A Note on Hash Functions

Peter Occil

1 A Note on Hash Functions

This version of the document is dated 2023-06-13.

A *hash function* is a function that maps an input of arbitrary size into an output with a fixed number of bits, called a *hash code*.

Hash functions are used for the following purposes:

- For random number generation. See my article on **random generator recommendations**.
- In hash tables to speed up data lookup by mapping data to a sequence of bits in a way that avoids hash collisions. Here, hash codes are usually small (for example, 32 bits long).
- For data lookup in databases and file systems (for example, to partition many files into a small number of directories to speed up file lookup). Here, hash codes generally are 64 bits or more long and can use cryptographic hash functions (for example, SHA2-256, BLAKE2) and other hash functions that tend to produce wildly dispersed hash codes for nearby inputs.
- For information security. Only cryptographic hash functions are appropriate here, and they serve as building blocks for many kinds of security algorithms, such as:
  - Checksums and authentication codes for data integrity.
  - Pseudorandom functions, for deriving secrets from other secrets.
  - Key derivation functions for verifying secrets without storing the secrets themselves. Because the secret may be easy to guess, like a password, some of these functions are designed to deliberately take time to calculate.

For the use in hash tables, (Richter et al. 2015)[1] recommends multiply-then-shift hashing over more complicated hash functions in most cases. ***Fibonacci hashing*** is a special case of multiply-then-shift hashing that serves to improve hash codes.

There are security attacks that serve to trigger worst-case performance on hash

[1] Richter, Alvarez, Dittrich, "A Seven-Dimensional Analysis of Hashing Methods and its Implications on Query Processing", *Proceedings of the VLDB Endowment* 9(3), 2015.

tables via carefully chosen keys, and keyed hash functions (such as SipHash) have been developed to mitigate them, but not everyone believes such hash functions should be used in hash tables (for example, see R. Urban's SMHasher fork).

Bret Mulvey has written a page on **how hash functions are built**. According to him, hash functions have as their core a *mixing function* (a function that maps N-bit inputs to N-bit outputs), and for hash functions the "best" mixing functions tend to produce wildly dispersed outputs for similar inputs.

2 Notes