© Columbia Engineering; Eileen Barroso

Steven M. Bellovin
Columbia University

What Should Crypto Look Like?

If there's one thing we know about cryptography, it's that it's hard. In fact, it's hard at all levels: the primitive encryption mechanisms, the protocols, the implementation, the rules for using it—it's all very, very hard, and mistakes are gifts to the attackers. Many of these problems have drawn considerable attention from the technical community. Submissions to the National Institute of Standards and Technology contests for encryption or hash functions must include a security analysis, including demonstrable resistance to known classes of attack such as differential cryptanalysis.

The risks inherent in poorly designed protocols have been suspected since Needham and Schroeder's 1978 paper, the first on the subject in the open literature. Similarly, the literature is rife with solutions for timing attacks, cache line attacks, and more. Most of these attacks are minor, though; they tend to require large amounts of intercepted data—itself a significant hurdle for many attackers—and a sophisticated attacker. One class of problem, though—user mistakes—has drawn very little attention outside the usability community. This is a serious omission; user errors were and are a serious threat, since they can and do result in messages being sent in plaintext despite the users' intentions.

It's not as if there was no warning. Codes have long been cracked because the clerks composing messages didn't insert enough nulls or make proper use of ciphertext homophones. German errors in using the World War II–era Enigma machine, and poor choice of what today we would call a session key, helped British cryptanalysts. In more recent years, Whitten and Tygar's classic "Why Johnny Can't Encrypt" sounded a clear warning, but follow-ons like Garfinkel and Miller's "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express" have either been presented to the usable security community—the choir to such a sermon—or concentrated at least as much on the protocol issues as the usability issues. This is wrong; usability failures are the leading technical cause of phishing attacks and unintended plaintext emails, and share much of the blame for the problems with the Web's public-key infrastructure.

Let's take a closer look at cryptographically protected email: what should it look like? Should senders have to request encryption? What if a user sometimes receives messages on a device that doesn't have his or her key? If the crypto is on by default, should senders be able to disable it? What if some recipients of a message are known to be able to handle encryption while others are either known not to be or are of unknown capabilities? How should this be shown? We know that users don't notice subtle indicators, such as a lock icon; we also know that vendors are fond of drastic UI changes—compare iOS 6 with iOS 7—for reasons of their own, so that won't work.

Recipients have similar problems. How should they be told—in clear, unambiguous terms—that a given message was received encrypted? How should the absence or presence of a digital signature be shown? How should the cryptographically verified sender, as opposed to the "From:" line, be shown? What if they disagree, or perhaps disagree in obviously trivial ways? These questions may have comparatively simple answers if we could flash-cut every mailer in the world—but of course we can't.

Key handling presents its own challenges. How do I securely determine someone else's public key? How can this be done so that I don't have to worry or even know about it? How do they get my key, when they may not even understand crucial concepts like certificates and PKI? How do I protect my private key, given the insecurity of today's operating systems? What about exceptions, such as key revocation?

None of these questions have simple answers. It isn't clear that some of them can even have answers. But we won't know unless and until the security community as a whole starts paying attention to this seemingly simple question: what should crypto look like? ■

Steven M. Bellovin is a professor of computer science at Columbia University. Contact him via www.cs.columbia.edu/~smb.