



UNIVERSITÄT
DES
SAARLANDES

FAKULTÄT FÜR MATHEMATIK UND INFORMATIK

MODULHANDBUCH

Cybersicherheit BSc

1. Oktober 2026

Liste der Modulbereiche und Module

1 Grundlagen der Mathematik	3
1.1 Mathematik für Informatiker 1	4
1.2 Mathematik für Informatiker 2	6
2 Grundlagen der Informatik	8
2.1 Big Data Engineering	9
2.2 Elements of Machine Learning	12
2.3 Grundzüge der Theoretischen Informatik	14
2.4 Grundzüge von Algorithmen und Datenstrukturen	16
2.5 Nebenläufige Programmierung	17
2.6 Programming 1	19
2.7 Programming 2	21
2.8 Statistics Lab	23
2.9 Systemarchitektur	25
3 Praktika	27
3.1 Software Engineering Lab (Softwarepraktikum)	28
3.2 Software Engineering Lab (Softwarepraktikum) LA LAB und Sek I+II	30
4 Spezialisierter Bereich Cybersicherheit	32
4.1 Cryptography	33
4.2 Cybersecurity Project	34
4.3 Foundations of Cybersecurity 1	35
4.4 Foundations of Cybersecurity 2	36
5 Seminare der Cybersicherheit	37
5.1 Proseminar Informatik	38
5.2 Seminar Informatik	40
6 Vertiefungsvorlesungen der Cybersicherheit	42
6.1 Advanced Public Key Cryptography	43

6.2	Algorithms in Cryptanalysis	44
6.3	Automated Debugging	45
6.4	Ethics for Nerds	46
6.5	Foundations of Web Security	48
6.6	Generating Software Tests	49
6.7	Machine Learning in Cybersecurity	50
6.8	Obfuscation	51
6.9	Parameterized Verification	52
6.10	Physical-Layer Security	53
6.11	Privacy-Enhancing Technologies	55
6.12	Reactive Synthesis	57
6.13	Recht der Cybersicherheit – Datenschutzrechtliche Aspekte	58
6.14	Recht der Cybersicherheit – Strafrechtliche Aspekte	59
6.15	Reverse Engineering and Exploit Development for Embedded Systems	60
6.16	Secure Web Development	61
6.17	Side-Channels Attacks & Defenses	62
6.18	Usable Security and Privacy	63
7	Bachelor-Seminar und -Arbeit	65
7.1	Bachelor-Seminar	66
7.2	Bachelor-Arbeit	67

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
1	6	jedes Wintersemester	1 Semester	6	9

Modulverantwortliche/r Prof. Dr. Joachim Weickert

Dozent/inn/en Prof. Dr. Joachim Weickert
 Prof. Dr. Mark Groves
 Prof. Dr. Henryk Zähle
 Prof. Dr. Christian Bender

Zulassungsvoraussetzungen keine

Leistungskontrollen / Prüfungen

- Teilnahme an den Übungen und Bearbeitung der wöchentlichen Übungsaufgaben (50 Prozent der Übungspunkte werden zur Klausurteilnahme benötigt)
- Bestehen der Abschlussklausur oder der Nachklausur

Lehrveranstaltungen / SWS 4 SWS Vorlesung
 + 2 SWS Übung
 = 6 SWS

Arbeitsaufwand 90 h Präsenzstudium
 + 180 h Eigenstudium
 = 270 h (= 9 ECTS)

Modulnote Wird aus Leistungen in Klausuren, Übungen und praktischen Aufgaben ermittelt. Die genauen Modalitäten werden vom Modulverantwortlichen bekannt gegeben.

Sprache Deutsch und Englisch

Lernziele / Kompetenzen

- Erarbeitung von mathematischem Grundlagenwissen, das im Rahmen eines Informatik- bzw. Bioinformatikstudiums benötigt wird
- Fähigkeit zur Formalisierung und Abstraktion
- Befähigung zur Aneignung weiteren mathematischen Wissens mit Hilfe von Lehrbüchern

Inhalt

Die Zahlen geben die Gesamtzahl der Doppelstunden an.

DISKRETE MATHEMATIK UND EINDIMENSIONALE ANALYSIS

- A. Grundlagen der diskreten Mathematik (8)
1. Mengen (1)
 2. Logik (1)
 3. Beweisprinzipien, incl. vollst. Induktion (1)
 4. Relationen (1)
 5. Abbildungen (2)
 - injektiv, surjektiv, bijektiv
 - Mächtigkeit, Abzählbarkeit
 - Schubfachprinzip
 6. Primzahlen und Teiler (1)
 7. Modulare Arithmetik (1)

B. Eindimensionale Analysis (22)

B.1 Zahlen, Folgen und Reihen (8)

- 8. Axiomatik der reellen Zahlen, $\sup$, $\inf$ (1)
- 9. Komplexe Zahlen (1)
- 10. Folgen (1 1/2)
- 11. Landau'sche Symbole (1/2)
- 12. Reihen: Konvergenzkriterien, absolute Kgz. (2)
- 13. Potenzreihen (1/2)
- 14. Zahlendarstellungen (1/2)
- 15. Binomialkoeffizienten und Binomialreihe (1)

B.2 Eindimensionale Differentialrechnung (8)

- 16. Stetigkeit (1)
- 17. Elementare Funktionen (1)
- 18. Differenzierbarkeit (1 1/2)
- 19. Mittelwertsätze und L'Hospital (1/2)
- 20. Satz von Taylor (1)
- 21. Lokale Extrema, Konvexität, Kurvendiskussion (2)
- 22. Numerische Differentiation (1)

B.3 Eindimensionale Integralrechnung (6)

- 23. Das bestimmte Integral (2)
- 24. Das unbestimmte Integral und die Stammfunktion (1)
- 25. Uneigentliche Integrale (1)
- 26. Numerische Verfahren zur Integration (1)
- 27. Kurven und Bogenlänge (1)

Literaturhinweise

Bekanntgabe jeweils vor Beginn der Vorlesung auf der Vorlesungsseite im Internet

Weitere Informationen

Dieses Modul ist inhaltsgleich mit dem englischsprachigen Modul *Mathematics for Computer Scientists 1*.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
2	6	jedes Sommersemester	1 Semester	6	9

Modulverantwortliche/r Prof. Dr. Joachim Weickert

Dozent/inn/en Prof. Dr. Joachim Weickert
 Prof. Dr. Mark Groves
 Prof. Dr. Henryk Zähle
 Prof. Dr. Christian Bender

Zulassungsvoraussetzungen Mathematik für Informatiker 1 (empfohlen)

Leistungskontrollen / Prüfungen

- Teilnahme an den Übungen und Bearbeitung der wöchentlichen Übungsaufgaben (50 Prozent der Übungspunkte werden zur Klausurteilnahme benötigt)
- Bestehen der Abschlussklausur oder der Nachklausur

Lehrveranstaltungen / SWS 4 SWS Vorlesung
 + 2 SWS Übung
 = 6 SWS

Arbeitsaufwand 90 h Präsenzstudium
 + 180 h Eigenstudium
 = 270 h (= 9 ECTS)

Modulnote Wird aus Leistungen in Klausuren, Übungen und praktischen Aufgaben ermittelt. Die genauen Modalitäten werden vom Modulverantwortlichen bekannt gegeben.

Sprache Deutsch und Englisch

Lernziele / Kompetenzen

- Erarbeitung von mathematischem Grundlagenwissen, das im Rahmen eines Informatik- bzw. Bioinformatikstudiums benötigt wird
- Fähigkeit zur Formalisierung und Abstraktion
- Befähigung zur Aneignung weiteren mathematischen Wissens mit Hilfe von Lehrbüchern

Inhalt

Die Zahlen geben die Gesamtzahl der Doppelstunden an.

LINEARE ALGEBRA

C. Algebraische Strukturen (5)

- 29. Gruppen (2)
- 30. Ringe und Körper (1)
- 31. Polynomringe über allgemeinen Körpern (1/2)
- 32. Boole'sche Algebren (1/2)

D. Lineare Algebra (21)

- 33. Vektorräume (2)
 - Def., Bsp.,
 - lineare Abb.
 - Unterraum,
 - Erzeugnis, lineare Abhängigkeit, Basis, Austauschsatz

- 34. Lineare Abb. (Bild, Kern) (1)
- 35. Matrixschreibweise für lineare Abbildungen (1 1/2)
 - Interpretation als lineare Abbildungen
 - Multiplikation durch Hintereinanderausführung
 - Ringstruktur
 - Inverses
- 36. Rang einer Matrix (1/2)
- 37. Gauss-Algorithmus für lineare Gleichungssysteme: (2)
 - Gausselimination (1)
 - Lösungstheorie (1)
- 38. Iterative Verfahren für lineare Gleichungssysteme (1)
- 39. Determinanten (1)
- 40. Euklidische Vektorräume, Skalarprodukt (1)
- 41. Funktionalanalytische Verallgemeinerungen (1)
- 42. Orthogonalität (2)
- 43. Fourierreihen (1)
- 44. Orthogonale Matrizen (1)
- 45. Eigenwerte und Eigenvektoren (1)
- 46. Eigenwerte und Eigenvektoren symmetrischer Matrizen (1)
- 47. Quadratische Formen und positiv definite Matrizen (1)
- 48. Quadriken (1)
- 50. Matrixnormen und Eigenwertabschätzungen (1)
- 51. Numerische Berechnung von Eigenwerten und Eigenvektoren (1)

Literaturhinweise

Bekanntgabe jeweils vor Beginn der Vorlesung auf der Vorlesungsseite im Internet

Weitere Informationen

Dieses Modul ist inhaltsgleich mit dem englischsprachigen Modul *Mathematics for Computer Scientists 2*.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
4	6	jedes Sommersemester	1 Semester	4	6

Modulverantwortliche/r Prof. Dr. Jens Dittrich

Dozent/inn/en Prof. Dr. Jens Dittrich

Zulassungsvoraussetzungen *Programming 1, Programming 2, Softwarepraktikum oder Projektpraktikum, Mathematik für Informatiker 1, sowie Grundzüge von Algorithmen und Datenstrukturen (jeweils empfohlen).*

Leistungskontrollen / Prüfungen Erfolgreiche Teilnahme an den Übungen/Projekt berechtigt zur Teilnahme an der Abschlussklausur.

Lehrveranstaltungen / SWS 2 SWS Vorlesung
+ 2 SWS Übung
= 4 SWS

Arbeitsaufwand 60 h Präsenzstudium
+ 120 h Eigenstudium
= 180 h (= 6 ECTS)

Modulnote Wird aus Leistungen in Klausuren, Übungen, und ggf. Projekt ermittelt. Die genauen Modalitäten werden vom Modulverantwortlichen bekanntgegeben.

Sprache Englisch

Lernziele / Kompetenzen

Die Vorlesung vermittelt grundlegende Kenntnisse über fundamental Konzepte von Datenmanagement und Datenanalyse im Kontext von Big Data und Data Science.

Im Rahmen der Übungen kann während des Semesters ein durchgehendes Projekt durchgeführt. Dies kann zum Beispiel ein soziales Netzwerk (im Stil von Facebook) sein bzw. jedes andere Projekt, in dem Techniken des Datenmanagements eingeübt werden können (z.B. naturwissenschaftliche Daten, Bilddaten, andere Webapplikationen, etc.). Zunächst wird dieses Projekt in E/R modelliert, dann umgesetzt und implementiert in einem Datenbankschema. Danach wird das Projekt erweitert, um auch unstrukturierte Daten verwalten und analysieren zu können. Insgesamt werden so an einem einzigen Projekt alle fundamentalen Techniken gezeigt, die für das Verwalten und Analysieren von Daten wichtig sind.

Inhalt

1 Einführung und Einordnung

Einordnung und Abgrenzung: Data Science
Wert von Daten: Das Gold des 21. Jahrhunderts
Bedeutung von Datenbanksystemen
Architekturen: 2-Tier, 3-Tier, etc
Was sind eigentlich Daten?
Modellierung vs Realität
Kosten mangelhafter Modellierung
Datenbanksystem nutzen vs selbst entwickeln
Positive Beispiele für Apps
Anforderungen
Literaturhinweise
Vorlesungsmodus

- 2 Datenmodellierung
 - Motivation
 - E/R
 - Relationales Modell
 - Hierarchische Daten
 - Graphen und RDF
 - Redundanz, Normalisierung, Denormalisierung
 - Objektrelationale DBMS
- 3 Anfragesprachen
 - Relationale Algebra
 - Hierarchische Anfragesprachen
 - Graphorientierte Anfragesprachen
- 4 SQL
 - Grundlagen
 - Zusammenhang mit relationaler Algebra
 - PostgreSQL
 - Integritätsbedingungen
 - Transaktionskonzept
 - ACID
 - Sichten (und access control lists)
- 5 Implementierungstechniken
 - Übersicht
 - vom WAS zum WIE
 - Kosten verschiedener Operationen
 - EXPLAIN
 - Physisches Design
 - Indexe, Tuning
 - Datenbank-Tuning
 - Regelbasierte Anfrageoptimierung
 - Kostenbasierte Anfrageoptimierung
 - Machine Learning als Anfrageoptimierungstechnik
- 6 Zeitliche und räumliche Daten
 - als Teil des Schemas
 - as of/time travel
 - append-only und Streaming
 - Versioning
 - Snapshotting (Software und OS-basiert)
 - Differential Files/LSM et al
 - Publish/Subscribe
 - Indexstrukturen
- 7 Recovery, Durability, Archivierung
 - Grundproblematik
 - Vergessen vs Komprimieren vs Kondensieren
 - Heiße vs kalte Daten
 - Archivierung
 - Redundanz
 - Implementierungsaspekte
 - UNDO/REDO
 - Logging
- 8 Nebenläufigkeitskontrolle
 - Serialisierbarkeitstheorie
 - Isolationslevels
 - Verteilte Datenbanksysteme: Sharding, HP, VP, permissioned Blockchains
 - Implementierungsaspekte

9 ETL und Data Cleaning

- Datenbankschnittstellen: JDBC et al
- Textdatenbanken: CSV, SQLite
- Data Warehousing
- Schema Matching
- Reporting
- Data Cleaning
- Denormalisierung, Caching, Materialisierung
- Workflows
- ETL und Data Science in Data Science und Machine Learning

10 Big Data

- Was ist eigentlich Big Data?
- Big Data vs Privatheit
- Beispiele: Zusammenführen von Daten
- Physische Barrieren

11 NoSQL

- Key/Value Stores
- KeyDocument Stores: MongoDB
- MapReduce
- Flink
- Spark

Literaturhinweise

Bekanntgabe jeweils vor Beginn der Vorlesung auf der Vorlesungsseite im Internet.

Weitere Informationen

Dieses Modul wurde früher auch unter dem Namen *Informationssysteme* geführt. Dieses Modul ist inhaltsgleich mit dem englischsprachigen Modul *Big Data Engineering*.

Informationen für "Mathematik und Informatik BSc": Diese Veranstaltung ist Teil des Wahlpflicht-Bereichs der Grundlagen der Informatik (siehe Studienordnung §5 (2) 4)

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
3	6	every winter semester	1 semester	4	6

Modulverantwortliche/r Prof. Dr. Jilles Vreeken
Prof. Dr. Isabel Valera

Dozent/inn/en Prof. Dr. Jilles Vreeken
Prof. Dr. Isabel Valera

Zulassungsvoraussetzungen 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).

Leistungskontrollen / Prüfungen 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.

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

Modulnote Will be determined from performance in exams.

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

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

Literaturhinweise

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.

Weitere Informationen

Informationen für "Mathematik und Informatik BSc": Diese Veranstaltung ist Teil des Wahlpflicht-Bereichs der Grundlagen der Informatik (siehe Studienordnung §5 (2) 4)

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
3	6	jedes Wintersemester	1 Semester	6	9

Modulverantwortliche/r Prof. Dr. Raimund Seidel

Dozent/inn/en Prof. Dr. Raimund Seidel
Prof. Dr. Bernd Finkbeiner
Prof. Dr. Kurt Mehlhorn
Prof. Dr. Markus Bläser

Zulassungsvoraussetzungen *Programming 1* und *2* und *Mathematik für Informatiker 1* und *2* oder vergleichbare Veranstaltungen der Mathematik sind empfohlen.

Leistungskontrollen / Prüfungen Erfolgreiche Bearbeitung der Übungsaufgaben berechtigt zur Klausurteilnahme.

Lehrveranstaltungen / SWS 4 SWS Vorlesung
+ 2 SWS Übung
= 6 SWS

Arbeitsaufwand 90 h Präsenzstudium
+ 180 h Eigenstudium
= 270 h (= 9 ECTS)

Modulnote Wird aus Leistungen in Klausuren, Übungen und praktischen Aufgaben ermittelt. Die genauen Modalitäten werden vom Modulverantwortlichen bekannt gegeben.

Sprache Englisch

Lernziele / Kompetenzen

Die Studierenden kennen verschiedene Rechenmodelle und ihre relativen Stärken und Mächtigkeiten.

Sie können für ausgewählte Probleme zeigen, ob diese in bestimmten Rechenmodellen lösbar sind oder nicht.

Sie verstehen den formalen Begriff der Berechenbarkeit wie auch der Nicht-Berechenbarkeit.

Sie können Probleme aufeinander reduzieren.

Sie sind vertraut mit den Grundzügen der Ressourcenbeschränkung (Zeit, Platz) für Berechnungen und der sich daraus ergebenden Komplexitätstheorie.

Inhalt

Die Sprachen der Chomsky Hierarchie und ihre verschiedenen Definitionen über Grammatiken und Automaten; Abschlusseigenschaften; Klassifikation von bestimmten Sprachen („Pumping lemmas“);

Determinismus und Nicht-Determinismus;

Turing Maschinen und äquivalente Modelle von allgemeiner Berechenbarkeit (z.B. μ -rekursive Funktionen, Random Access Machines) Reduzierbarkeit, Entscheidbarkeit, Nicht-Entscheidbarkeit;

Die Komplexitätsmaße Zeit und Platz; die Komplexitätsklassen P und NP;

Grundzüge der Theorie der NP-Vollständigkeit

Literaturhinweise

Bekanntgabe jeweils vor Beginn der Vorlesung auf der Vorlesungsseite im Internet.

Weitere Informationen

Dieses Modul ist inhaltsgleich mit dem englischsprachigen Modul *Introduction to Theoretical Computer Science*.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
3	6	jedes Wintersemester	1 Semester	4	6

Modulverantwortliche/r Prof. Dr. Raimund Seidel

Dozent/inn/en Prof. Dr. Raimund Seidel
Prof. Dr. Kurt Mehlhorn
Prof. Dr. Markus Bläser

Zulassungsvoraussetzungen *Programming 1* und 2, und *Mathematik für Informatiker 1* und 2 oder vergleichbare Veranstaltungen der Mathematik sind empfohlen.

Leistungskontrollen / Prüfungen Erfolgreiche Bearbeitung der Übungsblätter berechtigt zur Klausurteilnahme.

Lehrveranstaltungen / SWS 2 SWS Vorlesung
+ 2 SWS Übung
= 4 SWS

Arbeitsaufwand 60 h Präsenzstudium
+ 120 h Eigenstudium
= 180 h (= 6 ECTS)

Modulnote Wird aus Leistungen in Klausuren, Übungen und praktischen Aufgaben ermittelt. Die genauen Modalitäten werden vom Modulverantwortlichen bekannt gegeben.

Sprache Englisch

Lernziele / Kompetenzen

Die Studierenden lernen die wichtigsten Methoden des Entwurfs von Algorithmen und Datenstrukturen kennen: Teile-und-Herrsche, Dynamische Programmierung, inkrementelle Konstruktion, „Greedy“, Dezimierung, Hierarchisierung, Randomisierung. Sie lernen Algorithmen und Datenstrukturen bzgl. Zeit- und Platzverbrauch für das übliche RAM Maschinenmodell zu analysieren und auf Basis dieser Analysen zu vergleichen. Sie lernen verschiedene Arten der Analyse (schlechtester Fall, amortisiert, erwartet) einzusetzen.

Die Studierenden lernen wichtige effiziente Datenstrukturen und Algorithmen kennen. Sie sollen die Fähigkeit erwerben, vorhandene Methoden durch theoretische Analysen und Abwägungen für ihre Verwendbarkeit in tatsächlich auftretenden Szenarien zu prüfen. Ferner sollen die Studierenden die Fähigkeit trainieren, Algorithmen und Datenstrukturen unter dem Aspekt von Performanzgarantien zu entwickeln oder anzupassen

Inhalt

Literaturhinweise

Bekanntgabe jeweils vor Beginn der Vorlesung auf der Vorlesungsseite im Internet.

Weitere Informationen

Dieses Modul ist inhaltsgleich mit dem englischsprachigen Modul *Fundamentals of Data Structures and Algorithms*.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
4	6	jedes Sommersemester	1 Semester	4	6

Modulverantwortliche/r Prof. Dr.-Ing. Holger Hermanns

Dozent/inn/en Prof. Dr.-Ing. Holger Hermanns
Prof. Bernd Finkbeiner, Ph.D
Prof. Dr. Verena Wolf

Zulassungsvoraussetzungen *Programming 1 und 2, Softwarepraktikum, und Grundzüge der Theoretischen Informatik* (empfohlen)

Leistungskontrollen / Prüfungen Zwei Klausuren (Mitte und Ende der Vorlesungszeit), praktisches Projekt.
Eine Nachklausur für die Zwischenklausur findet vor der Hauptklausur statt, eine Nachklausur für die Hauptklausur findet in den letzten Wochen vor Vorlesungsbeginn des Folgesemesters statt.

Lehrveranstaltungen / SWS 4 SWS Vorlesung
+ 2 SWS Übung
= 6 SWS

Arbeitsaufwand 50 h Präsenzstudium
+ 130 h Eigenstudium
= 180 h (= 6 ECTS)

Modulnote Wird aus Leistungen in den Klausuren, sowie den Prüfungsvorleistungen ermittelt.
Die genauen Modalitäten werden vom Modulverantwortlichen bekannt gegeben.

Sprache Englisch

Lernziele / Kompetenzen

Die Teilnehmer lernen die Nebenläufigkeit von Berechnungen als ein weitreichendes, grundlegendes Prinzip in der Theorie und Anwendung der modernen Informatik kennen. Durch die Untersuchung und Verwendung unterschiedlicher formaler Modelle gewinnen die Teilnehmer ein vertieftes Verständnis von Nebenläufigkeit. Dabei lernen die Teilnehmer wichtige formale Konzepte der Informatik korrekt anzuwenden. Das im ersten Teil der Veranstaltung erworbene theoretische Wissen wird in der zweiten Hälfte in der (Programmier-)Praxis angewendet. Dabei lernen die Teilnehmer Verwendung der Programmierparadigmen „Shared Memory“ und „Message Passing“ zuerst gemeinsam in der Programmiersprache *pseuCo*, bevor sie dann diese Fähigkeiten auf Java und teilweise Rust übertragen. Außerdem lernen die Teilnehmer verschiedene Phänomene des nebenläufigen Programmierens in den formalen Modellen zu beschreiben und mit deren Hilfe konkrete Lösungen für die Praxis abzuleiten. Des Weiteren untersuchen die Teilnehmer in der Praxis existierende Konzepte auf ihre Verlässlichkeit hin. Ein spezifischer Aspekt dieser beruflichen Praxis ist das taktisch adäquate Reagieren auf Problemstellungen der Nebenläufigkeit unter engen Zeitvorgaben.

Inhalt

Nebenläufigkeit als Konzept

- Potentieller Parallelismus
- Tatsächlicher Parallelismus
- Konzeptioneller Parallelismus

Nebenläufigkeit in der Praxis

- Objektorientierung

- Betriebssysteme
- Multi-core Prozessoren, Coprozessoren
- Programmierte Parallelität
- Verteilte Systeme (Client-Server, Peer-to-Peer, Datenbanken, Internet)

Die Schwierigkeit von Nebenläufigkeit

- Ressourcenkonflikte
- Fairness
- Gegenseitiger Ausschluss
- Verklemmung (Deadlock)
- gegenseitige Blockaden (Livelock)
- Verhungern (Starvation)

Grundlagen der Nebenläufigkeit

- Sequentielle vs. Nebenläufige Prozesse
- Zustände, Ereignisse und Transitionen
- Transitionssysteme
- Beobachtbares Verhalten
- Determinismus vs. Nicht-Determinismus
- Algebren und Operatoren

CCS: Der Kalkül kommunizierender Prozesse

- Konstruktion von Prozessen: Sequenz, Auswahl, Rekursion
- Nebenläufigkeit und Interaktion
- Strukturelle operationelle Semantik
- Gleichheit von Beobachtungen
- Implementierungsrelationen
- CCS mit Datentransfer

Programmieren von Nebenläufigkeit

- pseuCo
- Message Passing in pseuCo und Go
- Shared Memory in pseuCo und Java
- Monitore und Semaphoren
- Shared Objects und Threads in Java
- Shared Objects und Threads als Transitionssysteme

Programmier- und Analyseunterstützung

- Erkennung von Verklemmungen
- Zusicherung von Sicherheit und Lebendigkeit
- Model-Basiertes Design von Nebenläufigkeit
- Software Architekturen für Nebenläufigkeit

Literaturhinweise

Bekanntgabe jeweils vor Beginn der Vorlesung auf der Vorlesungsseite im Internet.

Weitere Informationen

Dieses Modul ist inhaltsgleich mit dem englischsprachigen Modul *Concurrent Programming*.

Informationen für "Mathematik und Informatik BSc": Diese Veranstaltung ist Teil des Wahlpflicht-Bereichs der Grundlagen der Informatik (siehe Studienordnung §5 (2) 4)

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
1	6	every winter semester	1 semester	6	9

Modulverantwortliche/r Prof. Dr.-Ing. Holger Hermanns

Dozent/inn/en Prof. Dr.-Ing. Holger Hermanns
Prof. Dr. Sebastian Hack
Prof. Dr. Martina Maggio

Zulassungsvoraussetzungen none

Leistungskontrollen / Prüfungen

- Weekly exercises and tests
- Mid-term and final exam
- Re-examination at end of the semester

Lehrveranstaltungen / SWS 4 h lectures
+ 2 h tutorial
= 6 h (weekly)

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

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

Sprache English and German

Lernziele / Kompetenzen

Programming 1 is an introduction to the foundations of computer science. Using the functional programming language OCaml, students learn to write programs, to understand precisely what programs mean, to prove them correct, and to analyse their running time. From the first day on, students work hands-on with an interpreter, which serves as a laboratory for experimenting with programs and with the programming language itself.

Upon completion of the module, students

- can write well-structured functional programs in OCaml, using recursion, tail recursion and accumulators, higher-order and polymorphic functions, pattern matching, algebraic data types, options, and exceptions;
- have acquired practical programming skills: experimenting with an interpreter, reading and reacting to type errors, systematic testing and debugging, and using the standard library;
- can design and implement recursive algorithms on numbers, lists, strings, and trees (e.g. sorting algorithms, search trees, tries, prime factorisation);
- understand static typing, polymorphism, and type inference;
- can structure programs with modules, signatures, abstract data types, representation invariants, and functors;
- can program with mutable state (references, arrays, loops, imperative data structures) and understand how data is represented in linear memory;
- can specify functions mathematically, reason about termination, and prove recursive programs correct by induction (natural, structural, and well-founded induction);
- can determine and classify the time complexity of algorithms (running-time functions, recurrences, asymptotic complexity classes);
- understand the structure of programming languages and their formal description by grammars and inference rules (abstract and concrete syntax, static and dynamic semantics);
- can implement the components of a language processor for a small functional language: lexer, a simple parser, type checker, and evaluator;
- are able to use the basic mathematical structures underlying computer science: sets, propositions, tuples, relations, directed graphs, functions, and well-founded orders.

Inhalt

see above

Literaturhinweise

Lecture notes (in English) are provided to the participants.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
2	6	every summer semester	1 semester	6	9

Modulverantwortliche/r Prof. Dr. Sebastian Hack

Dozent/inn/en Prof. Dr. Sebastian Hack
Prof. Dr. Holger Hermanns

Zulassungsvoraussetzungen *Programming 1* and *Mathematics for Computer Scientists 1* and mathematics courses in the study semester or comparable knowledge from other mathematics courses are recommended.

Leistungskontrollen / Prüfungen

- Practical part: students independently implement a series of programming projects in RISC-V assembly, C, and Java. The projects practise the language concepts of the lecture and introduce more complex algorithms and data structures. Automatic tests check the quality of the implementations.
- Lecture part: short written tests (minitests) accompanying the lecture, and a written end-of-term exam (with re-exam).
- Admission to the written exam requires successful participation in the programming projects and the minitests. The exact thresholds are announced at the beginning of the module.

Lehrveranstaltungen / SWS 4 h lectures
+ 2 h tutorial
= 6 h (weekly)

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

Modulnote Will be determined from performance in the written exams. Successful participation in the programming projects and minitests is a prerequisite for admission to the exams. The exact modalities will be announced at the beginning of the module.

Sprache English and German

Lernziele / Kompetenzen

Programming 2 introduces the foundations of imperative and object-oriented programming. The course follows the software stack from the machine level (computer arithmetic, RISC-V machine code) via the systems language C to the object-oriented language Java, and closes the loop with the formal semantics of a small imperative language and a simple compiler that translates it back to machine code. A central goal is that students can identify the different levels of abstraction at work in programming and move between them deliberately, so that they can reason about the behaviour, performance, and security of programs rather than relying on unexamined habits.

After successfully completing the module, students

- understand how computers represent numbers and execute programs, and can read, write, and debug small programs in RISC-V assembly, including function calls, the stack, and a calling convention;
- can implement, debug, and test small to mid-size C programs, and understand the C memory model, pointers, dynamic memory management, translation units, and the consequences of undefined behaviour;
- can design, implement, debug, and test mid-size Java programs using classes, interfaces, inheritance, subtyping, dynamic dispatch, encapsulation, and immutability, use the Java Collections Framework, and can judge when an object-oriented decomposition is appropriate;

- can implement fundamental data structures and algorithms (lists, trees, hash tables, dynamic programming) in an imperative language and choose suitable ones for a given programming task;
- know the formal syntax, operational semantics, and a type system of a small imperative language with pointers and scopes, and can explain type safety and undefined behaviour in these terms;
- can specify the functional correctness of programs, know the fundamental techniques and limits of testing (test oracles, black-box and white-box testing, coverage criteria, fuzzing), can prove simple imperative programs correct with Hoare logic and loop invariants, and understand how such proofs are mechanized via verification conditions;
- understand how a high-level language is translated to machine code by working through the phases of a simple compiler (syntactic analysis, type checking, syntax-directed code generation);
- have practical experience in independently completing programming projects of non-trivial size, including working with an existing code base, automatic tests, and debugging tools.

Inhalt

1. **Computer arithmetic:** positional notation, binary numbers, addition and subtraction, two's-complement integers, shifts
2. **Machine code:** the von Neumann architecture, the RV32I instruction set, the assembler, execution traces, the data segment, memory segmentation, linking, calling functions (stack, calling convention)
3. **Simple algorithms at the machine level:** number conversion, the sieve of Eratosthenes, insertion sort, evaluating polynomials with Horner's method, cycle detection (tortoise and hare)
4. **Introduction to C:** basics, execution traces, imperative variables, the C memory model, translation units, number types, control flow, expressions, calling functions, pointers, dynamic memory management, structs, input and output, undefined behaviour
5. **Algorithms and data structures:** lists, trees, hashing, dynamic programming
6. **Formal semantics:** the syntax of programming languages, abstract syntax and operational semantics of the language C0, extension by pointers (C0p) and scopes (C0b), a simple type system (C0t) and type safety
7. **Testing and verification:** functional correctness and specifications, failures, testing (the oracle problem, black-box testing, white-box testing and coverage, fuzzing), verification with Hoare logic, loop invariants, verification condition generation and mechanized verification
8. **Object-oriented programming with Java:** compilation, main method, and packages; types and reference types; classes; encapsulation; references, aliasing, and immutable objects; inheritance, subtyping, and dynamic dispatch; overriding and overloading methods; the Java Collections Framework; using object orientation properly; the expression problem and the visitor pattern; object-oriented programming in C
9. **A simple compiler:** overview of syntactic analysis, type checking, syntax-directed code generation from C0 to RISC-V

as well as lectures specifically designed for the individual programming projects (e.g. algorithmic background such as Dijkstra's algorithm for route planning or bounding volume hierarchies for ray tracing).

Literaturhinweise

Lecture notes: Sebastian Hack, *An Introduction to Imperative Programming*, <https://prog2.de/book>. Further literature will be announced before the start of the course on the course page on the Internet.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
2	6	jedes Sommersemester	1 Semester	4	6

Modulverantwortliche/r Prof. Dr. Verena Wolf
Prof. Dr. Vera Demberg

Dozent/inn/en Prof. Dr. Verena Wolf
Prof. Dr. Vera Demberg

Zulassungsvoraussetzungen keine

Leistungskontrollen / Prüfungen mündliche oder schriftliche Prüfung

Lehrveranstaltungen / SWS 2 SWS Vorlesung
+ 2 SWS Übung
= 4 SWS

Arbeitsaufwand 60 h Präsenzstudium
+ 120 h Eigenstudium
= 180 h (= 6 ECTS)

Modulnote 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.

Sprache Deutsch oder Englisch

Lernziele / Kompetenzen

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

Inhalt

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

Literaturhinweise

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
4	6	jedes Sommersemester	1 Semester	6	9

Modulverantwortliche/r Prof. Dr. Jan Reineke

Dozent/inn/en Prof. Dr. Jan Reineke

Zulassungsvoraussetzungen *Programming 1, Programming 2* (im selben Semester) und *Mathematik für Informatiker 1* oder vergleichbare Veranstaltungen der Mathematik sind empfohlen.

Leistungskontrollen / Prüfungen Prüfungsleistungen werden in zwei Teilen erbracht, die beide in die Endnote eingehen. Um die Gesamtveranstaltung zu bestehen, muss jeder Teil einzeln bestanden werden.

Im *Projektteil* müssen die Studierenden eine Reihe von Projekten selbstständig bearbeiten. Diese Projekte vertiefen das praktische Verständnis des Vorlesungsstoffes in den Bereichen Rechnerarchitektur und Betriebssysteme.

Im *Vorlesungsteil* müssen die Studierenden Klausuren absolvieren und Übungsaufgaben und/oder Minitests bearbeiten. Die erfolgreiche Bearbeitung der Übungsblätter beziehungsweise der Minitests ist Voraussetzung zur Teilnahme an der Klausur.

Lehrveranstaltungen / SWS 4 SWS Vorlesung
+ 2 SWS Übung
= 6 SWS

Arbeitsaufwand 90 h Präsenzstudium
+ 180 h Eigenstudium
= 270 h (= 9 ECTS)

Modulnote Wird aus Leistungen in Klausuren, Übungen, Minitests und praktischen Projekten ermittelt. Die genauen Modalitäten werden vom Modulverantwortlichen bekannt gegeben.

Sprache Englisch

Lernziele / Kompetenzen

Die Studierenden sollen die Funktionsweise und die wichtigsten Eigenschaften moderner Rechnerarchitekturen und Betriebssystemen kennenlernen.

Außerdem sollen die Studierenden die der Implementierung solcher Systeme zugrundeliegenden Entwurfsprinzipien verstehen.

Inhalt

1. Rechnerarchitektur
 - a. Boolesche Algebra und Schaltkreise
 - b. Zahlendarstellungen und arithmetische Schaltkreise
 - c. Befehlssatzarchitekturen
 - d. Mikroarchitekturen, insbesondere der Entwurf eines einfachen Reduced Instruction Set Computers, sowie Techniken zur Leistungsoptimierung.
2. Betriebssysteme
 - a. Virtualisierungsmechanismen
 - b. Planungsalgorithmen
 - c. Dateisysteme

Literaturhinweise

Bekanntgabe jeweils vor Beginn der Vorlesung auf der Vorlesungsseite im Internet.

Weitere Informationen

Dieses Modul ist inhaltsgleich mit dem englischsprachigen Modul *System Architecture*.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
2-3	6	lecture free time after SS	7 weeks	BLOCK	9

Modulverantwortliche/r Prof. Dr. Sven Apel

Dozent/inn/en Prof. Dr. Sven Apel
Dr. Norman Peitek

Zulassungsvoraussetzungen 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.

Leistungskontrollen / Prüfungen 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.

Lehrveranstaltungen / SWS Daily exercises and lectures (first few weeks)
Daily project work with tutoring

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

Modulnote ungraded

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- Software design
- Software testing
- Team work
- Debugging

Literaturhinweise

- Software Engineering. I. Sommerville, Addison-Wesley, 2004.
- Software Engineering: A Practitioner'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.

Weitere Informationen

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
2-3	6	lecture free time after SS	7 weeks	BLOCK	9

Modulverantwortliche/r**Dozent/inn/en**

Zulassungsvoraussetzungen Participation in the Software Engineering Lab requires extensive programming skills as taught in the courses *Programming 1* and *Programming 2*. Passing *Programming 2* is recommended before enrolling in this course.

Students are required to bring their own laptops.

Leistungskontrollen / Prüfungen 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.

Lehrveranstaltungen / SWS Daily exercises and lectures (first few weeks)
Daily project work with tutoring

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

Modulnote ungraded

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- Software design
- Software testing
- Team work
- Debugging

Literaturhinweise

- Software Engineering. I. Sommerville, Addison-Wesley, 2004.
- Software Engineering: A Practitioner'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.

Weitere Informationen

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
4	6	at least every two years	1 semester	6	9

Modulverantwortliche/r Dr. Nico Döttling

Dozent/inn/en Prof. Dr. Cas Cremers
 Dr. Nico Döttling
 Dr. Antoine Joux
 Dr. Lucjan Hanzlik
 Dr. Julian Loss

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

Leistungskontrollen / Prüfungen

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

Lehrveranstaltungen / SWS 4 h lectures
 + 2 h tutorial
 = 6 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Cybersecurity Project

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5	6	every semester	1 Semester	6	9

Modulverantwortliche/r Dr. Sven Bugiel

Dozent/inn/en Dozent/inn/en der Fachrichtung

Zulassungsvoraussetzungen Grundlegende Kenntnisse im jeweiligen Teilbereich der Informatik.

Leistungskontrollen / Prüfungen Projektarbeit, Projektdokumentation, Projektpräsentation

Lehrveranstaltungen / SWS 2 SWS Vorlesung
+ 4 SWS Praktikum
= 6 SWS

Arbeitsaufwand 30 h Präsenzstudium
+ 240 h Projektarbeit
= 270 h (= 9 ECTS)

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

Sprache Deutsch oder Englisch

Lernziele / Kompetenzen

Die Studierenden erwerben die Fähigkeit, im Team zu arbeiten und Probleme der Cybersicherheit zu lösen. Die Studierenden wissen, welche sicherheitskritischen Probleme auftreten können, und wie man damit umgeht. Sie sind vertraut mit Grundzügen der Cybersicherheit wie den grundlegenden kryptographischen Primitiven, der Schutz der Privatsphäre und der System-sicherheit, sie können Cyberangriffe erkennen und entsprechende Maßnahmen treffen.

Inhalt

Siehe Lernziele/Kompetenzen

Literaturhinweise

Die Literatur zum Modul kann englisch- und/oder deutschsprachig sein und wird zu Beginn der Veranstaltung bekanntgegeben.

Weitere Informationen

Die Unterrichtssprache ist deutsch oder englisch und wird zu Beginn der Veranstaltung bekannt gegeben.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
1	6	every winter semester	1 semester	6	9

Modulverantwortliche/r Dr. Ben Stock

Dozent/inn/en Dr. Ben Stock

Zulassungsvoraussetzungen keine

Leistungskontrollen / Prüfungen Students need to solve the exercise during the semester to be allowed to take the exam.

Lehrveranstaltungen / SWS 4 SWS Lecture
+ 2 SWS Tutorials
= 6 SWS

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

Modulnote 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.

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Weitere Informationen

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
2	6	every summer semester	1 semester	4	6

Modulverantwortliche/r Dr. Michael Schwarz

Dozent/inn/en Dr. Michael Schwarz

Zulassungsvoraussetzungen keine

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

Lehrveranstaltungen / SWS 2 SWS lectures
+ 2 SWS tutorial
= 4 SWS

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

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

Sprache Englisch

Lernziele / Kompetenzen

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

Inhalt

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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
3	6	jedes Semester	1 Semester	2	5

Modulverantwortliche/r Studiendekan der Fakultät Mathematik und Informatik
Studienbeauftragter der Informatik

Dozent/inn/en Dozent/inn/en der Fachrichtung

Zulassungsvoraussetzungen Grundlegende Kenntnisse im jeweiligen Teilbereich des Studienganges.

Leistungskontrollen / Prüfungen

- Thematischer Vortrag mit anschließender Diskussion
- Aktive Teilnahme an der Diskussion
- Gegebenenfalls kurze schriftliche Ausarbeitung und/oder Projekt

Lehrveranstaltungen / SWS 2 SWS Proseminar

Arbeitsaufwand 30 h Präsenzstudium
+ 120 h Eigenstudium
= 150 h (= 5 ECTS)

Modulnote Wird aus den Leistungen im Vortrag und der schriftlichen Ausarbeitung und/oder dem Seminarprojekt ermittelt. Die genauen Modalitäten werden von dem/der jeweiligen Dozenten/in bekannt gegeben.

Sprache Deutsch oder Englisch

Lernziele / Kompetenzen

Die Studierenden haben am Ende der Veranstaltung ein grundlegendes Verständnis aktueller oder fundamentaler Aspekte eines spezifischen Teilbereiches der Informatik erlangt.

Sie haben insbesondere grundlegende Kompetenz im eigenständigen wissenschaftlichen Recherchieren, Einordnen, Zusammenfassen, Diskutieren, Kritisieren und Präsentieren von wissenschaftlichen Erkenntnissen gewonnen.

Im Vergleich zum Seminar liegt der Fokus beim Proseminar auf der Aneignung der grundlegenden wissenschaftlichen Arbeitsweisen.

Inhalt

Unter Anleitung werden folgende Punkte praktisch geübt:

- Lesen und Verstehen wissenschaftlicher Arbeiten
- Diskutieren der Arbeiten in der Gruppe
- Analysieren, Zusammenfassen und Wiedergeben des spezifischen Themas
- Präsentationstechnik

Spezifische Vertiefung in Bezug auf das individuelle Thema des Seminars.

Der typische Ablauf eines Proseminars ist üblicherweise wie folgt:

- Vorbereitende Gespräche zur Themenauswahl
- Regelmäßige Treffen mit Diskussion ausgewählter Beiträge
- ggf. Bearbeitung eines themenbegleitenden Projekts
- Vortrag und ggf. Ausarbeitung zu einem der Beiträge

Literaturhinweise

Material wird dem Thema entsprechend ausgewählt.

Weitere Informationen

Die jeweils zur Verfügung stehenden Proseminare werden vor Beginn des Semesters angekündigt und unterscheiden sich je nach Studiengang.

Seminar Informatik

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5	6	jedes Semester	1 Semester	2	7

Modulverantwortliche/r Studiendekan der Fakultät Mathematik und Informatik
Studienbeauftragter der Informatik

Dozent/inn/en Dozent/inn/en der Fachrichtung

Zulassungsvoraussetzungen Grundlegende Kenntnisse im jeweiligen Teilbereich des Studienganges.

Leistungskontrollen / Prüfungen

- Thematischer Vortrag mit anschließender Diskussion
- Aktive Teilnahme an der Diskussion
- Gegebenenfalls schriftliche Ausarbeitung oder Projekt

Lehrveranstaltungen / SWS 2 SWS Seminar

Arbeitsaufwand 30 h Präsenzstudium
+ 180 h Eigenstudium
= 210 h (= 7 ECTS)

Modulnote Wird aus den Leistungen im Vortrag und der schriftlichen Ausarbeitung und/oder dem Seminarprojekt ermittelt. Die genauen Modalitäten werden von dem/der jeweiligen Dozenten/in bekannt gegeben.

Sprache Deutsch oder Englisch

Lernziele / Kompetenzen

Die Studierenden haben am Ende der Veranstaltung vor allem ein tiefes Verständnis aktueller oder fundamentaler Aspekte eines spezifischen Teilbereiches der Informatik erlangt.

Sie haben weitere Kompetenz im eigenständigen wissenschaftlichen Recherchieren, Einordnen, Zusammenfassen, Diskutieren, Kritisieren und Präsentieren von wissenschaftlichen Erkenntnissen gewonnen.

Inhalt

Weitgehend selbstständiges Erarbeiten des Seminarthemas:

- Lesen und Verstehen wissenschaftlicher Arbeiten
- Analyse und Bewertung wissenschaftlicher Aufsätze
- Diskutieren der Arbeiten in der Gruppe
- Analysieren, Zusammenfassen und Wiedergeben des spezifischen Themas
- Erarbeiten gemeinsamer Standards für wissenschaftliches Arbeit
- Präsentationstechnik

Spezifische Vertiefung in Bezug auf das individuelle Thema des Seminars.

Der typische Ablauf eines Seminars ist üblicherweise wie folgt:

- Vorbereitende Gespräche zur Themenauswahl
- Regelmäßige Treffen mit Diskussion ausgewählter Beiträge
- ggf. Bearbeitung eines themenbegleitenden Projekts
- Vortrag und ggf. Ausarbeitung zu einem der Beiträge

Literaturhinweise

Material wird dem Thema entsprechend ausgewählt.

Weitere Informationen

Die jeweils zur Verfügung stehenden Seminare werden vor Beginn des Semesters angekündigt und unterscheiden sich je nach Studiengang.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Nico Döttling

Dozent/inn/en Dr. Nico Döttling

Zulassungsvoraussetzungen Cryptography

Leistungskontrollen / Prüfungen Mündliche Prüfung oder Abschlussklausur

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Algorithms in Cryptanalysis

Studiensem.

5-6

Regelst.sem.

6

Turnus

occasional

Dauer

1 semester

SWS

4

ECTS

6

Modulverantwortliche/r Dr. Antoine Joux

Dozent/inn/en Dr. Antoine Joux

Zulassungsvoraussetzungen Good working knowledge of algebra and algorithms

Leistungskontrollen / Prüfungen Written exam.

Lehrveranstaltungen / SWS

Arbeitsaufwand

Modulnote Determined by the performance in exams.

Sprache

Lernziele / Kompetenzen

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.

Inhalt

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.

Literaturhinweise

Automated Debugging

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	at least every two years	1 semester	4	6

Modulverantwortliche/r Prof. Dr. Andreas Zeller

Dozent/inn/en Prof. Dr. Andreas Zeller

Zulassungsvoraussetzungen *Programming 1, Programming 2 and Softwarepraktikum*

Leistungskontrollen / Prüfungen Projects and mini-tests

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional / summer semester	1 semester	4	6

Modulverantwortliche/r Prof. Dr.-Ing. Holger Hermanns

Dozent/inn/en Prof. Dr.-Ing. Holger Hermanns
Kevin Baum
Sarah Sterz

Zulassungsvoraussetzungen 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.

Leistungskontrollen / Prüfungen 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.

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Modulnote 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.

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

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.

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Ben Stock

Dozent/inn/en Dr. Ben Stock

Zulassungsvoraussetzungen *Security or Foundations of Cybersecurity 1 and 2*

Leistungskontrollen / Prüfungen Projekt und schriftliche Abschlussklausur

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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)

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Prof. Dr. Andreas Zeller

Dozent/inn/en Prof. Dr. Andreas Zeller

Zulassungsvoraussetzungen Programming 1, Programming 2, Softwarepraktikum

Leistungskontrollen / Prüfungen Projekte und Mini-Tests

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Prof. Dr. Mario Fritz

Dozent/inn/en Prof. Dr. Mario Fritz

Zulassungsvoraussetzungen *Data Science/Statistics Course*

Leistungskontrollen / Prüfungen Übungen, Projekt und mündliche Prüfung

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Obfuscation

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	usually every year	1 semester	4	6

Modulverantwortliche/r Dr. Nico Döttling

Dozent/inn/en Dr. Nico Döttling

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

Leistungskontrollen / Prüfungen Passing a usually oral exam

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

Modulnote Determined by the performance in exams

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

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.

Literaturhinweise

Parameterized Verification

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Swen Jacobs

Dozent/inn/en Dr. Swen Jacobs

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

Leistungskontrollen / Prüfungen Passing a written exam (re-exam can be oral, if any)

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

Modulnote Determined by the performance in exams

Sprache

Lernziele / Kompetenzen

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.

Inhalt

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.

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Nils-Ole Tippenhauer

Dozent/inn/en Dr. Nils-Ole Tippenhauer

Zulassungsvoraussetzungen *Security or Foundations of Cyber Security I + II*

Leistungskontrollen / Prüfungen Übungen und schriftliche Abschlussklausur

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

- 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

Inhalt

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

Literaturhinweise

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

Weitere Informationen

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).

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Wouter Lueks

Dozent/inn/en Dr. Wouter Lueks

Zulassungsvoraussetzungen 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.

Leistungskontrollen / Prüfungen

- Programming projects
- Final exam (written or oral)
- Midterm

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial and office hours
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Swen Jacobs

Dozent/inn/en Dr. Swen Jacobs

Zulassungsvoraussetzungen *Grundzüge der Theoretischen Informatik*

Leistungskontrollen / Prüfungen Projekt und schriftliche Abschlussklausur

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	i.d.R. jedes Wintersemester	1 Semester	4	6

Modulverantwortliche/r Prof. Dr. Christoph Sorge

Dozent/inn/en Prof. Dr. Christoph Sorge

Zulassungsvoraussetzungen Keine

Leistungskontrollen / Prüfungen Abschlussklausur bzw. mündliche (Nach-)Prüfung

Lehrveranstaltungen / SWS 2 h Vorlesungen
+ 2 h Übungen
= 4 h (wöchentlich)

Arbeitsaufwand 60 h Präsenzstudium
+ 120 h Eigenstudium
= 180 h (= 6 ECTS)

Modulnote Wird aus Leistung in Abschlussklausur bzw. Nachprüfung ermittelt.

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

Lernziele / Kompetenzen

- 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

Inhalt

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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	i.d.R. jedes Sommersemester	1 Semester	4	6

Modulverantwortliche/r Dr. Stephanie Vogelgesang

Dozent/inn/en Dr. Stephanie Vogelgesang

Zulassungsvoraussetzungen Keine

Leistungskontrollen / Prüfungen Abschlussklausur bzw. mündliche (Nach-)Prüfung

Lehrveranstaltungen / SWS 2 h Vorlesungen
+ 2 h Übungen
= 4 h (wöchentlich)

Arbeitsaufwand 60 h Präsenzstudium
+ 120 h Eigenstudium
= 180 h (= 6 ECTS)

Modulnote Wird aus Leistung in Abschlussklausur bzw. Nachprüfung ermittelt.

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

Lernziele / Kompetenzen

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).

Inhalt

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

Literaturhinweise

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

Reverse Engineering and Exploit Development for Embedded Systems RExp4Em-Sys

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	2 weeks	BLOCK	6

Modulverantwortliche/r Dr. Ali Abbasi

Dozent/inn/en Dr. Ali Abbasi

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

Leistungskontrollen / Prüfungen

- 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.

Lehrveranstaltungen / SWS Daily lectures followed by daily tutorial and exercises

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

Modulnote 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.

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Nils-Ole Tippenhauer

Dozent/inn/en Dr. Nils-Ole Tippenhauer

Zulassungsvoraussetzungen keine

Leistungskontrollen / Prüfungen Projekt und schriftliche Abschlussklausur

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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)

Literaturhinweise

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

Weitere Informationen

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	occasional	1 semester	4	6

Modulverantwortliche/r Dr. Michael Schwarz

Dozent/inn/en Dr. Michael Schwarz

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

Leistungskontrollen / Prüfungen project and written exam

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

- 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

Literaturhinweise

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

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
5-6	6	every summer semester	1 semester	4	6

Modulverantwortliche/r Dr. Katharina Krombholz

Dozent/inn/en Dr. Katharina Krombholz

Zulassungsvoraussetzungen *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.

Leistungskontrollen / Prüfungen

- 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

Lehrveranstaltungen / SWS 2 h lectures
+ 2 h tutorial or office hours
= 4 h (weekly)

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

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

Sprache English

Lernziele / Kompetenzen

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.

Inhalt

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

Literaturhinweise

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

Weitere Informationen

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*.

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
6	6	jedes Semester	variabel	2	9

Modulverantwortliche/r Studiendekan der Fakultät Mathematik und Informatik
Studienbeauftragter der Informatik

Dozent/inn/en Dozent/inn/en der Fachrichtung

Zulassungsvoraussetzungen Erwerb von mindestens 120 CP

Leistungskontrollen / Prüfungen

- Schriftliche Ausarbeitung der Aufgabenstellung der Bachelorarbeit und der relevanten wissenschaftlichen Literatur
- Vortrag über die geplante Aufgabenstellung mit anschliessender Diskussion
- Aktive Teilnahme an der Diskussion

Lehrveranstaltungen / SWS 2 SWS Seminar

Arbeitsaufwand

- 30 h Präsenzstudium (Seminar)
- + 30 h Betreuung durch den Lehrstuhl
- + 210 h Eigenstudium
- = 270 h (= 9 ECTS)

Modulnote Wird aus den Leistungen im Vortrag und der schriftlichen Ausarbeitung ermittelt. Die genauen Modalitäten werden von dem/der jeweiligen Dozenten/in bekannt gegeben.

Sprache Deutsch oder Englisch

Lernziele / Kompetenzen

Im Bachelorseminar erwirbt der Studierende unter Anleitung die Fähigkeit zum wissenschaftlichen Arbeiten im Kontext eines angemessenen Themengebietes.

Am Ende des Bachelorseminars sind die Grundlagen für eine erfolgreiche Anfertigung der Bachelorarbeit gelegt, und wesentliche Lösungsansätze bereits eruiert.

Das Bachelorseminar bereitet somit die Themenstellung und Ausführung der Bachelorarbeit vor.

Es vermittelt darüber hinaus praktische Fähigkeiten des wissenschaftlichen Diskurses. Diese Fähigkeiten werden durch die aktive Teilnahme an einem Lesekreis vermittelt, in welchem die Auseinandersetzung mit wissenschaftlich anspruchsvollen Themen geübt wird.

Inhalt

Einarbeitung in ein wissenschaftliches Themengebiet innerhalb der Informatik.

Anfertigung einer schriftlichen Ausarbeitung der Aufgabenstellung der Bachelorarbeit und der relevanten wissenschaftlichen Literatur.

Fachvortrag über das Themengebiet und die geplante Aufgabenstellung der Bachelorarbeit.

Das Thema wird in enger Absprache mit dem anleitenden Dozenten definiert.

Literaturhinweise

Dem Themengebiet entsprechende wissenschaftliche Artikel in enger Absprache mit dem Dozenten

Bachelor-Arbeit

Studiensem.	Regelst.sem.	Turnus	Dauer	SWS	ECTS
6	6	jedes Semester	3 Monate	-	12

Modulverantwortliche/r Studiendekan der Fakultät Mathematik und Informatik
Studienbeauftragter der Informatik

Dozent/inn/en Professoren der Fachrichtung

Zulassungsvoraussetzungen Erfolgreicher Abschluss des *Bachelor-Seminars*

Leistungskontrollen / Prüfungen Schriftliche Ausarbeitung. Sie beschreibt sowohl das Ergebnis der Arbeit als auch den Weg, der zu dem Ergebnis führte. Der eigene Anteil an den Ergebnissen muss klar erkennbar sein. Außerdem Präsentation der Bachelorarbeit in einem Kolloquium, in dem auch die Eigenständigkeit der Leistung des Studierenden überprüft wird.

Lehrveranstaltungen / SWS keine

Arbeitsaufwand 30 h Betreuung durch den Lehrstuhl
+ 330 h Eigenstudium
= 360 h (= 12 ECTS)

Modulnote Beurteilung der Bachelorarbeit durch die Gutachter.

Sprache Deutsch oder Englisch

Lernziele / Kompetenzen

Die Bachelor-Arbeit ist eine Projektarbeit, die unter Anleitung ausgeführt wird. Sie soll der Kandidaten/die Kandidatin in der Lage versetzen, innerhalb einer vorgegebenen Frist ein Problem aus dem Gebiet der Informatik selbständig zu lösen und die Ergebnisse in wissenschaftlich angemessener Form zu dokumentieren.

Inhalt

Bearbeitung einer aktuellen Problemstellung aus der Informatik unter Anleitung. Adäquate Dokumentation der Ergebnisse in Form einer wissenschaftlichen Abschlussarbeit.

Das Thema wird in enger Absprache mit dem anleitenden Dozenten definiert.

Literaturhinweise

Dem Themengebiet entsprechende wissenschaftliche Artikel in enger Absprache mit dem anleitenden Dozenten.