Cryptographic Hash Functions (and Rainbow Tables)

Kevin Ye

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Questions

1. Who created MD5?
2. How many bits are in the output of MD5?
3. Is a reduction function an inverse of a hash function?

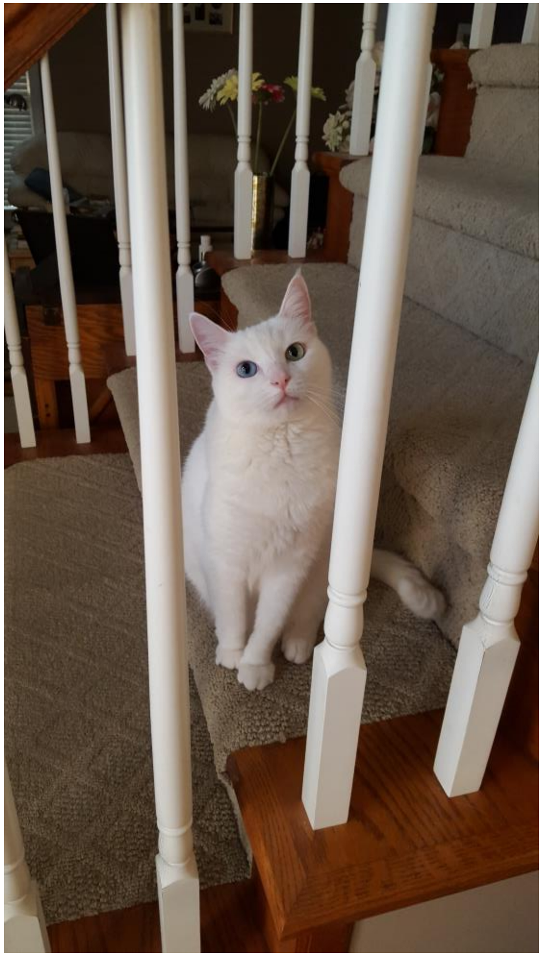
Kevin Ye

- Masters in CS
- Coursework only
- Intern at Siemens Molecular Imaging
- Interests:
  - Data Mining
  - Applications Development
  - *Haven't taken any crypto yet*

Kevin Ye (in the wild)

- Hometown:
  - Johnson City, TN
  - Overland Park, KS
- Hobbies:
  - Cook, bake, video games, play airsoft
- Projects:
  - Homemade furnace

Pets



Furnace

- Mark II:
  - Charcoal-fired furnace
  - Temperature > 1100℃
- Casting parts still WIP

Outline

- Overview – Definitions
- History
- Algorithms
- Applications
- Implementation
- Issues
- Discussion

Focus

- There are many cryptographic hash functions
- Going to focus on MD5

Definitions

Definitions

- Hash function – "projects a value from a set with many (or even infinite number of) members to a value from a set with a fixed number of (fewer) members".*
- One-way function – given H(x) = h, cannot find x given h.

*http://mathworld.wolfram.com/HashFunction.html

Definitions

- Cryptographic Hash Function – distributes output evenly across output space and resists collisions [1].
  - Targeted collisions
  - Pre-image resistance
  - Free collisions

Targeted Collision

- Given x
- Find x' such that $H(x) = H(x')$
- should require 2^n hashes
- n = bits in output[1]

Pre-Image Resistance

- Given y
- Finding x such that $H(x) = y$
- should require 2^n hashes.
- Different from Targeted Collision [1]
  - Breaking Pre-Image Resistance -> breaking targeted collision
  - Reverse is not true

Free Collisions

- Find x and x' such that:

$$x \neq x$$
$$H(x) = H(x')$$

- Should require 2^n hashes [1].

History

History

- 1979 – Ralph Merkle publishes the Merkle Damgard construction in his PhD thesis [4]
- 1989 – MD2 is published by Ron Rivest [5]
- 1990 – MD4 is published by Ron Rivest [6]
- 1992 – MD5 is published by Ron Rivest[7]
- 1995 – The NIST republishes SHA-0 as SHA-1 [8]

History

- 1996 – Non-fatal flaws are found in MD5 [9]
- 2003 – A team from Shangdong University and Rijmen and Oswald publish papers with collision pairs for SHA-1 [10]
- 2008 – A team at the 25th Annual Chaos Communication Congress create a rogue CA certificate[11]

Applications

Cryptographic Hash Function Uses

- Digital Signatures
- Proof of work
- Cryptographic Commitments [1]

MD5 Checksum

- Check integrity of downloaded file
- “Avalanche” – small change in file changes hash significantly [7].
- MD5 of my source code
- 61514232255154dbf020db99b1dab9e3f
- Added apostrophe
- 5977773600dffc8b5705c88920b9b181

Bloom Filters

- Can tell you if an item is in a set
- Always correct if item is not in set
- High probability of being correct if item is in a set [13]

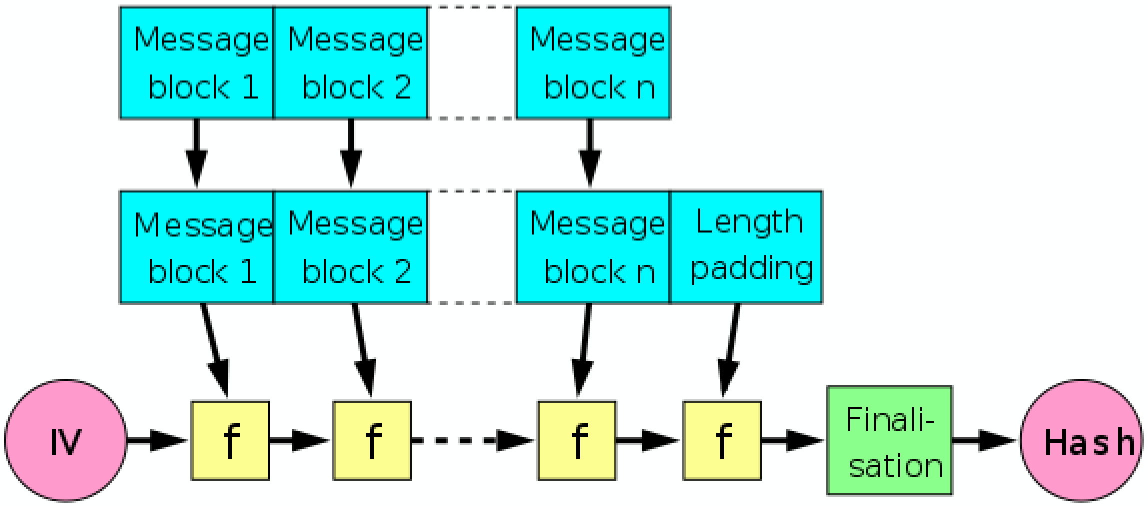
MinHash

- Metric for approximating how the similarity of two sets
- Jaccard Similarity. For two sets A and B, similarity is defined by $\frac{|A \cap B|}{|A \cup B|}$.
- Hash each element and look at min-hash
- Probability of this hash being equal is same as Jaccard Similarity

Algorithms

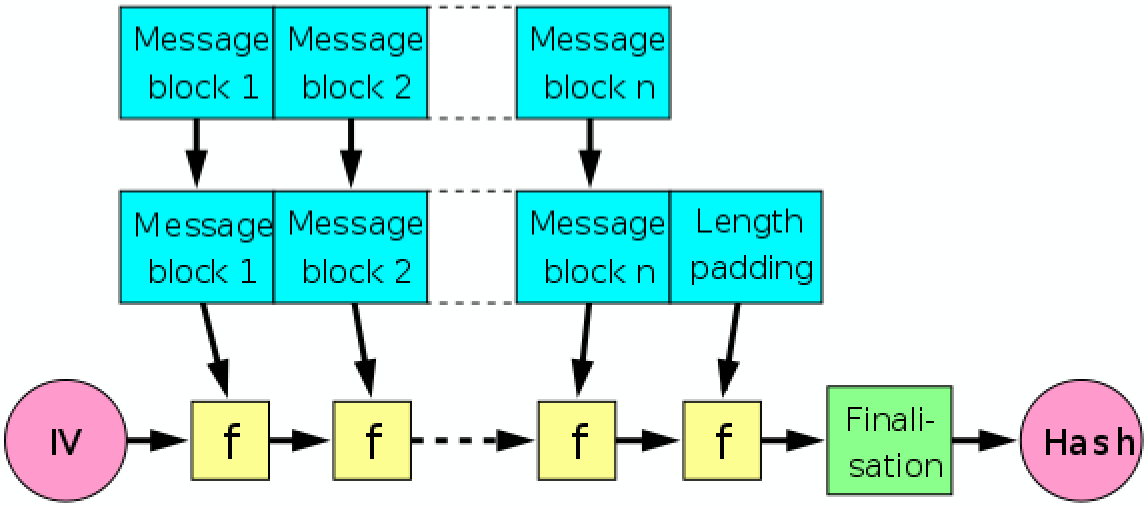
Merkle-Damgard Construction

- Use a compression function (limited input size) $f: \{0,1\}^L \rightarrow \{0,1\}^m$ [1][3].
- Generate hash function $h: \{0,1\}^* \rightarrow \{0,1\}^m$



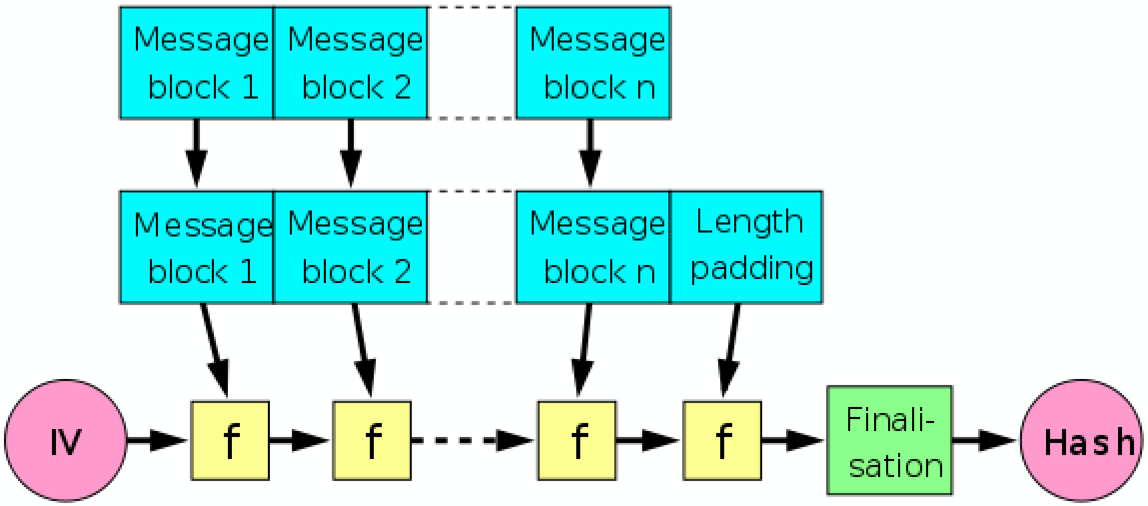
Merkle-Damgard Construction

- IV is a fixed value
- Divide input into blocks of size B
- B = ceil(L/n)



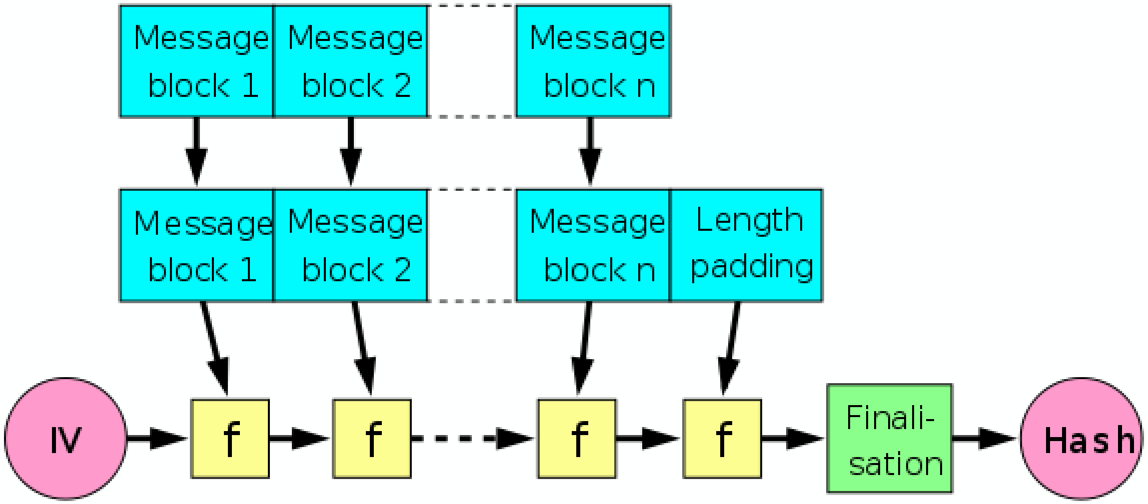
Merkle-Damgard Construction

- Pad input to multiple of n
- Each step, $z_i = f(z_{i-1} || x_i)$
- $H(x) = z_{B+1}$



Merkle-Damgard Construction

- IV is some default value
- $IV = z_0 = 0^n$



Merkle-Damgard Construction Security

- Theorem:
- If $f: \{0, 1\}^L \rightarrow \{0,1\}^m$ is collision resistant, then the Merkle-Damgard construction $h: \{0,1\}^* \rightarrow \{0,1\}^m$ is collision resistant [3].

MD5

- Message Digest 5 published by Ron Rivest 1992 [7]
- Digest (output) size = 128 bits
- Block size = 512
- Word size = 32 bits
- Built upon Merkle-Damgard Transformation

MD5

- Message m
- m is b bits long
- m is padded such that $b \equiv 448\ (mod\ 512)$
- Add 1, then add 0's till padding length is met [9].

MD5

- Initialize buffers A, B, C, D [9]
- A: 01 23 45 67
- B: 89 ab cd ef
- C: fe dc ba 98
- D: 76 54 32 10
- This is your IV

MD5

- Define four functions in [9]
- $F(X, Y, Z) = XY \vee (\neg X)Z$
- $G(X, Y, Z) = XZ \vee Y(\neg Z)$
- $H(X, Y, Z) = X \; xor \; Y \; xor \; Z$
- $I(H, X, Y, Z) = Y xor \; (X \vee (\neg Z))$

Rainbow Table

- Hardcoded tables
- T = [1…64] where T[I] = $floor(4294967296 \cdot \sin(I))$ where I is in radians [9].

Image credit: https://en.wikipedia.org/wiki/Rainbow#/media/File:Double-alaskan-rainbow.jpg

MD5 Algorithm

1. Pad input
2. Intialize buffers/tables
3. Divide message into 16-word blocks
4. Copy previous block
5. Perform message digest passes (4)
6. Repeat 4-5 until all of message is processed

Message Digest

```
/* Round 1. */
/* Let [abcd k s i] denote the operation
     a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
[ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
[ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
[ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]

/* Round 2. */
/* Let [abcd k s i] denote the operation
     a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
[ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
[ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
[ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]

/* Round 3. */
/* Let [abcd k s t] denote the operation
     a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
[ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
[ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
[ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]

/* Round 4. */
/* Let [abcd k s t] denote the operation
     a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
[ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
[ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
[ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]

/* Then perform the following additions. (That is increment each
   of the four registers by the value it had before this block
   was started.) */
A = A + AA
B = B + BB
C = C + CC
D = D + DD
```

How to Attack MD5

- Hash functions are one way
- Finding plaintext (pre-image attack) is $O(2^n)$
- Brute-force
- Lookup tables
- Rainbow Tables
- Find collisions (targeted collision attack)

Brute Force

- Given y,
- Find H(x) = y
- Iterate through all possible plaintexts and calculate hashes
- When hash matches, plaintext has been found
- $O(2^n)$ computational complexity

Lookup Table

- Given y
- Find H(x) = y
- Precompute all possible hashes (same as brute force)
- $O(2^n)$ computational complexity
- $O(2^n)$ memory complexity
- $O(1)$ lookup(!)

Rainbow Table

- Philippe Oeschlin - 2003
- Lookup table with a twist
- *Time/memory tradeoff*
- Use hash chaining to reduce memory footprint
- Cost: increase computation at lookup [12]

Rainbow Table

- “For a cryptosystem having N keys, this method can recover a key in $N^{\frac{2}{3}}$ operations using $N^{\frac{2}{3}}$ words of memory” [12].

Hash Chaining

- Find reduction function – generate key from ciphertext
- NOT inverse hash function
- $k_i \xrightarrow{S_k(P_0)} C_i \xrightarrow{R(C_i)} k_{i+1}$ [12]
- Chain hashing and reduction
- Multiple reduction functions

Rainbow Table Generation

- Lookup table:

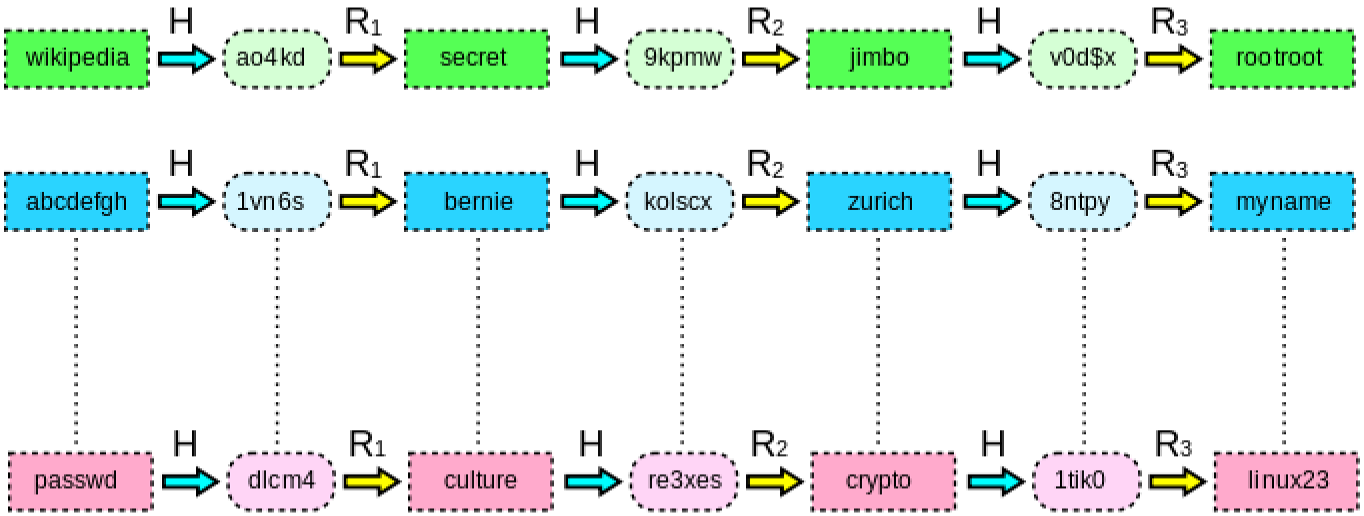
$$\color{red}{aaaa} \rightarrow 53abf8$$

- Rainbow table:

$$\boxed{\underset{P}{\color{red}{aaaa}}} \rightarrow \underset{H}{53abf8} \rightarrow \underset{R}{\color{orange}{cake}} \rightarrow \boxed{\underset{H}{42abcd}}$$

Rainbow Table Example

- Multiple reduction functions
- Reduces chain merging

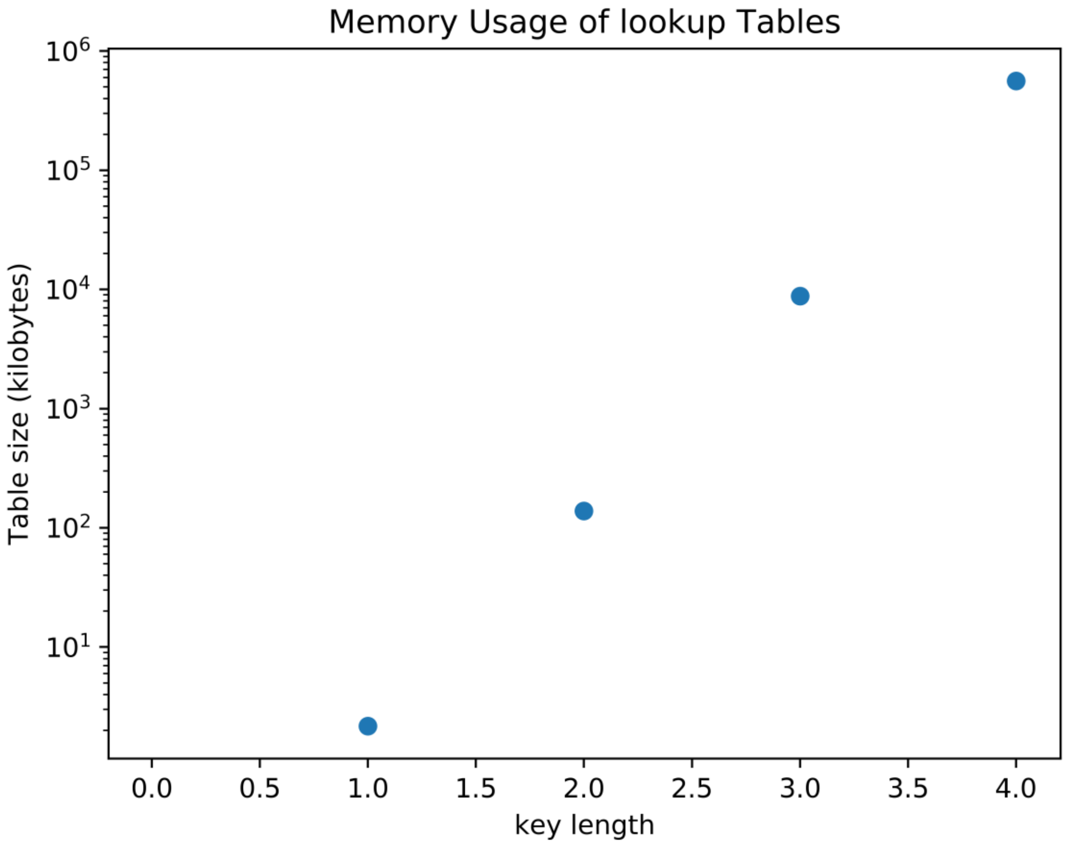


Implementation

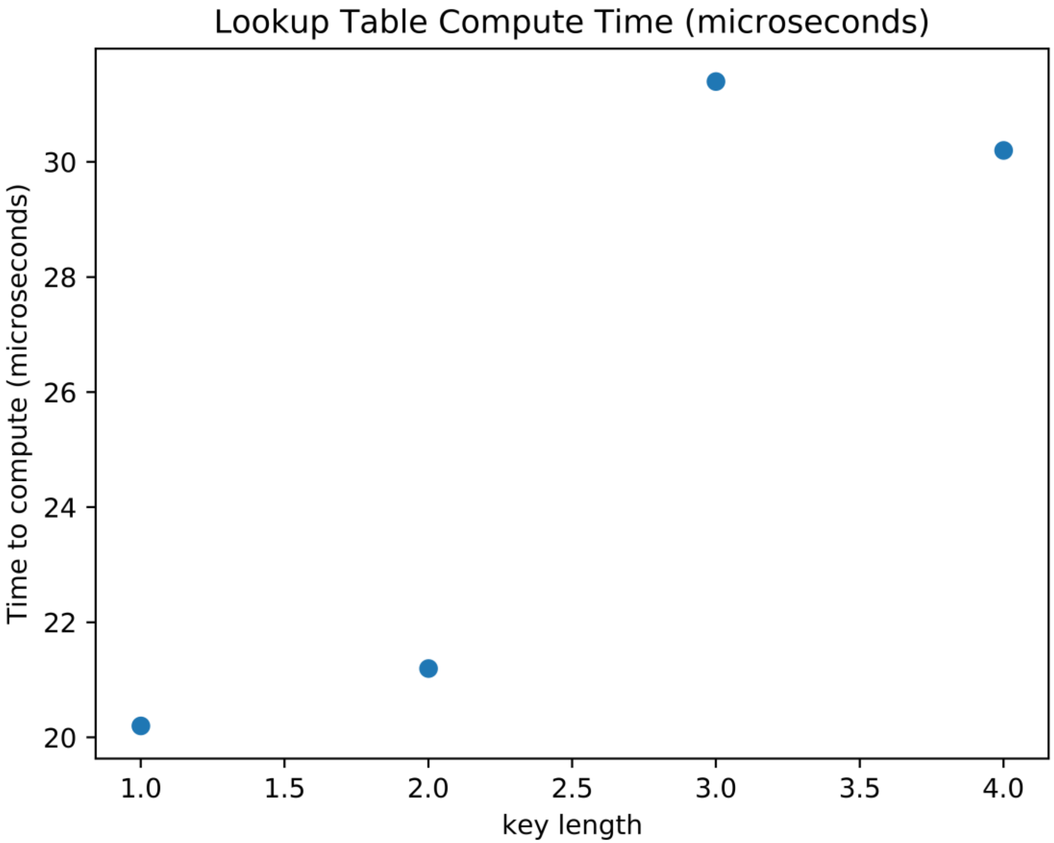
Simple Rainbow Table

- Generate lookup tables for mixed alphanumeric strings length 4
- Generate 1-hash chain rainbow tables
- Do brute-force lookup

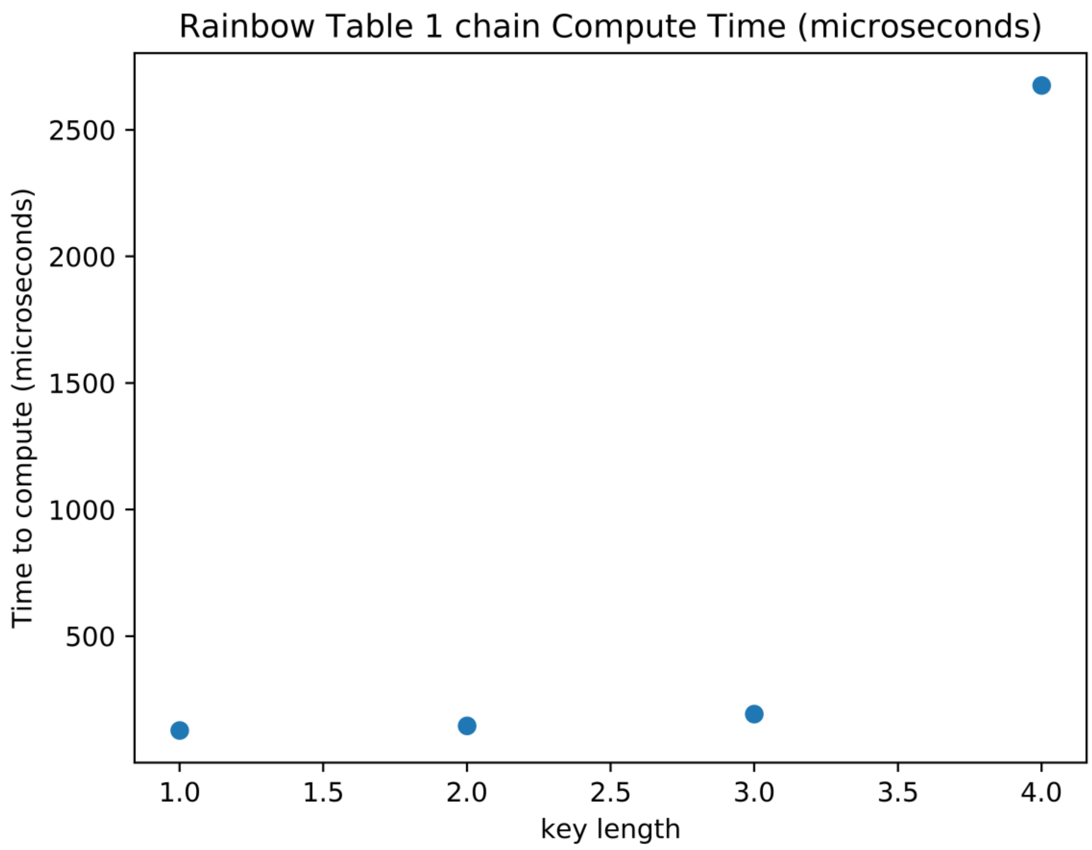
Memory Usage



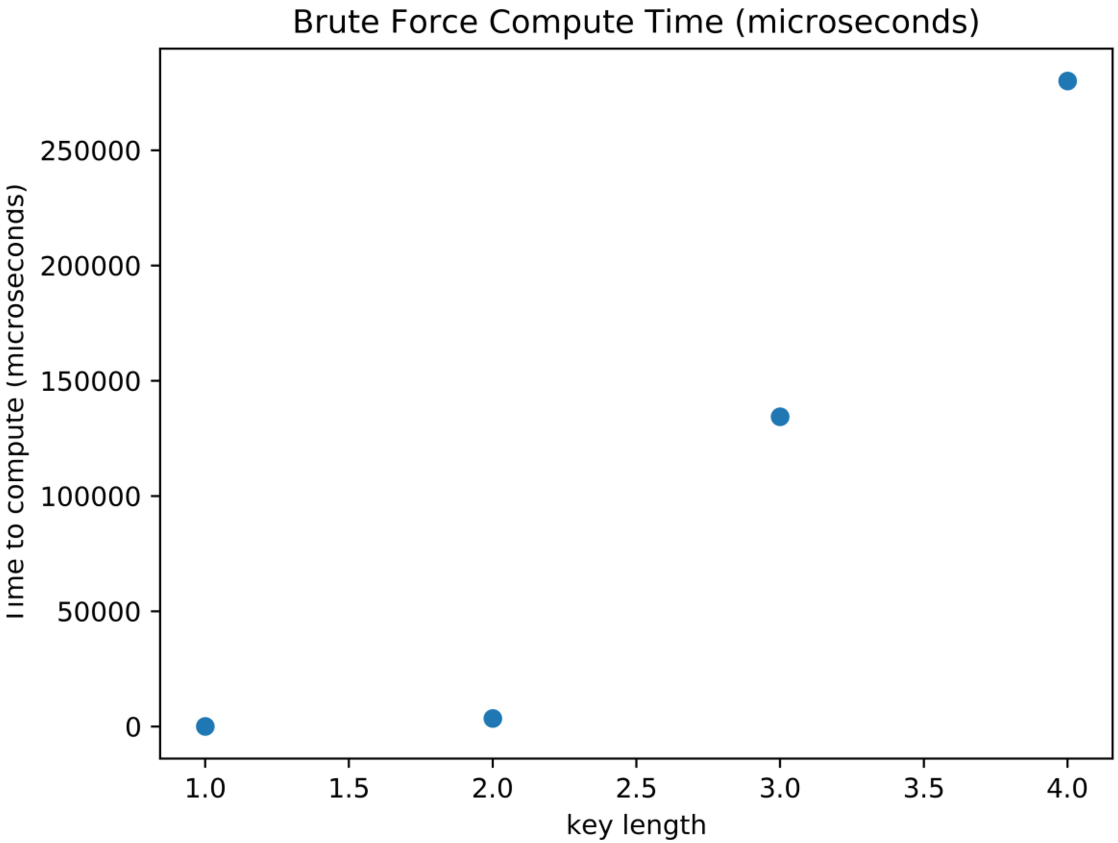
Compute Time - Lookup



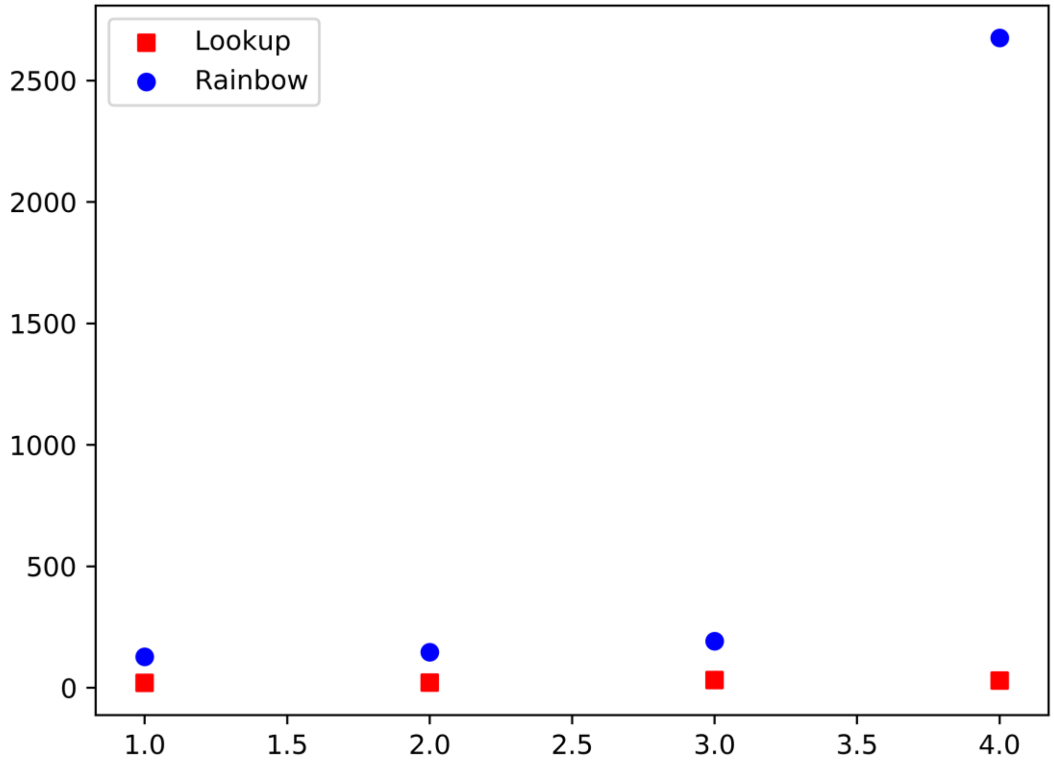
Compute Time - Rainbow



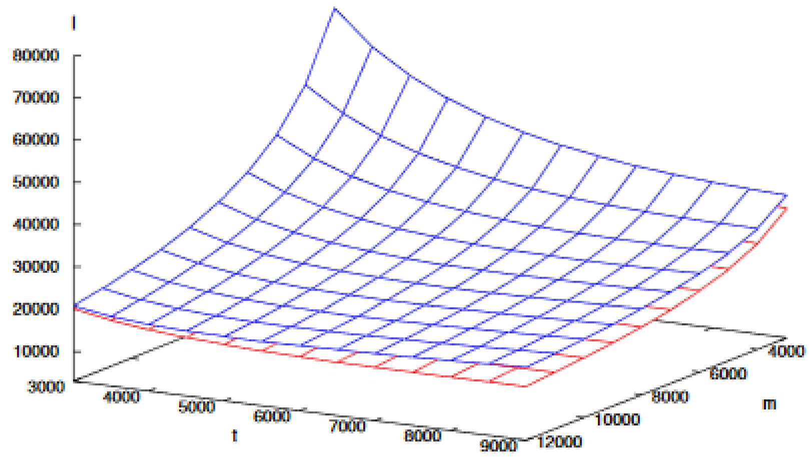
Compute Time – Brute Force



Comparison – Rainbow vs. Lookup



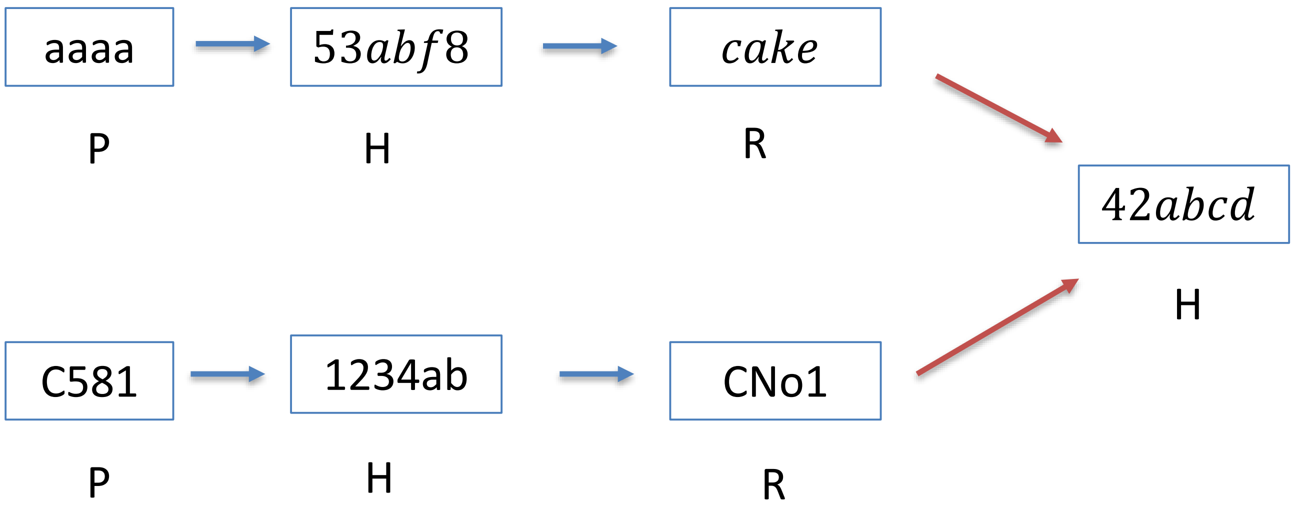
Success > 0.999 and min(Memory <1.4GB, Time < 110)



* **Fig. 3.** Comparison of the success rate of classical tables and rainbow tables. The upper surface represents the constraint of 99.9% success with classical tables, the lower surface is the same constraint for rainbow tables. For rainbow tables the scale has been adjusted to allow a direct comparison of both types of tables $m \rightarrow \frac{m'}{t}, \ell \rightarrow \frac{\ell'}{t}$

Issues

Chain Merging

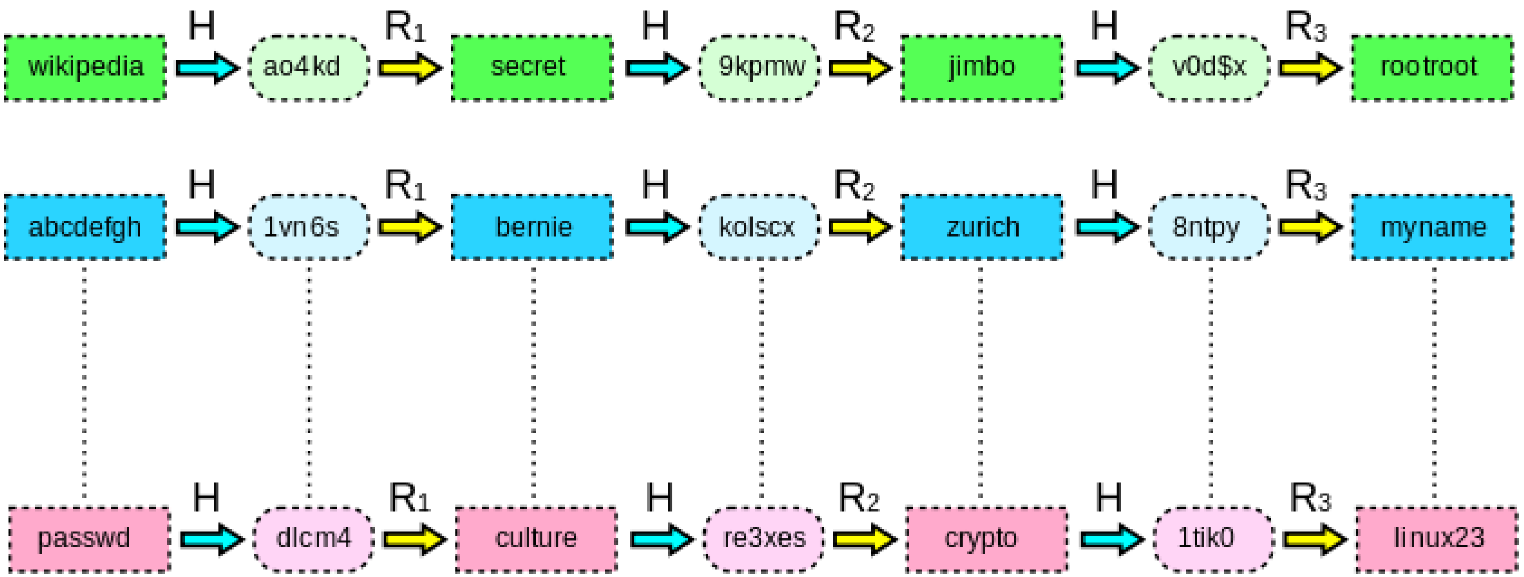


False Alarms

- Matching endpoint found
- Key is not in table
- Key chain has merged
- Have to generate more chains

Missing Hashes

- Hash "myname"
- Result isn't in the table?



References

- [1] M. Schuchard, Class Lecture, “Cryptographically Secure Hashing.” COSC483/583, Tickle College of Engineering, University of Tennessee Knoxville, 2017.
- [2] S. Northcutt, “Hash Functions”, Sans Technical Institute, Jan. 10, 2018. [Online]. Available: https://www.sans.edu/cyber-research/security-laboratory/article/hash-functions. [Accessed: Apr. 26, 2018]
- [3] M. Kantarcioglu, Class Lecture, “Cryptographic Hash Functions”, CS6377, Department of Computer Science, University of Texas Dallas, Feb 2012.
- [4] R. C. Merkle, “Secrecy, Authentication, and Public Key Systems”, Stanford Electronics Laboratories, Department of Electrical Engineering, Stanford Univeristy, Stanford, CA, USA, 1979.
- [5] R. Rivest, “The MD2 Message-Digest Algorithm”, *RSA Laboratories,* Apr. 1992. [Online]. Available: https://tools.ietf.org/html/rfc1319. [Accessed: Apr. 26, 2018].
- [6] R. Rivest, “The MD4 Message-Digest Algorithm”, MIT Laboratory for Computer Science and RDSA Data Security*,* Apr. 1992. [Online]. Available: https://tools.ietf.org/html/rfc1320. [Accessed: Apr. 26, 2018].
- [7] R. Rivest, “The MD5 Message-Digest Algorithm”, MIT Laboratory for Computer Science and RDSA Data Security*,* Apr. 1992. [Online]. Available: https://tools.ietf.org/html/rfc1321. [Accessed: Apr. 26, 2018].
- [8] P. A. DesAutels, “SHA1 Secure Hash Algorithm – Version 1.0”, W3C, Oct. 1997. [Online]. Available: https://www.w3.org/PICS/DSig/SHA1_1_0.html. [Accessed: Apr. 26, 2018].
- [9] X. Wang, and H. Yu, *How to break MD5 and other hash functions*, EUROCRYPT’05 Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Functions, May 22-26, 2005, Springer-Verlag Berlin, Heidelberg, Germany.
- [10] B. Schneier, “SHA-1 Broken”, *Schneier on Security*, Feb. 15, 2005. [Online]. Available: https://www.schneier.com/blog/archives/2005/02/sha1_broken.html. [Accessed: Apr. 26, 2018].
- [11] A. Sotirov, M. Stevens, et al. “MD5 considered harmful today”, Technische Universiteit Eindhoven, Dec. 30, 2008. [Online]. Available: http://www.win.tue.nl/hashclash/rogue-ca/. [Accessed: Apr. 26, 2018].
- [12] P. Oeschlin, “Making a Faster Cryptanalytic Time-Memory Trade-Off”, Annual International Cryptology Conference Advances in Cryptology – CRYPTO 2003, pp 617-630, 2003.
- [13] J. Plank, Class Lecture, “Bloom Filters”, Tickle College of Engineering, University of Tennessee Knoxville, Feb. 2016
- [14] J. Plank, Class Lecture, “MinHash”, Tickle College of Engineering, University of Tennessee Knoxville, Oct. 2017.

Discussion

- Favourite hash function?
  - E.g. SHA-2, MD2, the hash function that turns potatoes into hashbrowns
- Favourite attack?
- Do you trust MD5 checksums?

Questions

1. Who created MD5?
2. How many bits is the output of MD5?
3. Is a reduction function an inverse of a hash function?