4

Engineering at Scale and User Implications

Workshop participants next tackled various aspects of cryptographic agility in practice. Matthew Green of Johns Hopkins University offered a case study of benefits and drawbacks of agility mechanisms in the Transport Layer Security (TLS) protocol. Google's Adam Langley explained agility in the context of extensibility, touched on the international context, and explored how retiring outdated security mechanisms affects users. Sara "Scout" Sinclair Brody of Simply Secure offered a perspective informed by the field of user-centered design, explaining how security and agility can be enhanced by better defaults, communication, education, and transparency—not only for end users, but also for developers themselves.

TRANSPORT LAYER SECURITY AND THE DOWNSIDES OF AGILITY

Matthew Green, Johns Hopkins University

Matthew Green is an assistant professor at the Johns Hopkins Information Security Institute and also writes a well-known cryptography blog titled "A Few Thoughts on Cryptographic Engineering."[1] Like other speakers at the workshop, he offered examples of the benefits of agility (which he defined as having the ability in place to react quickly and

[1]The website for Green's blog is http://blog.cryptographyengineering.com/, accessed November 18, 2016.

appropriately to change), but he also highlighted some drawbacks. He used the frequent attacks on the TLS protocol as a case study.

When Agility “Saves the Day”

Green provided examples of how agility mechanisms have saved TLS over the years. Today, he said, “We are so used to TLS breaks . . . It has become a joke in our community.” But when the Browser Exploit Against SSL/TLS (BEAST) attack on TLS was first documented in 2011, it was “the beginning of an era.” In response to the attack, most users took advantage of the protocol’s built-in agility to move over to the Rivest Cipher 4 (RC4) stream cipher. However, RC4 was later found to be itself insecure and therefore a poor defense against BEAST attacks. In addition, the BEAST designers introduced another vulnerability called Compression Ratio Info-leak Made Easy (CRIME), which went after the preprocessing compression instead of attacking algorithms directly. Unfortunately, although cryptographers were theoretically aware of this weakness, those deploying and building servers did not know about it, and CRIME was able to decrypt TLS communication fairly quickly. When CRIME became known, it was contained by turning off compression on most TLS servers: “Agility saves the day,” Green summarized.

Newer, more secure cryptography platforms in subsequent versions of TLS brought mixed results. Cipher suites, authenticated encryptions, fast stream ciphers, and fast authenticated stream ciphers were all developed to increase security. However, Green explained, “it turns out that there were applications that we had not really considered.” RC4 was fast but not necessarily secure; to attain something fast *and* secure, designers are now looking at nonstandard cipher suites such as ChaCha20 and Poly1305 and considering whether it would be worth pursuing standardization of these suites by the Internet Engineering Task Force (IETF).

Agility as “Our Achilles’ Heel”

Green then turned to what he views as the drawbacks to agility, starting with the caveat that he does not quite agree with Richard George’s statement that people now try to find a way around cryptography instead of attacking it. He suggested that BEAST, CRIME, and the “padding oracle” attack be considered direct cryptography attacks. Arguing that many of today’s problems with TLS and Secure Sockets Layer (SSL) stem from decisions made in the 1990s, he said that even though our cryptographic algorithms and authenticated encryption have gotten significantly better, the built-in mechanisms have become “our Achilles’ heel.”

Green observed that a significant drawback in most systems is that legacy cryptography is not removed when new cryptography is added. As an example, Green said he was recently shown a paper describing vulnerabilities in several SSL stacks that involved the "completely obsolete" export-grade (512-bit) RSA. Despite its obsolescence, "It turns out that at the time of this vulnerability coming out, about 30 percent of the Internet browser-trusted certificates were actually supporting export RSA," Green said. Digging deeper, the researchers discovered that there had been an implementation vulnerability that allowed attackers to downgrade from a more secure protocol to export RSA, which could then be exploited—the Factoring Attack on RSA-EXPORT Keys CVE-2015-0204 (FREAK) vulnerability.

Noting that this same bug was found in three separate, and supposedly independent, TLS implementations, Green suggested that the bug may be in the protocol itself, rather than in the implementation. During the later discussion, Green speculated that one reason for the bug's persistence could be the persistence of features—and bugs—from previous versions of code that remain when a new version is created. Adam Langley, Google, Inc., added that the written specifications or instructions could inadvertently drive the implementer toward this bug. Eric Grosse, Google, Inc., noted that these examples underscore the importance of testing.

It is extremely challenging to prove today's protocols secure.

After discovering the vulnerability that allowed a downgrade to export-grade RSA, Green's team decided it would be wise to look at other ciphers supported in TLS/SSL and check for similar vulnerabilities. The team found a vulnerability, known as LogJam, that allows attackers to downgrade from a strong protocol to a weaker export protocol, even when the client does not support export. The cause appears to be a key exchange message that has the proper parameters for the server to talk to the client but is missing an identifier that tells whether it is a normal or export protocol. Paul Kocher, Cryptography Research Division, Rambus, Inc., interjected to accept the blame for this vulnerability. "Leaving aside who is to blame," Green replied, "the point here is that these things are incredibly difficult to get right. Even today, there are still vulnerabilities that probably we have not found."

Green then described another TLS attack, known as Decrypting RSA with Obsolete and Weakened eNcryption (DROWN). DROWN takes advantage of the continued use of SSLv2 to lead a cross-protocol attack to decrypt TLS communications. "This is lousy, and it leads to . . . all sorts of bad things that we should not be seeing in protocols in 2016," he said.

The Challenge of Complexity

A big problem, Green said, is that it is extremely challenging to prove today's protocols secure, or even to prove small parts of them secure, because they are so complex. As an example, Green pointed to a paper from 2012 that was the first to fully analyze *one* part of TLS, the Diffie-Hellman ephemeral protocol.[2] The next year, a paper analyzed the RSA handshake, another small aspect of TLS.[3] In both analyses, the authors were unable to analyze the last renegotiation. Green concluded, "It turns out that our agility is actually our weakest point."

In 2014, researchers "gave up on proving the entire protocol secure and they decided to have computers do it," Green said. Yet this effort was inconclusive because, as Green put it, "nobody can actually understand what the computers did." However, the paper[4] did provide some encouraging news: It showed that as long as the client and the server support only secure algorithms, TLS is a secure protocol. This is not actually what the TLS design community had been assuming, though; they had thought that as long as *the intersection* of the two sets of algorithms contained secure protocols, that was enough. However, he noted that this work showed that downgrade attacks can still find a way to use the insecure protocols, if they are also supported. "We need to improve TLS to deal with this kind of problem," Green concluded.

Practical Matters

Agility, then, is both good and bad, Green said. It allows us a way to move forward when attacks occur, but it also creates complexity in our protocols and enables new attacks. Whether we like agility or not is not the issue, Green stressed: "We are going to get agility no matter what we do because when we do not build agility into our protocols, other people do it for us, and they do it in really bad, dangerous ways." An example of this is the fact that most browsers support three TLS protocols as fallback maneuvers in case one fails, yet this agility creates insecure situations everywhere because of the possibility of downgrade attacks. "We have to design [agility] correctly because if we do not, we get the worst of all possible worlds," Green said.

Building on the argument—articulated many times throughout the workshop—that implementation is often the source of vulnerabilities, Bob Blakley, CitiGroup, Inc., posited that the situation is made even worse by the current convention that a reference

[2]T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, 2012, On the Security of TLS-DHE in the Standard Model, pp. 273-293 in *Advances in Cryptology – CRYPTO 2012* (R. Safavi-Naini and R. Canetti, eds.), Lecture Notes in Computer Science, Vol 7417, Springer, Berlin, https://eprint.iacr.org/2011/219.pdf.

[3]H. Krawczyk, K.G. Paterson, and H. Wee, 2013, On the Security of the TLS Protocol: A Systematic Analysis, pp. 429-448 in *Advances in Cryptology – CRYPTO 2013* (R. Canetti and J.A. Garay, eds.), Security and Cryptology, Vol 8042, Springer, Berlin, https://eprint.iacr.org/2013/339.pdf.

[4]K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, P. Strub, and S. Zanella-Béguelin, 2014, Proving the TLS Handshake Secure (As It Is), pp. 235-255 in *Advances in Cryptology – CRYPTO 2014* (J.A. Garay and R. Gennaro, eds.), Security and Cryptology, Vol 8616, Springer, Berlin, https://eprint.iacr.org/2014/182.pdf.

implementation guide is released before the official specification guide is finalized. By virtue of being released first, Blakley asserted, this implementation guide will have errors that may be fixed in the final version, yet everyone will still refer to it simply because it is released first.

Tadayoshi Kohno, University of Washington, questioned whether anyone has tried to intentionally break backward compatibility mechanisms for a small set of users (on Google Chrome, perhaps) in order to study the results. Doing so, he suggested, might help to determine whether we need to support agility in certain circumstances or if it is acceptable to "move to something new" without stranding large numbers of users. In response, Green suggested that perhaps a more fruitful approach would be to tap into the developer community and the IETF, who have the clout to influence security decisions and build solutions.

Another participant mentioned the counterparts to agility (i.e., transition planning and execution), suggesting that any transition is going to break something and what matters is how one plans for that eventuality. Pointing to his closing slide, which featured a tweet about how much of the Internet is "rotted with age," Green reiterated how difficult it is to eliminate code and security mechanisms that have become outdated. For example, he said during the disclosure of the LogJam vulnerability, Green's team wanted to remove the 512-bit Diffie-Hellman from use and raise the minimum to 1,024 bits. Apple, Google, and others scanned the Internet and found that removing the 512-bit Diffie-Hellman would break 3 percent of the Internet, including many older Internet of Things devices that cannot be upgraded. That experience underscored Green's conclusion that "we are stuck with these kinds of legacy devices that are dragging us back into the past." Adam Langley noted that Google did remove the 512-bit Diffie-Hellman modules internally, though the decision had some negative repercussions for at least one Google team.

Recalling Richard George's story about the multiple decades spent designing, testing, and implementing VINSON for radio communication, Steven Lipner, independent consultant, inquired about whether there is a way to achieve a similar but faster level of validation for security instruments being developed today—perhaps enabled by more people working in parallel. Green suggested the best way forward was for security-focused people to gain more traction in large companies and to fight for the resources to do it right. The current reliance on open source developers and research cryptographers to do this work has left us "in bad shape," he said. Langley agreed that formal verification could create good systems and good software, and that the field of verification is improving. He pointed to the National Aeronautics and Space Administration and space shuttle software as an example. Green countered that instead of a slow but steady process, it is going to take a major breakthrough for a significant portion of cryptography to be formally verified.

EXTENSIBILITY AND AGILITY

Adam Langley, Google, Inc.

Adam Langley is a principal software engineer at Google, Inc. He discussed extensibility, which he defined as the ability to make additions to a protocol without requiring sweeping, simultaneous global updates (a model, he noted, that is generally impossible anyway). Extensibility is a necessary component of agility, Langley said, because it provides a practical approach to updating already-deployed systems. He also offered examples illustrating how extensibility plays out in real-world situations—for better or worse.

Extensibility in Transport Layer Security

Langley described examples of extensibility mechanisms in TLS. When two computers negotiate which version of the protocol to use, it is the extensibility mechanism that allows the computer with an updated protocol to communicate with a computer with the older protocol. It is a system Langley described as "commendably simple," yet it does not always work. Sometimes the second computer does not recognize the updated information, preventing the communication from taking place.

When such failures occur, Langley said, "blame attaches to the last thing that changed." If Google updates Chrome's security mechanisms, for example, and suddenly triggers a bug in a bank's website, users would blame the Chrome update for blocking access to their bank's website, even though the update was meant to *increase* the users' security. Google's solution to such problems is to rely on an older "fallback" version while attempting to mitigate the disconnect. This intermediate step can last for a very long time; Langley said the browser community spent 15 years trying to fix several such bugs while relying on a fallback version to ensure continuity of service for users. Fallbacks are only used if the breaks caused by updated mechanisms will affect a substantial portion of traffic. If they only affect a small portion of users, say 0.1 percent of traffic, Langley said, "we just break things." Although developers attempt to build in many extensibility and agility mechanisms, often these supposed points of flexibility wind up becoming "rusted solid with bugs," Langley said.

Implications for Agility

From these experiences, Langley learned to "have one joint and keep it well oiled." Repairs should focus on one specific area, such as compliance suites or implementations, and they should be well tested in order to predict performance and interoperability. Testing the extensibility mechanism on a virtually continual basis can help ensure whatever "joint" you are depending on "is going to bend when you need it to," Langley said.

Experiences with extensibility underscore one of the workshop's broader themes, which is that agility should be approached "with hesitation, at least," Langley said, outlining the many costs of diverse options. Diversity, he said, generally results in more code and more bugs, thus reducing the ability for researchers and testers to focus deeply on any one of the many moving parts. As more interactions are possible, more bugs are created. He pointed to the process of defining primitives for use in TLS as an example of agility's secondary costs. More than 25 different symmetric cipher primitives have been defined so far; although only 9 are currently in use, each of the 25 were put through all the necessary IETF processes, a large expense of time and resources. Agreeing with Matthew Green's earlier statements, Langley asserted that many of today's challenges stem from decisions made in the 1990s, when technologists were distracted by the "crypto war" with the U.S. government: "An awful lot of these mistakes got baked in really deep, and we are still fighting them today," he said.

The International Context

Langley then turned to the international context, in particular the question of whether TLS should include ciphers for specific nations, which he termed "national pride cipher suites." Ciphers for Japan and South Korea, for example, were included in TLS. He views such cipher suites as "all cost and no benefit" and recommended that they be "fervently resisted." Though there might be nothing inherently wrong with them—and indeed, he conceded that to his knowledge, none, including those from Russia and China, contain anything deliberately malicious—he asserted that they offer no *technical* advantages over the National Institute of Standards and Technology–recommended Advanced Encryption Standard-Galois/Counter Mode (AES-GCM).

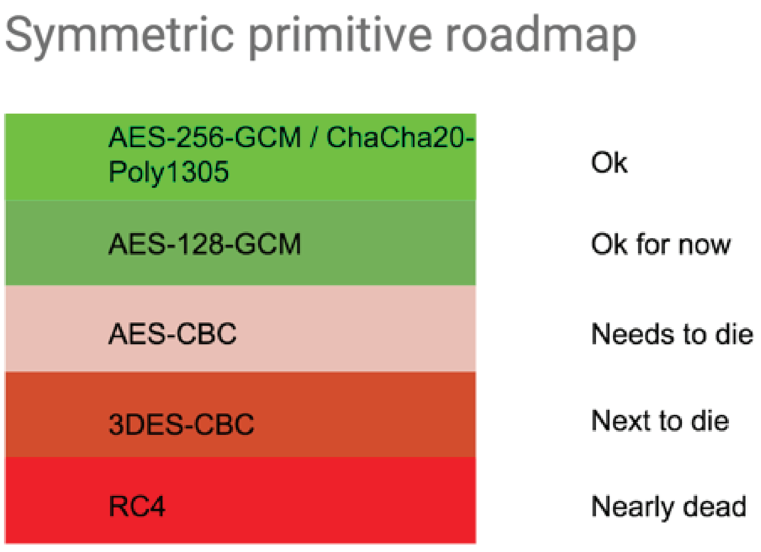
Returning to this issue in the discussion, Peter Swire, Georgia Institute of Technology, suggested that national cipher suites offer other advantages to nations, such as local expertise to better configure cryptography or local intelligence to better disable it. Agreeing that national security goals are likely part of the motivation for these cipher suites, Langley cautioned that the approach can backfire, arguing that making their own primitives, for example, would likely leave countries in a worse situation. Another drawback is that accommodating national cipher suites increases costs to everyone: While the nation in question may pay to create the cipher, the world bears the cost of supporting it. He then clarified that while he believes national ciphers should be resisted, it is unlikely to be feasible or necessary to "absolutely banish" them. Long-term Evolution (LTE) security uses a Chinese cipher, for example; however, Chrome does not support any national ciphers.

The Imperative to Retire Old Security Mechanisms

Building on previous workshop discussions around the issue of legacy cryptography, Langley described a stacked "conveyer belt" of security, with the newest and safest mechanisms at the top and the older, more vulnerable ones at the bottom (Figure 1 illustrates such a conveyer belt of symmetric cryptography primitives). "It is very important that we occasionally turn the crank on this conveyor belt so something falls off the bottom," Langley asserted.

Retiring outdated security mechanisms is extremely difficult because it means that some devices, connections, or software products will no longer work. For example, it is easy to envision how removing one of the older security mechanisms could cause hundreds of thousands of televisions to stop working, something that many find unacceptable since users expect televisions to last a long time. However, Langley suggested that expecting a complex electronic product to have a lifespan of a decade or more is "increasingly nonviable."

A significant downside of agility is that supporting outdated security can encourage its persistence, and new products may even be released today with cryptography from the *bottom* of the stack. People buying these products are unaware that they might have a very short lifespan because "they have jumped on this conveyor belt and we are about to turn the crank and they are about to fall off." Making such decisions, Langley said, "is not a job you want if you wish to be well liked." While large numbers of people will benefit from overall improved security, a concentrated group of users will pay the cost.



Symmetric primitive roadmap

AES-256-GCM / ChaCha20-Poly1305	Ok
AES-128-GCM	Ok for now
AES-CBC	Needs to die
3DES-CBC	Next to die
RC4	Nearly dead

FIGURE 1 **"Conveyer belt" of symmetric cryptography primitives.** SOURCE: Courtesy of Adam Langley, Google, Inc., presentation to the workshop.

During the discussion, Paul Kocher asked whether strategies for tying new features to new security—or making software with outdated security slower or otherwise less desirable—can help to motivate users to upgrade. Noting that Google has sometimes been able to offer "carrots" to users if they make updates, Langley said this approach can be helpful. However, it is not without problems: Hypertext Transfer Protocol Version 2 (HTTP/2), for example, was built to be usable only with modern cipher suites, but it ended up creating significant configuration problems. While he had good intentions, Langley said, "I am unsure whether that was a good idea."

Noting that the focus of his presentation is on symmetric ciphers in TLS, Langley said that the same conveyer belt exists for asymmetric ciphers. In one sense, TLS is a bit simple because it is a negotiated protocol where two computers can communicate in order to reach a security agreement. In non-negotiated protocols, such as Public-Key Cryptography Standard #12 (PKCS#12), things are much harder because there is no conversation. PKCS#12, in practice, still uses Rivest Cipher 2 (RC2) (a 64-bit block cipher with known vulnerabilities) because that was how it was originally created. A similar situation exists in Secure/Multipurpose Internet Mail Extensions (S/MIME) (for encrypting email). If S/MIME is not communicating with a known channel, the recipient may not be able to decrypt the emails unless he agrees to use the *lowest* security mechanism they share in common.

Wrapping up, Langley pointed to the typically decade-long timelines of retiring security mechanisms such as Message-Digest Algorithm-5 (MD5), RSA 1024, and Secure Hash Algorithm-1 (SHA-1) in certificates. Preparations for these transitions are often years in the making, and use of the retired mechanisms often persists for years past their supposed final date.

Looking toward the future, with regard to the threat of quantum computing, Langley said only that he hopes he is retired before the threat becomes a reality. As a cynical upside, he posited that software itself has so many bugs that attackers might "not even bother" to attack the cryptography.

Who Bears the Costs?

Participants engaged in a lively discussion about the costs that are borne when products or devices become obsolete, and they explored how those costs might be allocated more fairly or equitably.

Prompted by a comment from Grosse, Langley described a situation in which Google disabled an older version of software used by televisions that caused a break for a tiny fraction of users—approximately 0.001 percent. After user complaints, Google relented and postponed the switch, giving the manufacturer a deadline to have the devices updated before the old version would be permanently disabled.

Deirdre Mulligan, University of California, Berkeley, noted that such experiences raise consumer protection questions—consider, for example, the implications of a break if the users who are most affected are also the least able to afford a replacement television. She pointed to the National Vaccine Injury Compensation Program as a model of one potential solution. Because vaccines are so widely beneficial for public health, the program provides compensation to anyone who is injured as a result of receiving a vaccine. Could a similar system be instituted to support the common good of cybersecurity, where subsidies could be given to those needing new devices who cannot afford to buy them?

Langley agreed that this is a challenge, describing the "painful realization" that often the users most likely to be affected by a break are the poorest in society, who are more likely to be using older devices and software. Expressing his view that subsidizing new device purchases could make removing legacy cryptology significantly easier, Langley conceded that he did not know how such a plan could work from a political standpoint. One partial solution Google supports is to allow manufacturers to test new products with Google's test server to ensure that their devices work with the latest cryptography. The idea is that if a product does not work, the manufacturers will fix it, thus preventing new devices from being released with security that is at the "bottom of the conveyer belt."

Who bears the cost when products or devices become obsolete?

Delving more deeply into the question of who should bear the cost of replacing defunct products, participants noted the many players involved, including users, manufacturers, and service providers. Sara Brody, Simply Secure, questioned how a system in which society at large bears the responsibility for upgrades would affect the incentive structures for manufacturers, who already reap profits from upgrades and new device purchases. Mulligan added that if a manufacturer decides to stop providing upgrades or services to a product, perhaps it should be required to release its code so that others could repair the flaws or install updates to keep those devices functioning. Building on this point, Brody noted that this problem is particularly acute for Internet of Things devices, especially when they are sold separately from the service that is required for them to connect online. If the company goes out of business, the service will be cut off and the device will be rendered useless. Grosse noted that one reason SSLv2 is widely supported overseas, especially in Asia, is because low-end "feature phones" use it to communicate with online banking sites. It is difficult to conceive of a system for replacing so many devices currently in use by a wide array of people that could continue to support the needed functionality but provide upgraded security, he said.

THE IMPORTANCE OF THE HUMAN FACTOR IN CRYPTOGRAPHIC AGILITY

Sara "Scout" Sinclair Brody, Simply Secure

Sara "Scout" Sinclair Brody is the executive director at Simply Secure, a nonprofit that aims to increase privacy and security through solutions that are human centered. Based on her background in the field of usable security, she posited that unless a system being built will use solely machine-to-machine communication, without contact with people at any point, human factors must be considered when designing the system's security and agility mechanisms.

What role do humans play in cryptographic agility? Brody delineated two groups of users: the end users, who actually use the products, and the developers who create and apply the security and agility tools. In considering this second group of users, she urged attendees to include not only elite developers, such as those in attendance at the workshop, but also the vast majority of developers who are simply not experts in cryptography or agility. It is those developers, she noted, who are working on start-ups whose products may be insignificant now but could quickly balloon to include millions of users: "Ultimately, they are sort of the front line of cryptographic agility." Beyond solving the very large problems being discussed at the workshop, Brody asserted the need to communicate effectively with this vast developer community about agility principles and policies.

Moving Toward a More Nuanced Conversation

Security is a secondary task in the minds of most users and most developers, Brody said. Both users and developers are far more focused on the primary task of a given product, such as sending and receiving messages. Given this mindset and the limited resources most organizations devote to security, non-experts tend to think of security in a binary way: either a piece of software is secure, or it is not. They—and this includes both developers and end users—do not see the "nuances of gray" or think of security as a spectrum dependent on contexts or threat models.

Security experts sometimes reinforce this binary thinking, she said, by flatly rejecting or endorsing certain systems, or failing to recognize how design constraints or threat models contribute to a more nuanced landscape of options and decisions. Brody posited that a better approach is to "create more nuance in the conversation" around security needs by reinforcing that security is not a binary property. She also cautioned against letting "the perfect be the enemy of the good."

Providing a Basis for Better Decisions

How can this be accomplished? Recognizing that many security decisions are situation dependent, Brody said that being able to see a stack ranking or prioritization list could help developers make better decisions about security. Another way to help developers would be if the cryptography community did something it is unaccustomed to doing: pass value judgment on the algorithms. “We tend to say, ‘You have to make that decision for yourself,’” she said. “No; we are the experts. Our community needs to be willing to say, ‘These algorithms are definitely better than these other algorithms.’” Brody pointed to Langley’s conveyer belt image as a way to guide a developer’s cryptography design choices and explain what cryptography under certain circumstances would require.

Another way to convey these messages is through the education process. Brody suggested students should be learning about—or perhaps be taught more effectively about—not just system building, but also recognizing and planning for obsolescence, cryptographic agility, and system maintenance. In the discussion, Steven Lipner noted that in his experience, most developers do not know much about cryptography and are not taught about software vulnerability, even in the top engineering schools. On one hand, he said he recognized that not every developer can also be a security expert, because it would distract them from their primary task of building software, but on the other hand, developers do need a basic level of knowledge in order to best serve their end users. Brody pointed out that developers would also benefit from at least having access to people and resources to help with decision making. Lipner cautioned that “access to people” is not a scalable option, though improved access, in addition to a lot of information and the motivation to consume it, would be useful.

Security is not a binary property.

Recognizing Real-World Constraints

Developers are often concerned about how different governments are going to react to their use of cryptography, which Brody said was understandable given that developers are humans with real concerns and legitimate fears. “The legal aspects of cryptography are very scary to them,” she said. But while it may be easier to ignore the legal landscape of these issues, it is far more useful to engage in the debate on a deeper level and to truly understand both the technical and legal implications.

On the subject of legacy cryptography, she agreed with previous speakers that continuing to run insecure software, especially somewhere with sensitive data to protect, like a hospital, is a scary proposition. However, she noted that it is important to recognize that sometimes users have no choice. A hospital using a magnetic resonance imaging machine running Windows XP, for example, cannot necessarily be expected to acquire a new machine in order to get the latest software.

Insight about developers, their constraints, and their motivations could help to surface better solutions, she said. An ethnographic study of developers, for example, could help reveal how they think about security, what their concerns are, and what barriers they face in implementing agility. In the discussion, Peter Swire suggested that the field of human-computer interaction is relevant in this context, though he had not previously considered applying human-computer interaction techniques to figure out how to get developers and others to adopt certain practices. Brody reiterated that however much technology there is in cryptography and agility, these are still, in the end, "human systems," built and implemented by humans to serve human needs. These systems need agility, but what really makes agility happen is the developers, the standards bodies, and the research organizations—all of which are comprised of humans.

Systemic Improvements

Good defaults are important for developers and end users alike. Developers cannot create good user defaults when they are themselves offered too many security options without clear advice or explication of their differences. In such situations, developers often end up passing the decision on to the end user, who is, Brody noted, "not exactly the most qualified person to be deciding what cryptographics [they] should be using." Providing good defaults first for developers, and then for users, would be far more helpful.

If good defaults cannot be offered, the next best thing is clear recommendations, she said. Even a simple drop-down menu with cryptography suites rated "recommended" and "not recommended" would be preferable to confusing acronyms that mean nothing to the non-expert. End users would appreciate seeing security options in binary terms and are in fact more likely to make good decisions this way.

Automatic updates that push important patches out to end users are also crucial, said Brody, though she recognized that they are "scary on some levels." She suggested that the best practice is to push the update automatically and notify the user, although an easy opt-out mechanism should still be included with the alert.

The Benefits of Transparency

In closing, Brody stressed the value of transparency. End users, she said, should be able to determine which security library their various applications are running in order to evaluate the threat posed to them personally by a new vulnerability or attack. Such transparency would also be helpful to citizens in the context of national cipher suites and potential government surveillance.

Opening the discussion, Bob Blakley agreed that transparency is a good thing, but he said many end users do not have the education or background to make the best decisions when it comes to complex computer security issues. Brody agreed, clarifying that she would not expect average users to engage with this information regularly. Rather, she suggested having at least the *ability* to investigate one's computer security is helpful both for the general population and for populations whose security might be most at risk, such as journalists or human rights activists who might face threats of government surveillance. Those groups could benefit from increased security transparency because they have a particular need or interest in knowing the safety of their communications. If transparency becomes the standard, accountability could improve and software and applications would be updated regularly.