Permutation functions

Most of cryptography is achieved using invertible functions, commonly called "permutation functions" in cryptography. In this module we will review several definitions related to functions, investigate the notion of a "random function", and see several ways of constructing an invertible function mathematically and in C.

Cryptographic algorithms are data-intensive and computationally expensive. The C programming language is well-suited for such requirements. In fact, the most used cryptographic software library, OpenSSL, is written in C. One goal of this module is to review the most important parts of C for manipulating data.

This module assumes that you took CSC 60 and can write, compile, and debug simple C programs. We will *not* significantly review C programming, but will review the parts of it used in cryptography (eg, memory and bitwise manipulation).

Learning objectives

By the end of this module you should be able to...

- Explain and give examples of mathematical functions that are onto, one-to-one, and invertible;
- Calculate probabilities of function outputs when the function is randomly chosen;
- Explain what a distinguishing game is and calculate the probability of success in one;
- Explain the notion of diffusion and identify operations with good diffusion;
- Write invertible functions in C and/or Java that use the Add/Xor/Rotate or Feistel paradigm.
- write a C and/or Java function that receives memory as a parameter and manipulates the memory, such as to reverse the order of a buffer of bytes in memory;
- manipulate data using bitwise operations, such as to read or set a bit inside a variable, circular shift a variable's bits, or reverse the order of bytes in a variable; and
- explain little- and big-endian memory accesses, and implement big-endian reads and writes on a little-endian computer.