Use cryptography correctly

Cryptography is one of the most important tools for building secure systems. Through the proper use of cryptography, one can ensure the confidentiality of data, protect data from unauthorized modification, and authenticate the source of data. Cryptography can also enable many other security goals as well. Cryptography, however, is not a panacea. Getting cryptography right is extremely hard. Some common pitfalls are listed below.

- **Rolling your own cryptographic algorithms or implementations**. Designing a cryptographic algorithm (including protocols and modes) requires significant and rare mathematical skills and training, and even trained mathematicians sometimes produce algorithms that have subtle problems. There are also numerous subtleties with implementing cryptographic algorithms. For example, the order of operations involved when exponentiating a number — something common in cryptographic operations — can leak secret information to attackers. Standard algorithms and libraries are preferable.
- **Misuse of libraries and algorithms**. Even when using strong libraries, do not assume that just using the libraries will be sufficient. There have been numerous instances in which standard libraries were used, but the developers using the libraries made incorrect assumptions about how to use the library routines. In other situations, developers don't choose the right algorithm or use the algorithm incorrectly. For example, an encryption scheme may protect the confidentiality of data, but may not protect against malicious modifications to the data. As another example, if an algorithm requires an initialization vector (IV), then choosing an IV with certain properties may be required for the algorithm to work securely. Understanding the nuances of algorithm and library usage is a core skill for applied cryptographers.
- **Poor key management**. When everything else is done correctly, the security of the cryptographic system still hinges on the protection of the cryptographic keys. Key management mistakes are common, and include hard-coding keys into software (often observed in embedded devices and application software), failure to

allow for the revocation and/or rotation of keys, use of cryptographic keys that are weak (e.g., keys that are too short or that are predictable), and weak key distribution mechanisms.

- **Randomness that is not random**. Confusion between statistical randomness and cryptographic randomness is common. Cryptographic operations require random numbers that have strong security properties. In addition to obtaining numbers with strong cryptographic randomness properties, care must be taken not to re-use the random numbers.
- **Failure to centralize cryptography.** Numerous situations have been observed in which different teams within an organization each implemented their own cryptographic routines. Cryptographic algorithms often don't interact nicely. Best practices indicate getting it "right" once and reusing the component elsewhere.
- **Failure to allow for algorithm adaptation and evolution**. See Bullet "*Design for changes in the security properties of components beyond your control*" in "*Be flexible when considering future changes to objects and actors*" section.

Cryptography is so hard to get right that it *always* makes sense to work with an expert if you can. Note that expertise in applied cryptography is not the same as being a mathematician and having a mathematical understanding of cryptography. At the highest level, make use of proven algorithms and libraries, but realize that *just* the use of such things does not guarantee security — it is easy to accidentally misuse these things. Have a cryptography expert work with your designers to provide an API abstraction around a strong library, so that that your developers are not making decisions on algorithms and cipher modes, and so that if you need to change algorithms behind that abstraction layer you can.