarise the students with the variety of algorithmic techniques that are used in cryptanalysis and with the mathematical background underlying these techniques.

content

The course will be arranged around three main directions:

- Presentation of the cryptographic motivation
- Description of relevant algorithmic techniques
- Application of the algorithms in the cryptographic context

The techniques covered in the course will range from fundamental algorithms such as sorting which are essential in many cryptanalyses to advanced factorisation and discrete logarithm algorithms on finite field and elliptic curves, requiring a working knowledge of number theory.

literature & reading

Foundations of Web Security — FoWS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Ben Stock

lecturers Dr. Ben Stock

entrance requirements *Security* or *Foundations of Cybersecurity 1 and 2*

assessments / exams Projekt und schriftliche Abschlussklausur

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistung bestanden wurde.

language English

aims / competences to be developed

The students will acquire a practical understanding of the security threats a modern Web application is faced with. The students fully comprehend the attack surface of applications and know the necessary countermeasures and mitigations for a wide range of attacks.

content

- Historical evolution of the Web
- Client-side security (e.g., Cross-Site Scripting, Cross-Site Script Inclusion, Cross-Site Request Forgery)
- User-centric security (e.g., Clickjacking & Phishing)
- Server-side security (e.g., SQL injections, command injections)
- Infrastructure security (e.g., HTTPS & attacks against it)

literature & reading

The teaching material will be in English and it will be announced at the beginning of the lecture.

Generating Software Tests GST

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Prof. Dr. Andreas Zeller

lecturers Prof. Dr. Andreas Zeller

entrance requirements Programming 1, Programming 2, Softwarepraktikum

assessments / exams Projekte und Mini-Tests

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
  60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistung bestanden wurde.

language English

aims / competences to be developed

Software has bugs and catching bugs can involve lots of effort. Yet, finding bugs is important especially when these bugs are critical vulnerabilities. This course addresses this problem by automating software testing, specifically by generating tests automatically. Students learn the basics of general testing and security testing and explore the most important tools and techniques for generating software tests.

content

- Introduction to Software Testing
- Fuzzing: Breaking Things with Random Inputs
- Mutation-Based Fuzzing
- Greybox Fuzzing
- Search-Based Fuzzing
- Fuzzing with Grammars
- Parsing Inputs
- Probabilistic Grammar Fuzzing
- Fuzzing with Generators
- Reducing Failure-Inducing Inputs
- Mining Input Grammars
- Concolic Fuzzing
- Symbolic Fuzzing
- Testing APIs
- Testing Web Applications
- Testing Graphical User Interfaces
- When To Stop Fuzzing

literature & reading

The teaching material consists of text, Python code, and Jupyter Notebooks from the textbook “The Fuzzing Book” (https://www.fuzzing-book.org/) in English.

Machine Learning in Cybersecurity — MLCySec

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Prof. Dr. Mario Fritz

lecturers Prof. Dr. Mario Fritz

entrance requirements *Data Science/Statistics Course*

assessments / exams Übungen, Projekt und mündliche Prüfung

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistungen bestanden wurden.

language English

aims / competences to be developed

Students know about the opportunities and risks of applying machine learning in cyber security. They understand a range of attacks and defense strategies and are capable of implementing such techniques. Students are aware of privacy risks of machine learning methods and understand how such risks can be mitigated.

content

- Machine learning methodology in the context of cyber security
- Applications and opportunities of learning in cyber security
- Risks and attacks on machine learning in cyber security
- Malware classification
- Anomaly detection
- Intrusion detection
- Evasion attacks
- Model stealing
- Privacy risks and attacks
- Privacy protection

literature & reading

The teaching material will be in English and it will be announced at the beginning of the lecture.

Mobile Security — MOS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Sven Bugiel

lecturers Dr. Sven Bugiel

entrance requirements *Foundations of Cybersecurity 1 and 2*, *Programmierung 2* (recommended)

assessments / exams Schriftliche Abschlussklausur

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistung bestanden wurde.

language English

aims / competences to be developed

This advanced lecture deals with different, fundamental aspects of mobile operating systems and application security, with a strong focus on the popular, open-source Android OS and its ecosystem. In general, the awareness and understanding of the students for security and privacy problems in this area is increased. The students learn to tackle current security and privacy issues on smartphones from the perspectives of different security principals in the smartphone ecosystem: end-users, app developers, market operators, system vendors, third parties (like companies).

Central questions of this course are:

- What is the threat model from the different principals' perspectives?
- How are the fundamental design patterns of secure systems and security best practices realized in the design of smartphone operating systems? And how does the multi-layered software stack (i.e., middleware on top of the OS) influence this design?
- How are hardware security primitives, such as Trusted Execution Environments, and trusted computing concepts integrated into those designs?
- What are the techniques and solutions market operators have at hand to improve the overall ecosystem's hygiene?
- Which problems and solutions did security research in this area identify in the past half-decade?
- Which techniques have been developed to empower the end-users to protect their privacy?

The lectures are accompanied by exercises to re-enforce the theoretical concepts and to provide an environment for hands-on experience for mobile security on the Android platform. Additionally, a short course project should give hands-on experience in extending Android's security architecture with a simple custom mechanism for access control enforcement.

content

- Security concepts and introduction to Android's security architecture
- Access control and permissions
- Role of Binder IPC in the security architecture
- Mandatory access control
- Compartmentalization
- Advanced attacks and problems

- SSL and WebViews
- Application-layer security extensions
- Smart Home IoT
- Hardware-based mobile platform security
- Course project: Security extension to the Android Open Source Project

literature & reading

The teaching material will be in English and it will consist of slides as well as book chapters.

Obfuscation

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**usually every year**	**1 semester**	**4**	**6**

responsible Dr. Nico Döttling

lecturers Dr. Nico Döttling

entrance requirements While there are no strict requirements to attend this course beyond being interested in the topic, having taking the core-lecture cryptography is recommended.

assessments / exams Passing a usually oral exam

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Determined by the performance in exams

language English

aims / competences to be developed

Obtain a fundamental understanding of how obfuscation can be defined and constructed using cryptographic notions and techniques. Study the mathematical structures and underlying hardness assumptions on which current obfuscation candidates are based.

content

In software design, obfuscation generally refers to various techniques which make computer code unintelligible, or make it hard to reverse engineer program code. Such techniques have been used for decades in an attempt to protect proprietary algorithms in commercial software. Unfortunately, commercially available obfuscation tools are typically broken within a very short time of their introduction.

From a scientific perspective, this raises the question whether the task of obfuscation is possible at all, or whether any conceivable obfuscation scheme can be broken. To approach this question, we first need to agree on a suitable notion of what it means to break an obfuscation scheme. This question was first addressed by a seminal work of Barak et al. (CRYPTO 2001) who considered several ways of defining security for obfuscation schemes.

In this course, we will take a comprehensive tour through the realm of cryptographically secure obfuscation. We will start by surveying the initial impossibility results, and see how they can be circumvented by weakening the security requirements in a meaningful way. We will proceed to show how obfuscation became a central hub of modern cryptography, on which essentially any advanced notion of proof systems and encryption can be based.

literature & reading

Parameterized Verification

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Swen Jacobs

lecturers Dr. Swen Jacobs

entrance requirements The course picks up on some of the topics of the core lecture ”Verification”, which is a recommended prerequisite for this course.

assessments / exams Passing a written exam (re-exam can be oral, if any)

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Determined by the performance in exams

language

aims / competences to be developed

The course is aimed at students interested in the theoretical concepts behind parameterized verification, which generalize system models, specification formalisms and proof methods from standard verification approaches.

content

We consider the problem of providing correctness and security guarantees for systems that scale with some parameter, e.g., the number of nodes in a network, the number of concurrent processes in a multi-threaded program, or the size of a data structure that a program operates on. Most systems are expected to scale in one or several parameters, but correctness and security guarantees are usually only given for fixed parameter values. In contrast, parameterized verification is the problem of obtaining correctness guarantees for all parameter values. In this course, we will look at methods for parameterized verification and investigate their capabilities and limitations.

literature & reading

The course is based on ”Decidability of Parameterized Verification” by Bloem et al., augmented with selected research papers.

Physical-Layer Security — PHYSEC

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Nils-Ole Tippenhauer

lecturers Dr. Nils-Ole Tippenhauer

entrance requirements *Security* or *Foundations of Cyber Security I + II*

assessments / exams Übungen und schriftliche Abschlussklausur

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistung bestanden wurde.

language English

aims / competences to be developed

- Classify and describe common physical-layer attacks and countermeasures
- Apply known side-channel attacks, e.g., simple power analysis
- Model, analyze, and simulate physical-layer attacks and defenses for wireless communications (e.g., eavesdropping, jamming, manipulation)
- Classify and describe countermeasures such as distance bounding protocols to prevent relay attacks
- Evaluate the security of existing cyber-physical systems against physical-layer attacks
- Classify and describe security issues and solutions for industrial control systems

content

The lecture will cover three main topic areas: attacks (and countermeasures) that leverage physical channels (e.g., side-channel attacks), attacks (and countermeasures) involving wireless communications (e.g., jamming, manipulation, and forwarding), and security for cyber-physical systems (such as industrial control systems).

Selected list of topics:

- Relay attacks
- Distance Bounding
- Physical-Layer Identification
- Wireless eavesdropping and manipulations
- GPS spoofing and countermeasures
- Industrial Control System security, attacks and countermeasures
- Security issues related to PLC logic applications, proprietary industrial protocols and end devices

literature & reading

The teaching material will be in English and will be announced at the beginning of the lecture.

additional information

While the lecture will touch physical-layer concepts such as (wireless) signal processing, no background in that area is assumed. Exercises will require students to run Linux applications (e.g., via a virtual machine).

Privacy-Enhancing Technologies PETS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Wouter Lueks

lecturers Dr. Wouter Lueks

entrance requirements A basic understanding of security and cryptography (as taught for example in *Foundations of Cybersecurity 1* and *2* or *Security*) is essential to be able to follow the material in this course. A larger course in cryptography (for example the core lecture *Cryptography*) would help.

assessments / exams
- Programming projects
- Final exam (written or oral)
- Midterm

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial and office hours
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Digital technologies have become an essential part of our day to day live. While often beneficial, these technologies also bring great privacy risks. In this course you will learn how to mitigate these risks by design privacy-friendly systems and how to evaluate the privacy-protections offered by systems.

To reason about the privacy of systems you will learn how to define desirable privacy properties and how to reason about privacy attackers. Privacy can be violated both at the application level (i.e., what data parties exchange) as well as on the meta-data level (i.e., how parties exchange data). You will learn about techniques to offer protection at both of these layers.

On the application layer, we'll discuss cryptographic techniques such as secure multi-party computation, homomorphic encryption and anonymous authentication that together can be used to ensure privacy at the application layer. We will also discuss data anonymisation techniques such as k-anonymity and differential privacy to enable privacy-friendly data publishing. On the meta-data level, we'll explore techniques for anonymous communication, censorship resistance, (browser) tracking and location privacy.

At the end of this course you will be able to:

- Explain basic building-blocks for designing privacy-friendly systems
- Combine these building blocks to solve simple problems while maintaining privacy
- Evaluate the privacy of simple proposed systems.

content

- Introduction to privacy
- Secure Multi-party computation

- (Fully) Homomorphic encryption
- Privacy-preserving authentication
- Anonymous communication
- Censorship resistance
- Protected data release and differential privacy
- Tracking
- Location Privacy

literature & reading

The teaching material will be in English and it will be announced at the beginning of the lecture.

Reactive Synthesis RS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Swen Jacobs

lecturers Dr. Swen Jacobs

entrance requirements *Grundzüge der Theoretischen Informatik*

assessments / exams Projekt und schriftliche Abschlussklausur

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistung bestanden wurde.

language English

aims / competences to be developed

Students will gain an understanding of reactive synthesis in its full breadth, ranging from its theoretical formalization as an infinite game to efficient algorithms and data structures to solve the synthesis problem, and in the implementation of state-of-the-art algorithms for practically relevant and challenging problems.

content

- State of the art in reactive synthesis
- Formalization of reactive synthesis problems as an infinite game
- Different types of infinite games
- Solving infinite games
- Efficient algorithms and data structures for solving games
- Implementation of reactive synthesis tools/game solvers

literature & reading

The teaching material will be in English and it will be announced at the beginning of the lecture.

Reverse Engineering and Exploit Development for Embedded Systems RExp4Em-Sys

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**2 weeks**	**BLOCK**	**6**

responsible Dr. Ali Abbasi

lecturers Dr. Ali Abbasi

entrance requirements An extensive background on software security and a background on embedded systems are recommended.

assessments / exams
- Regular attendance at classes and tutorials.
- Successful completion of a course final project (Project due approximately 2 weeks).
- Score at least 50% on the final oral exam (which is based on your final project).
- To be admitted to the exam, you must achieve at least 50% of the points from the exercises.

course types / weekly hours Daily lectures followed by daily tutorial and excercises

total workload
```
  60 h of lectures and exercises
+ 120 h project work
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in oral exam, exercise tasks and final project. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

In this course, we will work toward understanding the fundamentals of developing software/hardware exploits against embedded systems. We will cover topics such as firmware extraction modification, and different hardware serial protocols. We also cover topics such as exploit development for embedded devices and write exploits for vulnerabilities such as uninitialized stack variables, off-by-one bugs, Use-after-free, and utilize techniques such as ROP, Signal-oriented programming, to attack embedded systems. We also attack micro-controllers and try to extract secrets from them by utilizing reverse-engineering techniques and firmware patching. Finally, we perform fuzz-testing on embedded firmware via re-hosting.

content

1. Software security vulnerabilities in embedded systems 1
2. Software security vulnerabilities in embedded systems 2
3. Software security vulnerabilities in embedded systems 3
4. Introduction to Firmware and Peripheral Register Configuration
5. Embedded Hardware Peripherals
6. Binary and Firmware Emulation
7. Ghidra and Reverse Engineering
8. Fuzzing and Firmware Patching
9. Fuzzing 2
10. Real-World Firmware Exploitation and Project QA

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Secure Web Development — SecWebDev

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Nils-Ole Tippenhauer

lecturers Dr. Nils-Ole Tippenhauer

entrance requirements none

assessments / exams Projekt und schriftliche Abschlussklausur

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Das Modul ist insgesamt bestanden, wenn die Prüfungsleistung bestanden wurde.

language English

aims / competences to be developed

Students will learn principles, best-practices, and tools to build secure web applications. Also, Students will acquire deep understanding of existing vulnerabilities and security threats.

content

- Basics on secure software engineering and development life-cycle
- Architecture of modern web application
- Secure coding and coding patterns
- Security of the HTTP message processing pipeline
- Known threats and vulnerabilities
- (Mini) BiBiFi challenges (Build it, Break it, Fix it)

literature & reading

Teaching material and notes will be in English and announced at the beginning of the lecture.

additional information

Given the limited resources available for this lecture, the course is limited to 20 seats.

Side-Channels Attacks & Defenses SCAD

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Dr. Michael Schwarz

lecturers Dr. Michael Schwarz

entrance requirements A background in the basics of operating systems and in programming C is recommended

assessments / exams project and written exam

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams, exercises, and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Students will acquire both a theoretical and practical understanding of microarchitectural attacks, such as side-channel attacks, transient-execution attacks, and software-based fault attacks. The students will understand the attack surface for these types of attacks and learn how such attacks can be mitigated on the hardware, operating system, and software layer. Moreover, students will acquire a more in-depth understanding of how modern CPUs work internally.

The lectures are accompanied by exercises to apply the theoretical concepts in a practical setting and get hands-on experiences with side-channel attacks and their mitigations.

content

- Basic introduction to the CPU microarchitecture and side channels
- Software-based side-channel attacks (e.g., cache attacks, timing attacks)
- Trusted execution environments and their attack surface (e.g., controlled-channel attacks)
- Transient execution attacks (e.g., Meltdown, Spectre, ZombieLoad)
- Software-based fault attacks (e.g., Rowhammer, Plundervolt)
- Overview of various other types of side channels
- Mitigation strategies in software and hardware

literature & reading

The teaching material will be in English and it will be announced at the beginning of the lecture.

Usable Security and Privacy — USP

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**every summer semester**	**1 semester**	**4**	**6**

responsible Dr. Katharina Krombholz

lecturers Dr. Katharina Krombholz

entrance requirements *Foundations of Cybersecurity 1* and *2* or the core lecture *Security* and knowledge in statistics are highly recommended. A deep understanding of the topics covered in these lectures as well as a good understanding of statistics is necessary.

assessments / exams

- Regular attendance of classes and tutorials & office hours.
- Assignments during the semester
- Final exam
- A re-exam takes place before the start of lectures in the following semester for students who failed in the final exam or did not participate in the final exam

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial or office hours
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in the exam and the exercises. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

In this lecture, students will learn about human-centric aspects of IT security. Besides research and design methods, students will learn about hot topics in usable security such as authentication, confidentiality and privacy. In particular, they will learn to

- design user studies to study how humans interact with security & privacy technology with respect to threat models,
- collect, understand, evaluate qualitative & quantitative data,
- interpret results and draw conclusions based on your data,
- design new security and privacy technology that is better tied to the users' needs and values.

content

- Qualitative Research Methods
- Quantitative Research Methods
- Statistics
- User Study Design and Ethics
- Design Methods
- Surveillance
- Privacy
- Authentication
- Encryption

literature & reading

Will be announced before the start of the course on the course page on the internet.

additional information

Occasionally, this course is offered as a three week block course before the lectures of the summer semester. In this case, lecture and tutorial distribution as well as examination will vary and be communicated with the students on the course page on the internet. This module was formerly also known as *Usable Security*.

Module Category 7

Complementary Topics in Cybersecurity

Automated Debugging

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**at least every two years**	**1 semester**	**4**	**6**

responsible Prof. Dr. Andreas Zeller

lecturers Prof. Dr. Andreas Zeller

entrance requirements *Programmierung 1*, *Programmierung 2* and *Softwarepraktikum*

assessments / exams Projects and mini-tests

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
  60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade The module is passed in its entirety if the examination performance has been passed.

language English

aims / competences to be developed

Finding and fixing software bugs can involve lots of effort. This course addresses this problem by automating software debugging, specifically identifying failure causes, locating bugs, and fixing them. Students learn the basics of systematic debugging, and explore tools and techniques for automated debugging.

content

- Tracking Problems
- The Scientific Method
- Cause-Effect Chains
- Building a Debugger
- Tracking Inputs
- Assertions and Sanitizers
- Detecting Anomalies
- Statistical Fault Localization
- Generating Tests
- Reducing Failure-Inducing Inputs
- Mining Software Archives
- Fixing the Defect
- Repairing Bugs Automatically
- Managing Bugs

literature & reading

The teaching material consists of text, Python code, and Jupyter Notebooks from the textbook “The Debugging Book” (https://www.debuggingbook.org/), also in English.

Big Data Engineering — BDE

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**every summer semester**	**1 semester**	**4**	**6**

responsible Prof. Dr. Jens Dittrich

lecturers Prof. Dr. Jens Dittrich

entrance requirements *Programming 1*, *Programming 2*, *Software Engineering Lab*, *Mathematics for Computer Scientists 1*, as well as *Fundamentals of Algorithms and Data Structures* (all recommended)

assessments / exams Successful participation in the exercises/project entitles the student to take part in the final exam.

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams, exercises, and (optionally) practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The lecture provides basic knowledge of fundamental concepts of data management and data analysis in Big Data Engineering.

As part of the exercises, a project can be carried out during the semester. This can be, for example, a social network (Facebook style) or any other project where data management techniques can be practiced (e.g., natural science data, image data, other web applications, etc.). First, this project will be modeled in E/R, then realized and implemented in a database schema. Then the project is extended to manage and analyze unstructured data as well. Altogether, all fundamental techniques that are important for managing and analyzing data are thus demonstrated on a single project.

content

```
1 Introduction and classification
    Classification and delimitation: "Big Data"
    Value of Data: The gold of the 21st century
    Importance of database systems
    What is data?
    Modeling vs Reality
    Costs of inadequate modeling
    Using a database system vs developing it yourself
    Positive examples for apps
    Requirements
    References
    Lecture mode

2 Data modeling
    Motivation
```

```
    E/R
    Relational Model
    domains, attributes
    entity type vs entity
    relation type vs relation
    Hierarchical Data
    keys, foreign keys
    inheritance
    Redundancy, normalization, denormalization

3 query languages
    Relational Algebra
    Graph-oriented query languages

4 SQL
    Basics
    Relationship to relational algebra
    CRUD-style vs analytical SQL
    SQL standards
    joins, grouping, aggregation, having
    PostgreSQL
    Integrity constraints
    Transaction concept
    ACID
    Views

5 Basic query optimization
    Overview
    from WHAT to HOW
    Costs of different operations
    EXPLAIN
    Physical Design
    Indexes, Tuning
    Database tuning
    Rule-based query optimization
    Cost-based query optimization

6 Automatic Concurrency control
    Serializability theory
    Isolation levels
    Pessimistic concurrency control
    lock-based approaches, 2PL-variants

7 Grahical Data
    recursion in SQL, WITH RECURSIVE
    graph-oriented query languages: e.g. Cypher, Neo4J

8 Database Security
    SQL injection
    passwords
    salt and pepper

9 Ethical Aspects of Big Data
    mass surveillance
    NSA
    the "big data arithmetic"
    counter measures
```

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module was formerly also known as *Informationssysteme*. This module is identical in content to the German language module *Big Data Engineering*.

Concurrent Programming CP

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**every summer semester**	**1 semester**	**4**	**6**

responsible Prof. Dr.-Ing. Holger Hermanns

lecturers Prof. Dr.-Ing. Holger Hermanns
Prof. Dr. Bernd Finkbeiner
Prof. Dr. Verena Wolf

entrance requirements *Programming 1* and *2*, *Software Engineering Lab*, and *Introduction to Theoretical Computer Science* (recommended).

assessments / exams Two exams (mid-term and end-term), practical project.

Re-exams take place within the last weeks before the start of lectures of the following semester.

course types / weekly hours **Element T - Theory (2 SWS):**
8 lectures: 6 weeks
4 exercises: 6 weeks

Element A - Application (2 SWS):
9 lectures: 6 weeks
4 exercises: 6 weeks

Element P - Practice (private study):
8 written reflections during the semester (preparatory work for exams), followed by project work over approx. 2 weeks

= 4 SWS

total workload **Element T:**
24 h of classes, 36 h private study

Element A:
26 h of classes, 34 h private study

Element P:
60 h private study

```
   50 h of classes
+ 130 h private study
= 180 h (= 6 ECTS)
```

grade Is determined from performance in written examinations (following elements T and A), as well as the preparatory examinations (element P). The exact modalities will be announced by the person responsible for the module. All module elements must be successfully completed within one examination period.

language English / Deutsch

aims / competences to be developed

The participants of this course get acquainted with concurrency in computation as a far-reaching and foundational principle with respect to both theory and application of modern computing sciences. By analysing and applying different formal models, the participants gain a deeper understanding of concurrency, and learn to apply formal computing concepts correctly. The theoretical knowledge acquired in the first half of the lecture is in the second half applied to practical programming. Therein, participants learn using the programming paradigms “shared memory” and “message passing” starting off with the programming language pseuCo before applying their skills to Java and (partially) to Go. In addition, participants learn to describe various phenomena of concurrent programming using formal models, and to derive concrete solutions for practical

problems from them. Moreover, the participants examine existing practitioneer's concepts with respect to their reliability. A specific aspect of this professional practice is the tactically adequate reaction to concurrency problems under tight time constraints.

content

Concurrency as a Concept

- potential parallelism
- actual parallelism
- conceptional parallelism

Concurrency in Practice

- object orientation
- operating systems
- multi-core processors, coprocessors
- programmed parallelism
- distributed systems (client-server, peer-to-peer, databases, the Internet)

Problems of Concurrency

- resource conflicts
- fairness
- mutual exclusion
- deadlock
- livelock
- starvation

Foundations of Concurrency

- sequential vs. concurrent processes
- states, events and transitions
- transition systems
- observable behaviour
- determinism vs. non-determinism
- algebras and operators

CCS - The Calculus of Communicating Systems

- constructing processes: sequence, choice, recursion
- concurrency and interaction
- structural operational semantics
- equivalence of observations
- implementation relations
- CCS with message passing

Programming Concurrency

- pseuCo
- message passing in pseuCo and Go
- shared memory in pseuCo and Java
- shared objects and threads in Java
- shared objects and threads as transition systems

Programming and Analysis Support

- deadlock detection
- verification of safety and liveness
- model-based design supporting concurrency
- software architectures supporting concurrency

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Nebenläufige Programmierung*.

Elements of Statistical Learning

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**				

responsible	
lecturers	
entrance requirements	none
assessments / exams	
course types / weekly hours	
total workload	
grade	
language	

aims / competences to be developed

content

literature & reading

Ethics for Nerds — E4N

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**occasional / summer semester**	**1 semester**	**4**	**6**

responsible Prof. Dr.-Ing. Holger Hermanns

lecturers Prof. Dr.-Ing. Holger Hermanns
Kevin Baum
Sarah Sterz

entrance requirements We expect basic knowledge of propositional and first-order logic, an open mind, and interest to look at computer science in ways you probably are not used to.

assessments / exams The details of exam admission and grading are announced at the beginning of each iteration. Typically, participant are graded based on

- an exam or a re-exam (the better mark counts),
- a short essay where the participant has to argue for or against a moral claim in a topic from computer science.

To get the exam admission, participants usually have to get 50% of the points on weekly exercise sheets.

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

(may be adjusted before the start of each iteration of the course)

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined based on exam performance, essay performance, and possibly exercise outcomes. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Many computer scientists will be confronted with morally difficult situations at some point in their career – be it in research, in business, or in industry. This module equips participants with the crucial assets enabling them to recognize such situations and to devise ways to arrive at a justified moral judgment regarding the question what one is permitted to do and what one should better not do. For that, participants will be made familiar with moral theories from philosophy, as well as different Codes of Ethics for computer scientists. Since one can quickly get lost when talking about ethics and morals, it is especially important to talk and argue clearly and precisely. In order to do prepare for that, the module offers substantial training regarding formal and informal argumentation skills enabling participants to argue beyond the level of everyday discussions at bars and parties. In the end, succesful participants are able to assess a morally controversial topic from computer science on their own and give a convincing argument for their respective assessments.

The module is intended to always be as clear, precise, and analytic as possible. What you won't find here is the meaningless bla-bla, needlessly poetic language, and vague and wordy profundity that some people tend to associate with philosophy.

content

This course covers:

- an introduction to the methods of philosophy, argumentation theory, and the basics of normative as well as applied ethics;
- relevant moral codices issued by professional associations like the ACM, the IEEE, and more;
- starting points to evaluate practices and technologies already in use or not that far away, including for instance: filter bubbles and echo chambers, ML-algorithms as predictive tools, GPS-tracking, CCTV and other tools from surveillance, fitness trackers, big data analysis, autonomous vehicles, lethal autonomous weapons systems and so on;
- an outlook on more futuristic topics like machine ethics, roboethics, and superintelligences;
- and more.

The content of the course is updated regularly to always be up-to-date and cover the currently most relevant topics, technologies, policies, and developments.

literature & reading

Will be announced before the start of the course on the course page.

Recht der Cybersicherheit – Datenschutzrechtliche Aspekte RdC-DS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**i.d.R. jedes Wintersemester**	**1 Semester**	**4**	**6**

responsible Prof. Dr. Christoph Sorge

lecturers Prof. Dr. Christoph Sorge

entrance requirements Keine

assessments / exams Abschlussklausur bzw. mündliche (Nach-)Prüfung

course types / weekly hours
```
  2 h Vorlesungen
+ 2 h Übungen
= 4 h (wöchentlich)
```

total workload
```
   60 h Präsenzstudium
+ 120 h Eigenstudium
= 180 h (= 6 ECTS)
```

grade Wird aus Leistung in Abschlussklausur bzw. Nachprüfung ermittelt.

language i.d.R. Deutsch; wird zu Beginn der Veranstaltung bekannt gegeben

aims / competences to be developed

- Erarbeitung grundlegender juristischer Methodenkenntnisse, daraus ableitend grundlegende Befähigung sich weiteres juristisches Grundlagenwissen mit Hilfe von Literatur anzueignen
- Vermittlung von Kenntnissen in rechtlichen Teilbereichen, schwerpunktmäßig im Datenschutzrecht, aber auch von einzelnen Aspekten des Urheber-, Patent- und IT-Sicherheitsrechts

content

- Grundlagen juristischer Methodik
- Einführung in das europäische Datenschutzrecht
- Grundlagen des IT-Sicherheitsrechts
- Grundlagen des Urheber- und Patentrechts

literature & reading

Bekanntgabe im Rahmen der Vorlesung, sowie auf der Website der Vorlesung.

Recht der Cybersicherheit – Strafrechtliche Aspekte RdC-SR

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**i.d.R. jedes Sommersemester**	**1 Semester**	**4**	**6**

responsible Dr. Stephanie Vogelgesang

lecturers Dr. Stephanie Vogelgesang

entrance requirements Keine

assessments / exams Abschlussklausur bzw. mündliche (Nach-)Prüfung

course types / weekly hours
```
  2 h Vorlesungen
+ 2 h Übungen
= 4 h (wöchentlich)
```

total workload
```
   60 h Präsenzstudium
+ 120 h Eigenstudium
= 180 h (= 6 ECTS)
```

grade Wird aus Leistung in Abschlussklausur bzw. Nachprüfung ermittelt.

language i.d.R. Deutsch; wird zu Beginn der Veranstaltung bekannt gegeben

aims / competences to be developed

Die Vorlesung soll Informatikern und Studierenden verwandter Fächer einen Einblick in das juristische Denken und Arbeiten geben. Neben allgemeinen Konzepten werden exemplarisch Rechtsgebiete, die für berufliche Tätigkeiten im Bereich Cybersicherheit besonders relevant sein dürften, behandelt.

Die Vorlesung dient auch der Umsetzung des Anspruchs, den die Gesellschaft für Informatik in ihren ethischen Leitlinien formuliert: „Vom Mitglied wird erwartet, dass es die einschlägigen rechtlichen Regelungen kennt, einhält und gegebenenfalls an ihrer Fortschreibung mitwirkt." Sie hat hingegen nicht den Anspruch, den Besuch von Rechtsvorlesungen zu ersetzen (etwa im Nebenfach Rechtsinformatik). Sie kann jedoch auch aufzeigen, welche Rechtsgebiete für eine Vertiefung von Interesse sein könnten und wann es sich in der Praxis lohnt oder angebracht ist, sich einen Rechtsbeistand zu besorgen.

Nach einer allgemeineren Einführung wird ein umfassender Einblick in das Strafrecht vermittelt. Neben allgemeinen strafrechtlichen Normen werden insbesondere Delikte des sogenannten „Cyberstrafrechts" betrachtet. Dabei wird ein Teil der Veranstaltung die spezifisch strafrechtliche Bewertung von Cyberangriffen darstellen. Abschließend wird das Strafprozessrecht beleuchtet (u.a. Aspekte der Beschlagnahmung und Durchsuchung).

content

- Überblick über Rechtsgebiete
- Grundlagen juristischer Methodik
- Einführung in das Strafrecht und Strafprozessrecht
- Überblick über Cyberangriffe sowie deren strafrechtliche Bewertung

literature & reading

Bekanntgabe im Rahmen der Vorlesung, sowie auf der Website der Vorlesung.

Topics in Algorithmic Data Analysis — TADA

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-5	**6**	**every summer semester**	**1 semester**	**2**	**6**

responsible Prof. Dr. Jilles Vreeken

lecturers Prof. Dr. Jilles Vreeken

entrance requirements a background in statistics, machine learning, and/or data mining is strongly recommended (e.g. *Elements of Machine Learning*, *Elements of Statistical Learning*, *Machine Learning*, or *Information Retrieval and Data Mining*)

assessments / exams oral exam and written assignments

course types / weekly hours
```
  2 h lectures
= 2 h (weekly)
```

total workload
```
   30 h of classes
+ 150 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in examinations and exercises. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

- Thorough understanding of selected advanced topics in data analysis.
- Ability to quickly understand the main gist in scientific literature, without getting lost in details, critically assessing claims, seeing through the hype.
- Ability to comparatively analyse and reason about (seemingly disparate) concepts and methods, quickly developing meta-level understanding of advanced topics.

content

During the course we consider hot topics in machine learning and data mining that are also important to understand deeply. The exact topics we will cover will differ per year, but for example often include aspects of Pattern Discovery, Dependency Discovery, Causal Inference, and Fairness.

literature & reading

Recent scientific publications from the top venues in machine learning and data mining.

Module Category 8

Bachelor's Seminar and Thesis

Bachelor's Seminar

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
6	6	every semester	1 semester	2	9

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Minimum acquisition of 120 CP.

assessments / exams
- Written formulation of the task of the bachelor's thesis and the relevant scientific literature.
- Presentation of the planned assignment with subsequent discussion
- Active participation in the discussion

course types / weekly hours 2 h seminar

total workload
```
  30 h of classes (seminar)
+ 30 h mentoring by the chair
+ 210 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from the performance in the lecture and the written report. The exact modalities will be announced by the respective instructor.

language English or German

aims / competences to be developed

In the Bachelor's seminar, the student acquires the ability to work scientifically in the context of an appropriate subject area under supervision.

At the end of the Bachelor's seminar, the foundations for the successful completion of the Bachelor's thesis are laid and essential approaches to solving the problem are already determined.

The Bachelor's seminar thus prepares the topic and execution of the Bachelor's thesis.

It also teaches practical skills of scientific discourse. These skills are taught through active participation in a reading circle, in which the discussion of scientifically challenging topics is practised.

content

Familiarisation with a scientific subject area within the field of computer science.

Preparation of a written elaboration of the task of the Bachelor thesis and the relevant scientific literature.

Presentation of the subject area and the planned task of the Bachelor's thesis.

The topic is defined in close consultation with the supervising lecturer.

literature & reading

Scientific articles appropriate to the subject area in close consultation with the supervising lecturer

Bachelor's Thesis

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
6	**6**	**every semester**	**3 months**	**-**	**12**

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Successful completion of the *Bachelor's Seminar*.

assessments / exams Written elaboration. It describes both the result of the work and the path that led to the result. The student's own contribution to the results must be clearly recognisable. In addition, presentation of the Bachelor's thesis in a colloquium, in which the independence of the student's performance is also examined.

course types / weekly hours none

total workload

```
  30 h supervision by the chair
+ 330 h private study
= 360 h (= 12 ECTS)
```

grade Assessment of the Bachelor's thesis by the reviewers.

language English or German

aims / competences to be developed

The Bachelor's thesis is a project work that is carried out under supervision. It is intended to enable the candidate to independently solve a problem from the field of computer science within a given period of time and to document the results in a scientifically appropriate form.

content

Work on a current problem from the field of computer science under supervision. Adequate documentation of the results in the form of a scientific thesis.

The topic is defined in close consultation with the instructing lecturer.

literature & reading

Scientific articles appropriate to the subject area in close consultation with the instructing lecturer.