What is Birthday attack??

Technical Report · February 2015

DOI: 10.13140/2.1.4915.7443

1 author:

Ganesh Gupta
The Maharaja Sayajirao University of Baroda
2 PUBLICATIONS **1** CITATION

SEE PROFILE

What is Birthday attack??

- By Ganesh Gupta

Abstract

In this Paper we will discuss about birthday attack which is mainly based on birthday problem .Birthday problem is basically a probability problem in which situation is, if there are 23 people in a room, the probability of two people having birthday on same date is slightly more than 0.50. If there are 30 people, the probability increases to 0.70. This is called the birthday paradox. We use this problem in a generalized form in cryptography which we call as Birthday attack. Consider a variation of the same problem, suppose there are two rooms, each with 30 people, what is the probability that someone in the first room has the same birthday as someone in the second room?

Birthday attack can even be used to find collisions for hash functions if the output of the hash function is not sufficiently large.

In this paper we shall see what hash function is and why birthday attack is so important for hash function.

The paper has 3 main segments,

1) Introduction – Two sub parts
   a) Terminology – Definition of the terms used in birthday attack.
   b) Method – What is birthday attack Algorithm? What is its use in real life? A discussion on variation of birthday attack problem.
2) Background- In this section we will discuss extensively about past work.
3) Conclusion

Introduction

Terminology

Birthday problem – It is basically a probability problem which was first discussed By Harlod Davenport. Richard Von mises proposed an earlier version of what we consider today to be the birthday problem
The problem is following: Consider a room with few people. What is the probability that two of them have same birthday?
We will generalize this problem mathematically and make use some facts that are situation-specific (e.g.: number of people in the room, span of days, which is 365 in this case) for problem solving.
We start with one example and try to solve that problem.

Problem: - Consider a room with 60 people, what is the probability that at least one of them will have birthday same as another person in the same room.

To solve this problem we take alternative way; we try to find the number of ways for a person to not share a birthday with any of the other people. This could be thought of as putting 60 balls into 365 slots in order so that each slot can have at most one ball, divided by 365, the total number of possible locations to choose from.
So,
E is the event in which At least one person has same birthday as another person in same group
So $\bar{E}$ is the event No same birthday

$$P(E) = 1 - P(\overline{E})$$
$$= 1 - \left(\frac{365}{365} \times \frac{365-1}{365} \times \frac{365-2}{365} \times \ldots \times \frac{365-59}{365}\right)$$
$$\approx 0.2739$$

So if number of people in the room is 365, the probability becomes 1; but if it is 366 then does not actually reach 1, due to the pigeonhole principle, where if the number of pigeons exceeds the number of holes, then there must be at least one pair of pigeons in one of the holes.
For k persons in the room and n=365 the formula is

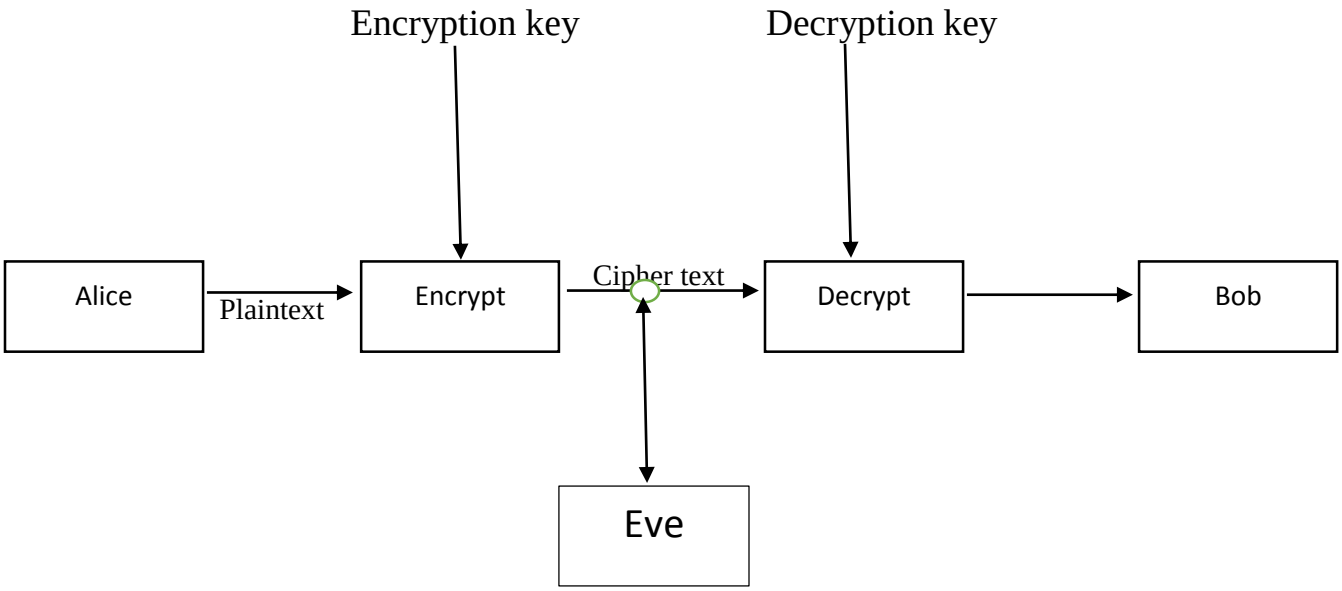
$$1 - \left(\frac{365}{365} \times \frac{365-1}{365} \times \ldots \times \frac{365-(k-1)}{365}\right)$$

This is basically a platform for birthday attack.

Cryptography- Cryptography has a long and fascinating history, with usage dating as back as 4000 years by Egyptians. It saw its extensive utilization during the

period of World wars. Completed in 1963, Kahn's book covers those aspects of the history which were most significant (up to that time) to the development of the subject. The predominant practitioners of cryptography were those associated with the military, the diplomatic service and government in general. Cryptography was used as a tool to protect national secrets and strategies.
Today cryptography is more than secret writing, more than encryption and decryption. Authentication has become fundamental part of our lives. We use authentication throughout everyday life, when we sign to some documents for instance, and as we move to a world where our decision and agreements are communicated electronically, we need to replicate these authentication procedure.

The most basic process of encrypted message passing is shown by following diagram



In the basic communication scenario, depicted in Figure , there are two parties, we'll call them Alice and Bob, who want to communicate with each other. A third party, Eve, is a potential eavesdropper. When Alice wants to send a message, called the plain text, to Bob, she encrypts it using a method prearranged with Bob. Usually, the encryption method is assumed to be known to Eve; what keeps the message secret is the **key**. When Bob receives the encrypted message, called the cipher text, he changes it back to the plaintext using a decryption key.

Eve makes use of many attacks to decipher the encrypted message, one of them being **brute force method**.
Brute force attack is mainly a strategy that can be used against any encrypted data by an attacker who is unable to take advantage of any weakness in an encryption system that would otherwise make his task easier. It involves systematically checking all possible keys until the correct key is found.
The key length used in the encryption determines the practical feasibility of performing a brute force attack, with longer keys exponentially more difficult to crack the shorter ones.

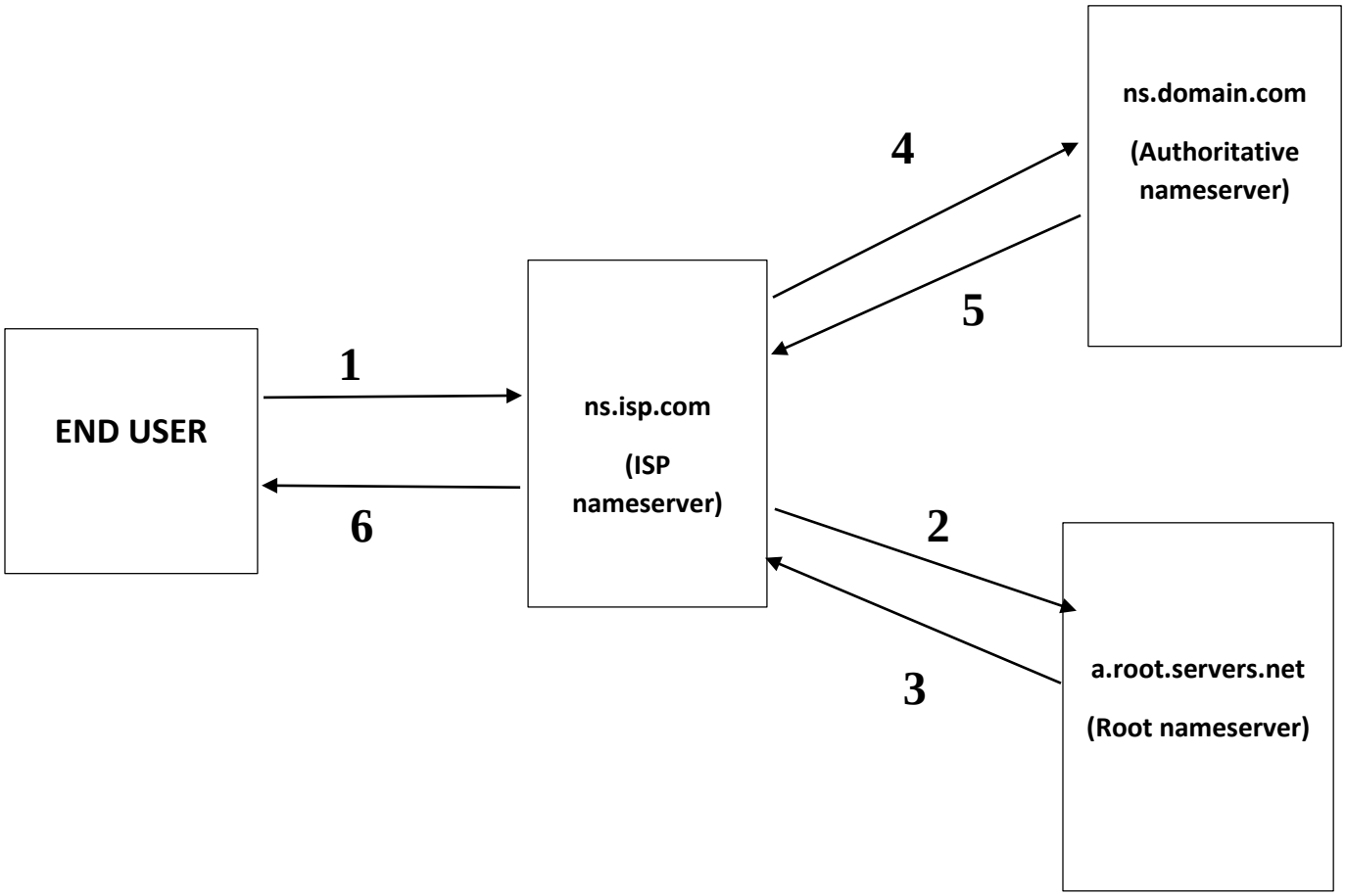
Hash function – it is a computationally efficient function of mapping binary strings of arbitrary length to binary strings of some fixed length, called *hash-values h* .
Basic properties of Hash functions is follow:-

1) The input can be of any of length.
2) The output has a fixed length.
3) H(x) is easy to compute for any given x
4) H(x) is one –way
5) H(x) is collision free

One-way – If it is hard to invert it means that given a hash value h, it is computationally infeasible to find some input x such that $H(x) = h$
Collision-free – If given a message x, it is computationally infeasible to find a message y not equal to x such that $H(x) = H(y)$ the H is said to be a weakly collision-free hash function. A **strong collision-free** hash function H is one for which it is computationally infeasible to find any two message x and y such that $H(x) = H(y)$.
The main role of a cryptographic hash function is in the provision of digital signature.
Digital signature –Digital signature are used to sign electronic documents .such a signature have properties of similar to handwritten signatures.
If Alice sign a document with her handwritten signature, then everybody who sees the documents and who knows the Alice's signature can verify that Alice has in fact signed the documents.

DNS (Domain Name system) server
Basically a user wants to find the IP address of a webserver would first query her local DNS server. This server is designed to make queries on her behalf to as many nameservers as needed in order to find the answer.

Illustration of DNS protocol is following:
Briefly it is 6 step work
Step 1) user ask ISP nameserver to look up the IP address of www.doamin.com
Step 2) ISP nameserver queries root nameserver to find out who is authoritative for domain.com
Step 3) Root nameserver answers: ns.domain.com is authoritative for domain.com
Step 4) ISP nameserver queries ns.domain.com for IP address of www.domain.com
Step 5) ns.domain.com answers "www.domain.com is at 1.2.3.4"
Step 6) ISP nameserver sends reply to user "www.domain.com is at 1.2.3.4"



In order to verify authenticity of the reply, the DNS system uses transaction IDs (Sometimes called query IDs). A 16-bit number is generated by the nameserver or Resolver client who is issuing a query. Any reply from the nameserver must contain this transaction ID. As long as the TCP or UDP port number, IP address and transaction ID from the remote host are correct, the reply is considered to be legitimate.

DNS spoofing (DNS Cache Poisoning) – It is basically a Computer hacking attack, whereby data is introduced into a domain name system resolver's cache, causing the name server to return an incorrect IP address, diverting traffic to the attacker's computer,
To perform a cache poisoning attack, the attackers exploits flaws in the DNS software. Server should correctly validate DNS responses to ensure that they form an authoritative sources, otherwise the server might end up caching the incorrect entries locally and server them to other users that make the same request .This attack can be used to direct users from a website to another site of the attacker's choosing.

BIND (Berkeley Internet Name Domain) It is a software which is used by nearly all of the nameservers on the Internet. BIND did not randomize its transaction IDs – they were purely sequential.

Method

Birthday paradox in cryptography is same as Birthday attack. We've already told you many things about birthday attack, so now, we will come straight forward to the method. There is not any General algorithm for this method. it is basically *Algorithm-independent attacks* that can be applied to any hash function. This attack is used in hash function to produce a collision if the output of the hash function is not sufficiently large.
Given a function H , Find two different inputs x_1, x_2 have same output value $H(x_1) = H(x_2)$ this pair (x_1, x_2) is called a Collision. Know by birthday problem we define birthday attack terms.
For k number of people and n number of days the formula is

$$p(n,k) = 1 - \left(\frac{n!}{(n-k)!\, n^k}\right)$$

$$= 1 - \left[\left(1 - \frac{1}{n}\right) \times \left(1 - \frac{2}{n}\right) \times \ldots \times \left(1 - \frac{k-1}{n}\right)\right]$$

By calculus fact we know that $(1 - x) \leq e^{-x}$ then we have

$$\begin{aligned} p(n,k) &> 1 - [e^{-\frac{1}{n}} \times e^{-\frac{2}{n}} \times e^{-\frac{3}{n}} \times \ldots \times e^{\frac{-(k-1)}{n}}] \\ &> 1 - e^{-\left[(\frac{1}{n} + \frac{2}{n} + \cdots + (\frac{k-1}{n})\right]} \\ &> 1 - e^{-\frac{k(k-1)}{2n})} \\ p(n,k) &\approx 1 - e^{-\frac{k(k-1)}{2n}} \end{aligned}$$

We would like to know when $p(n,k) > 0.5$(we want to get more than 0.5 probability to find collision in hash function)

We compute

$$0.5 = 1 - e^{-\frac{k(k-1)}{2n}}$$

$$\ln(2) = \frac{k(k-1)}{2n}$$

For larger values k we can replace $k \times (k-1) by\ k^2$ giving

$k = \sqrt{2(ln2)n} = 1.174\sqrt{n} \approx \sqrt{n}$

Application of Birthday attack

Now let's look at this in terms of hash codes. Remember a hash code is a function That takes a variable length message M and produces a fixed length message digest. Assuming the length of the digest is m then there are 2^mpossible message digests. The length of M will generally be greater than m; this Implies that more than one message will be mapped to the same digest. Idea is to make it computationally infeasible to find two messages that map to the same digest. However if we apply k random messages to our hash code what must the value of k be so that there is the probability 0.5 that at least one duplicate.

So by last equation in which $k = \sqrt{n}$ by using this equation $k = \sqrt{2^m} = 2^{\frac{m}{2}}$

It is clearer by following table

Bits	Possible outputs	Attempts for Desired probability of random collision	
		50%	75%
16	65,536	300	430
32	4.3×10^{9}	77,000	110,000
64	1.8×10^{19}	5.1×10^{9}	7.2×10^{9}
128	3.4×10^{38}	2.2×10^{19}	3.1×10^{19}
256	1.2×10^{77}	4.0×10^{38}	5.7×10^{38}
384	3.9×10^{115}	7.4×10^{57}	1.0×10^{58}
512	1.3×10^{154}	1.4×10^{77}	1.9×10^{77}

For example if you have $2^{16} bits$ then there are approximately 65,536 possible outputs and for producing collision we use above equation $k \approx 1.1774\sqrt{2^{16}} \approx 1.1774 \times 2^{\frac{16}{2}} \approx 1.1774 \times 256 \approx 301.4144$ so the desired probability of random collision 50% is around 301 attempts or by specified calculation we can say that 300

So we can say that for n bits of code we use $2^{\frac{n}{2}}$ for computation.

In a short summary we can say that if there are *n* possibilities and we have a list of length $\sqrt{n}$ then there is a good chance of match. If we want to increase the chance of match, we can make the list have length$2\sqrt{n}$ *or* $5\sqrt{n}$. The main point is that a length of a constant times $\sqrt{n}$.

This is one of the most basic application of birthday attack, similarly, suppose we have two sets of input S and T. if we compute $h(s)$ for approximately $\sqrt{n}$ randomly chosen $s \in S$ and $h(t)$ for approximately $\sqrt{n}$ randomly chosen $t \in T$.This same situation will arise in an attack on Digital signature where S will be a set of good documents and T will be a set of fraudulent documents. For understand Digital signature, we shall first discuss Yuval's Birthday attack; it is same as birthday attack but with some variation, in which, while drawing elements randomly, with replacement, from a set of n elements, with high probability a repeated element will be encountered after $O(\sqrt{n})$ (asymptotic upper bound:$f(n) = O(g(n))$ if there exist a positive constant c and a positive integer n_o

such that $0 \leq f(n) \leq cg(n) \; \forall \; n \geq n_o)$ Selections . This attack also called Square roots Attacks

Yuval's Birthday attack algorithm

Input
Two types of messages - legitimate message x_1 ; fraudulent message x_2 ; m bit length; one -way hash function H

Output
x_1', x_2' Is a minor modification of x_1, x_2 with $H(x_1') = H(x_2')$.

1) Generate $t = 2^{\frac{m}{2}}$ minor modifications of x_1' of x_1.
2) Hash each such modified message, and store the hash-values such that they can be subsequently searched on hash -values. This can be done in $O(t)$ total time using conventional hashing.
3) Generate minor modifications $x_2' \; of \; x_2$, computing $H(x_2')$ for each and checking for any matches with any x_1' above; continue until a match is found.

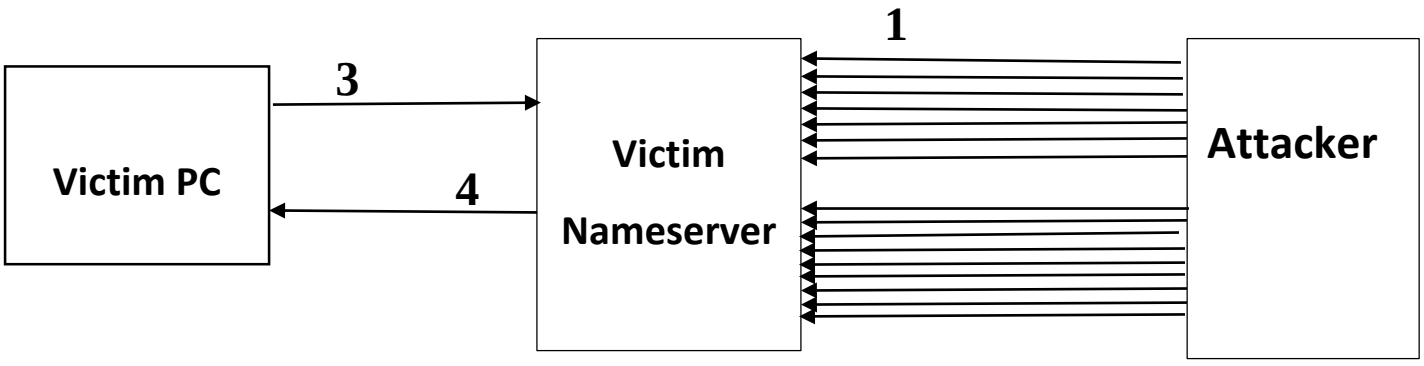
Yuval's birthday Attack is the Algorithm which we use in Digital signature forgery that can be explained by following example

Alice is going to sign a document electronically by using one of the signature schemes to sign the hash of the document. Suppose the hash function produces an output of 50 bits. She is worried that Fred will try to trick her into signing an additional contract, perhaps for swamp land in Florida, but she feels safe because the chance of a fraudulent contract having the same hash as the correct document is 1 out of 2^{50}, which is approximately 1 out of 10^{15}.
Fred can try several fraudulent contracts, but it is very unlikely that he can find one that has the right hash. So at this place Fred used Birthday problem. He finds 30 places where he can make a slight change in the document: adding a space at the end of a line, changing a wording slightly, etc. At each place, he has two choices: Make the small change or leave the original. Therefore, he can produce 2^{30} documents that are essentially identical with the original. Surely, Alice will not object to any of these versions. Now, Fred computes the hash of each of the 2^{30} versions and stores them. Similarly, lie makes 2^{30} versions of the fraudulent contract and stores their hashes. Fred finds the match and asks Alice to sign the good version. He plans to append her signature to the fraudulent contract, too.

Since they have the same hash, the signature would be valid for the fraudulent one, so Fred could claim that Alice agreed to buy the swamp land.
So Fred did a Birthday attack.
In Digital signature we can fail the Birthday attack by before signing any electronic document, make a slight change in generalization you can say that the output length of the hash function used for a signature scheme can be chosen large enough so that the birthday attack becomes computationally infeasible.

BIND birthday attack -
In 2002, Vagner Sacramento released an advisory showing another problem with BIND's implementation of the DNS protocol. He found that BIND would send multiple simultaneous recursive queries for the same IP address. Because of this a mathematical phenomenon comes into play known as the "Birthday Paradox". This causes the probability of a successful attack to rise to near 100% with only a few hundred packets

The attacker merely needs to send a few hundred queries to the ISP nameserver asking for the IP address of the domain name to be hijacked. At the same time, he will send the same number of replies formulated to look as if they were sent from the authoritative nameserver. In each packet he will assign a random transaction ID. In order to be successful, one of his spoofed packets transaction IDs, source and destination IP addresses and ports must match a legitimate recursive query packet from the victim nameserver.
This attack to DNS Protocol psuedo-random number sequences, such as the one that generates transaction IDs in BIND. With conventional spoofing, we would send *n* spoofed replies for one query. Our probability of success is $\frac{n}{65535}$ (65535 is the number of possible ID). With the BIND birthday attack, we send *n* number of spoofed replies for *n* queries. For this, we can predict the probability of success using the above formula where *k* is the total number of possible values in the master set, and *n* is the number of values in the spoofing subset
$[p(n,k) \approx 1 - e^{-\frac{k(k-1)}{2n}}]$



The Bind Birthday attack Illustration step wise:-
Step-1 Attackers send a large number of quires to the victim nameserver, all of the same domain name
Step-2 Attackers sends spoofed replies giving fake answer for the quires it made
Step-3 At a later time, victim PC sends a request for the quires it made
Step-4 Victim nameserver return fake information to victim PC

Conclusion:-

Birthday attack is Brute force attack and algorithm independent attack that is basically based on birthday problem, Birthday problem is probability problem that helps in finding probability of people having birthdays on the same day in a group. In cryptography it plays an important role which help to find the message between two persons as an eavesdropper but it is basically a hash based function in which we trying to find Collison in hash function by finding two different input for which hash functions output is the same. We can say that it is the shortest method of attack which we use for attack on cryptographic algorithm .It is computationally feasible so that's why we use in hash function to finding collision and one-way function.
It is also used in Digital signature for frauds by converting original documents into fraudulent document. We trying to create huge variations *m*' in original documents then we apply the hash function to all these variations until we finds a version of the fair contract and a version of the fraudulent contract which have the same hash value, $h(m) = h(m')$. After that we present the fair version to Signer for signing. After Signer has signed, we take the signature and attach it to the fraudulent contract. This signature then "proves" that the signer signed the fraudulent contract.
We use Yuval's birthday attack algorithm for Digital signature forgery in this we repeat the Yuval's birthday step until a match is found which is expected after $2^{\frac{n}{2}}$

steps. It is also used in DNS server for failing system .Birthday attack have many modifications and its vast variations with changing of birthday problem. This attack is used on many objects like multi collision, MAC, Block cipher, DES. So Birthday attack has basically an easy and useful term that we use to finding secrets and stealing something or many things.

References:-

1) http://en.wikipedia.org/wiki/Birthday_attack
2) http://x5.net/faqs/crypto/q95.html
3) http://www.facweb.iitkgp.ernet.in/~sourav/Attacks_on_cryptosystems.pdf
4) Hand Book of applied cryptography - Alfred J. Menezes , Paul C. van Oorschot , Scott A. Vanstone
5) Introduction to cryptography with coding theory – Wade Trappe
6) Introduction to cryptography – Johannes buchmann

7) http://www.secureworks.com/resources/articles/other_articles/dns-cache-poisoning/