ATTACKS ON CRYPTOGRAPHIC HASH FUNCTIONS

Abhishek Prajapati

Cryptography Research Paper

Prof. Brian Garnett

May 2, 2017

Introduction

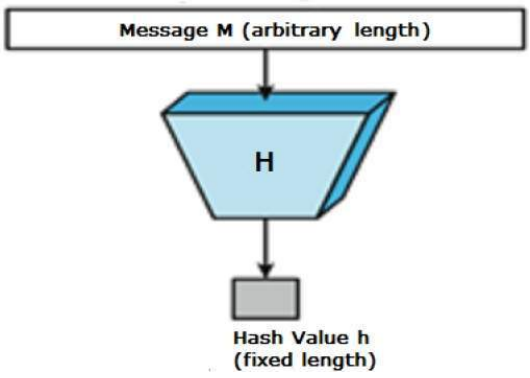
Confidentiality and authenticity have been big concerns in communication and information security for centuries. The idea to encrypting messages in such a way so that no other parties can intercept the message has been going on since the ancient Roman time. Julies Caesar was the first person who encrypted one of the most important messages with a secret number N by cyclically substituting each letter with the Nth next letter in the alphabet, that is if N=2 then A goes to C, B to D, C to E and so on (Stevens 3). This cipher is known as the "Caesar Cipher", which became one of the most important ciphers in ancient history, although it is not used anywhere today. Since then authenticating messages and information has become a significant part of today's modern computerized world.

The advancement of technology opened up a range of possibilities in processing information, which lead to the whole new world where everything is digitalized (Stevens 4). With the rise of the digital world, sending and storing of information has become quick, cheaper, and easier. Therefore, more and more people are starting to depend on the technology. Today majority of the people make use of services like Google, Facebook, and Dropbox to store and share their information because it provides lots of flexibility and it is much easier to use then mail service to deliver information. However, the evolution of computer presents new security requirements such as being able to verify the integrity of an information or a file. As more and more people make use of the advanced technology to store their information online, the process of protecting online information has also gotten much better than just using simple ciphers. One of the building blocks of modern cryptography that gave rise to new ideas and approaches is cryptographic Hash Functions, also known as the Hash Functions. Hash Functions were created to protect the authenticity of information and it is used with many cryptographic algorithms and

protocols in the context of the information security as a whole. In today's digitalized world, hash functions have become very important part of protecting privacy and authenticating information, however, as technology advances and the computer gets faster it is becoming easier and easier to break the hash functions.

Hash Functions and its applications

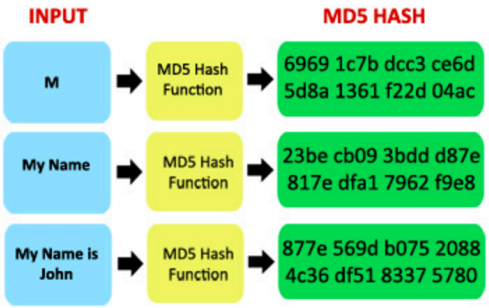
"A hash function H: {0,1}* -> {0,1}n is a mathematical function that converts a string or a numerical input value of length m into another compressed string or a numeric value of n bits" (Preneel). For example, input "abcd" turns into something like this "e2fc714c4727ee9395f324cd2e7f331f". The hash function has an input of arbitrary length but the resulting string or number is always of the same length. In most cases the hash output is much smaller than the input, therefore it is consistent, faster and cheaper then encrypting (Preneel). Hashing is not the same thing as encrypting; in encryption, you are required to have some kind of secret key to decipher the text, while in hash functions there is no keys or secret, and it's only a one-way operation so there is no way to decipher the hash function. In order for the hash function to be efficient and faster, it has to follow three properties: Pre-Image Resistance, Second Pre-Image Resistance, and Collision Resistance. Pre-image resistance states that given a message M, the computation of hash function h = hash(M) is fast and easy to do, however, there is no way to get the input from the hashed output (Preneel). For instance, say a file includes some numbers 2,5,7,9; and the hash function is the sum of those number (not a very good hash function), then the output of the hash function will be 2+5+7+9 = 23. Now, given 23 as the output, there is no way of telling that the



file contains those such values; and if someone were to change one of the digits from the file then the hash output would also change, for instance, changing 5 to 7 in the file would result in hash output of 25. Therefore, hash functions are very good at detecting intercepted files. Second pre-image resistance states that "given an input M1 and its hash H, it is hard to find any other input with the same hash (i.e. hash (M1) does not equal hash(M2))" (Preneel). And the final property collision resistance states that "it should be hard to find two different inputs m1 and m2 of any length that has the same hash output" (Stevens 9).

Hash functions are very important to cryptography and are used all over the internet to secure information like passwords or verify the integrity of the file without exposing or opening the file. For example, if you want to check the file you received is not intercepted by anyone, then all you have to do is compare the original hash with the hash of the file you just received. If the hash values are same then it's the same file, otherwise, someone has intercepted the file and changed information. Some of the most used and widely used algorithms in this category, that are known as the cryptographic hash functions include MD5 and SHA-1.

MD5, message digest algorithm, was created by Ronald Rivest in 1992 (wang). And SHA-1, secure hash algorithm, was designed by United States' National Security Agency



(Steven). Both algorithms follow the properties of hash functions in similar manner, except the size of the hashed message. MD5 produces 128-bit hash value, which is 32-digit hex-decimals, and SHA-1 produces 160-bit hash value, which is 40-digits long hex-decimals. Both algorithms are based upon Merkle Damgard, which was used to create collision resistance hash functions (Steven). Both algorithms were widely used because it

offered fast and secure way to protect and check the information. However, in today's modern world with supercomputers and far more advanced technology, there are many hidden risks involved in storing information securely without anyone else intercepting it.

Attacks on hash functions

Just like everything else in the world, nothing is perfect including the most secure hash functions. Even though there is no way of getting the input from the hashed output, hash functions can still be broken. A problem in hash function appears when it fails the collision resistance property, which is when two input produces the same hash output. When two inputs produce the same hash function, it can allow hackers to smuggle the malicious file because it contains the same hash output (Brandom).

There is a mathematical principle called pigeonhole principle that says that if there are 10 pigeons and 9 pigeonholes then there must be one hole which contains two pigeons. Similarly, there are incredible number of documents out in the world and hash is just a 32 or 40-digit long number, which means there might be some files who has the same hashed value. However, that is not a problem because the odds against are really unlikely that it's never going to happen. But if one can artificially create a hash collision, then he or she can gain access to malicious file. This is one of the major security concern in today's digitalized world. For instance, let's say NASA sends a very important document to its employees that has the information about who will be going to Mars along with its original hash to verify the integrity of the message. Now let's say I can intercept the file because I know way to create a file with the same hash, then I can change or remove everyone's name from the list and put Donald Trump on the list and send him to mars and maybe leave him there. This is just a minor example of what could go wrong if someone is able to create a file with the same hash. Hacker could also use this technique to

collect users' password to hack into their bank accounts or steal their personal information. Now, this would be quite difficult and would require lots of computer code but even the most widely used algorithms like MD5 and SHA-1 are prone to collisions like this which break the hash function making them insecure to use in today's technologized world where computers are getting faster and faster every year.

The simplest way to define hash function collision is to look at the birthday attack. The main idea behind birthday attack is that in a room fool of 23 people the probability that at least two people have a common birthday exceeds 50% (Preneel). "This can be exploited to attack a hash function in the following way: an adversary generates r1 variation on a bogus message and r2 variation on a genuine message" (Preneel). Then the probability of finding both messages that

$1 - \exp\ (-(r1 * r2) \div 2^n)\ \ for\ r1 * r2 = O(2^{\frac{n}{2}})$

have the same hash is about 63% when $r_1 = r_2 = 2^{n/2}$. With this methodology it is possible to crack any hash function, however, it is not easy because it requires tremendous time and computer power.

In 1993, Bosselaers “found a kind of pseudo-collision for MD5 which consisted of the same message with two different sets of initial value” (Wang). While trying to find collision for MD5 researchers were looking to find a pair of message, “each consists of two blocks, that produces collision” (Wang). That is finding a pair of message (m1, m2) and ($m1^{'}$, $m2^{'}$) such that

- (a, b, c, d) = MD5(a_0, b_0, c_0, d_0, m_0),
- ($a^{'}$, $b^{'}$, $c^{'}$, $d^{'}$) = MD5(a_0, b_0, c_0, d_0, $m_0^{'}$),
- MD5 (a, b, c, d, m1) = MD5($a^{'}$, $b^{'}$, $c^{'}$, $d^{'}$, $m1^{'}$),

where a_0, b_0, c_0, d_0 are initial values for MD5 (Wang). Researchers found that such collisions can be efficiently found, where finding the first block takes about 2^{39} MD5 operations, and finding the second block takes about 2^{32} MD5 operations (Wang).

One of the most important method to analysis or find collision in hash functions is differential attack. Differential attack searches for cipher text or plaintext whose difference is constant, and then investigate the difference of two messages m1 and m2, which is defined as m1 XOR m2 (Wang). "The combination of both kinds of differences gives us more information than each of them keep by itself" (Wang). For example, when the modular integer subtraction difference X' – X = 26 for some value of X, the XOR difference X' XOR X can have many possibilities, such as one-bit difference, two-bit difference, three-bit difference, similarity, or negative difference (Wang). The differential for two messages M and M' is defined as follows:

$$\Delta H_0 \xrightarrow{(M_0, M_0')} \Delta H_1 \xrightarrow{(M_1, M_1')} \Delta H_2 \xrightarrow{(M_2, M_2')} \cdots\cdots \Delta H_{k-1} \xrightarrow{(M_{k-1}, M_{k-1}')} \Delta H,$$

where if ΔH0 is the initial value difference which equals to 0 (Wang). ΔH is the difference of two messages and if ΔH = 0, then there is a collision between m1 and m2.

From the given $H_0 \xrightarrow{(M_0, M_0'), 2^{-37}} \Delta H_1 \xrightarrow{(M_1, M_1'), 2^{-30}} \Delta H = 0.$ it is very easy to find a collision on MD5 using following algorithm. First we repeat he following three steps until we find the first block.

- Select a random message M_0 (Wang).
- Modify M_0 by the message modification techniques, then M_0 and $M_0' = M_0 + \Delta M_0$ produces the first differential $\Delta M_0 \rightarrow (\Delta H_1, \Delta M_1)$ with probability 2^{-37} (Wang).
- Test if all the characteristics really hold by applying the compression function on M_0 and M_1 (Wang).

Then repeat the following three steps until the collision is found.

- Select a random message M_1 (Wang).
- Modify M_1 by the message modification techniques, then M_1 and $M_1 + \Delta M_0$ generates the second iteration differential $(\Delta H_1, \Delta M_1) \rightarrow \Delta H = 0$ with probability 2^{-30} (Wang).
- Test if the pair message lead to a collision (Wang).

It is easy to find collisions between messages m1 and m2 with the running time 239 on MD5 operations, which takes about an hour or so on IBM's p690 computer (Wang).

Because we can now easily create MD5 collisions, MD5 now is considered broken because you can have a file where it is possible to send something malicious and have it come out with the same hash. This is where the computer speed comes in; if the hash is too slow no one would want to use it because it would simply too long. But if the hash is too fast then you can create a new file or a document in a few processor cycles then one can easily create a document that matches a particular hash. And because the computers today are fast enough its is very easy and quick to create hash collisions. Since MD5 was broken back in 2005, everyone moved to SHA-1, however, recently google and CWI successfully found the first collision for SHA-1.

Google and CWI institute's researchers were successful in finding a SHA-1 collision, which took about $2^{63.1}$ SHA-1 cycles and tremendous about of computer power and time (Steven). Researchers used similar technique as MD5 collision (differential attack), where they use two blocks in which difference in the first block cause a small difference and the output chaining value is canceled by the difference in the second block leading to collision (Steven). Although, creating a SHA-1 collision is not easy as it sounds. It cost google between $75000-$120000 using the computer power from amazon's EC2 server over a period of few months

(Stevens). That is about 10 quintillion SHA-1 operations and over 6500 years of single CPU computation and 110 years of single GPU computations (Steven). That being said, anyone who has money and tremendous computer power can break the SHA-1 hash functions.

It shows that both MD5 and SHA-1 have weakness and can be easily broken now. The core principle of detecting collision is to detect the last near-collision block of collision attack and use two key observation:

1. "There are only a small number of possible message block differences that may lead to feasible near-collision attack" (Steven, 188).
2. Both MD5 and SHA-1 uses differential attack that at some state has no differences at all in the working state (Steven, 188).

From this observations, one can easily check for collision given only one message of colliding pair of messages (Steven, 188).

Conclusion

With computers getting faster and with more people connecting to the internet every year, security has become a major concern in today's digitalized world. With technology advancement, it has become easier to find collisions in both MD5 and SHA-1 which is used all over the internet the protect and authenticate information. A broken hash function can also break the HTTPS, the encryption that is used today to protect more then half of the internet (Brandon). Since MD5 and SHA-1 have become insecure, industry has moved to SHA-2 also known as SHA-256, secure hash algorithm 2, which produces 64-digit long hash output. But however, as computer gets faster and smarter every year, SHA-2 might also be broken in next 10-15 years. The security of the hash function depends on its speed. If the hash is too slow then no would use it simply because it takes too long. However, if the hash is too fast then it is easier to break the

hash functions because one can create hash files rather quickly to match the hash of malicious file. Therefore, as technologies advanced, hash functions also need to produce larger output. Currently, SHA-2, for the time being, is secure, and SHA-3 is going through the process of being the industry standard and in a few years that will be the standard for hash functions.

References

Stevens, Marc Martinus Jacobus. Attacks on hash functions and applications. Mathematical Institute, Faculty of Science, Leiden University, 2012.

Wang, Xiaoyun, and Hongbo Yu. "How to break MD5 and other hash functions." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2005.

Brandon, Russell. "Google Just Cracked One of the Building Blocks of Web Encryption (but Don't Worry)." The Verge. The Verge, 23 Feb. 2017. Web. 29 Apr. 2017. <http://www.theverge.com/2017/2/23/14712118/google-sha1-collision-broken-web-encryption-shattered>.

Preneel, Bart. "Cryptographic hash functions." Transitions on Emerging Telecommunications Technologies, 5(4) (1994): 431-448.

Stevens, Marc, et al. "The first collision for full SHA-1." URL: https://shattered. it/static/shattered. pdf (2017).