Steven M. Bellovin
Columbia University

The Key to the Key

We're hearing a lot about encryption these days. It might be the US Federal Bureau of Investigation complaining about encrypted iPhones or privacy advocates bemoaning the lack of encryption on breached databases. These discussions frequently miss a vital point: What about the keys? Where do they come from, how are they handled, and who has access to them? Whenever someone says that something is encrypted, the first question you should ask—before you ask about algorithms, data formats, or protocols—is, "What about the keys?"

Protecting keys has been a priority throughout the history of cryptography. When spies stole codebooks, they were actually stealing keys. Additive tables, a form of superencipherment, are themselves a form of key, intended to encrypt the codebook. Both, of course, were targeted by cryptanalysts who were really trying to recover the keys.

However, there's a subtle point here: a cryptosystem is a system. The surrounding context is vital—how the system is used, what other information is available to cryptanalysts, and how people actually use the system. Thus, to crack the Enigma machine during World War II, the British exploited standard messages ("Nothing to report"); the same message sent in other, weaker ciphers; and cipher clerks' propensity to pick bad session keys ("cillies"). All of this matters—it's not just the combinatorics of the key space.

Cryptographers understand this as well as cryptanalysts do, of course. In recent decades, a major goal of encryption system design has been to introduce more keys, so that recovery or compromise of one doesn't expose an entire communications network. That's why the Enigma procedures required use of session keys: to limit the damage if a key was recovered.

In recent years, *key freshness*—ensuring that no key is reused and that both parties can verify this—has been a major cryptographic protocol design goal. Sometimes, though, the systems issue is neglected, especially by those who don't deal with cryptanalysis daily.

We see this in the current debate about the challenges strong cryptography poses. Governments around the world are demanding that developers provide ways around strong cryptography. In the US, its proponents have called this a "golden key." They're right to be thinking about the keys—but they're neglecting the systems aspect of the problem. Will these schemes expose only the right information, and to only the right parties? Many suggestions have been made, such as encrypting the data encryption key with law enforcement's public key. There are many reasons to be skeptical, starting with the fact that we're dealing with a *system*.

Consider the problem of buggy code. Code is frequently buggy, including the programming to export keys "only" to law enforcement. I don't have to speak hypothetically; it's happened in the past. Look at it this way: exporting keys to third parties is by definition a security breach. The intent is to limit the scope of the breach—but did the programmer get the limits right? Even if he or she did, the basic export code can still be abused if some other security problem exists. According to published reports, abusing a Greek cell phone switch to tap calls took only 6,500 lines of code—but that's because the basic tapping software had already been built in for law enforcement's use.

There's another issue: How does the software get the proper law enforcement key? If it's compiled in, how are other countries' needs handled? If it's supplied at device activation time—assuming that the device needs to be activated—where is the device when activation takes place? How does the vendor's activation software know the device's real location? How does the vendor reliably and securely get these public keys from the police?

There are many more such issues, but my fundamental point is simple: protecting keys requires protecting the entire system. ■

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

Copublished by the IEEE Computer and Reliability Societies