Smalltalk: The Birthday Problem

Brian Heinold

Mount St. Mary's University

September 7, 2017

The Birthday Problem: How many people need to be in a room in order for there to be a 50-50 chance that some two people in the room have the same birthday?

The Birthday Problem

The Birthday Problem: How many people need to be in a room in order for there to be a 50-50 chance that some two people in the room have the same birthday?

A Different Problem: How many people have to be in a room with you in order for there to be a 50-50 chance that someone has the same birthday as you?

The Birthday Problem

The Birthday Problem: How many people need to be in a room in order for there to be a 50-50 chance that *some two people in the room have the same birthday*?

A Different Problem: How many people have to be in a room with you in order for there to be a 50-50 chance that *someone has the same birthday as you*?

The Birthday Problem: How many people need to be in a room in order for there to be a 50-50 chance that *some two people in the room have the same birthday*? **23**

A Different Problem: How many people have to be in a room with you in order for there to be a 50-50 chance that *someone has the same birthday as you*? **253**

Why it's so few people

- Imagine we have 10 people in a room with different birthdays. Someone new walks into the room. What is the probability their birthday matches someone else's in the room?

Why it's so few people

- Imagine we have 10 people in a room with different birthdays. Someone new walks into the room. What is the probability their birthday matches someone else's in the room?

$$\frac{10}{365} \approx 3\%$$

- Another person walks in, another 3% chance of a match.

Why it's so few people

- Imagine we have 10 people in a room with different birthdays. Someone new walks into the room. What is the probability their birthday matches someone else's in the room?

$$\frac{10}{365} \approx 3\%$$

- Another person walks in, another 3% chance of a match.
- Another person walks in, another 3% chance of a match.

Why it's so few people

- Imagine we have 10 people in a room with different birthdays. Someone new walks into the room. What is the probability their birthday matches someone else's in the room?

$$\frac{10}{365} \approx 3\%$$

- Another person walks in, another 3% chance of a match.
- Another person walks in, another 3% chance of a match.
- Etc.

Why it's so few people

- Imagine we have 10 people in a room with different birthdays. Someone new walks into the room. What is the probability their birthday matches someone else's in the room?

$$\frac{10}{365} \approx 3\%$$

- Another person walks in, another 3% chance of a match.
- Another person walks in, another 3% chance of a match.
- Etc.
- How many 3% chances can we keep taking before one of them succeeds?

Why it's so few people

- Imagine we have 10 people in a room with different birthdays. Someone new walks into the room. What is the probability their birthday matches someone else's in the room?

$$\frac{10}{365} \approx 3\%$$

- Another person walks in, another 3% chance of a match.
- Another person walks in, another 3% chance of a match.
- Etc.
- How many 3% chances can we keep taking before one of them succeeds?
- And with each new person, that 3% gradually increases.

Another way to think about it

- With 23 people in room, there are $\binom{23}{2} = \frac{23 \cdot 22}{2} = 253$ possible pairs of people.

Another way to think about it

- With 23 people in room, there are $\binom{23}{2} = \frac{23 \cdot 22}{2} = 253$ possible pairs of people.
- A could have a match with B, C, $\ldots$, W
  B could have a match with C, D, $\ldots$, W
  C could have a match with D, E, $\ldots$, W
  Etc.

Another way to think about it

- With 23 people in room, there are $\binom{23}{2} = \frac{23 \cdot 22}{2} = 253$ possible pairs of people.
- A could have a match with $B, C, \ldots, W$
  B could have a match with $C, D, \ldots, W$
  C could have a match with $D, E, \ldots, W$
  Etc.
- In order for there to be no shared birthday, all 253 of those pairs need to work.

Another way to think about it

- With 23 people in room, there are $\binom{23}{2} = \frac{23 \cdot 22}{2} = 253$ possible pairs of people.
- A could have a match with $B, C, \ldots, W$
  B could have a match with $C, D, \ldots, W$
  C could have a match with $D, E, \ldots, W$
  Etc.
- In order for there to be no shared birthday, all 253 of those pairs need to work.
- Things just grow from there:
  - With 50 people there are 1225 possible pairs.
  - With 100 people there are 4950 possible pairs

Let's look at a simulation.

Simulation code

```
from random import choice
from time import sleep

months = ["January", "February", "March","April", "May",
          "June", "July", "August", "September", "October",
          "November", "December"]

days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

days_of_year = []
for i in range(12):
    for j in range(1, days[i]+1):
        days_of_year.append(months[i] + ' ' + str(j))

prev = set()
c = 0
while True:
    x = choice(days_of_year)
    print('{:2d}. {:12s}   {:3.1f}% chance of repeat on next person'\
          .format(c, x, 100*(c+1)/365))
        if x in prev:
        break
    prev.add(x)
    c += 1
    sleep(.3)
```

Simulation 1

```
 0. April 11      0.3% chance of repeat on next person
 1. May 8         0.5% chance of repeat on next person
 2. May 1         0.8% chance of repeat on next person
 3. November 16   1.1% chance of repeat on next person
 4. September 17  1.4% chance of repeat on next person
 5. January 30    1.6% chance of repeat on next person
 6. February 2    1.9% chance of repeat on next person
 7. July 8        2.2% chance of repeat on next person
 8. August 17     2.5% chance of repeat on next person
 9. February 8    2.7% chance of repeat on next person
10. April 11      3.0% chance of repeat on next person
```

Simulation 2

```
 0. August 6        0.3% chance of repeat on next person
 1. January 15      0.5% chance of repeat on next person
 2. April 14        0.8% chance of repeat on next person
 3. February 18     1.1% chance of repeat on next person
 4. July 10         1.4% chance of repeat on next person
 5. August 31       1.6% chance of repeat on next person
 6. September 28    1.9% chance of repeat on next person
 7. July 30         2.2% chance of repeat on next person
 8. March 20        2.5% chance of repeat on next person
 9. December 30     2.7% chance of repeat on next person
10. May 1           3.0% chance of repeat on next person
11. July 23         3.3% chance of repeat on next person
12. April 19        3.6% chance of repeat on next person
13. April 2         3.8% chance of repeat on next person
14. June 19         4.1% chance of repeat on next person
15. October 24      4.4% chance of repeat on next person
16. May 22          4.7% chance of repeat on next person
17. December 9      4.9% chance of repeat on next person
18. July 26         5.2% chance of repeat on next person
19. April 19        5.5% chance of repeat on next person
```

Simulation 3

```
 0. June 9         0.3% chance of repeat on next person
 1. January 15     0.5% chance of repeat on next person
 2. March 16       0.8% chance of repeat on next person
 3. April 4        1.1% chance of repeat on next person
 4. November 8     1.4% chance of repeat on next person
 5. September 26   1.6% chance of repeat on next person
 6. February 13    1.9% chance of repeat on next person
 7. January 10     2.2% chance of repeat on next person
 8. October 24     2.5% chance of repeat on next person
 9. May 23         2.7% chance of repeat on next person
10. January 13     3.0% chance of repeat on next person
11. March 22       3.3% chance of repeat on next person
12. May 11         3.6% chance of repeat on next person
13. January 26     3.8% chance of repeat on next person
14. December 13    4.1% chance of repeat on next person
15. May 11         4.4% chance of repeat on next person
```

Simulation 4

```
 0. February 25    0.3% chance of repeat on next person
 1. June 23        0.5% chance of repeat on next person
 2. September 12   0.8% chance of repeat on next person
 3. January 10     1.1% chance of repeat on next person
 4. May 9          1.4% chance of repeat on next person
 5. September 25   1.6% chance of repeat on next person
 6. October 20     1.9% chance of repeat on next person
 7. January 24     2.2% chance of repeat on next person
 8. April 14       2.5% chance of repeat on next person
 9. April 20       2.7% chance of repeat on next person
10. June 11        3.0% chance of repeat on next person
11. November 10    3.3% chance of repeat on next person
12. March 30       3.6% chance of repeat on next person
...
45. June 8         12.6% chance of repeat on next person
46. February 24    12.9% chance of repeat on next person
47. November 11    13.2% chance of repeat on next person
48. April 26       13.4% chance of repeat on next person
49. March 3        13.7% chance of repeat on next person
50. June 20        14.0% chance of repeat on next person
51. August 6       14.2% chance of repeat on next person
52. January 12     14.5% chance of repeat on next person
53. April 14       14.8% chance of repeat on next perso
```

Probabilities

# in room	Probability of two people sharing a birthday
5	2.7%
10	11.7%
20	41.1%
30	70.6%
40	89.1%
50	97.0%
60	99.4%
70	99.9%
80	99.99%
90	99.999%
100	99.99997%

Probability comparison

# in room	Any two	exactly yours
5	2.7%	1.4%
10	11.7%	2.7%
20	41.1%	5.3%
30	70.6%	7.9%
40	89.1%	10.4%
50	97.0%	12.8%
60	99.4%	15.2%
70	99.9%	17.5%
80	99.99%	19.7%
90	99.999%	21.9%
100	99.99997%	24.0%

Where these probabilities come from

With k people in a room, the probability p of a match is

$$p = 1 - \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdot \cdots \cdot \frac{365-(k-1)}{365}.$$

Where these probabilities come from

With k people in a room, the probability p of a match is

$$p = 1 - \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdot \cdots \cdot \frac{365-(k-1)}{365}.$$

This comes from computing the complement of the event that all the birthdays are different.

Where these probabilities come from

With k people in a room, the probability p of a match is

$$p = 1 - \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdot \cdots \cdot \frac{365-(k-1)}{365}.$$

This comes from computing the complement of the event that all the birthdays are different.

Use the multiplication rule to get that probability.

Some math

- With k people in a room, the probability p of no shared birthdays is

$$\begin{aligned}
p &= 1 - \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdot \cdots \cdot \frac{365-(k-1)}{365} \\
&= 1 - \left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)\left(1 - \frac{3}{365}\right)\cdots\left(1 - \frac{k-1}{365}\right) \\
&\approx 1 - e^{-1/365} e^{-2/365} \cdots e^{-(k-1)/365} \\
&= 1 - e^{-k(k-1)/(2\cdot 365)} \\
&\approx 1 - e^{-k^2/(2\cdot 365)}
\end{aligned}$$

Some math

- With k people in a room, the probability p of no shared birthdays is

$$
\begin{aligned}
p =& 1 - \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdot \dots \cdot \frac{365-(k-1)}{365} \\
=& 1 - \left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)\left(1 - \frac{3}{365}\right) \dots \left(1 - \frac{k-1}{365}\right) \\
\approx& 1 - e^{-1/365} e^{-2/365} \dots e^{-(k-1)/365} \\
=& 1 - e^{-k(k-1)/(2 \cdot 365)} \\
\approx& 1 - e^{-k^2/(2 \cdot 365)}
\end{aligned}
$$

- Invert this to get the number of people needed for there to be a probability p of a repeat:

$$k \approx \sqrt{2 \cdot 365 \ln\left(\frac{1}{1-p}\right)}$$

Generalizing the birthday problem

There's nothing special about birthdays.
If we generate randomly from a set of n items, the probability of a repeat after k items is

$$p = 1 - \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{n-3}{n} \cdot \cdots \cdot \frac{n-(k-1)}{n}$$

Generalizing the birthday problem

There's nothing special about birthdays.
If we generate randomly from a set of n items, the probability of a repeat after k items is

$$p = 1 - \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{n-3}{n} \cdot \dots \cdot \frac{n-(k-1)}{n}$$

The number of things we need to generate before there's a probability p of a repeat is roughly

$$k = \sqrt{2 \cdot n \ln\left(\frac{1}{1-p}\right)}$$

Generalizing the birthday problem

There's nothing special about birthdays.
If we generate randomly from a set of n items, the probability of a repeat after k items is

$$p = 1 - \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{n-3}{n} \cdot \cdots \cdot \frac{n-(k-1)}{n}$$

The number of things we need to generate before there's a probability p of a repeat is roughly

$$k = \sqrt{2 \cdot n \ln\left(\frac{1}{1-p}\right)}$$

And, the most commonly used rule of thumb is that after $\sqrt{n}$ things are generated, repeats are fairly likely.

An example

Generate random numbers from 1 to 1000.

1. What's the probability of seeing some number twice after generating only 10 numbers?

An example

Generate random numbers from 1 to 1000.

1. What's the probability of seeing some number twice after generating only 10 numbers?

$$p = 1 - \frac{999}{1000} \cdot \frac{998}{1000} \cdot \frac{997}{1000} \cdot \dots \cdot \frac{991}{1000} \approx 5\%$$

An example

Generate random numbers from 1 to 1000.

1. What's the probability of seeing some number twice after generating only 10 numbers?

$$p = 1 - \frac{999}{1000} \cdot \frac{998}{1000} \cdot \frac{997}{1000} \cdot \cdots \cdot \frac{991}{1000} \approx 5\%$$

2. About how many will we have to generate before there's a 25% chance of a repeat?

An example

Generate random numbers from 1 to 1000.

1. What's the probability of seeing some number twice after generating only 10 numbers?

$$p = 1 - \frac{999}{1000} \cdot \frac{998}{1000} \cdot \frac{997}{1000} \cdot \cdots \cdot \frac{991}{1000} \approx 5\%$$

2. About how many will we have to generate before there's a 25% chance of a repeat?

$$\sqrt{2 \cdot 1000 \ln\left(\frac{1}{1-.25}\right)} \approx 24$$

An example

Generate random numbers from 1 to 1000.

1. What's the probability of seeing some number twice after generating only 10 numbers?

$$p = 1 - \frac{999}{1000} \cdot \frac{998}{1000} \cdot \frac{997}{1000} \cdot \cdots \cdot \frac{991}{1000} \approx 5\%$$

2. About how many will we have to generate before there's a 25% chance of a repeat?

$$\sqrt{2 \cdot 1000 \ln\left(\frac{1}{1 - .25}\right)} \approx 24$$

3. About how many before a repeat is likely?

An example

Generate random numbers from 1 to 1000.

1. What's the probability of seeing some number twice after generating only 10 numbers?

$$p = 1 - \frac{999}{1000} \cdot \frac{998}{1000} \cdot \frac{997}{1000} \cdot \cdots \cdot \frac{991}{1000} \approx 5\%$$

2. About how many will we have to generate before there's a 25% chance of a repeat?

$$\sqrt{2 \cdot 1000 \ln\left(\frac{1}{1-.25}\right)} \approx 24$$

3. About how many before a repeat is likely?

   Quick estimate: $\sqrt{1000} \approx 32$

This happened to me

- I took my CD library and put it onto my computer—about 5000 songs in total. I set it to randomly play songs. After only about 50 songs, I was hearing repeats.

This happened to me

- I took my CD library and put it onto my computer—about 5000 songs in total. I set it to randomly play songs. After only about 50 songs, I was hearing repeats.

- Let's calculate the chance of a repeat after only 50 songs:

$$p = 1 - \frac{4999}{5000} \cdot \frac{4998}{5000} \cdot \frac{4997}{5000} \cdot \cdots \cdot \frac{4951}{5000} \approx 22\%$$

This happened to me

- I took my CD library and put it onto my computer—about 5000 songs in total. I set it to randomly play songs. After only about 50 songs, I was hearing repeats.
- Let's calculate the chance of a repeat after only 50 songs:

$$p = 1 - \frac{4999}{5000} \cdot \frac{4998}{5000} \cdot \frac{4997}{5000} \cdot \cdots \cdot \frac{4951}{5000} \approx 22\%$$

- Quick estimate: repeats are likely after around $\sqrt{5000} \approx 70$ songs

One more example

- When playing a card game where you are dealt 5 cards, there are about 2.6 million possible hands.

One more example

- When playing a card game where you are dealt 5 cards, there are about 2.6 million possible hands.
- Whatever the next 5-card hand you are dealt, there is only a 1 in 2.6 million chance that you will ever be dealt that exact hand ever again.

One more example

- When playing a card game where you are dealt 5 cards, there are about 2.6 million possible hands.
- Whatever the next 5-card hand you are dealt, there is only a 1 in 2.6 million chance that you will ever be dealt that exact hand ever again.
- But how likely is it that over the course of your life, you will be dealt a hand that you had been dealt at some time in the past?

One more example

- When playing a card game where you are dealt 5 cards, there are about 2.6 million possible hands.
- Whatever the next 5-card hand you are dealt, there is only a 1 in 2.6 million chance that you will ever be dealt that exact hand ever again.
- But how likely is it that over the course of your life, you will be dealt a hand that you had been dealt at some time in the past?
- That's the birthday problem.

One more example

- When playing a card game where you are dealt 5 cards, there are about 2.6 million possible hands.
- Whatever the next 5-card hand you are dealt, there is only a 1 in 2.6 million chance that you will ever be dealt that exact hand ever again.
- But how likely is it that over the course of your life, you will be dealt a hand that you had been dealt at some time in the past?
- That's the birthday problem.
- After roughly $\sqrt{2{,}600{,}000} \approx 1600$ hands, repeats are likely.

The quick estimate $\sqrt{2{,}600{,}000} \approx 1600$ is nice because it gives an order of magnitude for when we should expect repeats. For example:

Probability	**# of hands for a repeat**
5%	517
20%	1078
39%	1603
50%	1899
75%	2685
99%	4894
99.999%	7738

A security example

- Suppose you are designing a system where every user transaction is assigned a random key.

A security example

- Suppose you are designing a system where every user transaction is assigned a random key.
- You decide to use a 64-bit key, which means keys are random numbers between 1 and about 18,000,000,000,000,000,000.

A security example

- Suppose you are designing a system where every user transaction is assigned a random key.
- You decide to use a 64-bit key, which means keys are random numbers between 1 and about 18,000,000,000,000,000,000.
- It seems unlikely that two transactions will ever have the same key.

A security example

- Suppose you are designing a system where every user transaction is assigned a random key.
- You decide to use a 64-bit key, which means keys are random numbers between 1 and about 18,000,000,000,000,000,000.
- It seems unlikely that two transactions will ever have the same key.
- But the birthday problem matters. After only $\sqrt{18{,}000{,}000{,}000{,}000{,}000{,}000} \approx 4$ billion transactions, a repeat is likely.

A security example

- Suppose you are designing a system where every user transaction is assigned a random key.
- You decide to use a 64-bit key, which means keys are random numbers between 1 and about 18,000,000,000,000,000.
- It seems unlikely that two transactions will ever have the same key.
- But the birthday problem matters. After only $\sqrt{18{,}000{,}000{,}000{,}000{,}000} \approx 4$ billion transactions, a repeat is likely.
- If this is a large internet site, 4 billion transactions is quite possible.

Hash functions

- Hash functions are used all over in cryptography and security.

Hash functions

- Hash functions are used all over in cryptography and security.
- Basic idea is that you feed them a string and they return a fixed length output.

Hash functions

- Hash functions are used all over in cryptography and security.
- Basic idea is that you feed them a string and they return a fixed length output.
- MD5 is one well-used hash function that returns 64-bit outputs. Example hashes:

  "smalltalk"
  `90945bf1d2c52618e38eada42b86086b`

  Chapter 1 of *A Tale of Two Cities*
  `ff5eb755a31ea99fca07b9fba8dd1d07`

  Chapter 1 of *A Tale of Two Cities* with first letter changed to *Z* and everything else left intact
  `a5d7c988d5d87d1b28e1d85e77110cd1`

- The output of hash functions is pretty random and even small changes in the input totally change the hash.

Hash functions, continued

- The output of hash functions is pretty random and even small changes in the input totally change the hash.
- For these reasons, hash functions are used as fingerprints.

Hash functions, continued

- The output of hash functions is pretty random and even small changes in the input totally change the hash.
- For these reasons, hash functions are used as fingerprints.
- For instance, the hash of Chapter 1 of *A Tale of Two Cities*, `ff5eb755a31ea99fca07b9fba8dd1d07`, is highly unlikely to be the output of any other string created by humans in history.

Hash functions, continued

- The output of hash functions is pretty random and even small changes in the input totally change the hash.
- For these reasons, hash functions are used as fingerprints.
- For instance, the hash of Chapter 1 of *A Tale of Two Cities*, `ff5eb755a31ea99fca07b9fba8dd1d07`, is highly unlikely to be the output of any other string created by humans in history.
- Why? A 64-bit hash function has $2^{64} \approx$ 18,000,000,000,000,000,000 possible outputs, far more than the number of strings created by humans in history.

Hash functions and the birthday problem

- Hashes are used in digital signatures: A person sending you a document can compute its hash, and you can compute its hash when you get the document. If the hashes match, then you know the document you got is the same as what the person sent.

Hash functions and the birthday problem

- Hashes are used in digital signatures: A person sending you a document can compute its hash, and you can compute its hash when you get the document. If the hashes match, then you know the document you got is the same as what the person sent.
- Birthday problem: If we generate $\sqrt{18 \text{ quintillion}} \approx 4$ billion strings, it is likely that some pair of them have the same MD5 hash.

Hash functions and the birthday problem

- Suppose I want to defraud you:
  - I write up a contract agreeing to pay you $100 for something

Hash functions and the birthday problem

- Suppose I want to defraud you:
  - I write up a contract agreeing to pay you \$100 for something
  - I also write up a fraudulent version agreeing to only pay \$10 for that same thing.

Hash functions and the birthday problem

- Suppose I want to defraud you:
  - I write up a contract agreeing to pay you \$100 for something
  - I also write up a fraudulent version agreeing to only pay \$10 for that same thing.
  - I generate a few billion subtle variations on the \$100 contract (adding extra spaces, words, whatever).

Hash functions and the birthday problem

- Suppose I want to defraud you:
  - I write up a contract agreeing to pay you \$100 for something
  - I also write up a fraudulent version agreeing to only pay \$10 for that same thing.
  - I generate a few billion subtle variations on the \$100 contract (adding extra spaces, words, whatever).
  - I generate a few billion subtle variations on the \$10 contract as well.

Hash functions and the birthday problem

- Suppose I want to defraud you:
  - I write up a contract agreeing to pay you \$100 for something
  - I also write up a fraudulent version agreeing to only pay \$10 for that same thing.
  - I generate a few billion subtle variations on the \$100 contract (adding extra spaces, words, whatever).
  - I generate a few billion subtle variations on the \$10 contract as well.
- Eventually, I will get a hash collision where one of the \$100 variations matches one of the \$10 variations.

Hash functions and the birthday problem

- Suppose I want to defraud you:
  - I write up a contract agreeing to pay you $100 for something
  - I also write up a fraudulent version agreeing to only pay $10 for that same thing.
  - I generate a few billion subtle variations on the $100 contract (adding extra spaces, words, whatever).
  - I generate a few billion subtle variations on the $10 contract as well.
- Eventually, I will get a hash collision where one of the $100 variations matches one of the $10 variations.
- Since they both have the same hash, you might think you have the real one, but you actually have a fake one.

Hash functions and the birthday problem

- For this reason, 64-bit hash functions are no longer considered secure (even though they are still widely used).

Hash functions and the birthday problem

- For this reason, 64-bit hash functions are no longer considered secure (even though they are still widely used).
- Even 128-bit hash functions (with 10^{38} possible outputs) are not considered secure.

Hash functions and the birthday problem

- For this reason, 64-bit hash functions are no longer considered secure (even though they are still widely used).
- Even 128-bit hash functions (with 10^{38} possible outputs) are not considered secure.
- This is because $\sqrt{2^{128}} = 2^{64} = 10^{18}$, which is a lot of variations to make, but within the reach of well-funded nation states.

Hash functions and the birthday problem

- For this reason, 64-bit hash functions are no longer considered secure (even though they are still widely used).
- Even 128-bit hash functions (with 10^{38} possible outputs) are not considered secure.
- This is because $\sqrt{2^{128}} = 2^{64} = 10^{18}$, which is a lot of variations to make, but within the reach of well-funded nation states.
- So 256-bit hash functions are recommended.

Wifi and the birthday problem

- The original security scheme for Wifi is called WEP.

Wifi and the birthday problem

- The original security scheme for Wifi is called WEP.
- It used something called a stream cipher to encrypt communications.

Wifi and the birthday problem

- The original security scheme for Wifi is called WEP.
- It used something called a stream cipher to encrypt communications.
- The one thing to know about stream ciphers is: *Do not reuse the encryption keys*.

Wifi and the birthday problem

- The original security scheme for Wifi is called WEP.
- It used something called a stream cipher to encrypt communications.
- The one thing to know about stream ciphers is: *Do not reuse the encryption keys*.
- If you do, it's trivially easy to crack.

Wifi and the birthday problem, continued

- For WEP, the wireless access point and your computer share a master key.

Wifi and the birthday problem, continued

- For WEP, the wireless access point and your computer share a master key.
- To make a different key for each message, a random number called an initialization vector (IV) is combined with the master key.

Wifi and the birthday problem, continued

- For WEP, the wireless access point and your computer share a master key.
- To make a different key for each message, a random number called an initialization vector (IV) is combined with the master key.
- This would be fine, except they used 24-bit IVs.

Wifi and the birthday problem, continued

- For WEP, the wireless access point and your computer share a master key.
- To make a different key for each message, a random number called an initialization vector (IV) is combined with the master key.
- This would be fine, except they used 24-bit IVs.
- A 24-bit IV would mean 16 million possible encryption keys

Wifi and the birthday problem, continued

- For WEP, the wireless access point and your computer share a master key.
- To make a different key for each message, a random number called an initialization vector (IV) is combined with the master key.
- This would be fine, except they used 24-bit IVs.
- A 24-bit IV would mean 16 million possible encryption keys
- Now the birthday problem comes in: $\sqrt{2^{24}} = 2^{12} = 4096$

Wifi and the birthday problem, continued

- For WEP, the wireless access point and your computer share a master key.
- To make a different key for each message, a random number called an initialization vector (IV) is combined with the master key.
- This would be fine, except they used 24-bit IVs.
- A 24-bit IV would mean 16 million possible encryption keys
- Now the birthday problem comes in: $\sqrt{2^{24}} = 2^{12} = 4096$
- After 4096 messages, repeated keys are likely.

Wifi and the birthday problem, continued

- For WEP, the wireless access point and your computer share a master key.
- To make a different key for each message, a random number called an initialization vector (IV) is combined with the master key.
- This would be fine, except they used 24-bit IVs.
- A 24-bit IV would mean 16 million possible encryption keys
- Now the birthday problem comes in: $\sqrt{2^{24}} = 2^{12} = 4096$
- After 4096 messages, repeated keys are likely.
- This is the amount of traffic you get on a typical network in a few minutes.

DNS and the birthday problem

- DNS is the system that translates human-readable domain names like `www.msmary.edu` into numerical IP addresses like `216.230.103.23` that computers use.

DNS and the birthday problem

- DNS is the system that translates human-readable domain names like `www.msmary.edu` into numerical IP addresses like `216.230.103.23` that computers use.
- When you want to go to a website, you ask a machine run by your service provider to do a DNS lookup for you.

DNS and the birthday problem

- DNS is the system that translates human-readable domain names like `www.msmary.edu` into numerical IP addresses like `216.230.103.23` that computers use.
- When you want to go to a website, you ask a machine run by your service provider to do a DNS lookup for you.
- That machine asks other machines on the internet for the answer.

DNS and the birthday problem

- DNS is the system that translates human-readable domain names like `www.msmary.edu` into numerical IP addresses like `216.230.103.23` that computers use.
- When you want to go to a website, you ask a machine run by your service provider to do a DNS lookup for you.
- That machine asks other machines on the internet for the answer.
- If an attacker can return an answer before the real answer arrives, then the attacker's answer will be accepted.

DNS and the birthday problem

- DNS is the system that translates human-readable domain names like `www.msmary.edu` into numerical IP addresses like `216.230.103.23` that computers use.
- When you want to go to a website, you ask a machine run by your service provider to do a DNS lookup for you.
- That machine asks other machines on the internet for the answer.
- If an attacker can return an answer before the real answer arrives, then the attacker's answer will be accepted.
- The attacker can use this to make it so that when you go to `gmail.com`, you are actually directed to a fake gmail site, where they can phish your username and password.

DNS and the birthday problem, continued

- However, DNS queries have an ID that the attacker must guess right.

DNS and the birthday problem, continued

- However, DNS queries have an ID that the attacker must guess right.
- There are $2^{16} = 65536$ possible IDs, so guessing is hard.

DNS and the birthday problem, continued

- However, DNS queries have an ID that the attacker must guess right.
- There are $2^{16} = 65536$ possible IDs, so guessing is hard.
- One more thing: the machine your ISP uses stores the answers to the DNS queries it does, so that if someone else with the same provider as you recently went to `msmary.edu`, the DNS machine will store the resulting IP address to save time from having to ask remote machines for the answer.

DNS and the birthday problem, continued

- Now the birthday problem comes into play.

DNS and the birthday problem, continued

- Now the birthday problem comes into play.
- An attacker creates a bunch of fake requests that it asks the service provider's DNS machine to make.

DNS and the birthday problem, continued

- Now the birthday problem comes into play.
- An attacker creates a bunch of fake requests that it asks the service provider's DNS machine to make.
- At the same time, it sends back a bunch of bogus replies to those queries.

DNS and the birthday problem, continued

- Now the birthday problem comes into play.
- An attacker creates a bunch of fake requests that it asks the service provider's DNS machine to make.
- At the same time, it sends back a bunch of bogus replies to those queries.
- Most of those won't have matching IDs, but the attacker just needs one fake answer to match one fake request and then that result will be stored in memory for a while, affecting many of the service provider's customers.

DNS and the birthday problem, continued

- Now the birthday problem comes into play.
- An attacker creates a bunch of fake requests that it asks the service provider's DNS machine to make.
- At the same time, it sends back a bunch of bogus replies to those queries.
- Most of those won't have matching IDs, but the attacker just needs one fake answer to match one fake request and then that result will be stored in memory for a while, affecting many of the service provider's customers.
- By the birthday problem, they need only create around $\sqrt{2^{16}} = 2^8 = 256$ bogus requests, which is easy.

In conclusion…

A fun little math problem turns out to have big consequences in computer security. The examples shown here are just a few of many more.

Thanks for your attention!