Investigating cryptographic hash functions with a computer algebra system

Alasdair McAndrew

School of Engineering and Science,
Victoria University, Melbourne Australia

alasdair.mcandrew@vu.edu.au

Abstract

Cryptographic hash functions sit at the core of almost every cryptographic protocol, yet often they are treated only briefly, if at all, in an introductory course. A cryptographic hash function is designed to produce a sort of "digital fingerprint" of its input data, and to produce a fixed length output (often 256 bits) no matter the size of the input. For security such a function must be *pre-image resistant*, meaning that given a hash output it should not be possible to find the input, and also *collision resistant*, meaning that it should not be possible to find two different inputs with the same hash. Hash functions are used, for example, in digital signature algorithms, where the algorithms used are slow to the extent that it is far more efficient to sign the hash of the message than to sign the message itself.

Hash functions may be divided into two classes: those which are *provably secure*, and which are based on number theoretic constructions, and faster bit-oriented hash functions.

The complexity of many of the standard functions means that it is very difficult for beginning students to gain insight into how hash functions work, and to explore them. A computer algebra system can be used to great effect to

1. Build simple hash functions of both types (provably secure, and bit-oriented).
2. Provide an environment in which students can explore such functions.
3. Show how hash functions can be used in a large cryptographic protocol, for example, a digital signature algorithm.

We show how the CAS Sage (http://www.sagemath.org) can be used to support all the above, and how we have used it in an introductory cryptography course.