The Mathematics Behind the Birthday Attack

Luke M. Kim

Abstract

The following paper briefly examines the origin and the mathematics behind the "Birthday Paradox" in a non-rigorous manner and aims to provide a simple explanation of how this idea is applied and used for the Birthday Attack, which is a form of cryptographic attack that threatens message integrity. The mathematics involves simple probability calculations, and any background knowledge that is needed to understand the Birthday Attack is explained.

1 Introduction

In the age where computers and electronic devices have taken up a prominent role, it has become very important to maintain the security and integrity of information transmitted via these devices. It is an inconvenient truth that technological progress and progress in cyberattacks go hand in hand. As we invent more efficient and secure methods of transmitting messages and maintaining their integrity, it seems that the attackers too are able to come up with ways of circumventing our system, or are able to develop an algorithm to extract or modify information in the most efficient manner. An example of this is the "Birthday Attack" - an attack designed to threaten message integrity and possibly abuse communication between two or more parties. In another sense, it is a type of brute force cryptographic attack based on very simple probability calculations that are famously known as the "Birthday Paradox". Although methods of preventing Birthday Attacks have already been invented[1], it is interesting to discuss the Birthday Attack since we can observe how often simple mathematical ideas may be applied to be used as a form of a cyberattack. Furthermore, as a student interested in Mathematics as well as Computer Security and Cryptography, I would like this brief and informative article to be of a non-technical introduction to a topic in Cryptography for readers who are mathematically inclined.

2 What is the Birthday Attack?

2.1 Message Integrity

Before we begin discussing the Birthday Attack, we have to be aware that we have a goal of maintaining message integrity in computer security, i.e. the message that a person 'A' sends to another person 'B' should not be tampered with or changed to contain false or modified information. Message integrity is normally maintained via the protocol called "MAC" or Message Authentication Code. To briefly explain its mechanism, let us use an example of a communication between two people, in which there is a sender and a receiver. The sender and the receiver both have a shared key, k, that they can use for the MAC signing algorithm. Now, the sender generates a 'tag'[2] that goes along with the message that he/she has sent. When the receiver receives the message and the tag, the receiver runs the MAC verification tag and

[1] For interested readers, the article **"Foiling Birthday Attacks in Length-Doubling Transformations"** by Aiello and Venkatesan outlines in detail how we may counter Birthday Attacks

[2] The tag constitutes of a key and message pair.

this verification algorithm outputs 'yes' or 'no' depending on the validity of the message. To describe this whole procedure step by step using some cryptographic notations:

1. Sender generates a signing tag $S(k, m)$, where k is the shared key, and m is the message.
2. The receiver runs a verification algorithm, defined by $V(k, m, tag) = V(k, m, S(k, m))$.
3. If the sender signs with a particular key, and the receiver verifies with the same key, then the result of $V(k, m, S(k, m)) =$ 'yes' which shows that message integrity has been maintained and the receiver is able to view the correct information that the sender transmitted.

2.2 Hash Collisions

I have presented to you how message integrity is maintained in a nutshell (actually the above is a very basic model and there is more but delving too deep into different types of Message Authentication Codes would be a digression). An important concept here is that when a tag is generated, it involves 'Hash Functions', which assign Hash Values to piece of messages. Here, a 'Hash' (or some say 'cryptographic hash') is a type of a signature for a stream of data that represents the contents. In other words, it acts as a form of temper-evident seal on a software. Even a small change in the file may create a huge difference in the Hash Value (If you are running Unix or Linux you can test it using the md5sum program. In fact, this effect is most evident by hashing two files with nearly identical contents). Now, imagine that we have a large message space, M that contains the message that the sender wants to send to the receiver. If the sender generates a tag, it means that there is a hash function H such that it maps the messages from the message space to the tag space, T i.e. $H : M \rightarrow T$.Note here that both the message space $|M|$ and the tag space $|T|$ are finite and the message space is much larger than the tag space ($|M| >> |T|$). This is generally true for all message and tag spaces, and this means that every cryptographic hash functions are inherently vulnerable to collisions. A collision (typically called 'Hash Collisions') is when some message m_i and m_j have the same Hash Value in the tag space. The collision is inevitable from the pigeonhole principle, since the message space is much larger than the tag space, so some messages must share the same Hash Value.

2.3 Some Implications of Hash Collisions and The Birthday Attack Algorithm

So why is it important that a collision occurs in the tag space? Consider the following scenario, in which a person named Eve wants to abuse the communication between person A and B. And let's say that person A and B decided to sign a contract, and that Eve has obtained the digital copy of the contract. From this document that Eve obtained, Eve created a document of the same hash value (but with different content by finding messages with hash collisions). Eve shows the unmodified document to A, and A signs it with his digital signature. Eve copies the digital signature on the unmodified contract that A signed, and pastes it onto the modified document with the same Hash Value. Eve then shows this to B, claiming that A signed this document, and B believes this because the digital signature matches the document hash.To provide the readers with an analogy, let's say that people spray a particular perfume on some objects and the scent from the object (that has perfume sprayed on it) defines what object this is. For instance object x smells like x'. Now, if someone can create a duplicate of this perfume and sprays it on object y, then y will smell like x' and people will think that y is in fact x (assume that people can only distinguish by scent). Here, the scent is analogous to the Hash Value. As such collisions may be used maliciously to destroy message integrity. The problem for the attackers at this point is "how to find messages with the same Hash Values?". The safest and surest way seems to be brute forcing. However, the attackers don't have all the time in the world and they need to find the most efficient brute forcing algorithm that can give them results as soon as possible. This is where the 'Birthday Attack' comes into play. The following is the general algorithm for the Birthday Attack and in the next section I will discuss the Birthday Paradox, which is a problem that gave birth to the Birthday Attack algorithm.

1. Let $H : M \to \{0, 1\}^n$ be a hash function. From this we know that the size of the tag space is $\approx 2^n$ bits and that $|M| >> 2^n$.
2. We choose $2^{\frac{n}{2}}$ random messages in M, i.e. $m_1, m_2, ..., m_{2^{\frac{n}{2}}} \in M$.
3. For $i = 1, 2, ..., 2^{\frac{n}{2}}$ compute $t_i = H(m_i)$, where t_i is the Hash Value in the tag space.
4. We then search for any collisions, i.e. $t_i = t_j$ for $i, j \in \{1, 2, ..., 2^{\frac{n}{2}}\}$. If this is not found, we go back to step 1 and repeat with different message samples.

This algorithm in fact is the most efficient algorithm to find the collision, and this runs in time O($2^{\frac{n}{2}}$), Where O stands for the big-O notation. The most important thing to note is why we take $2^{\frac{n}{2}}$ sample messages from the message space, and this is explained by examining the Birthday Paradox.

3 The Birthday Paradox

3.1 The Problem

The Birthday Problem asks "how many people do we need in a room such that the probability that at least two people in the room share the same birthday is at least $\frac{1}{2}$?". The answer to this question is in fact 23. If we have 23 people, there is chance greater than $\frac{1}{2}$ that at least two people will share the same birthday. This is often called a paradox, since compared to the number of possible days (365) in a year, only a small number of people are needed to fulfill the condition of the problem. In fact, by statistical analysis people were able to devise a generic formula.

Statement : Let $k_1, k_2, ..., k_n \in 1, ..., x$ where k_i is a random variable in the interval from 1 to x. If we are given that these random variables are independent of one another and distributed identically, then when the sample size $n = 1.2 * \sqrt{x} \approx \sqrt{x}$, the probability that $\exists i \neq j, k_i = k_j \geq \frac{1}{2}$.

In other words, when we have $\sqrt{x}$ samples, then there must be an identical variable with the variables being distinct. The number 1.2 is from statistical observation of data. We will now try to prove this fact, and note that we get the answer 23 for the problem above too if we follow the same line of logic that is about to be presented. Before we proceed to proving, there are two important pre-assumptions:

1. the random variables are independent
2. we will only consider the cases of uniform distribution - in fact it turns out that if the distribution type is not uniform, we even need less number of samples than $1.2 * \sqrt{x}$ (why would this be the case?)

Proof : The probability $P[\exists i \neq j : k_i = k_j] = 1$ - $P[\forall i \neq j : k_i \neq k_j]$. In other words, we find the probability such that no collision occurs among the n samples we choose and subtract this from 1. Hence,

$P[\exists i \neq j : k_i = k_j] = 1 - (\frac{x}{x})(\frac{x-1}{x})(\frac{x-2}{x})...(\frac{x-(n-1)}{x})$
$= 1 - \prod_{i=1}^{n-1}(1 - \frac{i}{x})...(*)$

Now we make use of the inequality $1 - x \leq e^{-x}$. This is true because the Taylor Expansion of $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \ldots$ and $\frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \ldots$ is a positive value. Applying this inequality directly to $(*)$, we obtain

$P[\exists i \neq j : k_i = k_j] = 1 - \prod_{i=1}^{n-1}(1 - \frac{i}{x}) \geq 1 - \prod_{i=1}^{n-1} e^{\frac{i}{x}} = 1 - e^{\frac{-1}{x}\sum_{i=1}^{n-1} i}$ Since the product of exponents is the sum of the exponent. Note that we may approximate $1 - e^{\frac{-1}{x}\sum_{i=1}^{n-1} i} \approx 1 - e^{\frac{-n^2}{2x}}$. Since the statement says that $n = 1.2 * \sqrt{x}$, $\frac{n^2}{2x} = 0.72$. With a hand held calculator, we may easily evaluate that $1 - e^{-0.72} \approx 0.53 > \frac{1}{2}$ and our proof is complete.

3.2 Comments

The proof only involves multiplying probabilities and manipulating simple inequalities, yet it gives us an important statistical information. Indeed, if we let $x = 365$, we have $n = 1.2\sqrt{356} \approx 23$. It is interesting to note that the probability drops very quickly if we have less than 23 people, but on the other hand, the probability also reaches 1 very quickly if we have more than 23. In fact, the probability is nearly 1 with around 70 people.An important conclusion that we can draw from this is that if our hash function outputs n bit outputs, there will always be an attack algorithm that runs in time $2^{\frac{n}{2}}$. For example, if we have 128 bits of output, then the collision can be found in time $\approx 2^{64}$, which is considered not sufficiently secure from a computer security aspect. Another remark that I would like to make is that statistics shows that Birthdays are actually not uniformly distributed - it is often biased towards certain months and certain days. Yet, the Birthday problem has been constructed with the assumption that the birthdays are uniformly distributed.

4 Concluding Remarks

The Birthday Attack, used to exploit the fact that there are collision points in the tag space, may be hindered or significantly slowed by various methods as well (which the readers may find out through independent study). Computer security is a field where there is a constant cat and mouse game between the people who want to prevent information from being tampered with and those who want to gain (illegal) access to them to fulfill whatever purposes they have. However,there are many interesting mathematical ideas underlying this conflict - ideas particularly related to number theory and statistics. The Birthday Attack is just one simple example, and other mathematical ideas underlying computer security and cryptography are definitely worth exploring.

5 Bibliography

1. Coursera. (n.d.). Coursera. Retrieved from Stanford Cryptography Coursera Videos: https://class.coursera.org/crypto-preview/lecture

2. Friedl, S. (n.d.). Unixwiz.net Tech Tips. Retrieved from http://www.unixwiz.net/techtips/iguide-crypto-hashes.html