Birthday attack in Cryptography

Birthday attack is a type of cryptographic attack that belongs to a class of brute force attacks. It exploits the mathematics behind the birthday problem in probability theory. The success of this attack largely depends upon the higher likelihood of collisions found between random attack attempts and a fixed degree of permutations, as described in the **birthday paradox problem**.

Birthday paradox problem –

Let us consider the example of a classroom of 30 students and a teacher. The teacher wishes to find pairs of students that have the same birthday. Hence the teacher asks for everyone's birthday to find such pairs. Intuitively this value may seem small. For example, if the teacher fixes a particular date say ***October 10***, then the probability that at least one student is born on that day is $1 - (364/365)^{30}$ which is about ***7.9%***. However, the probability that at least one student has the same birthday as any other student is around ***70%*** using the following formula:

$1 - 365!/((365 - n!) * (365^n))$ (substituting n = 30 here)

Derivation of the above term:

Assumptions –

1. Assuming a non leap year(hence 365 days).
2. Assuming that a person has an equally likely chance of being born on any day of the year.

Let us consider n = 2.

P(Two people have the same birthday) = 1 – P(Two people having different birthday)

= 1 – (365/365)*(364/365)
= 1 – 1*(364/365)
= 1 – 364/365
= 1/365.

So for n people, the probability that all of them have different birthdays is:

P(N people having different birthdays) = (365/365)*(365-1/365)*(365-2/365)*….(365-n+1)/365.

= 365!/((365-n)! * 365^n)

Hash function –

A hash function H is a transformation that takes a ***variable sized input m*** and returns a ***fixed size string*** called a ***hash value(h = H(m))***. Hash functions chosen in cryptography must satisfy the following requirements:

- The input is of variable length,
- The output has a fixed length,
- H(x) is relatively easy to compute for any given x,
- H(x) is one-way,
- H(x) is collision-free.

A hash function H is said to be one-way if it is hard to invert, where “hard to invert” means that given a hash value h, it is computationally infeasible to find some input x such that ***H(x) = h***.
If, given a message x, it is computationally infeasible to find a message y not equal to x such that ***H(x) = H(y)*** then H is said to be a weakly collision-free hash function.
A strongly collision-free hash function H is one for which it is computationally infeasible to find any two messages x and y such that ***H(x) = H(y)***.
Let **H: M => {0, 1}^{n}** be a hash function (**|M| >> 2^n**)
Following is a generic algorithm to find a collision in time ***O(2$^{n/2}$)*** hashes.
Algorithm:

1. Choose $2^{n/2}$ random messages in M: m_1, m_2, …., $m_{n/2}$
2. For i = 1, 2, …, $2^{n/2}$ compute $t_i = H(m_i) => \{0, 1\}^n$
3. Look for a collision ($t_i = t_j$). If not found, go back to step 1

We consider the following experiment. From a set of H values, we choose n values uniformly at random thereby allowing repetitions. Let ***p(n; H)*** be the probability that during this experiment at least one value is chosen more than once. This probability can be approximated as:

$$p(n; H) = 1 - ((365-1)/365) * (365-2)/365) * ...(365-n+1/365))$$

$$p(n; H) = e^{-n(n-1)/(2H)} = e^{-n2/(2H)}$$

Digital signature susceptibility –
Digital signatures can be susceptible to birthday attacks. A message ***m*** is typically signed by first computing ***H(m)***, where ***H*** is a cryptographic hash function, and then using some secret key to sign ***H(m)***. Suppose Alice wants to trick Bob into signing a fraudulent contract. Alice prepares a fair contract ***m*** and fraudulent one ***m’***. She then finds a number of positions where ***m*** can be changed without changing the meaning, such as inserting commas, empty lines, one versus two spaces after a sentence, replacing synonyms, etc. By combining these changes she can create a huge number of variations on m which are all fair contracts.
Similarly, Alice can also make some of these changes on ***m’*** to take it, even more, closer towards ***m***, that is ***H(m) = H(m’)***. Hence, Alice can now present the fair version ***m*** to Bob for signing. After Bob has signed, Alice takes the signature and attaches to it the fraudulent contract. This signature proves that Bob has signed the fraudulent contract.
To avoid such an attack the output of the hash function should be a very long sequence of bits such that the birthday attack now becomes computationally infeasible.