

Rethinking the Role of Security in Undergraduate Education

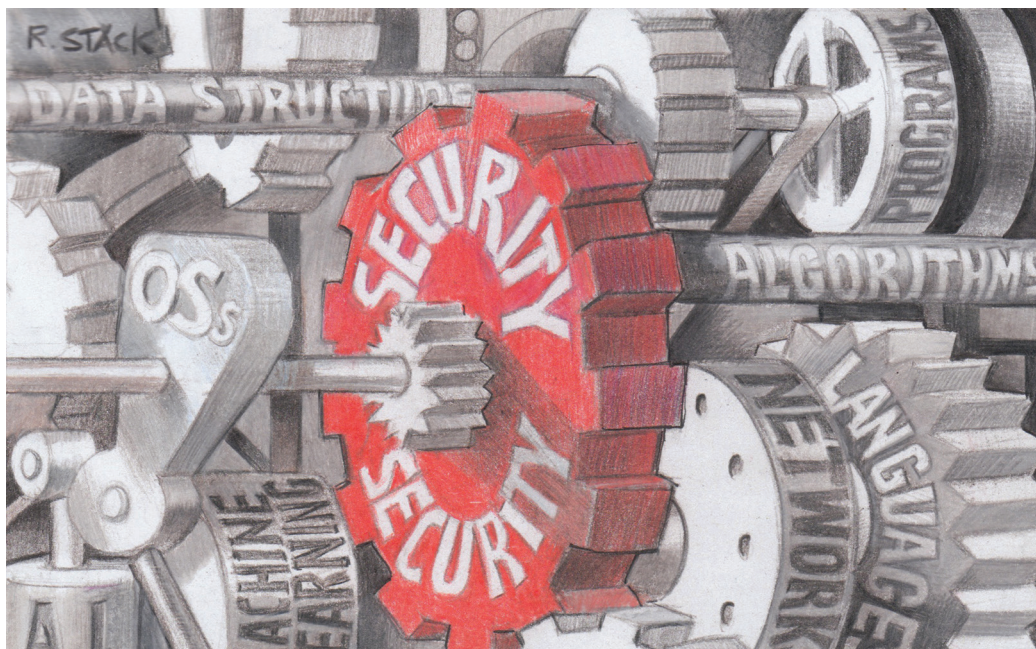
Sarah Zatzko | Cyber Independent Testing Laboratory

Security is largely an afterthought in software design and development. Because programmers often lack a baseline understanding of security, they don't always build security into their design and development process. Not surprisingly, the result is insecure code and systems with avoidable bugs and vulnerabilities that developers must patch.

The “security as an afterthought” mindset is a natural consequence of the way in which security is taught. Although security is integral to just about every computer science topic, curricula tend to treat it separately, often making it an advanced elective for interested seniors or graduate students. It's too easy for today's undergraduates to complete a computer science degree without learning about the security concepts relevant to their specialty.

This process is backward. How a subject is taught influences how its students think about it. There's been a major push to develop specialized security content for future security experts. Such work is valuable, but all computer science students should learn at least the basics of security. Rather than addressing security after the foundations of computer science are laid, curricula should integrate it into areas such as networking, programming languages, and OSs. Even introductory courses need applied security content.

Drawing on a survey of computer science department heads, I



present data on security's current visibility in general undergraduate curricula and the degree to which it's integrated into core classes.

Security in Today's Curricula

I sent a survey to the heads of 100 university computer science departments to determine

- the degree to which their undergraduate core curricula prioritized and integrated security content, and
- how applied the content was.

Because the survey had to be brief and address multiple measures,

I enlisted a Google data scientist's help in designing the survey to maximize response rate and reliability. Thirty-three department heads responded.

Priority Relative to Other Subjects

I ascertained security's ranking as a curriculum topic in two ways. First, the department heads reviewed a list of 10 core curriculum subjects. They then picked the five they believed were most important to include in an undergraduate curriculum. Only six of the 33 respondents chose security, resulting in a ninth-place ranking, just above compilers (see Figure 1).

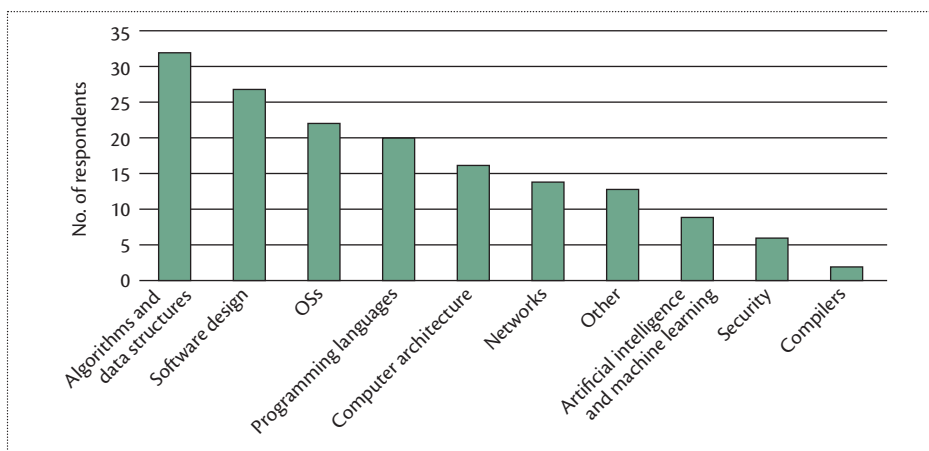


Figure 1. After reviewing a list of 10 computer science subjects, department heads chose which five were most important to include in an undergraduate core curriculum. Only six respondents included security.

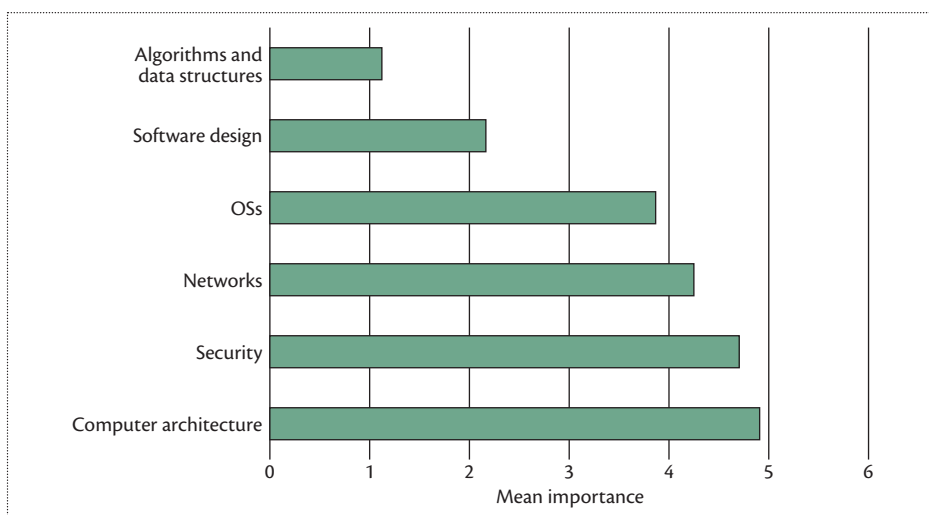


Figure 2. Ranking of the importance of six computer science subjects (1 = most important, and 6 = least important). Security ranked fifth.

Second, the department heads examined a list of six computer science subjects, which they ranked from most (1) to least (6) important. Security received a mean ranking of 4.7, making it the fifth most important subject (see Figure 2).

As these results show, security has a very low priority in most computer science curricula.

Integration into Other Subjects

The results indicate that when security is taught as a separate subject,

it's not highly prioritized in the overall curriculum design. But security is an integral facet of all other computing subjects—and could be treated as such. Thus, I asked the department heads to rate their university's curriculum in terms of how separate or integrated security was, with 1 being completely separate and 5 being totally integrated.

As Figure 3 illustrates, security was usually taught separately. Only six department heads reported that security was completely or mostly integrated in their university's

curriculum; 22 said that it was completely or mostly separate.

Theory versus Application

The data is even more telling when I factor in whether security content is theoretical or applied. The department heads rated how applied their university's content was for several subjects (with 1 = theoretical, and 5 = applied). Figure 4 plots the mean values of this rating against the previous integration ratings. Although the applied-versus-theoretical ratings tended to be moderate, security content clearly became less applied as it became more integrated.

The department heads also rated security as one of the most theoretical subjects; only algorithms and data structures was considered more theoretical. Theory is, of course, useful. But if computer science students aren't required to put theory into practice, how will they know how to practice good security in their careers?

Repetition helps students learn behaviors that don't come naturally. Otherwise, they quickly forget or omit these behaviors from their routines. For example, students review their essays for grammatical errors or pick the correct proof type for a given problem because these behaviors have been consistently reinforced. To become reflexive among graduates entering the workforce, good security practices must be emphasized and applied consistently. In neglecting this part of computer science education, we're doing students a vast disservice.

Unquestionably, the other computer science topics in the survey are important. Replacing one of them with a security course wouldn't make sense, because most universities have a separate, more advanced security course for students interested in specializing in that field. What's missing is meaningful, applied security content that's integrated into all the other core courses.

The Example of Environmental Engineering

Environmental engineering and its role in other engineering disciplines demonstrate how such integration can be achieved. Security and environmental engineering have similar origins. In both cases, a particular class of problems was recognized, and then a discipline was created to address those problems. Because environmental engineering has been around longer than security, engineers have worked through most of the “growing pains.” Thus, following environmental engineering’s lead could accelerate the integration of security into computer science curricula.

Like security, environmental engineering began as a separate field but eventually had to be folded back into the other engineering disciplines. Pretty much any type of engineering—be it mechanical, chemical, civil, and so on—requires grounding in environmental design. Without this background, environmental considerations become an afterthought, and easily avoidable issues become difficult and expensive to address after the products have reached the public. All engineers need enough environmental engineering knowledge to be able to evaluate their own designs.

The parallels with security are obvious. Security is a facet of just about every computer science subject and must be integrated into them. Computer scientists need enough security knowledge to evaluate the security of their own designs and to recognize situations requiring further expertise. Otherwise, security experts will find themselves patching the same mistakes repeatedly.

The change in environmental design’s treatment is due to multiple factors, all of which have direct analogs in computer security.

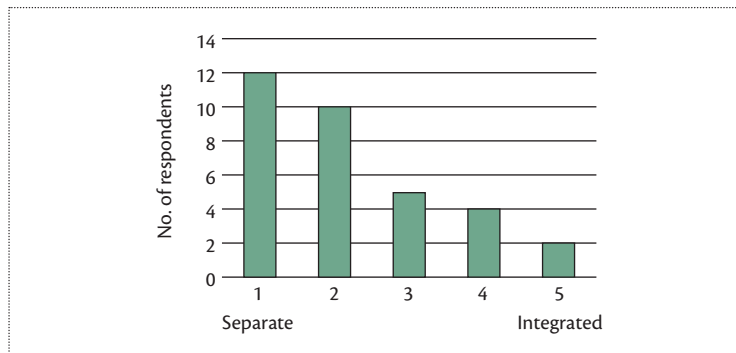


Figure 3. Integration of security in the core curriculum. Most department heads reported that security was taught completely or mostly separately.

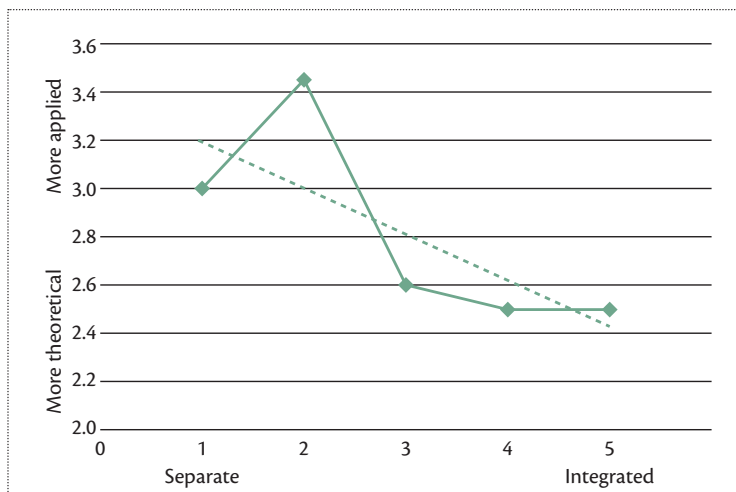


Figure 4. Rating of security content’s application versus integration. Content tended to become more theoretical with increased integration. The dashed line shows this linear trend.

Legislative Pressure and Changing Industry Standards

Various governing bodies worldwide have enacted legislation that regulates the environmental aspects of commercial practices and products. For example, in the EU, Directive 2006/66/EC regulates batteries’ manufacture and disposal, and Directive 2002/95/EC restricts the manufacture of products containing lead, mercury, and several other environmentally harmful substances.^{1,2}

As environmental legislation becomes more prevalent, engineers without the knowledge or skills to make compliant designs are increasingly a liability to employers. Thus,

some understanding of environmental design is now the industry standard. This, in turn, has put pressure on educators to incorporate environmental design content into other engineering disciplines so that their students have the knowledge necessary to be marketable in the workforce.

Similar regulatory pressures are affecting computer science. In the US, the Federal Trade Commission successfully sued Wyndham Hotels for negligent security practices, and 47 of the 50 states have passed data breach notification laws.^{3,4} Given the poor state of commercial security practices, we expect more attempts to enforce good security habits through

regulation. Of course, it's difficult to write accurate security legislation, so it would be preferable to migrate to better practices voluntarily.

Many major US companies have reported large data breaches in recent years. In 2014 alone, there were at least 19 breaches involving 30,000 or more customers.⁵ In some cases, the breach was quite costly for the companies involved. For example, Target's reported breach in 2013 resulted in major financial losses.⁶ If this trend continues, financial and legal pressures should cause industry standards to change.

Accreditation

In its "Criteria for Accrediting Engineering Programs," the Accreditation Board for Engineering and Technology (ABET) requires "an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability."⁷ Although not all programs are accredited, these criteria set an industry standard.

Intriguingly, ABET's "Criteria for Accrediting Computing Programs" starts out similarly but lacks the "within realistic constraints" clause.⁸ This omission is troubling. Not only are security and privacy not mentioned, but the constraints from the engineering list are missing as well. This implies that computer science students are held to a much more lenient standard than engineering students. Could this be contributing to computer and network security's current state? Most computer science graduates will enter the commercial world, working on systems that have constraints and implications beyond the strict programming specifications listed. The costly security and privacy failures that regularly make the news prove that computer science students must be more aware

of the realistic concerns constraining their work.

Why is computer science held to a lesser standard than engineering? Perhaps the ABET criteria for computer science were designed to adhere more closely to physical science criteria. This might have been appropriate in the past, when computer science was more theoretical. But most of today's computer science students will be working in situations more similar to engineering than to science. Thus, they must be equipped to design systems that work in the real world. And having computer security knowledge and experience will certainly not hurt students who later choose science-oriented paths.

ACM Guideline

In 2012, the ACM issued a new Curriculum Guideline Report that introduced information assurance and security as a knowledge area.⁹ To limit the core curriculum's growth, ACM integrated security content into existing courses. For example, it introduced practical security concepts such as input testing and sanitization, defensive coding, phishing, and privacy.

In the ACM curriculum guideline, many topics are included at the "familiarity" rather than the "usage" level, which means that students aren't getting hands-on experience. It's essential that topics such as the principles of secure design and coding be taught in the core curriculum, not as electives. If not, the majority of students who need these skills won't learn them. Along the same lines, the ability to find and remove vulnerabilities from existing code should be considered a core skill. Ideally, this skill will become less important as security becomes less of an afterthought.

Curriculum Recommendations

Integrating computer security into other courses doesn't require drastic

curriculum redesign. In fact, only minor changes might be required. However, it's important that the changes be comprehensive and consistent. Security must be introduced early and as part of every course. All topics should be presented in conjunction with the security grounding that teaches students how and when to use the skills they're learning.

Introducing a security concept once isn't enough. Only frequent repetition will lead to reflexive behaviors. Following are a few examples of how minimal changes to existing course material can weave security principles into general computer science topics. These types of changes can have a high impact on student understanding without being onerous to implement.

Networking Protocols

Whenever networking protocols are discussed in class, the conversation usually focuses on the protocols' features and capabilities. Although it's important to understand how protocols are intended to work, it's also important to understand how they fail: their blind spots, limits, and vulnerabilities. This gives students a realistic understanding of how to use protocols in real-world systems. More fundamental, understanding how something breaks or fails is critical to understanding how it was designed. Figure 5 shows a homework problem about Border Gateway Protocol (BGP) from the University of California, San Diego. The students, who were told that the nodes used a shortest-path algorithm, had to answer the following questions:

- What path would E take to reach B?
- What path would F take to reach C?
- Suppose autonomous system (AS) 3 and AS4 were in a corporate rivalry. Using only your knowledge of BGP, how could AS3 make sure none of its traffic traverses AS4?

These questions were designed to make students demonstrate their understanding of BGP in an ideal environment. But missing are questions that would make the students think about BGP’s limitations:

- Suppose AS1 wants to hijack traffic destined to AS7. What would it advertise?
- If people see their own AS listed in a path, they will disregard it as a loop. How could this be used maliciously?

Adding such changes throughout the course would grant students a more well-rounded understanding of networking concepts as well as a greater ability to use those tools and concepts effectively and securely. Students would learn to ask the right security questions when familiarizing themselves with a new system.

Algorithms and Worst-Case Runtimes

The second example comes from a University of Montana algorithms class. Students were given the simple hash function shown in Figure 6, which takes a letter *k* and computes $h(k) = k \bmod 13$. This function builds a hash table from user-supplied data.

The students were asked initially about worst-case runtimes. But the assignment’s wording could be changed subtly to highlight the situation’s security:

- If attackers wanted to slow down operations, what is the longest runtime their denial-of-service attack could force?
- Give an example of an input that would achieve this.
- How could you change the hash function to avoid this vulnerability?

Any worst-case runtime is also a potential attack vector. Rather than presenting worst cases as the result

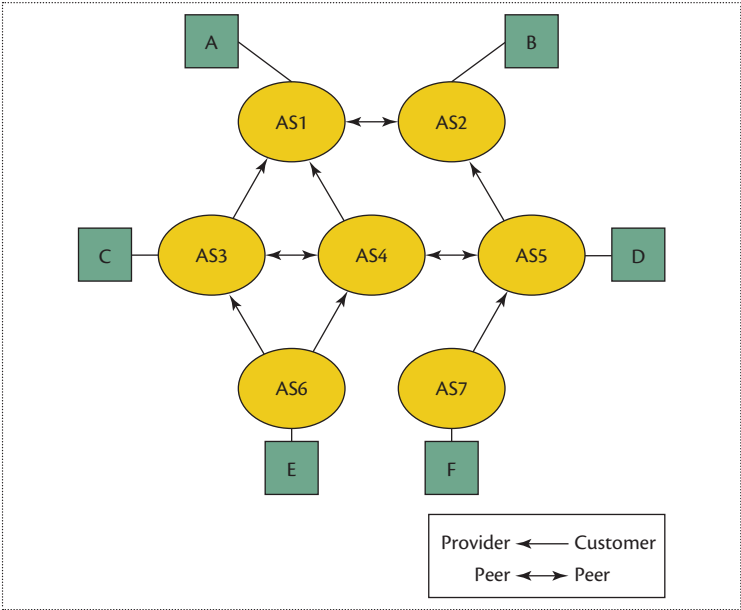


Figure 5. Network diagram from a Border Gateway Protocol homework problem from the University of California, San Diego. AS is autonomous system.

Key	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
mod 13	0	1	2	3	4	5	6	7	8	9	10	11	12	0	1	2	3	4	5	6	7	8	9	10	11	12

Figure 6. Simple hash function to be applied to letters of the alphabet, from a University of Montana algorithm course. $h(k) = k \bmod 13$.

of bad luck, instructors should stress that malicious users can construct bad inputs purposely for denial-of-service attacks.

If every discussion of worst-case runtimes emphasizes security risks, students will be trained to recognize similar situations later in life. The first step in secure design is recognizing which situations are riskier than others and require more careful consideration. A simple change in wording could go a long way toward developing students’ awareness early on.

Programming Assignments

A few schools teach defensive coding practices. Teaching defensive coding as an isolated exercise is better than not teaching it at all, but isn’t going to make defensive coding a habit.

In most programming courses, at least one or two assignments will

require students to perform input validation. However, they aren’t necessarily expected to continue doing so after the assignments are completed. Thus, students might conclude that input validation is useful to know but not something they need to think about consistently.

Students’ programming assignments, whether security focused or not, should be subjected to input fuzzing, with possible points taken off for segmentation faults, crashes, or other instances of lack of responsible program behavior for hostile environments.

Without penalties for bad coding practices in school, students won’t learn the lessons they need to program successfully in their careers. What students should learn is that security and robustness are default considerations—part of the design process from the beginning.

Goals and Next Steps

Such curriculum changes aim to achieve the following outcomes:

- Students can evaluate their designs' security.
- Students can recognize when they need further security expertise.
- Students can describe security issues to other professionals and security specialists.

This last point is particularly important. Too often, computer science professionals recognize that security should be a bigger priority in their design, but they lack the security knowledge to articulate the problem to others. At best, this makes it hard to seek out the right kind of expertise. At worst, it makes them unable to distinguish a qualified security professional from a charlatan.

These curriculum changes are necessary, but several challenges must first be addressed. Most pressing, not all computer science professors understand their fields' security issues. Teaching materials and other aids must be developed to enable professors to teach these concepts. I hope to partner with universities to develop new core computer science courses that integrate security concepts.

Security is a mindset—one that undergraduates must learn. Although it's important to provide dedicated offerings for students who are going to specialize in security, all students need fundamental grounding in the topic. Adding a security course for nonspecialists merely increases their already packed course load. Integrating applied, practical security content into other courses is a more logical and effective way to ensure that students gain the security knowledge they'll need later in life.

The changes required to fold security into existing courses need not be drastic. What's important is that the concepts are repeated throughout courses, so students understand that computer and information security must always be considered, not just saved for special occasions. Curriculum changes can be subtle, but they must be consistent and sweeping to instill the proper mindsets, instincts, and reflexes.

Some security situations will still require an expert's skill. But if programmers know what security questions to ask, they'll be better able to recognize such situations, find the specialist they need, and avoid costly security problems in the future. ■

Acknowledgments


My thanks to the professors who participated in the survey, to Neal Patel for his help in writing the survey questions, and to Bruce Schneier for his help with editing. *The Psychology of Survey Response* was a valuable reference for designing the survey questions.¹⁰

References

1. Directive 2006/66/EC of the European Parliament and of the Council of 6 September 2006 on Batteries and Accumulators and Repealing Directive 91/157/EEC (Text with EEA relevance), OJ L 266, 26 Sept. 2006, pp. 1–14.
2. Directive 2002/95/EC of the European Parliament and of the Council of 27 January 2003 on the Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment, OJ L 37, 13 Feb. 2003, pp. 19–23.
3. J. Greenwald, "FTC's Cyber Security Win against Wyndham May Lead to More Enforcement Actions," *Business Insurance*, 15 Dec. 2015; www.businessinsurance.com/article/20151215/NEWS06/151219906.

4. K. Alonso and A. Leopard, "Commonwealth of Kentucky Enacts Data Breach Notification Law," *Inside Counsel*, 26 June 2014; www.insidecounsel.com/2014/06/26/commonwealth-of-kentucky-enacts-data-breach-notifi.
5. D. McCandless, "World's Biggest Data Breaches," *Information Is Beautiful*, 2 Oct. 2015; www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks.
6. E. Harris, "Data Breach Hurts Profit at Target," *New York Times*, 26 Feb. 2014; www.nytimes.com/2014/02/27/business/target-reports-on-fourth-quarter-earnings.html?_r=0.
7. "Criteria for Accrediting Engineering Programs," Accreditation Board for Engineering and Technology, 26 Oct. 2013; www.abet.org/wp-content/uploads/2015/04/E001-14-15-EAC-Criteria.pdf.
8. "Criteria for Accrediting Computing Programs," Accreditation Board for Engineering and Technology, 26 Oct. 2013; www.abet.org/wp-content/uploads/2015/04/C001-14-15-CAC-Criteria.pdf.
9. M. Sahami et al., "Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science," Joint Task Force on Computing Curricula, ACM/IEEE Computer Society, 20 Dec. 2013; www.acm.org/education/CS2013-final-report.pdf.
10. R. Tourangeau et al., *The Psychology of Survey Response*, Cambridge Univ. Press, 2000.

Sarah Zatzko is chief scientist at the Cyber Independent Testing Laboratory. Contact her at ZatzkoSarah@gmail.com.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.