Hash Functions

November 24, 2008

Hash Functions

A hash function h compresses an arbitrary length input message to a fixed length message digest. Such a function should satisfy the following requirements:

- For any given messge m, the digest $h(m)$ can be computed quickly.
- Given any possible digest y, it is computationally infeasible to find a message m such that $h(m) = y$. (So h is a one-way function.)
- It is computationally infeasible to find messages m_1 and m_2 with $h(m_1) = h(m_2)$. (Such an h is said to be strongly collision free.)

Of course, many collisions must exist by simple counting. So the third requirement says that it must be hard to find examples.

Weakly collision-free

Sometimes people weaken the third condition. A hash function is weakly collision free if for any GIVEN x, it is computationally infeasible to find a second $x' \neq x$ such that $h(x) = h(x')$.

Popular hash functions don't work well!

- In 2004 Wang, Feng, Lai, and Yu found many examples of collisions in the popular hash functions MD4, MD5, HAVAL-128, etc. So hashing has recently fallen into a state of disgrace.
- It seems that hashing is actually harder than many people thought.
- Why hash? One important reason is to protect data integrity. Suppose you wish to download a large file. How do you know it downloaded correctly? If the person providing the file also provides its hash value, then you can recompute the hash value after download. If the two hashes don't agree, the download was corrupted.

EXAMPLE: Discrete Log Hash

- Choose a large prime p such that $q = (p-1)/2$ is also prime.
- Choose two primitive roots α, β for $\mathbb{F}_p$.
- Define $h(m) = \alpha^{x_0}\beta^{x_1} \pmod{p}$ where $m = x_0 + x_1 q$ with $0 \leq x_0, x_1 \leq q-1$.

This satisfies the last two requirements, but it is too slow to be used in practice.

A SIMPLE EXAMPLE

- Let m be a message. Break it into n-bit blocks, where n is fairly small, typically 160 bits. (The last block is padded with zeros if necessary so that all blocks are of the same size.)
- Say that the blocks are $m_1, m_2, \ldots, m_k$. Think of each block as an n-bit binary number, and let $c_i =$ XOR of all the ith bits. Then the binary number $c_1 c_2 \ldots c_n$ is the hash $h(m)$.
- This is too simple to be practical, since it is easy to construct collisions. But this idea can be combined with bit rotation and other bitwise manipulations, to make something which is more collision resistant.

The Birthday Attack

- In a room of 23 people, there is a better than 50% chance that two people have the same birthday. In a room of 30 people, the probability goes up to about 70%.
- The probability that all 23 people have different bithdays is

$$(1 - \frac{1}{365})(1 - \frac{2}{365}) \cdots (1 - \frac{22}{365}) = .493$$

  and thus the probability that two are the same is $1 - .493 = .507$.
- More generally, if we have N objects, and each of r people chooses an object (with replacement). Then the probability of a match is approximately $1 - e^{-r^2/2N}$.

The Birthday Attack

- A variation is to suppose two rooms, each with 30 peaople. What is the probability that someone in the first room has the same birthday as someone in the second room?
- More generally, there are N objects and two groups of r people, who choose an object (with replacement). What is the probability that someone in the first group chooses the same object as someone in the second group?
- ANSWER: $P \approx 1 - e^{-r^2/N}$
- Used to find collisions for hash functions if the output size is too small. Suppose h is an n-bit hash function. There are $N = 2^n$ possible outputs from h. List $h(x)$ for about $r = \sqrt{N}$ random values of x. There is a good chance of finding two values x_1, x_2 with the same hash value. Make the list 10 times as long, and the probability of a match goes up.
- Works well for $n \leq 60$ or so.