# Lecture 2 - Cryptographic Hash Functions

**Rylan Schaeffer and Vincent Yang**

**April 7, 2016**

Note: This lecture is based on Princeton University's BTC-Tech: Bitcoin and Cryptocurrency Technologies Spring 2015 course.

## Terms

- Set : group of objects represented as a unit
- Alphabet : finite, non-empty set
- String : finite sequence of characters from common alphabet, including empty string $\varepsilon$
- Language : set of strings over common alphabet

## Hash Functions

- $H : \{0,1\}^* \rightarrow \{0,1\}^k$, for fixed k e.g. 256
- Should be efficiently computable O(n)
- Examples: mod operator, separate chaining hash (draw)
- Homework 1 - *SHA-256*

## Cryptographic Hash Function

Two additional properties

- Collision Resistant: Computationally infeasible to find $x, y$ such that $x \neq y$ and $H(x) = H(y)$

  mod operator is not collision resistant

  collisions exist by pigeonhole principle - hence, computationally infeasible

  Avalanche Effect - when one letter changes, everything changes

  can also call "binding," since once hash is published, you cannot replace input value with another input value without modifying the hash output

- Hiding: Computationally infeasible to find $x$ given $H_{given}$ such that $H(x) = H_{given}$

  Frequently, cryptographic hash functions will be called one-way hash functions (trap door functions)

  Frequently, message space is too small. Append nonce (i.e. random value) r to grow message space such that computationally infeasible to find $x$ such that $H(x|r) = H_{given}$

- Birthday paradox reduces difficulty of finding collisions

- How many people do you need in a room for there to be a 50% chance that two have the same birthday? (A: 23)
- Exponents aren't intuitive
- With 23 people we have 253 pairs - $\frac{23 \cdot 22}{2} = 253$.
- Chance of 2 people having diff. birthdays is $1 - \frac{1}{365} = \frac{364}{365} = .997260$.
- If all pairs are different, this is like having 253 heads in a row. $Probability = (\frac{364}{365})^{253} = .4995$.
- SHA-256: Let's say we have a "perfect" hash function with output size $n$, with $p$ messages to hash. Probability of collision is $\frac{p^2}{2^{n+1}}$.
- This means, if we have $n = 256$ as in SHA-256, and 1 billion messages ($p = 10^9$) then the probability is still only $4.3 \cdot 10^{-60}$.
- A mass murdering space rock happens about once every 30 million years. The probability of it happening in the next second is about $10^{-15}$. This is 45 times more likely than a SHA-256 collision.

# Applications

- Message Digest

  Create summary (or "digest") of block of text

  Suppose I have $msg$ and $H$ is a cryptographic hash function. Then I know that $H(msg)$ or perhaps $H(msg|r)$ (where r is a random value and is needed because the message is predictable), will produce a hash value that no other block of text will.

  Example: cryptographic checksums

- Commitments

  Analogous to sealed envelope on the table

  Hiding ensures no one can "reverse engineer" the contents. Collision-resistant guarantees to the other party that you are bound to the value you initially put in.

# Puzzle Friendliness

- Search Puzzle

  Given $H$, target set $Y$, and value $x$

  Goal: find r such that $H(x|r) \in Y$

- Puzzle friendly if no solving strategy for puzzle other than trying random guesses at $r$

- Examples: $0|\{0,1\}^{k-1}$, $00|\{0,1\}^{k-1}$, $000|\{0,1\}^{k-1}$

  P(l leading zeroes) $= \frac{1}{2^l}$, can use geometric distribution's cumulative distribution function to model likelihood of observing a "hit" after a given number of failures

- Useful for mining, which we will get to later

# Hash Structures

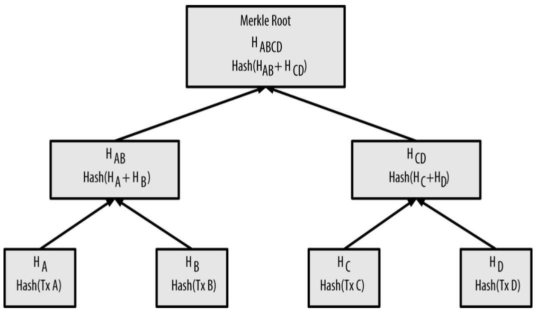
- Hash pointer : hash of data. Gives way to verify information hasn't changed, much like pointer gives a way to retrieve location of information

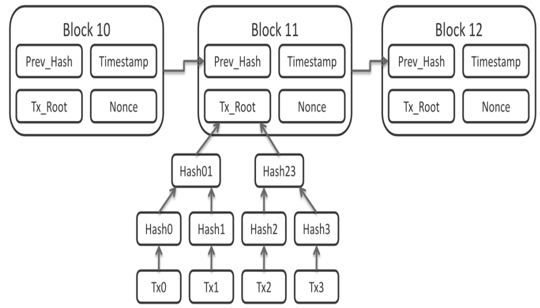
- Hash linked list (block chain) : Each block has hash of previous block plus new data. Head is hash of most recent block.

  Tamper-evident log

- Hash tree (Merkle Tree) : binary tree of data blocks. Proof of membership and proof of non-membership in $log(n)$, so faster than hash linked list. Can also sort.

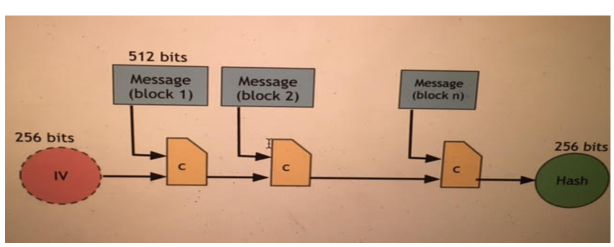
  Verification in $O(log(n))$.
- Can combine. Block chain is usually hash linked list of hash trees (draw picture on board)




## SHA-256

- Merkle-Damgard Transform: change arbitrary length to fixed length - compression function
- Takes input length $m$ and output length $n$.
  1. Split input into blocks of length $m - n$.
  2. Pass each block in with output of previous block
  3. Use IV (initialization vector) for first block
  4. Return last block's output
  5. Padding:
     (a) Append 1 to the end
     (b) Append $k$ bits of 0 such that $k$ is the smallest integer that fulfills $(l + 1 + k)\ mod\ 512 = 448$; $l$ is length of message.
     (c) Add length $l < 2^{64}$ represented with 64 bits, added to the end.
     (d) Message is *always* padded.
- Feistel Network/Cipher (2 rounds; SHA is 8)
  1. F is the round function, and K0, K1, ... Kn are the keys for rounds 0, 1, ... n.
  2. Split plaintext to 2 blocks L0 and R0
  3. For reach round i = 0, 1, ..., n, compute:
     $$L_{\mathrm{i}+1} = R_{\mathrm{i}}$$
     $$R_{\mathrm{i}+1} = R_{\mathrm{i}+1} \oplus F(R_{\mathrm{i}}, K_{\mathrm{i}}).$$
  4. Ciphertext is $(R_{\mathrm{n}+1}, L_{\mathrm{n}+1})$
- After padding from above, split into smaller parts, XOR with each other, left bitshift.




Other Sources: (More SHA) `http://www.quadibloc.com/crypto/mi060501.htm`