Hash 2

Crypto Hash Function

- Crypto hash function $h(x)$ must provide
  - **Compression** — output length is small
  - **Efficiency** — $h(x)$ easy to compute for any x
  - **One-way** — given a value y it is infeasible to find an x such that $h(x) = y$
  - **Weak collision resistance** — given x and $h(x)$, infeasible to find $y \neq x$ such that $h(y) = h(x)$
  - **Strong collision resistance** — infeasible to find ***any*** x and y, with $x \neq y$ such that $h(x) = h(y)$
- Lots of collisions exist, but hard to find ***any***

Hashes and Birthdays

- If $h(x)$ is N bits, 2^N different hash values are possible
- So, if you hash about $2^{N/2}$ random values then you expect to find a collision
  - Since $\text{sqrt}(2^N) = 2^{N/2}$
- **Implication:** secure N bit symmetric key requires 2^{N-1} work to "break" while secure N bit hash requires $2^{N/2}$ work to "break"
  - Exhaustive search attacks, that is

Popular Crypto Hashes

- **MD5** — invented by Rivest
  - 128 bit output
  - Note: MD5 collisions easy to find
- **SHA-1** — A U.S. government standard, inner workings similar to MD5
  - 160 bit output
- Many other hashes, but MD5 and SHA-1 are the most widely used
- Hashes work by hashing message in blocks

Crypto Hash Design

- Desired property: **avalanche effect**
  - Change to 1 bit of input should affect about half of output bits
- Crypto hash functions consist of some number of rounds
- Want security and speed
  - Avalanche effect after few rounds
  - But simple rounds
- Analogous to design of block ciphers

Hash Uses

- Authentication (HMAC)
- Message integrity (HMAC)
- Message fingerprint
- Data corruption detection
- Digital signature efficiency
- Anything you can do with symmetric crypto
- Also, many, many clever/surprising uses…

HMAC

- Can compute a MAC of the message M with key K using a "hashed MAC" or **HMAC**
- HMAC is a **keyed hash**
  - Why would we need a key?
- How to compute HMAC?
- Two obvious choices: h(K,M) and h(M,K)
- Which is better?

HMAC

- Should we compute HMAC as $h(K,M)$?
- Hashes computed in blocks
  - $h(B_1,B_2) = F(F(A,B_1),B_2)$ for some F and constant A
  - Then $h(B_1,B_2) = F(h(B_1),B_2)$
- Let $M' = (M,X)$
  - Then $h(K,M') = F(h(K,M),X)$
  - Attacker can compute HMAC of M' without K
- Is $h(M,K)$ better?
  - Yes, but… if $h(M') = h(M)$ then we might have $h(M,K)=F(h(M),K)=F(h(M'),K)=h(M',K)$

The Right Way to HMAC

- Described in RFC 2104
- Let B be the block length of hash, in bytes
  - $B = 64$ for MD5 and SHA-1 and Tiger
- $\text{ipad} = 0\text{x}36$ repeated B times
- $\text{opad} = 0\text{x}5\text{C}$ repeated B times
- Then

$$\text{HMAC}(M,K) = h(K \oplus \text{opad}, h(K \oplus \text{ipad}, M))$$

Online Bids

- Suppose Alice, Bob and Charlie are bidders
- Alice plans to bid A, Bob B and Charlie C
- They don't trust that bids will stay secret
- A possible solution?
  - Alice, Bob, Charlie submit **hashes** $h(A)$, $h(B)$, $h(C)$
  - All hashes received and posted online
  - Then bids A, B, and C submitted and revealed
- Hashes don't reveal bids (one way)
- Can't change bid after hash sent (collision)
- But there is a flaw here…