Cryptographic Hashes

Yan Huang

Credits: David Evans, CS588
Recap: CPA

1. \( k \leftarrow \text{KeyGen}(1^n) \). \( b \leftarrow \{0, 1\} \). Give \( \text{Enc}(k, \cdot) \) to \( \mathcal{A} \).
2. \( \mathcal{A} \) chooses as many plaintexts as he wants, and receives the corresponding ciphertexts via \( \text{Enc}(k, \cdot) \).
3. \( \mathcal{A} \) picks two plaintexts \( M_0 \) and \( M_1 \). (Picking plaintexts for which \( A \) previously learned ciphertexts is allowed!)
4. \( \mathcal{A} \) receives the ciphertext of \( M_b \), and continues to have accesses to \( \text{Enc}(k, \cdot) \).
5. \( \mathcal{A} \) outputs \( b' \).

\( \mathcal{A} \) wins if \( b' = b \).
Recap: CPA

For all efficient adversary $\mathcal{A}$,

\[ | \Pr[ b=b' ] - 1/2 | \text{ is “negligible”} \]
Recap: Message Integrity Game

1. \( k \leftarrow \text{Gen}(1^n) \).

2. \( \mathcal{A} \) is given polynomial time and an oracle access to query \( \text{Mac}(k, \cdot) \). Let \( t_i = \text{Mac}(k, m_i) \) and \( Q = \{(m_1, t_1), \ldots, (m_q, t_q)\} \).

3. \( \mathcal{A} \) outputs \((m, t)\).

\( \mathcal{A} \) wins the game if \( \text{Vrfy}(m, t) = 1 \) and \((m, t) \notin Q\).
Recap: Message Integrity

$(\text{Gen}, \text{Mac}, \text{Vrfy})$ --- a message authentication code scheme.

Def: $(\text{Gen}, \text{Mac}, \text{Vrfy})$ is a **Secure Message Authentication Code** if for all “efficient” $\mathcal{A}$:

$$\text{Adv}_{\text{Mac}}[\mathcal{A}] = \Pr[\text{Chal. outputs 1}] \text{ is “negligible.”}$$
Normal CS Hashing

\[ H (\text{char } s[]) = (s[0] - 'a') \mod 10 \]
Magic Function $f$

• One Way:
  – For every integer $x$, easy to compute $f(x)$
  – Given $f(x)$, hard to find any information about $x$

• Collision Resistant:
  – “Impossible” to find pair $(x, y)$ where $x \neq y$ and $f(x) = f(y)$
Regular Hash Functions

1. Many-to-one: maps a large number of values to a small number of hash values

2. Evenly distributed: for typical data sets,
   \[ \Pr (H(x) = n) = \frac{1}{N} \]
   where \( N \) is the number of hash values and \( n = 0 .. N – 1 \).

3. Efficient: \( H(x) \) is easy to compute.

   How well does
   \[ H(\text{char } s[]) = (s[0] - 'a') \mod 10 \]
   satisfy these properties?
Cryptographic Hash Functions

Collision resistance (even for malicious adversary):

**Preimage** resistance: for a uniformly chosen $v$, it is hard to find $x$ such that $H(x) = v$.

**Second-preimage** resistance: given $x$, it is hard to find $y \neq x$ such that $H(y) = H(x)$.

**Collision resistance:** it is hard to find any $x$ and $y$ such that $y \neq x$ and $H(x) = H(y)$. 
Merkle-Damgård Transform

\[ m = (m_1, m_2, m_3) \]

Compressing by a single bit is as easy (or as hard) as compressing by an arbitrary amount.
Example Real World Hash Functions

• MD5 is broken
• SHA-1 is phasing out
• SHA-256
  • M-D Transform
  • 256-bit output
  • 512-bit block size
  • 64 rounds
  • a combination of AND, OR, XOR, ADD, RotR, ShR
  • 128 bit security

One iteration in a SHA-2 family compression function. The blue components perform the following operations:

\[
\begin{align*}
    \text{Ch}(E, F, G) &= (E \land F) \oplus \neg(E \land G) \\
    \text{Ma}(A, B, C) &= (A \land B) \oplus (A \land C) \oplus (B \land C) \\
    \Sigma_0(A) &= (A \gg 2) \oplus (A \gg 13) \oplus (A \gg 22) \\
    \Sigma_1(E) &= (E \gg 6) \oplus (E \gg 11) \oplus (E \gg 25)
\end{align*}
\]

The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256. The red is addition modulo $2^{32}$. 
Authentication through Hash and Mac

If (Gen’, Mac’, Vrfy’) is a MAC for fixed length messages,

• Gen: Gen’
• Mac: $t = \text{Mac’}(k, H(m))$
• Vrfy: outputs 1 if and only if $Vrfy’(k, H(m), t) = 1$
Other Applications of Hashing

- Fingerprinting
- Authenticated Data Structures
- Coin tossing
IOU Request Protocol

Alice

Bob

Judge can subpoena for $K$

$K$

$m$

$Mac_K[H(m)]$
Attacking IOU Request Protocol

Alice

K

$m_1$

$Mac_K[H(m_1)]$

Bob

$m_2$

$Mac_K[H(m_1)]$

Bob picks $m_1$ and $m_2$ such that $H(m_1) = H(m_2)$.

Judge
can subpoena for $K$
Finding $m_1$ and $m_2$

Bob generates different agreeable $m_1$ messages:

I, \{Alice | Alice Hacker | Alice P. Hacker | Ms. A. Hacker\}, \{owe | agree to pay\} Bob \{the sum of | the amount of\} \{$2 | $2.00 | 2 dollars | two dollars\} \{by | before\} \{January 1^{\text{st}} | 1 Jan | 1/1 | 1-1\} \{2016 | 2016 AD\}.

How many different-text messages are there?
Finding $m_1$ and $m_2$

Bob generates $2^{10}$ different agreeable $m_2$ messages:

I, {Alice  |  Alice Hacker  |  Alice P. Hacker  |  Ms. A. Hacker}, {owe  |  agree to pay} Bob {the sum of  |  the amount of} {$2$ quadrillion  |  $2000000000000000000$  |  2 quadrillion dollars  |  two quadrillion dollars} {by  |  before} {January 1$^{st}$  |  1 Jan  |  1/1  |  1-1} {2016  |  2016 AD}.
Bob’s Quadrillionaire Plan

• For each message $m_{1,i}$ and $m_{2,i}$, Bob computes $H(m_{1,i})$ and $H(m_{2,i})$.

• If $H(m_{1,i}) = H(m_{2,j})$ for some $i$ and $j$, Bob sends Alice $m_{1,i}$, gets $\text{Mac}_K[H(m_{1,i})]$ back.

• Bob sends the judge $m_{2,j}$ and $\text{Mac}_K[H(m_{1,i})]$. 
Chances of Success

• Assume the Hash function $H$ is good (uniform randomly distributed outcome)

What is the probability that $H(m_{1,i}) = H(m_{2,j})$ for some $i$ and $j$?
Birthday “Paradox”

What is the probability that two people in this room have the same birthday?
Birthday Paradox

Ways to assign $k$ different birthdays without duplicates:

$$N = 365 \times 364 \times \ldots \times (365 - k + 1)$$

$$= 365! / (365 - k)!$$

Ways to assign $k$ different birthdays with possible duplicates:

$$D = 365 \times 365 \times \ldots \times 365 = 365^k$$
Birthday “Paradox”

Assuming real birthdays assigned randomly:

\[ \frac{N}{D} = \text{probability there are no duplicates} \]

\[ 1 - \frac{N}{D} = \text{probability there is a duplicate} \]

\[ = 1 - \frac{365!}{(365 - k)!365^k} \]
Generalizing Birthdays

\[ P(n, k) = 1 - \frac{n!}{(n - k)! \, n^k} \]

Given \( k \) random selections from \( n \) possible values, \( P(n, k) \) gives the probability that there is at least 1 duplicate.
Applying to Birthdays

- For $n = 365$, $k = 20$:
  \[ P(365, 20) \approx 0.4114 \]
- For $n = 365$, $k = 40$:
  \[ P(365, 40) \approx 0.8912 \]
Is 128 bits enough for hash output?

- For $n = 2^{128}, k = 2^{40}$: $P (2^{128}, 2^{40}) > 1.77 \times 10^{-15}$
- For $n = 2^{128}, k = 2^{60}$: $P (2^{128}, 2^{60}) > 1.95 \times 10^{-3}$
- For $n = 2^{128}, k = 2^{65}$: $P (2^{128}, 2^{60}) > 0.86$

A 10 thousand core machine can brute-force $2^{65}$ hashes in about 50 days (assuming $10^9$ hashes per second on each core).

Assumes you hash function is perfect (e.g., MD5 was not broken merely as a result of bruteforce).
A Most Disturbing Program!


#!/usr/bin/perl -w
use strict;
use Digest::MD5 qw(md5_hex);

# Create a stream of bytes from hex.
my @bytes1 = map {chr(hex($_))} qw(d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c 2f ca b5 87
    12 46 7e ab 40 04 58 3e b8 fb 7f 89 55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 71 41 5a 08 51 25 e8 f7
cd c9 9f d9 1d bd f2 80 37 3c 5b d8 82 3e 31 56 34 8f 5b ae 6d ac d4 36 c9 19 c6 dd 53 e2 b4 87 da
    03 fd 02 39 63 06 d2 48 cd a0 e9 9f 33 42 0f 57 7e e8 ce 54 b6 70 80 a8 0d 1e c6 98 21 bc b6 a8 83
    93 96 f9 65 2b 6f f7 2a 70);

my @bytes2 = map {chr(hex($_))} qw(d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c 2f ca b5 07
    12 46 7e ab 40 04 58 3e b8 fb 7f 89 55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 f1 41 5a 08 51 25 e8 f7
cd c9 9f d9 1d bd 72 80 37 3c 5b d8 82 3e 31 56 34 8f 5b ae 6d ac d4 36 c9 19 c6 dd 53 e2 34 87
da 03 fd 02 39 63 06 d2 48 cd a0 e9 9f 33 42 0f 57 7e e8 ce 54 b6 70 80 28 0d 1e c6 98 21 bc b6 a8
    83 93 96 f9 65 ab 6f f7 2a 70);

# Print MD5 hashes
print md5_hex(@bytes1), "\n", md5_hex(@bytes2), "\n";
    79054025255fb1a26e4bc422aef54eb4
    79054025255fb1a26e4bc422aef54eb4