CS 553

CRYPTOGRAPHY

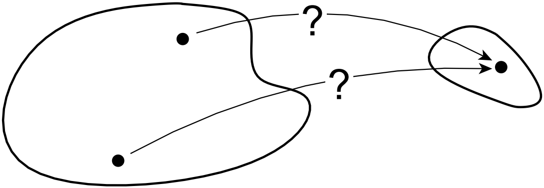
Lecture 19

Hash Collisions &
The Birthday Paradox

Instructor
Dr. Dhiman Saha

Image Source: Google

- Finding collisions
  - Get $x \neq x'$ such that $h(x) = h(x')$



Can we do better?

- We know how second preimage routine can be used to find collision
- But complexity is 2^n

Much easier to find matching objects than finding a particular object.

- A very famous problem in this regard:

The fundamental idea behind collision algorithms

The Birthday Problem
or
The Birthday Paradox

Consider the following questions

In a random group of 40 people:

- What is the probability that someone has the **same** birthday as **you**?
- What is the probability that at least two people share the **same** birthday?

Any guesses

- Are the answers to these questions similar
- Or very different

Consider the following questions

In a random group of 40 people:

- What is the probability that someone has the **same** birthday as **you**?
- What is the probability that at least two people share the **same** birthday?

Any guesses

- Are the answers to these questions similar
- **Or very different** ✓

What is the probability that someone has the **same** birthday as **you**?

A Common Mistake[1]

- Probability of one person sharing your birthday $= \frac{1}{365}$

[1] Think how this scales with the number of people

What is the probability that someone has the **same** birthday as **you**?

A Common Mistake[1]

- Probability of one person sharing your birthday $= \frac{1}{365}$
- Then in a crowd of 40 people, the probability of someone having your birthday is approximately

$$\frac{40}{365} \approx 11\% \quad \text{Overestimate!!!}$$

[1] Think how this scales with the number of people

What is the probability that someone has the **same** birthday as **you**?

A Common Mistake[1]

- Probability of one person sharing your birthday $= \frac{1}{365}$
- Then in a crowd of 40 people, the probability of someone having your birthday is approximately

$$\frac{40}{365} \approx 11\% \quad \text{Overestimate!!!}$$

- Double counts the occurrences of more than one person in the crowd sharing your birthday.

[1] Think how this scales with the number of people

The Right Way to Count

Consider the complementary event

None of the people share your birthday.

$$
\begin{aligned}
\Pr\left(\begin{array}{l}\text{someone has}\\ \text{your birthday}\end{array}\right) &= 1 - \Pr\left(\begin{array}{l}\text{None of the 40 people}\\ \text{have your birthday}\end{array}\right)\\
&= 1 - \prod_{i=1}^{40} \Pr\left(\begin{array}{l}i^{th}\text{ person does not}\\ \text{have your birthday}\end{array}\right)\\
&= 1 - \left(\frac{364}{365}\right)^{40}\\
&\approx 10.4\%
\end{aligned}
$$

Question 2

What is the probability that at least two people share the same birthday?

Again — Right way to count

Compute the probability that all 40 people have **different** birthdays.

- i^{th} person should have a birthday that is different from all of the previous $(i-1)$ peoples birthdays.

How to compute?

- Among the 365 possible birthdays, the previous $(i-1)$ people have taken up $(i-1)$ of them.
- Probability that the i^{th} person has his or her birthday among the remaining $365-(i-1)$ days is

$$\frac{365-(i-1)}{365}$$

$$
\begin{aligned}
\Pr\begin{pmatrix}\text{two people have}\\ \text{the same birthday}\end{pmatrix} &= 1-\Pr\begin{pmatrix}\text{all 40 people have}\\ \text{different birthdays}\end{pmatrix}\\
&= 1-\prod_{i=1}^{40}\Pr\begin{pmatrix}i^{th}\text{ person does not}\\ \text{have the same birthday}\\ \text{as any of the previous}\\ (i-1)\text{ people}\end{pmatrix}\\
&= 1-\prod_{i=1}^{40}\frac{365-(i-1)}{365}\\
&= 1-\frac{365}{365}\cdot\frac{364}{365}\cdot\frac{363}{365}\cdots\frac{326}{365}\\
&\approx 89.1\%
\end{aligned}
$$

Counter-intuitive

Among 40 strangers, there is almost a 90% chance that two of them share a birthday!!!

Why Paradox?

- General assumption that Question 1 & 2 have essentially the same answer.

The Birthday Paradox — To put things in perspective

- It requires only **23** people to have a better than **50%** chance of a matched birthday while
- It takes **253** people to have better than a **50%** chance of finding someone who has your birthday

Can you see the link of the **Birthday Paradox** with the problem of **collision finding** in hash functions?

https://pudding.cool/2018/04/birthday-paradox/

- Let us try to find Pr(Collision) using k out of n messages

Collision Probability in General

Derivation of Pr(Collision) using k out of n message and the relation with the number of messages has been shown in class.

$$\Pr(\text{Collision}) \approx 1 - e^{-\frac{k(k-1)}{2n}} \qquad \leftarrow \text{for large } n$$

- For $\Pr(\text{Collision}) = \frac{1}{2}$

$$k \approx 1.1774\sqrt{n} \qquad \leftarrow \text{for large } k$$

- Given N messages and as many hash values
- Total number of **potential** collisions producible
- Considering each pair of two hash values

$$\binom{N}{2} = \frac{N \times (N-1)}{2} \rightarrow O(N^2)$$

Connect this with the result from the previous slide.

A Comparative Understanding

Preimage Search

N messages only give N candidate preimages

Collision Search

Same N messages give $\approx N^2$ potential collisions

Observe

- With N^2 instead of N there are quadratically more chances to find a solution.
- The complexity of the search is in turn quadratically lower.

The Verdict

In order to find a collision, # messages needed $\rightarrow \sqrt{2^n}$

$$2^{\frac{n}{2}} \text{ instead of } 2^n$$

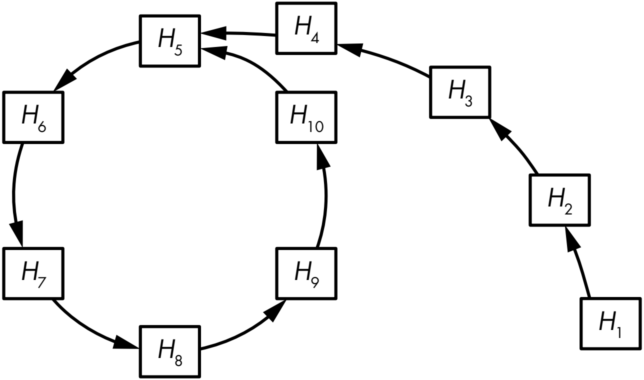
The Naive Birthday Attack

Simplest Way

- Compute $2^{\frac{n}{2}}$ hashes of $2^{\frac{n}{2}}$ arbitrarily chosen messages and store all the message/hash pairs in a list
- Sort the list with respect to the hash value
- Search the sorted list to find two consecutive entries with the same hash value

Complexity

- Huge memory: $2^{\frac{n}{2}}$ message/hash pairs
- Sorting: $O(n2^n)$



$H_{i+1} = \boxed{Hash(H_i)}$ ← Baby Step

$H'_{i+1} = \boxed{Hash(Hash(H'_i))}$ ← Giant Step

Note

Memory requirement is negligible

- The Rho method takes about $2^{\frac{n}{2}}$ operations to succeed
- On average, the **cycle** and the **tail** each include about $2^{\frac{n}{2}}$ hash values
- n is the bit length of the hash values.

 # Hash evaluations to find a collision

$$\geq \left(2^{\frac{n}{2}} + 2^{\frac{n}{2}}\right)$$