International Journal of Advance Engineering and Research Development
Scientific Journal of Impact Factor (SJIF): 4.72
Special Issue SIEICON-2017,April -2017
e-ISSN : 2348-4470 p-ISSN : 2348-6406

Birthday Attack on Cryptography Applications

Concept and Real Time Uses

Keyur Patel[1], Digvijaysinh Mahida[2]

[1]Asst. Prof. Department of Information Technology, Sigma Institute of Engineering
[2] Asst. Prof. Department of Information Technology, Sigma Institute of Engineering

Abstract — *In many network communications it is crucial to be able to authenticate both the contents and the origin of a message. This paper will study in detail the problem known in probability theory as the Birthday Paradox (or the Birthday problem), as well as its implications and different related aspects we consider relevant. The Birthday attack makes use of what's known as the Birthday paradox to try to attack cryptographic hash functions. Among other desirable properties of hash functions, an interesting one is that it should be collision-resistant, that is it should be difficult to find two messages with the same hash value. To find a collision the birthday attack is used, which shows that attacker may not need to examine too many messages before he finds a collision. The purpose of this paper is thus to give the reader a general overview of the general aspects and applications of the Birthday Paradox.*

Keywords-*Birthday Paradox, Hash Function, Cryptography Algorithm, Application.*

I. INTRODUCTION

Computer networks are used today for many applications (like banking, e-government etc.) where security is an absolute necessary. Changing or stealing data stored in electronic form is so widely spread that not having cryptographic tools to preserve the safety of information would make pointless any serious communication or data exchange. In general, the functions of security system are confidentiality, authentication, integrity and non-repudiation. The birthday problem concerns the probability that in a set of *n* random people, two of them will share the same birthday. It belongs in the field of probability theory and is also known as the birthday paradox, not because the answer is illogical, but because most people underestimate this probability thus perceiving it as a paradox. To overcome that problem hash functions that map a message to a small digest *h(M)* are used. Such hash function has to be collision resistant - it should be difficult to find two messages with the same hash value[2][8].

The basic idea behind this paper, as a deviation from widely spread methods, is to examine the birthday attack and hash functions that are not dependent on any particular algorithm. This method is universal, because the principles it is based on are applicable to various algorithms, independently of the mechanism they use.

1.1. Hash Function [2][7][9]

A hash function *h* should map strings of bits of variable length to fix-length strings of bits, called the hash value of the message $\{0,1\}^m \rightarrow \{0,1\}^t$, where $m > t$. Ideally it has the following properties:

- The length of *h(M)* should be small so that messages can be signed efficiently.
- The function *h* should be a publicly known one-way function – it should be hard to find a message that hashes to a pre-specified value.
- It should destroy algebraic relationships between messages and signatures.
- It should be collision-resistant, that is it should be difficult to find two messages with the same hash value, or more precisely: an attacker should not be able to find a pair of messages $M \neq M'$ such that $h(M) = h(M')$ with less than about $2^{t/2}$ work.
- Preimage-resistance: An attacker given a possible output value for the hash *Y* should not be able to find an input *X* so that $Y = h(X)$ with less than about 2^t work.
- Second preimage-resistance: An attacker given one message M should not be able to find a second message, *M'* to satisfy $h(M) = h(M')$ with less than about 2^t work.

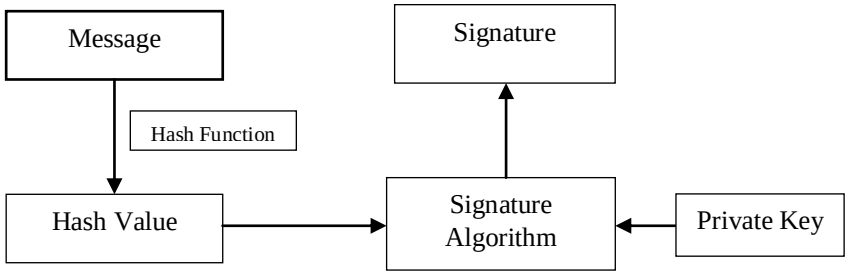

Figure 1. Hash Function

1.2. Birthday Paradox

In the probability theory, the birthday problem pertains to the probability that in a set of *n* randomly chosen people some pair of them will have the same birthday. Contrary to the naive intuition, the required number *n* of people

that will make the probability of some pair having the common birthday greater than 0.5 is not around 180, it is only 23. For 57 people, the probability of some pair having common birthday is more than 99%[2][8].

The birthday problem concerns the probability that in a set of *n* random people, two of them will share the same birthday. It belongs in the field of probability theory and is also known as the birthday paradox, not because the answer is illogical, but because most people underestimate this probability thus perceiving it as a paradox[1].

In fact, most people wrongly expect probabilities to be linear, because they only take into account the scenarios that they're involved in. That means they instinctively consider the probability that someone shares a birthday with them, while in this case the question is clearly about any two people in the room. The answer to our question from the first paragraph is actually 50% (which we will later demonstrate with the mathematical calculations). In a room of 23 people there is a 50% chance that two people will share the same birthday, but as we said most people would underestimate this number thus creating a seemingly paradoxical situation.

Another element that comes into play is the pigeonhole principle. It states that "if *n + 1* objects are put into *n* boxes, then at least one box contains two or more objects". For instance, if there are 13 people in a room, at least two of them will have their birthday on the same month. According to this logic, the probability of two people having the same birthday would reach 100% when there are 367 people in a room (if we consider February 29), since there are 366 possible birth dates. However, this logic is faulty, since we actually have near 100% probability with far less people, as we will further demonstrate[1].

II. THE BIRTHDAY PARADOX IN CRYPTOGRAPHY

Cryptography is known as the art of protecting information by transforming it into an unreadable format. Computing-wise, hashing in cryptography is a one way operation that transforms a stream of data into a more compressed form called a message digest. The entire message digests or hash values generated by a given hash function have the same size regardless of the size of the input value. Cryptographic hash functions are therefore a valuable tool in cryptography. They are applied in numerous areas of information security to guarantee the authenticity of messages. A hash function *H* is a transformation that takes a variable-size input m and returns a fixed-size output, which is called the hash value *h* (that is, *h = H(m)*).

The basic requirements for a cryptographic hash function are:

- The input can be of any length,
- The output has a fixed length,
- *H(x)* is relatively easy to compute for any given *x*

Hypothetically, there are no collisions in Hash functions, while the birthday paradox contradicts that. In fact, it is computationally possible to find any two messages x and y such that *H(x) = H(y)*, even when using a "strongly collision free hash function"[1][6].

2.1 Mathematical Term of Birthday Paradox

The Birthday attack makes use of what's known as the Birthday paradox to try to attack cryptographic hash functions. The birthday paradox can be stated as follows: What is the minimum value of *k* such that the probability is greater than 0.5 that at least two people in a group of *k* people have the same birthday? It turns out that the answer is 23 which is quite a surprising result. In other words if there are 23 people in a room, the probability that two of them have the same birthday is approximately 0.5. If there is 100 people (i.e. *k*=100) then the probability is .9999997, i.e. you are almost guaranteed that there will be a duplicate[3]. A graph of the probabilities against the value of *k* is shown in figure.

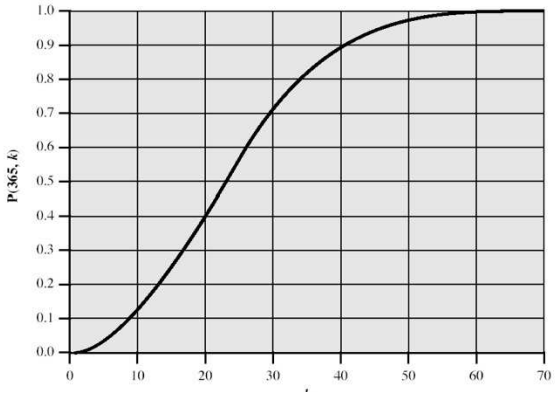

Figure 2. The Birthday Paradox

Although this is the case for birthdays we can generalize it for *n* equally likely values (in the case of birthdays *n* = 365). So the problem can be stated like this: Given a random variable that is an integer with uniform distribution between 1 and *n* and a selection of *k* instances *(k ≤ n)* of the random variable, what is the probability *P(n, k)*, that there is at least one duplicate[3].

It turns out that this value is

$$P(n,k) = 1 - \frac{n!}{(n-k)!\,n^k} \quad \ldots (1)$$

$$= 1 - [(1-\tfrac{1}{n}) \times (1-\tfrac{2}{n}) \times \ldots \times (1-\tfrac{k-1}{n})] \quad \ldots (2)$$

Take it that the following inequality holds

$$(1 - x) \leq e^{-x}$$

Then,

$$P(n,k) > 1 - [(e^{-1/n}) \times (e^{-2/n}) \times \ldots \times (e^{-(k-1)/n})]$$
$$> 1 - e^{-[(1/n)+(2/n)+(\overline{(k-1)}/n)]}$$
$$> 1 - e^{-(k \times (k-1))/2n} \quad \ldots (3)$$

Now let 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^m possible message digests. Normally however, because the length of *M* will generally be greater than *m* this implies that more than one message will be mapped to the same digest. Of course the 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 of 0.5 that at least one duplicate (i.e. *H(x) = H(y)* will occur for some inputs *x, y*). This is the same as the question we asked about the birthday duplicates[3].

Using this idea we can discuss the birthday attack as follows[3][4]:

- The source, A is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key.
- The opponent generates 2m/2 variations on the message, all of which convey essentially the same meaning. The opponent prepares an equal number of messages, all of which are variations of the fraudulent message to be substituted for the real one.
- The two sets of messages are compared to find a pair of messages that produce the same hash code. The probability of success is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.
- The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known. If we use a 64-bit hash code then the level of effort required is only on the order of 2m/2 = 264/2 = 232 which is clearly not sufficient to withstand today's computational systems.

The generation of many variations that convey the same meaning is not that difficult as figure 2 shows.

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}
Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}
Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}
responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}
in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}
to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the
market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the
{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}
of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}
as proof of identity. An order with his signature, which is {appended / attached}
{authorizes / allows} you to charge the cost to this company at the {above / head office}
address. We {fully / --} expect that our {level / volume} of orders will increase in
the {following / next} year and {trust / hope} that the new appointment will {be / prove}
{advantageous / an advantage} to both our companies.

Figure 3. A Letter in 2^{37} Variations

2.2 Implementation Attacks

Implementation attacks take on a different approach to the above for discovering the secret key. Instead of attacking the mathematical properties of the algorithm these form of attacks (also known as side channel attacks) take advantage of the physical phenomena that occurs when a cryptographic algorithm is implemented in hardware. Four side channel attacks are listed in the FIPS standard 140-2 "Security Requirements for Cryptographic Modules", Power Analysis, Timing Analysis, Fault Induction and TEMPEST. Here we will be interested mainly in Differential Power Analysis (DPA) as it applies to DES at Timing attacks[3].

2.2.1 Differential Power Analysis [3]

Power Analysis is a relatively new concept but has proven to be quite effective in attacking smartcards and similar devices. The smartcard is very susceptible to this form of attack mainly because it applies little or no power

filtering due to its small size. It was first demonstrated by Ernst Bovelander in 1997 but a specific attack strategy was not given. A year later it was brought to the general public's attention by Paul Kocher and the Cryptographic Research team in San Francisco. Kocher et al. provided an attack strategy that would recover the secret key from cryptographic systems running the DES algorithm. This caused great concern amongst the smartcard community and a search for an effective countermeasure began. To date a limited number of countermeasures have been proposed and none are fully effective. The attacks work equally well on other cryptographic algorithms as shown by Thomas Messages et al. who presented a great deal of supplementary research on the subject. Power analysis involves an analysis of the pattern of power consumed by a cryptographic module as it performs its operations. The purpose of this pattern analysis is to acquire knowledge about causal operations that is not readily available through other sources.

III. APPLICATION OF BIRTHDAY PARADOX

3.1 Practical Uses

3.1.1 Hash

If hash values have n bits, it should take 2n tries to find a key mapping to a given hash value, and $(2n)^2$ tries to find two keys that map to some common hash value. Therefore the one-way function is considered broken[1].

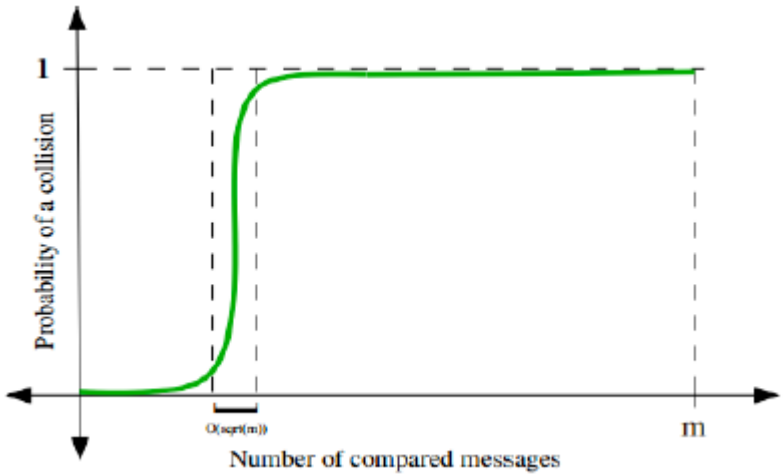

Figure 4. Hash Value of Collision Birthday Paradox

3.1.2 MAC (Message Authentication Code)

MAC is a short piece of information or a security code that is typed in by a computer user in order to access accounts or portals. Just like hash, MAC functions possess different security requirements[1]. This is how we compute an authentication tag:

Message authentication:
tag = MAC(key, M)
Sender sends (M, tag)
Receiver verifies that tag matches M
Authentication with HMAC:
tag=H(key | M) subject to extension attacks
tag=H(M | key) relies on collision resistance
HMAC: Compute tag = H(key | H(key | M))
M ≈ (key|M) looks random when key is secret

3.2 Other Applications

In substance, the birthday paradox shows that it is much easier to find a collision when both elements of the pair are not fixed. This is true when assuming a homogeneous probability distribution of the outcomes, but it also stands true for non-homogeneous distributions: actually, when the target distribution is not random, the collision hazard is even greater than in the case of pure randomness.

3.2.1 The pseudo random number generator (PRNG) application [1][5]

PRNG are used in a variety of computer-based applications to circumvent the limits of true random number generators. Although true random number generator can be (and are) conceived, they have to respect a number of requirements in order to achieve their goal. This makes them suboptimal for applications that require a high amount of random data.

PRNG are thus bridging the gap to fulfill this need by providing a reasonably random suite of numbers, while maintaining a good rate of (pseudo-) randomness availability. Also, they are used when large and reproducible random data is required – such as in symmetric key cryptography applications. Usually, PRNG take the form of a function, which is initialized with a true random value and then produces data by iterating indefinitely.

3.2.2 Applications of the birthday paradox in communication networks [1][6]

This use of the birthday paradox is interesting, because it takes advantage of the predictable risk of collision to achieve certain goals. One of them is to optimize the power consumption of communicating nodes of a wireless sensor network: the idea here is to optimize the duration and frequency of wake-up time of the transmission stages to minimize power consumption (or alternatively to maximize network lifetime), while maintaining a good probability of meet-up events in order for the nodes to be kept up-to-date (whether it is for transmitting or receiving data), all this without a synchronization clock readily available (because that would require power, reliable transmission media, etc.).

The birthday paradox, in this case, is used as a statistically reliable mean to guarantee that two unsynchronized devices can find a suitable communication slot, while preserving power.

IV. CONCLUSION

As we have seen throughout this work, the probability that in a set of *n* random people, two of them will share the same birthday, is a problem pertaining to probability theory with many unexpected applications. In the field of cryptography, namely Hash functions, we have seen how it is computationally possible to find two messages with the same hash value $H(x) = H(y)$, even when a "strongly collision-free hash function" is being used. A contemporary solution to this problem has been to increase the size of the hash value of hash functions, the latter needs to be large enough to make the possibility of a birthday attack very unlikely. With regards to other applications of the Birthday Paradox, we have also studied Pseudo-Random Number Generators (PRNG) as well as applications in communications networks. In communication networks, the birthday paradox is applied advantage of the predictable risk of collision to achieve certain goals.

REFERENCES

[1] Prof. Emil Simion, "The Birthday paradox: concept and real world applications", Operational Research and Optimization (Master EEJSI), December 2012.
[2] Milan Tuba, Nadezda Stanarevic, "Relation between Successfulness of Birthday Attack on Digital Signature and Hash Function Irregularity", WSEAS Transactions on Information Science and Applications, ISSN: 1790-0832, Issue 2, Volume 7, February 2010.
[3] Sourav Mukhopadhyay. "Attacks On Cryptosystems", Cryptography and Network Security, 2010.
[4] William Stallings, "Cryptography And Network Security Principles And Practice Fifth Edition", ISBN 13: 978-0-13-609704-4, 2006.
[5] NUNNIKHOVEN, Thomas. "A birthday problem solution for non uniform birth frequency". The American Statistician, Vol. 46, No. 4, November 1992.
[6] RAMAKRISHNA, Siva and BHAVANI, Ganga. "An Efficient Continuous Neighbour Discovery in Asynchronous Sensor Networks". International Journal of Computer Application, Issue 2, Volume 1, February 2012.
[7] STAMM, Stephanie. "Hash Functions and the Birthday Attack". United States Naval Academy. Midshipman 1/C. April 23, 2010.
[8] Tsang-Yean, L. and Huey-Ming, L. Encryption and decryption algorithm of data transmission in network security, WSEAS Transactions on Information Science and Applications. Vol. 3, no. 12, 2006, pp. 2557-2562.
[9] Talnor, J., Welsh, D., Complexity and Cryptography an Introduction, Cambridge University Press, 292 pages, 2006.

