International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-4, November 2019

Timestamp Injected Cryptographic Hash Function to Reduce Fabrication of Hash Collisions

Zulfany Erlisa Rasjid, Gunawan Witjaksono, Benfano Soewito, Edi Abdurahman

Abstract*: *Cryptographic hash functions are used in many applications. One important application is to ensure data integrity. Although there are many different types of hashing algorithms, MD5 is widely used to ensure data integrity in digital evidence. However, a weakness, where collisions can occur, has been found in the MD5 algorithm. With regards to digital evidence, this is a big issue. The integrity of the digital evidence becomes questionable due to collisions and hence it is not admissible in court. Many methods were used to find collisions, such as the Chosen-Prefix Collision and researchers have been improving collision finding algorithms. This paper concentrates on reducing the chances of collision by chopping the last 16 bits of the MD5 algorithm and injecting timestamp into the chopped parts. Experiments are performed to test this algorithm and the results show that the time taken to find collisions is longer using the MD5 with an injected timestamp. The chopping construction and the timestamp disrupt the iterative property of the hash function thus when dealing with digital evidence, there are less chances of hash collision and therefore the probability of the admissibility of the digital evidence in court is higher.

Index Terms*: *MD5, Digital evidence, Collision, Timestamp

I. INTRODUCTION

Cryptographic hash functions are functions that convert a variable length message to a fixed length data. One of the purposes of cryptographic hash functions is to ensure data integrity. There are several algorithms used for hashing. One of the most commonly used is the MD5 algorithm which was created by Ronald Rivest [1] in 1992. Digital forensic tools, software to extract evidence data from different storage media [2], mostly uses the Message Digest (MD5) hashing function to show the integrity of the data, thereby ensuring that the data have not been tampered with. However, a weakness has been found in the MD5 algorithm whereby collisions can be found. A collision is a condition in which two or more different files have the same hash value. The first discovery of the MD5 collision was by Wang, Feng, Lai, & Yu, 2004, and was presented at Eurocrypt, 2004. Since then cryptanalysts have been trying to find collisions in less time and using different strategies. Currently, MD5 is still used to prove the integrity of digital evidence. However when collisions are discovered in digital evidence, that evidence is not admissible in court, and cannot be used to prove innocence or guilt. Since the discovery of MD5 collisions, researchers have been developing techniques to improve the collision-finding algorithm [3], [4], [5], [6] rather than finding a solution or improving the hashing function itself. The purpose of this paper is to understand the nature of collisions and to create an algorithm that is based on the MD5 algorithm by injecting a timestamp into the hashed documents in order to reduce the creation of colliding files and therefore ensure the integrity of the data.

Manuscript published on November 30, 2019.
* Correspondence Author
Zulfany Erlisa Rasjid*, Computer Science Department, School of Computer Science, Doctor of Computer Science, BINUS Graduate Program, Bina Nusantara University, Jakarta, Indonesia.
Gunawan Witjaksono, Doctor of Computer Science, BINUS Graduate Program, Bina Nusantara University, Jakarta, Indonesia.
Benfano Soewito, Doctor of Computer Science, BINUS Graduate Program,Bina Nusantara University, Jakarta, Indonesia.
Edi Abdurahman, Computer Science Department, School of Computer Science, Doctor of Computer Science, BINUS Graduate Program,
Bina Nusantara University, Jakarta, Indonesia.

II. LITERATURE REVIEW

A. Cryptographic Hash Function

A hashing function is a function in which the purpose is to map any value or any size into a fixed size value [7]. The hashing function used in digital evidence integrity checks is a security-oriented hashing technique [7] where the purpose is to validate the integrity of the documents. There are several hashing functions. The most commonly used hashing functions used to validate digital evidence are the MD5 hash and the Secure Hash Algorithm (SHA) series. Hashing functions are one-way functions, meaning that it is practically impossible to reverse the hashing process. The cryptographic hash function is known to have an avalanche effect, meaning that a slight change in the input will result in a great change in the output. A cryptographic hash function must follow three properties: (1) pre-image resistance, (2) second pre-image resistance and (3) collision resistance.

B. Hash Collision

Cryptographic hash functions are irreversible, one-way functions. With regards to digital evidence, a document needs to be authenticated by the hash value. A document found at a crime scene is hashed before any further investigation is performed. After cloning, the hash is applied to the clone to ensure that the original and the clone are in fact, exactly the same [8].

A collision is a condition where two messages, m_1 and m_2, have the same hash value after applying a hash algorithm, thus $H(m_1) = H(m_2)$.. It is possible to find a collision using the Brute Force techniques, however, due to the length of the hash value, the complexity of finding collisions using the Brute Force algorithm is approximately $O(2^{n/2})$.

A cryptographic hash function has three properties: (1) Pre-image resistance, which is defined as "computationally infeasible" to obtain a value for x such that $H(x) = x'$ where $x \in Domain$. In other words, given x', it is difficult to find an x such that $H(x) = x'$; (2) Second pre-image resistance, which is defined as "computationally infeasible" to find a distinct $x' \in Domain$ for any $x \in Domain$ such that $H(x) = H(x')$. In other words, given x, it is difficult to find $x' \neq x$ such that $H(x) = H(x')$; and (3) Collision resistance, which is defined as "computationally infeasible" to find a distinct x and x', both in the $Domain$, such that $H(x) = H(x')$. A weak collision attack is an attack on the second pre-image resistance and a strong collision attack is an attack on the collision resistance property of a hash function [9]. Figure 1 and Figure 2 show illustrations of strong and weak collisions respectively.

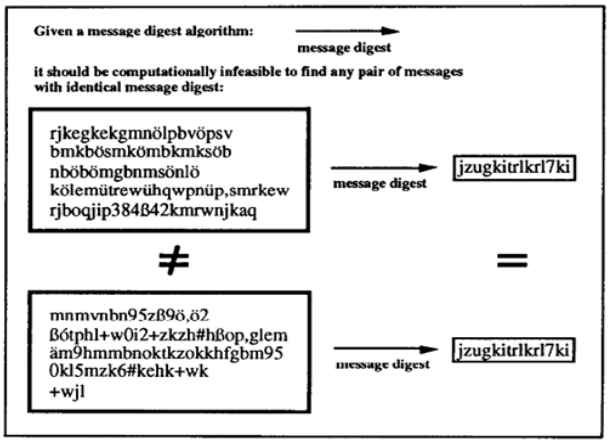

Fig. 1. Strong Collision [9]

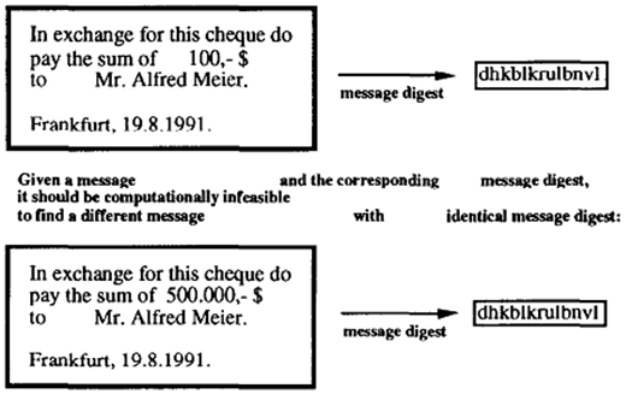

Fig. 2. Weak collision [9]

Differential cryptanalysis is used by researchers to create message collision. A hash function is applied to different inputs differences are analyzed. These differences are used to then build the attack. Any two message prefixes, for example P and P', and suffixes S and S' and are specifically selected, where $P \nu S$ and $P' \nu S'$ can be constructed in a way that would result in a collision [10]. A message digest always produces a fixed length value, where the original message can be much larger than the size of the hash values; therefore a collision can always exist, and according to the pigeonhole principle, several collisions may exist.

C. The Merkle-Damgard Construction

The Merkle-Damgard construction is the base of the cryptographic hash functions. It is an iterative function, wherein the output of one iteration will be included as the input for the next iteration. The input is a fixed value known as the initial value (IV). The message can be of arbitrary length. Each message is split into 512 block messages $m = m_1, m_2, \ldots . m_n$, and each block must be equal in length. If the last block's length is less than 512 blocks, then the last block will be padded until the length becomes equal to the other blocks. Functions are applied to each message block, with each result after applying the function to the message being included in the input for another function applied to the next block and so on until the last block. Message Digest Algorithms such as MD5 and the SHA series are based on this construction. MD5 and SHA series differ in terms of the resulting message length, the applied function f, and the number of rounds to which the function is applied (see Fig. 3).

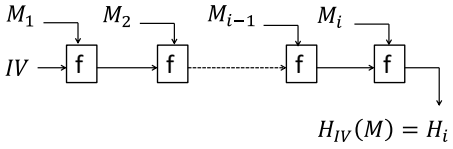

Fig. 3. The Merkle-Damgard Construction

D. MD5 Algorithm

The MD5 algorithm is a cryptographic hash function developed by Rivest in 1992 as an improvement to the MD4 algorithm [1]. The improvement was achieved by adding one more round to the MD4 algorithm and interchanging rounds 2 and 3. MD4 was considered harmful due to the fact that a full collision was found by Dobbertin [4] (Dobbertin, 1996a). The MD5 algorithm uses the Merkle-Damgard construction with four functions as follows:

$F(X,Y,X) = XY \vee X'Z$

$G(X,Y,X) = XZ \vee Y\,Z'$

$H(X,Y,X) = X \oplus Y \oplus Z$

$I(X,Y,X) = Y \oplus (X \vee Z')$

A weakness where a collision existed was later found in MD5 by Wang et al. (2004). This weakness discovery was presented at Eurocrypt, 2004. Ever since the collision was found, cryptanalysts have been trying to improve the performance for getting collisions and finding other techniques to find collisions.

E. MD5 Collision Attack

The first collision attack in MD5 was discovered by Boer and Bosselaers (1994). In their research they proved that the last step of MD5, allows for the creation of a collision [11]. This is caused by the fact that the result of the previous step was used in the addition process in the next step.

However, this discovery was not considered harmful, as it only affected some parts of the MD5 step (pseudo-collision), as stated by Dobbertin. The collision was found by using two different messages with a chosen initial value (IV).

Due to the collision found in MD5, Gauravaram, Millan, Dawson, & Viswanathan, 2006 developed a new hash function called the 3C construction which is an improvement on the Merkle-Damgard construction [12]. Through an enhancement to the block cipher using the chopping construction introduced by Coron, Yevgeniy, and Puniya (2002). They claimed that because of the bits chopping, collisions were reduced [13]. Using this construction, multi-block collision can still be found [14]. Wang et. al (2004) discovery centered around the construction of the MD5 collision using the initial value IV_0 and finding the differences in messages denoted by Δc_1 in order to find a message M'_k which is equal to another message M_k plus the differences as written in the equation below:

$IV_0 = A_0 = 0x01234567, B_0 = 0x89abcdef, C_0 = 0xfedcba98, D_0 = 0x76543210$ [1]

$M'_k = M_k + \Delta c_1, \Delta c_1 = (0,0,0,0, 2^{31}, 0,0,0,0,0,0. 2^{15}, 0,0, 2^{31}, 0), s = 4,11,14$ [2]

$M'_k = M_k + \Delta c_1 + \Delta c_2, \Delta c_2 = (0,0,0,0, 2^{31}, 0,0,0,0,0,0, 2^{15}, 0,0, 2^{31}, 0), s = 4,11,14$ [3]

Such that

$MD5(M, N_i) = MD5(M', N'_i)$ [4]

Collisions can also be created for other hash functions such as MD4, HAVAL-128 and RIPEMD.

In 2006, Black and Cochran (2006) found more collisions using differential paths, as was described by Wang et al. (2004), with an improvement in the performance using a technique to create the differential path. The collision had a slow performance and Klima (2005) speed up the collision search by 6 times by generating the first message colliding blocks more quickly [15].

Collisions found up to 2006 had no meaning, a colliding message that was found on any two files had no concrete importance. At this point this had no effect on digital evidence. However, later in 2006, researcher Kashyap (2006), used techniques to create collisions and develop those collisions into a valid and meaningful ones [16]. This attack resulted in two different files with different behavior, which had the same hash values.

With regards to performance, Wang's method for finding collisions has a complexity of . An improvement developed by Yu Sasaki and Naito (2005) created a collision with a complexity of 2^{30} . Stevens et al. (2009) constructed an MD5 collision using the Chosen-Prefix method. Two random messages that would create different Intermediate Hash Values (IHV) was selected and appended in a way that causes messages to collide [17], [10]. Nowadays, with the availability of high performance computers and parallel processing, such collisions can be found much faster. A single block collision was found by Kuznetsov using a parallel algorithm that took 11 hours, compared to the original program that took 3 weeks to find a collision [18]. Chiriaco, Franzen, Thayil and Zhang (2017) used Brute Force parallel programming to find a partial hash collision and improved performance by 50 percent.

F. Improvements to Hash Functions

Several improvements to hash functions have been proposed by researchers. It has been proposed to increase the size of the message digest to 512 bits instead of 128 bits [19]. However, with the increase of size, the performance of the hash function decreases. For example, a new hash function could be created based on a bitwise XOR operation from the additive constant in SHA-512 and the sine constant in MD5 [20]. A few months later a researcher designed a new hash function based on combining SHA-256 and MD5, and it was found to reduce weaknesses [21]. Another improvement performed by Zhang, Zhang and Yu (2017) used initial values that are processed though several rounds of iteration resulting in the improvement of the hash function with confusion and diffusion properties.

G. *Timestamp*

Timestamp is the current time recorded by the computer whenever a file is created, accessed, or modified. Different types of times are recorded in different file systems. Mac operating systems record the creation time of a file as well as the last accessed time and last modified time. In the Linux file system, the times that can be extracted are the access time, and modified time. Timestamp plays an important role when dealing with integrity, as it can be used to recognize that a file has been accessed or modified. Timestamp is used in many applications concerning the authentication of files, such as in message authentication codes to prevent the distribution of corrupted data/messages [22]. Timestamp-based mechanisms are used to ensure the integrity of distributed file systems [23]. Timestamp is also used in a Buyer-Seller Protocol to ensure the authenticity and integrity of a digital content [24]. In digital investigations, time plays an important role throughout the chain of custody of digital evidence, for example, the time that the evidence was found, the time of last accessed, the time of the last modification [25]. The operating systems provide the timestamp of each and every document. Concerning digital evidence, the different timestamps of a file show that the file has been accessed at different times and that there is the possibility that someone has accessed the file and probably tampered with it. Therefore, a file's timestamp plays an important role in data integrity. Timestamps can be extracted and used as a means to ensure that data has not been tampered with.

III. MATERIAL AND METHODS

A. Timestamp Injected Hash Function

Hash functions such as MD5 and SHA series have been compromised when collisions can be created. Hash collisions can be created based on a chosen prefix and appended in such a way that the two messages will collide [10], [26]. The Timestamp Injected Hash function was developed based on the Merkle-Damgard construction with the MD5 algorithm. The message/file is applied to MD5 algorithm, giving a fixed length value.

The resulting fixed length hash value is cut off as per the length of the timestamp (chopping) and the timestamp is injected at the last 16 bits of the resulting hash value which is the final value of the Timestamp Injected Hash function.

The timestamp is then extracted from the message itself and the extraction function records the timestamp up to 0.001 seconds. Therefore, it would be difficult for someone to duplicate the timestamp. Even though the time is known, it would be difficult to duplicate the timestamp up to the accuracy required hence the probability of getting a collision is reduced. It is estimated that it would take longer to get collision compared to the standard MD5 function. Experiments were conducted using 20 different files and by applying the standard MD5 algorithm and the Timestamp Injected MD5 algorithm. The time taken to find a collision was observed. The collision-finding algorithm for MD5 was developed by Marc Stevens [27] under a project called HashClash. The algorithm was downloaded from https://marc-stevens.nl/p/hashclash/ [27]. The algorithm uses the Chosen Prefix method. The same method is then applied to the Timestamp Injected Hash function. The time taken to find a collision based on the original MD5 and the Timestamp Injected MD5 was observed. The timestamp is considered in this situation because of the fact that when digital evidence is compromised, the file will have a different timestamp than the original file found at the crime scene. Fig. 3 shows the design of the Timestamp Injected Hash Algorithm. The basic algorithm is an MD5 algorithm and the extracted timestamp which is injected at the grey area in Fig. 5, just after the final round of the MD5 process. A hash collision is created based on a chosen prefix and is appended in a way that the two messages will collide [28], [29]. Using the MD5 algorithm, the resulting hash value is cut off as per the length of the timestamp and the time stamp is injected at the last 16 bits of the resulting hash value. The timestamp itself was extracted from the file (evidence). Greenwich Mean Time (GMT) was used as the standard for the timestamp and converted to a hexadecimal and hashed before it was appended to the chopped MD5 value. It would be difficult for someone to duplicate the timestamp and hence the probability of getting a collision is reduced. It is estimated that it would take longer to get a collision compared to the standard MD5 function. The time stamp method is shown in Fig. 5. First, the timestamp is extracted from the file. In this algorithm, the extracted timestamp is in the GMT format. The extracted timestamp is converted to a hexadecimal number and hashed. This way it would be more difficult to find the original timestamp due to the one-way property of the hash function. The last 16 bits of the hashed file are cut off so that the iterative pattern of the Merkle-Damgard construction is broken. The hashed timestamp is injected in place of the chopped 16 bits (Fig. 4)

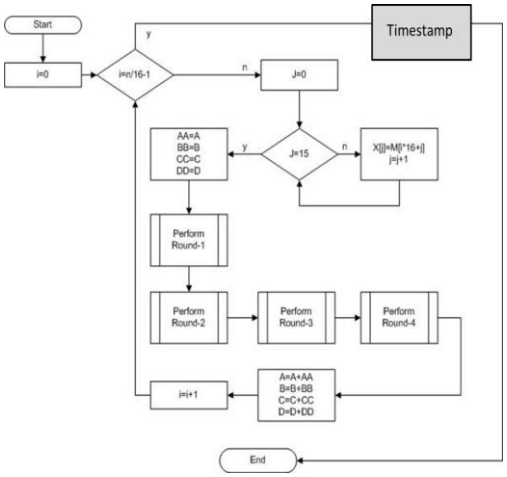

Fig. 4. Timestamp Injection

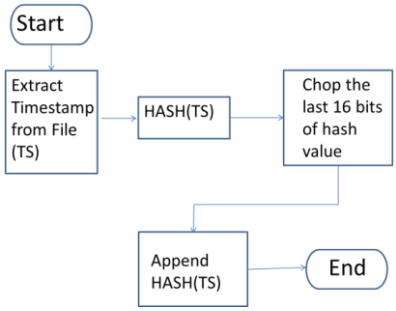

Fig. 5. Time stamping procedure

B. Experiments

Experiments were conducted using 20 different files with the standard MD5 algorithm applied and the Timestamp Injected MD5 algorithm. The file was used to find the prefix. The timestamp was converted into a hexadecimal number and hashed before appending it to the hash value. Fig. 5 shows the time stamping flowchart.

A Linux operating system was used for this development, although other operating systems will do as well. The timestamp was extracted using the gmtime() function, which retrieved the date and time of the accessed file up to a hundred milliseconds at GMT. Unfortunately, in Linux, only the accessed time and modified time can be extracted. However, for digital evidence, this is not an issue. The important thing is that the evidence is hashed at a particular timestamp before it is cloned for investigation. The cloned version was hashed again using the same timestamp. Therefore the create date of the file was not necessary to check the digital evidence integrity. Any timestamp will do as it is only required to see whether there is a difference between the timestamp in the original file and the cloned file. A difference in timestamp would mean that the document is a different document or the same document that has undergone changes.

IV. RESULTS AND DISCUSSIONS

Table 1 shows the results of the first 10 files that were used to create the collision. The file was used to obtain the prefix. Each run on the same file produced different files that collided. The time taken to produce a set of colliding files was recorded.

The time taken can be different for each files produced, even though the same file was used to obtain the prefix. However with the Timestamp Injected MD5 (TIMD5), after 20 minutes the colliding files were still not found. The timestamp was injected at the last part of the hash value, and therefore some the last part of the original hash value was removed in order to preserve the length. The Merkle-Damgard construction used the output in the intermediate hash value as the input for the next round. Because of the last part of the hash value was removed and replaced with the timestamp, the pattern was broken, thus making it more difficult to search for a colliding pair of files.

Table 1. Results of finding a collision.

No	File Name	Original	TIMD5	MD5 (seconds)	TIMD5 (seconds)
1	MD5.cpp	8a8b4d49d581d05f872ac594e92a26	022bb4f88709366d244ad2f715170324	29.159	>1200
2	Lecture Notes.docx	2189746542388b2e38a997fe0c23adfe	6fdf8f8e8713170324	18.5996	>1200
3	OutDec.txt	15a3b72f7b9bb1054dd141350619c379	15a3b72f7b9bb1054dd1413515170324	46.2141	>1200
4	try	943d795e3cea690ee763f2781ea8be9	54ed306eb0590d5c130b4cd215170324	46.2082	> 1200
5	try	943d795e3cea690ee763f2781ea8be9	54ed306eb0590d5c130b4cd215170324	54.2801	>1200
6	3May.png	08e169e4f93c866a6dbc3989a1de44e8	83f232752f4c58bd05d8d51f15170324	1.36682	>1200
7	Screenshot.png	6e8fafdf635b1ee943b7a701eca94013	a63f3611a9476deff938ff0215170324	1.40382	> 1200
8	fastcoll3	df066ada7ab2eb2face1d67cb7b96dc8	a0b5a049820095af12dc17ca15170324	0.443407	> 1200
9	buku cp.pdf	2ea233d0c7fe3013bb1abe23101957e2	ffe06c92ddc75c5382bc0d3415170324	26.6589	> 1204
10	bubble.exe	2a595417722b4534ee7f9eb71cb2b3b9	b1cc6cae190a4563fadc1ba91517024	67.6935	> 1205

The collision finding program was run twice using the same file used as the prefix. The program found two files that are different, which were checked and verified by the contents. This file was then hashed using MD5 to show that those two files collided. All of the resulting pairs of files had the same hash value. The results of the first three runs (using MD5) are summarized in Table 2.

Table 2. Hash values of resulting files

Run number	Hash Values	File Name
1	5b9f1c403880cc10d9e36647317055bb	msg1
	5b9f1c403880cc10d9e36647317055bb	msg2
2	5f39068638a6684fed3b2ad87a4fef7f	msg1_1
	5f39068638a6684fed3b2ad87a4fef7f	msg2_1
3	09cc45680045af9ae8374cd31f99f671	msg2_2
	09cc45680045af9ae8374cd31f99f671	msg1_2

This experiment shows that colliding pairs of files was created when MD5 was used, however, using the Timestamp Injected Hash function, colliding files were not created, even after a waiting time of 60 minutes or more. Therefore, the creation of a pair of colliding files using TIMD5 takes much longer (over 60 minutes). The graph in Fig. 6 shows the files used as prefixes against the time required to create colliding files. It can be seen that the time depends highly on the prefix used. When running using the Timestamp Injected algorithm, it takes a very long time before the colliding files are found. Note that it is always possible to create a pair of colliding files under any hash algorithm when using Brute Force algorithm; however, when using that algorithm, the complexity for finding a collision is 2^n, where $n > 512$. A comparison graph of the time for finding colliding files using MD5 and the Timestamp Injected MD5 is shown in Fig. 6.

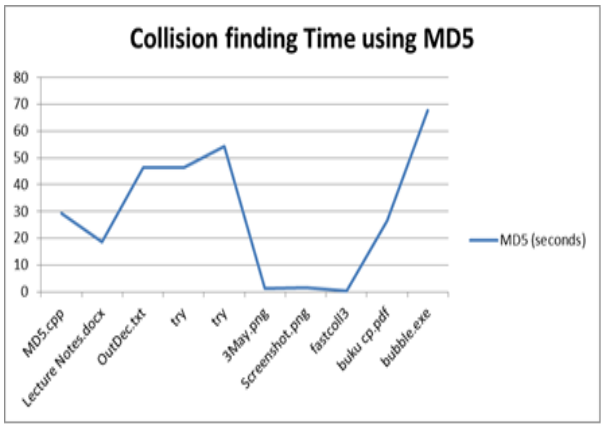

Fig. 5. Colliding files created using MD5

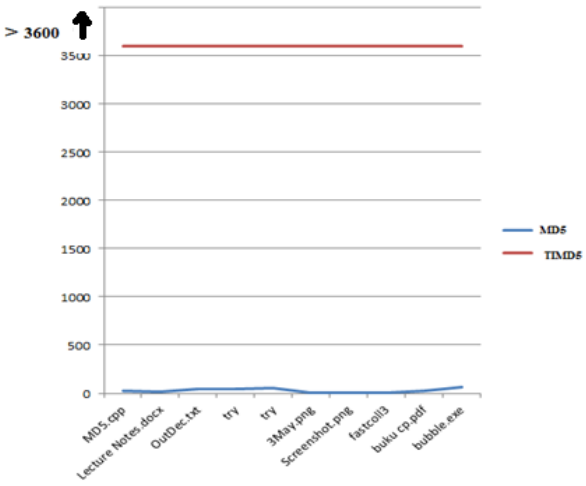

Fig. 6. Comparison using MD5 original and Timestamp Injected MD5.

The pseudo-code of the Timestamp Injected Hash function is as follows:

Pseudo-code : Timestamp Injected MD5 (TIMD5)
Purpose : Inject timestamp into MD5 hash function
Pre-condition:all the following function has been declared:
function MD5(file) : calculate message digest of file
function gmtime(buffer, mod_time): function to obtain modification
date and time of a file
Post-condition: none
Input: a file of any length (m)
Output: Message Digest of length 128 bits
BEGIN

1. $Let \quad m \leftarrow (m_1, m_2, ...m_{n-1}, m_n)$
2. $H_{org} \leftarrow MD5(m), H = \{0,1\}^{128}$
3. *call stat(m,buffer)*
4. $TS_m = gmmtime(buffer, st.mtime)$
5. $H_{TIMD5} \leftarrow$ *return the first n-16 bits of* $H_{org} \parallel HEX(TS_m)$

END

IV. CONCLUSION

MD5 hashing algorithm is widely used in making sure that the document's integrity is still intact. With regards to Digital Forensics, the evidence integrity must be protected so that the evidence is admissible in court. To ensure document's integrity, hash values are taken before and after the investigation. Collision shows that there are two or more files having the same hash values. When this happens, the evidence is considered to have been compromised and thus the evidence is not admissible in court, which can be fatal depending on the case. A Timestamp Injected hash function was developed by injecting timestamp to replace the chopped bits in the original hash function. Experiments are executed using different types of files using the original MD5 algorithm and the timestamp Injected hash. Using the Chosen Prefix algorithm in the HashClash project to create collisions, the experiment shows the results of using the Timestamp Injected hash function did not create colliding files in the first 20 minutes compared to using the original MD5 algorithm where colliding files were created within the range of 1 to 60 seconds. Future research should focus on improving the hash function such that collisions are hard or even impossible to generate.

REFERENCES

1. Rivest R. MD5 Message Digest. 1992. p. 1–21.
2. Yang CH, Yen PH. Fast deployment of computer forensics with USBs. In: Proceedings - 2010 International Conference on Broadband, Wireless Computing Communication and Applications, BWCCA 2010. 2010. p. 413–6.
3. Wang X, Feng D, Lai X, Yu H. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. IACR Cryptol ePrint Arch [Internet]. 2004;5(October):5–8. Available at: http://web.mit.edu/fustflum/documents/crypto.pdf
4. Dobbertin H. Cryptanalysis of MD4. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 1996;1039:53–69.
5. Xie T, Feng D. How To Find Weak Input Differences For MD5 Collision Attacks. Adv Cryptol - CRYPTO 2009. 2007;1–20.
6. Stevens M, Karpman P, Peyrin T. Freestart collision for full SHA-1. 2016;2012:1–21.
7. Chi L, Zhu X. Hashing Techniques: A Survey and Taxonomy. ACM Comput Surv [Internet]. 2017;50(1):1–36. Available at: http://dl.acm.org/citation.cfm?doid=3058791.3047307
8. Granja FM, Rafael GDR. Preservation of digital evidence: Application in criminal investigation. Proc 2015 Sci Inf Conf SAI 2015. 2015;1284–92.
9. Bauspiess F, Damm F. Requirements for cryptographic hash functions. Comput Secur. 1992;11(5):427–37.
10. Stevens M, Sotirov A, Appelbaum J, Lenstra A, Molnar D, Osvik DA, et al. Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In: Advances in Criptology CRYPT 2009. 2009. p. 55–69.
11. Boer B Den, Bosselaers A. Advances in Cryptology — EUROCRYPT '93. Adv Cryptology—EUROCRYPT'93. 1994;765:293–304.
12. Gauravaram P, Millan W, Dawson E, Viswanathan K. Constructing secure hash functions by enhancing Merkle-Damgard construction. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 2006;4058 LNCS(Iv):407–20.
13. Coron J-S, Dodis Y, Malinaud C, Puniya P. Merkle-Damgård Revisited: How to Construct a Hash Function. In: Advances in Cryptology. 2005. p. 430–48.
14. Joscák D, Tuma J. Multi-block Collisions in Hash Functions Based on {3C} and {3C+} Enhancements of the {Merkle}-{Damgard} Construction. Inf Secur Cryptol – ICISC 2006 [Internet]. 2006;4296:257–66. Available at: http://dblp.uni-trier.de/db/conf/icisc/icisc2006.html#JoscakT06
15. Klíma V. Finding MD5 Collisions – a Toy For a Notebook. Int Assoc Cryptologic Res. 2005;(February):1–7.
16. Kashyap N. A Meaningful MD5 Hash Collision Attack [Internet]. 2006. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.2659&rep=rep1&type=pdf
17. Stevens M, Lenstra AK, Weger B De. Chosen-prefix collisions for MD5 and applications. Int J Appl Cryptogr [Internet]. 2012;2(4):322. Available at: http://www.inderscience.com/link.php?id=48084
18. Kuznetsov AA. An algorithm for MD5 single-block collision attack using high- performance computing cluster. 2014;
19. Walia P, Thapar V. Implementation of New Modified MD5-512 bit Algorithm for Cryptography. Ijirae. 2014;1(6):87–97.
20. Sivakumar T. A New Symmetric Cryptosystem using Randomized Parameters of SHA-512 and MD5 Hash Functions. 2016;6(4):600–6.
21. Hakim S, Fouad M. SCITECH RESEARCH ORGANISATION | Improving Data Integrity in Communication Systems by Designing a New Security Hash Algorithm. 2017;6(2):638–47.
22. Mondal A. TDMAC : A Timestamp Defined Message Authentication Code for Secure Data Dissemination in VANET. Adv Networks Telecommun Syst (ANTS), 2016 IEEE Int Conf. 2016;
23. Watanabe, Hidenobu; Iwama, Tsukasa; Murata KT. Improvement of the Integrity Verification Application using Timestamp Mechanism System For Distributed System. 2014 IEEE 38th Annu Int Comput Softw Appl Conf Work. 2014;
24. Kumar A, Ghrera SP, Tyagi V. An ID-based secure and flexible buyer-seller watermarking protocol for copyright protection. Pertanika J Sci Technol. 2017;25(1):57–76.
25. Cosic J, Baca M. Do we have full control over integrity in digital evidence life cycle? Inf Technol Interfaces (ITI), 2010 32nd Int Conf. 2010;429–34.
26. Stevens M, Lenstra A, Weger B De. Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. Adv Cryptol – Eurocrypt 2007 [Internet]. 2007;vol. 4515(Lecture Notes in Computer Science):1–22. Available at: http://www.win.tue.nl/hashclash/EC07v2.0.pdf
27. Stevens M. Project HashClash [Internet]. [cited November 20, 2015]. Available at: https://marc-stevens.nl/p/hashclash/
28. Dobbertin H. Cryptanalysis of MD5 Compress. Ger Inf Secur Agency. 1996;
29. Black J, Cochran M. A Study of the MD5 Attacks : Insights and Improvements. Fast Softw Encryption. 2006;262–77.

AUTHORS PROFILE

Zulfany Erlisa Rasjid, B.Sc., MMSI., obtained her Magister of Information Management in 2010 from Binus University, and Bachelor of Science from New South Wales University, Sydney, Australia majoring in Computer Science and Mathematics in 1982. She started her career at PT. Astakona Indonesia (a distributor of Texas Instruments Computer) as a System Engineer and a few years later at PT. Metrodata Electronics, also as a system Engineer. She was the IT Manager of Otto International from 1992 until 2004. She was involved in several projects such as National Olympics (PON XII), Imaging Systems, such as Document Tracking Systems for the Regional Planning & Development Board (BAPPEDA - DKI). She started teaching at STTI (Sekolah Tinggi Teknologi Indonesia) as part time lecturer in 1998-2000 and was Head of Programme at Inti College Indonesia in 2008. She joined Binus University in 2002 as a part time lecturer and then became a Subject Content Specialist (Network Centric Division) in 2010 until present. She participated in several International conferences as speaker. She has written 6 journal articles of which 4 are indexed by Scopus.

Benfano Soewito, M.Sc., Ph.D., completed his bachelor level education in the FMIPA department of Physics, Airlangga University, Surabaya, Indonesia. He completed his master level education in the field of Computer Engineering at the Department of Electrical and Computer Engineering, Southern Illinois University, United States of America (USA) in 2004 and completed his doctoral level education in 2009 at the same faculty and university. After completed his undergraduate level education, Benfano worked at a company in Surabaya, which produces crystal oscillator as the head of engineering with the main task in the development of design specification, manufacturing technique and quality improvement of crystal oscillator unit. While completing his master and doctoral degree, he also worked as an assistant professor, research assistant, support computer specialist, and developed several websites inside the environment of Southern Illinois University, USA. After finished his doctoral studies, he chose to pursue a career in the education field by joining Bakrie University. Since 2013, he became full-time lecturer at the Faculty of Graduate Programs at Bina Nusantara University, Jakarta. He has a high interest toward researches in computer science with special interest in information technology as include Internet packet processing and scanning, router development as well as security and computer network.

Gunawan Witjaksono, B.Sc., M.Sc., Ph.D received the B.S. (magna cum laude) and M.S. degrees in electrical engineering from Michigan Technological University, Houghton, MI, USA in 1992, 1994, respectively, and the Ph.D. degree in Electrical and Computer Engineering from the University of Wisconsin-Madison, USA in 2002. In June 2002, he joined Denselight Semiconductors Pte Ltd in Singapore, where he developed high-speed, long wavelength, distributed feedback lasers. Following that, he was with Finisar Malaysia to develop uncooled, high-speed optical transceiver; Later he joined Department of Electrical Engineering, University of Indonesia in 2005 to 2007 before joining MIMOS when he held various key positions as Principal Researcher, Director of Research and Sensor System Architect until 2016. He is involved in IoT Research Interest Group (RIG) and a lecturer at Bina Nusantara University, and currently an Associate Professor at Electrical and Electronic Engineering, Universiti Teknologi PETRONAS, Malaysia and certified Information System Auditor (CISA) holding a Professional Engineer (IPM). He has published more than 70 scientific papers and invented more than 30 patents in the fields of optical sensors, information security system, and III-V based laser, fiber optical telecommunication, Internet of Things, as well as renewable energy.

Prof. Dr. Ir. Edi Abdurahman, MS., M.Sc., obtained his Professorship of Statistics 2008 and inaugurated at Bina Nusantara University in 2009. He graduated from Iowa State University, Ames, USA with a doctoral degree in Statistics in 1986. The Master of Science was conferred from the same university in Statistical Survey in 1983. He obtained Agricultural Engineering degree (Ir.) from IPB (Institute of Agriculture Bogor) with cum laude in 1978. Since 1986, he served as lecturer at Bina Nusantara University. His teaching expertise includes Linear Algebra, Information System Research Methods, Discrete Mathematics, Operation Research, Mathematical Statistics and others. He has more than 18 years experience as a professional Statistics Consultant at various companies in Indonesia and attended several national and international Statistical courses. He has been invited as a speaker for international conferences and various seminars.,e.g. Statistics and Data Utilization for Agricultural policy of Indonesia in Myanmar and the Economic Modeling for Agricultural Sector, the case of Predicting some Agricultural Strategic Commodities, the Office of Agricultural Economics (OAE) and JICA ASEAD in Bangkok, Thailand. He also published numerous research in Statistics. He is also a member of the American Statistics Association, the International Association of Engineers (IAENG) and honorary member of MU SIMA RHO Society.