Security (SEC)

Preamble

Computing supports nearly every facet of modern critical infrastructure: transportation, communication, healthcare, education, energy generation and distribution, to name a few. With rampant attacks on and breaches of this infrastructure, computer science graduates have an important role in designing, implementing, and operating software systems that are robust, safe, and secure.

The Security (SEC) knowledge area focuses on developing a *security mindset* into the overall ethos of computer science graduates so that security is embedded in all their work products. Computer science students need to learn about system vulnerabilities and understand threats against computer systems. The *Security* title choice was intentional to serve as a one-word umbrella term for this knowledge area, which also includes concepts to support privacy, cryptography, secure systems, secure data, and secure code.

The SEC knowledge area relies on shared concepts pervasive in all the other areas of CS2023. It identifies seven crosscutting concepts of cybersecurity: *confidentiality*, *integrity*, *availability*, *risk assessment*, *systems thinking*, *adversarial thinking*, and *human-centered thinking*. The seventh concept, *human-centered thinking*, is additional to the six crosscutting concepts originally defined in the Cybersecurity Curricula 2017 (CSEC2017) [1]. This addition reinforces to students that humans are also a link in the overall chain of security, a theme that is also covered in knowledge areas such as HCI. Principles of protecting systems (also in the DM, OS, SDF, SE and SF knowledge areas) include security-by-design, privacy-by-design, defense-in-depth, and zero-trust.

Another concept is the notion of assurance, which is an attestation that security mechanisms need to comply with the security policies that have been defined for data, processes, and systems. Assurance is tied in with the concepts of verification and validation in the SE knowledge area. Considerations of data privacy and security are shared with the DM (technical aspects) and SEP knowledge areas.

The SEC knowledge area thus sits atop several of the other CS2023 knowledge areas, while including additional concepts that are not present in those knowledge areas. The specific dependence on other knowledge areas is stated below, starting with the Core Hours table. CS2023 treats security as a crucial component of the skillset of any CS graduate, and the hours needed for security preparation come from all the other 16 CS2023 knowledge areas.

Changes since CS2013

The Security knowledge area is an updated name for CS2013's Information Assurance and Security (IAS) knowledge area. Since 2013, Information Assurance and Security has been rebranded as Cybersecurity, which has become a new computing discipline, with its own curricular guidelines (CSEC 2017) developed by a Joint Task Force of the ACM, IEEE Computer Society, AIS and IFIP in 2017.

Moreover, since 2013, other curricular recommendations for cybersecurity beyond CS2013 and CSEC 2017 have been made. In the US, the National Security Agency recognizes institutions as Centers of

Academic Excellence (CAE) in Cyber Defense and/or Cyber Operations if their cybersecurity programs meet the respective CAE curriculum requirements. Additionally, the National Initiative for Cybersecurity Education (NICE) of the US National Institute for Standards and Technologies (NIST) has developed and revised the Workforce Framework for Cybersecurity (NICE Workforce Framework), which identifies competencies (knowledge and skills) needed to perform tasks relevant to cybersecurity work roles. The European Cybersecurity Skills Framework (ECSF) includes a standard ontology to describe cybersecurity tasks and roles, as well as addressing the cybersecurity personnel shortage in EU member countries. Similarities and differences of these cybersecurity guidelines, viewed from the CS perspective, also informed the SEC knowledge area.

Building on CS2013's recognition of the pervasiveness of security in computer science, the CS2023 SEC knowledge area focuses on ensuring that students develop a *security mindset* so that they are prepared for the continual changes occurring in computing. One useful addition is the knowledge unit for security analysis, design, and engineering to support the concepts of security-by-design and privacy-by-design.

The importance of computer science in ensuring the protection of future computing systems and societal critical infrastructure will continue to grow. Consequently, it is imperative that faculty teaching computer science incorporate the latest advances in security and privacy approaches to keep their curriculum current.

Differences between CS2023 Security knowledge area and Cybersecurity

CS2023's SEC knowledge area focuses on those aspects of security, privacy, and related concepts important for computer science students. In comparison, CSEC 2017 characterizes similarities and differences in the cybersecurity book of knowledge using the disciplinary lenses of computer science, computer engineering, software engineering, information systems, information technology, and other disciplines. In short, the major goal of the SEC knowledge area is to ensure that computer science graduates can design and develop more secure code, ensure data security and privacy, and apply a security mindset to their daily activities.

Protecting what happens *within* the perimeter of a networked computer system is a core competency of computer science graduates. Although the computer science and cybersecurity knowledge units overlap, the demands upon cybersecurity graduates typically are to protect the perimeter. CSEC 2017 defines cybersecurity as a highly interdisciplinary field of study that covers eight areas (data, software, component, connection, system, human, organizational, and societal security) and prepares its students for both technical and managerial roles in cybersecurity.

The first five CSEC 2017 areas are technical and have overlaps with the CS2023 SEC knowledge area, but the intent of coverage is substantively different as computer science students bring to bear the core competencies described in all the 17 CS2023 knowledge areas. For instance, consider the SEC knowledge area's Secure Coding knowledge unit. The computer science student will need to view this knowledge unit from a computer science lens, as an extension of the material covered in the SDF, SE, and PDC knowledge areas, while the Cybersecurity student will need to view software security in the overall context of diverse cybersecurity goals. These viewpoints are not totally distinct and have

overlaps, but the lenses used to examine and present the content are different. There are similar commonalities and differences among CS2023 SEC knowledge units and corresponding CSEC 2017 knowledge units.

Core Hours

Knowledge Unit	CS Core	KA Core
Foundational Security	1 + 7 (DM, FPL, PDC, SDF, SE, OS)	7
Society, Ethics, and the Profession	1 + 4 (SEP)	2
Secure Coding	2 + 6 (FPL, SDF, SE)	5
Cryptography	1 + 8 (MSF)	4
Security Analysis, Design, and Engineering	1 + 4 (MSF, SE)	8
Digital Forensics	0	6
Security Governance	0	3
Total hours	**6**	**35**

The SEC knowledge area requires approximately 28 hours of CS Core hours from the other knowledge areas, either to provide the basis or to complement its content. Of these, MSF-Discrete, MSF-Probability, and MSF-Statistics are likely to be relied upon extensively in all the SEC knowledge units, as are SDF-Fundamentals, SDF-Algorithms, and SDF-Practices. The others are mentioned within each of the SEC knowledge units described below.

Knowledge Units

SEC-Foundations: Foundational Security

CS Core:

1. Developing a security mindset incorporating crosscutting concepts: confidentiality, integrity, availability, risk assessment, systems thinking, adversarial thinking, human-centered thinking
2. Basic concepts of authentication and authorization/access control
3. Vulnerabilities, threats, attack surfaces, and attack vectors (See also: OS-Protection)
4. Denial of Service (DoS) and Distributed Denial of Service (DDoS) (See also: OS-Protection)
5. Principles and practices of protection, e.g., least privilege, open design, fail-safe defaults, defense in depth, and zero trust; and how they can be implemented (See also: OS-Principles, OS-Protection, SE-Construction, SEP-Security)

6. Optimization considerations between security, privacy, performance, and other design goals (See also: SDF-Practices, SE-Validation, HCI-Design)
7. Impact of AI on security and privacy: using AI to bolster defenses as well as address increased adversarial capabilities due to AI (See also: AI-SEP, HCI-Design, HCI-SEP)

KA Core:

8. Access control models (e.g., discretionary, mandatory, role-based, and attribute-based)
9. Security controls
10. Concepts of trust and trustworthiness
11. Applications of a security mindset: web, cloud, and mobile devices (See also: SF-System Design, SPD-Common)
12. Protecting embedded and cyber-physical systems (See also: SPD-Embedded)
13. Principles of usable security and human-centered computing (See also: HCI-Design, SEP-Security)
14. Security and trust in AI/machine learning systems, e.g., fit for purpose, ethical operating boundaries, authoritative knowledge sources, verified training data, repeatable system evaluation tests, system attestation, independent validation/certification; unintended consequences from: adverse effect (See also: AI-Introduction, AI-ML, AI-SEP, SEP-Security)
15. Security risks in building and operating AI/machine learning systems (e.g., algorithm bias, knowledge corpus bias, training corpus bias, copyright violation) (See also: AI-Introduction, AI-ML, AI-SEP)
16. Hardware considerations in security, e.g., principles of secure hardware, secure processor architectures, cryptographic acceleration, compartmentalization, software-hardware interaction (See also: AR-Assembly, AR-Representation, OS-Purpose)

Illustrative Learning Outcomes:

CS Core:

1. Evaluate a system for possible attacks that can be launched by an adversary.
2. Design and develop approaches to protect a system from a set of identified threats.

KA Core:

3. Describe how harm to user privacy can be avoided.
4. Develop a system that incorporates various principles of security and privacy.
5. Compare the different access control models in terms of functionality and performance.
6. Show how an adversary could use machine learning algorithms to reduce the security of a system.
7. Show how a developer could improve the security of a system using machine learning algorithms.
8. Describe hardware (especially CPU) vulnerabilities that can impact software.

SEC-SEP: Society, Ethics, and the Profession

CS Core:

1. Principles and practices of privacy (See also: SEP-Security)
2. Societal impacts on breakdowns in security and privacy (See also: SEP-Context, SEP-Privacy, SEP-Security)
3. Applicability of laws and regulations on security and privacy (See also: SEP-Security)
4. Professional ethical considerations when designing secure systems and maintaining privacy; ethical hacking (See also: SEP-Professional-Ethics, SEP-Privacy, SEP-Security)

***KA-Core*:**

5. Security by design (See also: SF-Security, SF-Design)
6. Privacy by design and privacy engineering (See also: SEP-Privacy, SEP-Security)
7. Security and privacy implications of malicious AI/machine learning actors, e.g., identifying deep fakes (See also: AI-Introduction, AI-ML, SEP-Privacy, SEP-Security)
8. Societal impacts of Internet of Things (IoT) devices and other emerging technologies on security and privacy (See also: SEP-Privacy, SEP-Security)

Illustrative Learning Outcomes:

CS Core:

1. Calculate the impact of a breakdown in security of a given system.
2. Construct a system that conforms to security laws.
3. Apply a set of privacy regulations to design a system that protects privacy.

KA Core:

4. Evaluate the legal ramifications of a system not corresponding to applicable laws and regulations.
5. Construct a system that is designed to avoid harm to user privacy.

SEC-Coding: Secure Coding

CS Core:

1. Common vulnerabilities and weaknesses
2. SQL injection and other injection attacks
3. Cross-site scripting techniques and mitigations
4. Input validation and data sanitization (See also: OS-Protection, SDF-Fundamentals, SE-Validation)
5. Type safety and type-safe languages (See also: FPL-Types, FPL-Systems, OS-Protection, SDF-Fundamentals, SE-Validation)
6. Buffer overflows, stack smashing, and integer overflows (See also: AR-Assembly, FPL-Systems, OS-Protection)
7. Security issues due to race conditions (See also: FPL-Parallel, PDC-Evaluation)

KA Core:

8. Principles of noninterference and nondeducibility
9. Preventing information flow attacks
10. Offensive security techniques as a defense
11. AI-assisted malware detection techniques
12. Ransomware: creation, prevention, and mitigation
13. Secure use of third-party components (See also: SE-Construction, SE-Validation)
14. Malware: varieties, creation, reverse engineering, and defense against them (See also: FPL-Systems, FPL-Translation)
15. Assurance: testing (including fuzzing and penetration testing), verification, and validation (See also: OS-Protection, SDF-Fundamentals, SE-Construction, SE-Validation)
16. Static and dynamic analyses (See also: FPL-Analysis, MSF-Protection, PDC-Evaluation, SE-Validation)
17. Secure compilers and secure code generation (See also: FPL-Runtime, FPL-Translation)

Illustrative Learning Outcomes:

CS Core:

1. Identify underlying problems in given examples of an enumeration of common weaknesses and explain how they can be circumvented.
2. Apply input validation and data sanitization techniques to enhance security of a program.
3. Describe how the selection of a programming language can impact the security of the system being constructed.
4. Rewrite a program in a type-safe language (e.g., Java or Rust) originally written in an unsafe programming language (e.g., C/C++).
5. Evaluate a program for possible buffer overflow attacks and rewrite to prevent such attacks.
6. Evaluate a set of related programs for possible race conditions and prevent an adversary from exploiting them.
7. Evaluate and prevent SQL injections attacks on a database application.
8. Evaluate and prevent cross-site scripting attacks against a website.

KA Core:

9. Describe different kinds of malicious software.
10. Construct a program that tests for all input handling errors.
11. Explain the risks of misusing interfaces with third-party code and how to correctly use third-party code.
12. Discuss the need to update software to fix security vulnerabilities and the lifecycle management of the fix.
13. Construct a system that is protected from unauthorized information flows.
14. Apply static and dynamic tools to identify programming faults.
15. Evaluate a system for the existence of malware and remove it.
16. Implement preventive techniques to reduce the occurrence of ransomware.

SEC-Crypto: Cryptography

CS Core:

1. Differences between algorithmic, applied, and mathematical views of cryptography
2. Mathematical preliminaries: modular arithmetic, Euclidean algorithm, probabilistic independence, linear algebra basics, number theory, finite fields, complexity, asymptotic analysis (See also: MSF-Discrete, MSF-Linear)
3. Basic cryptography: symmetric key and public key cryptography (See also: AL-Foundational, MSF-Discrete)
4. Basic cryptographic building blocks, including symmetric encryption, asymmetric encryption, hashing, and message authentication (See also: MSF-Discrete)
5. Classical cryptosystems, such as shift, substitution, transposition ciphers, code books, and machines (See also: MSF-Discrete)
6. Kerckhoff's principle and use of vetted libraries (See also: SE-Construction)
7. Usage of cryptography in real-world applications, e.g., electronic cash, secure channels between clients and servers, secure electronic mail, entity authentication, device pairing, steganography, and voting systems (See also: NC-Security, GIT-Image)

KA Core:

8. Additional mathematics: primality, factoring, and elliptic curve cryptography (See also: MSF-Discrete)

9. Private-key cryptosystems: substitution-permutation networks, linear cryptanalysis, differential cryptanalysis, DES, and AES (See also: MSF-Discrete, NC-Security)
10. Public-key cryptosystems: Diffie-Hellman and RSA (See also: MSF-Discrete)
11. Data integrity and authentication: hashing, and digital signatures (See also: MSF-Discrete, DM-Security)
12. Cryptographic protocols: challenge-response authentication, zero-knowledge protocols, commitment, oblivious transfer, secure two- or multi-party computation, hash functions, secret sharing, and applications (See also: MSF-Discrete)
13. Attacker capabilities: chosen-message attack (for signatures), birthday attacks, side channel attacks, and fault injection attacks (See also: NC-Security)
14. Quantum cryptography; Post Quantum/Quantum resistant cryptography (See also: AL-Foundational, MSF-Discrete)
15. Blockchain and cryptocurrencies (See also: MSF-Discrete, PDF-Communication)

Illustrative Learning Outcomes:

CS Core:

1. Explain the role of cryptography in supporting security and privacy.
2. Discuss the risks of inventing one's own cryptographic methods.
3. Discuss the importance of prime numbers in cryptography and explain their use in cryptographic algorithms.
4. Implement and cryptanalyze classical ciphers.

KA Core:

5. Describe how crypto keys can be managed securely.
6. Compare the space and time performance of a given set of cryptographic methods.
7. Discuss how modern private-key cryptosystems work and ways to cryptanalyze them.
8. Discuss how modern public-key cryptosystems work and ways to cryptanalyze them.
9. Compare different cryptographic algorithms in terms of security.
10. Explain key exchange protocols and show approaches to reduce their failure.
11. Describe real-world applications of cryptographic primitives and protocols.
12. Discuss how quantum cryptography works and the impact of quantum computing on cryptographic algorithms.

SEC-Engineering: Security Analysis, Design, and Engineering

CS Core:

1. Security engineering goals: building systems that remain dependable despite errors, accidents, or malicious adversaries (See also: SE-Construction, SE-Validation, SEP-Security)
2. Privacy engineering goals: building systems that design, implement, and deploy privacy features and controls (See also: SEP-Privacy)
3. Problem analysis and situational analysis to address system security (See also: SE-Validation)
4. Engineering tradeoff analysis based on time, cost, risk tolerance, risk acceptance, return on investment, and so on (See also: PDC-Evaluation, SE-Validation)

KA Core:

5. Security design and engineering, including functional requirements, security subsystems, information protection, security testing, security assessment, and evaluation (See also: PDC-Evaluation, SE-Requirements, SE-Validation)
6. Security analysis, covering security requirements analysis; security controls analysis; threat analysis; and vulnerability analysis (See also: FPL-Analysis, PDC-Evaluation)
7. Security attack domains and attack surfaces, e.g., communications and networking, hardware, physical, social engineering, software, and supply chain (See also: NC-Security)
8. Security attack modes, techniques, and tactics, e.g., authentication abuse; brute force; buffer manipulation; code injection; content insertion; denial of service; eavesdropping; function bypass; impersonation; integrity attack; interception; phishing; protocol analysis; privilege abuse; spoofing; and traffic injection (See also: NC-Security, OS-Protection, SE-Validation)
9. Attestation of software products with respect to their specification and adaptiveness (See also: SE-Requirements, SE-Validation)
10. Design and development of cyber-physical systems
11. Considerations for trustworthy computing, e.g., tamper resistant packaging, trusted boot, trusted kernel, hardware root of trust, software signing and verification, hardware-based cryptography, virtualization, and containers (See also: SE-Construction, SE-Validation)

Illustrative Learning Outcomes:

CS Core:

1. Create a threat model for a system or system design.
2. Apply situational analysis to develop secure solutions under a specified scenario.
3. Evaluate a given scenario for tradeoff analysis for system performance, risk assessment, and costs.

KA Core:

4. Design a set of technical security controls, countermeasures, and information protections to meet the security requirements and security objectives for a system.
5. Evaluate the effectiveness of security functions, technical controls, and componentry for a system.
6. Identify and mitigate security vulnerabilities and weaknesses in a system.
7. Evaluate and predict emergent behavior in areas such as Data Science, AI, and Machine Learning.

SEC-Forensics: Digital Forensics

KA Core:

1. Basic principles and methodologies for digital forensics
2. System design for forensics
3. Forensics in different situations: operating systems, file systems, application forensics, web forensics, network forensics, mobile device forensics, use of database auditing (See also: NC-Security)
4. Attacks on forensics and preventing such attacks
5. Incident handling processes
6. Rules of evidence – general concepts and differences between jurisdictions (See also: SEP-Security)
7. Legal issues: digital evidence protection and management, chains of custody, reporting, serving as an expert witness (See also: SEP-Security)

Illustrative Learning Outcomes:

KA Core:

1. Explain what a digital investigation is and how it can be implemented (See also: SEP-Security)
2. Design and implement software to support forensics.
3. Describe legal requirements for using seized data and its usage. (See also: SEP-Security)
4. Describe and implement an end-to-end chain of custody from initial digital evidence seizure to evidence disposal. (See also: SEP-Privacy, SEP-Security)
5. Extract data from a hard drive to comply with the law (See also: SEP-Security)
6. Discuss a person's professional responsibilities and liabilities when testifying as a forensics expert (See also: SEP-Professional-Ethics)
7. Recover data based on a given search term from an imaged system
8. Reconstruct data and events from an application history, or a web artifact, or a cloud database, or a mobile device. (See also: SPD-Mobile, SPD-Web)
9. Capture and analyze network traffic. (See also: NC-Security)
10. Develop approaches to address the challenges associated with mobile device forensics.
11. Apply forensics tools to investigate security breaches.
12. Identify and mitigate anti-forensic methods.

SEC-Governance: Security Governance

KA Core:

1. Protecting critical assets from threats
2. Security governance: organizational objectives and general risk assessment
3. Security management: achieve and maintain appropriate levels of confidentiality, integrity, availability, accountability, authenticity, and reliability (See also: SE-Validation)
4. Security policy: organizational policies, issue-specific policies, system-specific policies
5. Approaches to identifying and mitigating risks to computing infrastructure
6. Data lifecycle management policies: data collection, backups, and retention; cloud storage and services; breach disclosure (See also: DM-Security)

Illustrative Learning Outcomes:

KA Core:

1. Describe critical assets and how they can be protected.
2. Differentiate between security governance, management, and controls, giving examples of each.
3. Describe a technical control and implement it to mitigate specific threats.
4. Identify and assess risk of programs and database applications causing breaches.
5. Design and implement appropriate backup strategies conforming to a given policy.
6. Discuss a breach disclosure policy based on legal requirements and implement the policy.
7. Identify the risks and benefits of outsourcing to the cloud.

Professional Dispositions

- **Meticulous:** students need to pay careful attention to details to ensure the protection of real-world software systems.

- **Self-directed**: students must be ready to deal with the many novel and easily unforeseeable ways in which adversaries might launch attacks.
- **Collaborative**: students must be ready to collaborate with others, as collective knowledge and skills will be needed to prevent attacks, protect systems and data during attacks, and plan for the future after the immediate attack has been mitigated.
- **Responsible:** students need to show responsibility when designing, developing, deploying, and maintaining secure systems, as their enterprise and society is constantly at risk.
- **Accountable**: students need to know that as future professionals they will be held accountable if a system or data breach were to occur, which should strengthen their resolve to prevent such breaches from occurring in the first place.

Mathematics Requirements

Required:

- MSF-Discrete
- MSF-Probability
- MSF-Statistics

Desired:

- MSF-Linear

Course Packaging Suggestions

There are two suggestions for course packaging, along with an additional suggestion for a more advanced course.

The first suggestion for course packaging is to infuse the CS Core hours of the SEC KA into appropriate places in other coursework that covers related security topics in the following knowledge units. As the CS Core Hours of the SEC KA are only 6 hours, coursework covering one or more of the following knowledge units could accommodate them.

- AI-SEP
- AL-SEP
- AR-Assembly
- AR-Memory
- DM-Security
- FPL-Translation
- FPL-Run-Time
- FPL-Analysis
- FPL-Types
- HCI-Design
- HCI-Accountability
- HCI-SEP

- NC-Security
- OS-Protection
- PDC-Communication
- PDC-Coordination
- PDC-Evaluation
- SDF-Fundamentals
- SDF-Practices
- SE-Validation
- SEP-Privacy
- SEP-Security
- SF-Design
- SF-Security
- SPD-Common
- SPD-Mobile
- SPD-Web

The second approach for course packaging is to create an additional full course focused on security that packages the following, building on the topics already covered in other knowledge areas.

Fundamentals of Computer Security:

- SEC-Foundations (6 hours)
- SEC-SEP (4 hours)
- SEC-Coding (7 hours)
- SEC-Crypto (5 hours)
- SEC-Engineering (4 hours)
- SEC-Forensics (2 hours)
- SEC-Governance (1 hour)
- AI-SEP (1 hour)
- AR-Assembly (1 hour)
- AR-Memory (1 hour)
- DM-Security (3 hours)
- FPL-Translation (1 hour)
- FPL-Run-Time (1 hour)
- FPL-Analysis (1 hour)
- FPL-Types (2 hours)
- HCI-Design (1 hour)
- HCI-Accountability (1 hour)
- HCI-SEP (1 hour)
- NC-Security (2 hours)
- OS-Protection (1 hour)
- PDC-Communication (1 hour)
- PDC-Coordination (1 hour)
- PDC-Evaluation (1 hour)
- SDF-Fundamentals (1 hour)

- SDF-Practices (1 hour)
- SE-Validation: (2 hours)
- SEP-Privacy (1 hour)
- SEP-Security (2 hours)
- SF-Design (2 hours)
- SF-Security (2 hours)
- SPD-Common (2 hours)
- SPD-Mobile (2 hours)
- SPD-Web: Web Platforms (2 hours)

The coverage exceeds 45 lecture hours, and so, in a typical course, instructors would need to decide what topics to emphasize and what not to cover without losing the perspective that the course should help students develop a *security mindset*.

Prerequisites: Depends on the selected topics, but appropriate coursework covering MSF, SDF, and SE knowledge areas is needed.

Course objectives: Students should develop a security mindset and be ready to apply this mindset to securing data, software, systems, and applications.

A third suggested packaging is to create an advanced course that develops a security architect/engineer's view by including the following:

Security Engineering:

- SEC-Foundations (6 hours)
- SEC-SEP (4 hours)
- SEC-Coding (6 hours)
- SEC-Crypto (2 hours)
- SEC-Engineering (10 hours)
- SEC-Forensics (2 hours)
- SEC-Governance (1 hour)
- DM-Security (2 hours)
- NC-Security (3 hours)
- OS-Protection (2 hours)
- PDC-Evaluation (2 hours)
- SDF-Fundamentals (1 hour)
- SDF-Practices (1 hour)
- SE-Validation (2 hours)
- SEP-Privacy (1 hour)
- SEP-Security (1 hour)
- SF-Design (2 hours)
- SF-Security (2 hours)
- SPD-Mobile (2 hours)
- SPD-Web (2 hours)

The coverage for all topics is over 45 lecture hours, and so instructors would need to decide what topics to emphasize and what not to cover without losing the perspective that the course should help students develop the security engineer's mindset. Laboratory time related to data and network security, web platform, secure coding and validation would be valuable aspects of this course.

Prerequisites: Depends on the selected topics, either the first or second packaging suggested above would be recommended based on degree program needs.

Course objectives: Computer science students should develop the mindset of a security engineer and be ready to apply this mindset to problems in designing and evaluating the security of a range of computing systems and information services.

Committee

Chair: Rajendra K. Raj, Rochester Institute of Technology, Rochester, NY, USA

Members:

- Vijay Anand, University of Missouri – St. Louis, St. Louis, MO, USA
- Diana Burley, American University, Washington, DC, USA
- Sherif Hazem, Central Bank of Egypt, Cairo, Egypt
- Michele Maasberg, United States Naval Academy, Annapolis, MD, USA
- Bruce McMillin, Missouri University of Science and Technology, Rolla, MO, USA
- Sumita Mishra, Rochester Institute of Technology, Rochester, NY, USA
- Nicolas Sklavos, University of Patras, Patras, Greece
- Blair Taylor, Towson University, Towson, MD, USA
- Jim Whitmore, Dickinson College, Carlisle, PA, USA

Contributors:

- Markus Geissler, Cosumnes River College, Sacramento, CA, USA
- Michael Huang, Rider University, Lawrenceville, NJ, USA
- Tim Preuss, Minnesota State Community and Technical College, Moorhead, MN, USA
- Daniel Zappala, Brigham Young University, Provo, UT, USA

References

1. Joint Task Force on Cybersecurity Education. 2017. Cybersecurity Curricula 2017. ACM, IEEE-CS, AIS SIGSEC, and IFIP WG 11.8. https://doi.org/10.1145/3184594