ICIC Express Letters
Part B: Applications
Volume **12**, Number **3**, March **2021**

ICIC International ©2021 ISSN 2185-2766
pp. 231–241

HASHING GENERATION USING RECURRENT NEURAL NETWORKS FOR TEXT DOCUMENTS

Raed Abu Zitar[1,*], Nidal Al-Dmour[1], Mirna Nachouki[1], Hanan Hussain[1] and Farid Alzboun[2]

[1]College of Engineering and Information Technology
Ajman University
University Street, Al Jurf-1, Ajman 346, United Arab Emirates
*Corresponding author: r.abuzitar@ajman.ac.ae

[2]Department of Information Technology
Public Sector
Fatemah Bent Al-Hasan St., Amman 11183, Jordan

Received September 2020; accepted November 2020

Abstract. *In this paper a novel technique is proposed to generate hashing values for text documents. The approach uses Recurrent Neural Networks (RNNs) for this purpose. RNNs are dynamic and temporal type of Neural Networks (NNs) that evolve continuously based on subsequent vectors of inputs. The capabilities of RNNs to incorporate present values of inputs with previous values exploiting relations and semantics of the text make it a competitive paradigm to discover the internal representations within text data in a unique way. Two types of RNNs are tested and compared to traditional methods. Adequate review has been done to existing techniques and the results obtained in this work demonstrate the applicability of this artificial intelligence paradigm in generating hashing values for plain text. RNNs are highly flexible, compact, and parallel in nature. Their capabilities are exploited in this paper as future competent technique in text hashing.*
Keywords: Recurrent neural network, Hashing methods, Collision probabilities, Intelligent paradigms, Message digest

1. Introduction.

Exchanging data among users through wired or wireless networks is today facilitated by the development of various communication technologies and the Internet. However, with the advancement of e-commerce and online banking, confidentiality and integrity constitute a serious issue to ensure that data sent via a specific communication network is not revealed to unauthorized entities and has not been altered in an illegal way. During data transmission, integrity is preserved by preventing insertion, deletion and substitution breaches. Hash functions are used to protect against any intentional or accidental alterations. A hash function is defined as the process that transforms a file of an arbitrary size to another file of small fixed size called hashed value, "message digest" or "fingerprint". The proposed technique is used to protect, secure, and preserve integrity of data. It is a one-way operation that transforms any input of plain text data into a particular output that is generally called cypher [1]. It should not be possible to retrieve the value of the input from the output generated by the hashing algorithm. Therefore, this hashing process provides transformation of data in one direction only, from the original file to a message digest data. There is no opportunity to retrieve back the original file as opposed to the encryption where data can be retrieved using a key.

To provide confidentiality and integrity for transmitted data, Saraireh et al. [2] have proposed to integrate different security algorithms, such as hashing function as well as cryptography and steganography techniques as methods to maintain security in data

DOI: 10.24507/icicelb.12.03.231

transmission [3]. The cryptography is used to convert initial data into incomprehensible one. Steganography consists of using multimedia to hide the message in a particular data carrier such as image, text, and video [4]. The hashing function is executed using Message Digest 5 algorithm (MD5) to generate 128-bit hash value, which is considered as a fast hashing algorithm [5] for providing data integrity services. This combination provides a secure and robust communication media that guarantees that hidden data is not visible over the cover images; this preserves the confidentiality and integrity of transmitted data against threats and attacks. New hashing techniques integrate convolutional neural network, discrete hash function learning, and ranking function optimizing into a unified framework [6]. Other techniques use hashing variable length algorithm to secure messages with the same length of results and also in addition to cryptography, this algorithm can also be used as message compression with very reliable security [7]. In [8], Discrete Multi-Graph Hashing (DMGH) is used to address a multi-graph learning technique to fuse multiple views, and adaptively learns the weights of each view. This techniques is applied for large-scale visual retrieval tasks. In [9], authors propose a robust image hashing based on Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) for copying detection. They used DCT to extract stable low-frequent coefficients for constructing feature matrix and exploiting DWT to convert the feature matrix to a compact hashing.

Hash functions play an important role in network security, cryptography as well as in digital signature to inspect the authenticity and reliability of transmitted data. They are used for key generation in symmetric and asymmetric key cryptosystems. Different algorithms such as secure hash algorithms: SHA-1, SHA-2, SHA-3, MD4, MD5 and Whirlpool provide different levels of security [10].

Hash algorithms with digital signatures have been used in the cloud-computing environment to ensure that the privacy and integrity of the stored data are preserved [11]. In this domain, users usually store their data in the cloud without worrying about storage management concerns. However, a crucial problem occurs related to data integrity: users do not have control on the physical storage of their data, which makes it difficult to be protected from being accessed by unauthorized entities. Different mechanisms generally ensure data integrity in the cloud. They mainly detect integrity violations using various techniques including checking sum, which is usually generated using a one-direction hash function. In that method it is not possible to restore the original file based on a specific check-sum value that could be modified or corrupted. For this reason, these mechanisms cannot recover data once a damage is detected [12]. Digital signature has been proved to ensure security and to authenticate access to data in a cloud-computing environment. Nevertheless, it is not possible to generate a message digest using those techniques. A comparison made between several hash functions indicated that in terms of security, it is preferable to apply SHA-512 digital signature than MD5, but in terms of speed, the latter one is much better than SHA-512. A signature scheme has been proposed to resolve data integrity issue in cloud-computing storage with better performance compared to traditional algorithms [11]. Singh et al. [13] have also investigated on how to preserve data integrity in cloud computing environment. Their mechanism depends on applying a hash function to generating a hash value of the file to be saved in the cloud before encryption. The next step is to store this hash value locally and to upload, encrypt and save the file on the cloud. When the user retrieves the decrypted data from the cloud, a hash value is calculated to ensure data integrity. If the new generated value matches the earlier calculated one, then data integrity is preserved; otherwise, data has been modified.

Hash functions have been also used to ensure privacy and integrity in web applications, which have experienced major development over the previous years and have changed the way businesses are provided and scattered [14,15]. However, preserving the security of data transmitted through web applications remains an issue, where various types of attacks may occur [16]. Commonly, a password is converted to its hashed cypher form before it is stored to ensure data integrity and security [17]. As part of the authentication process, when the user needs to access a web application, the password entered is first hashed and then converted to its cipher value before it is checked against its cipher value stored. Usually a function transforms this plain password text into cipher text in order to check data for its security, authenticity and integrity [18]. One of the most common cryptographic hashing functions used in web applications, and specifically in e-commerce applications, is MD5; it avoids data being forged or hacked and helps maintain data integrity and security during transmission as well as in storage [19]. As mentioned by Bhandari et al., the use of an enhanced MD5 algorithm to store data promotes user privacy and helps web applications to be more secured [1].

Hash algorithms are also used to detect viruses while transmitting data. In [20], the authors suggest to create and attach a code for each file using a cryptographic hash function. This code is used later to detect if the file has been altered. This technique, called virus localization, permits localizing the changes within the file. It consists on repeatedly applying a cryptographic hash function that identifies the most significant and recognized virus contamination procedures (such as rewriting, inserting, appending and prepending techniques) in the various subsections of the traditional file [20].

Robust hashing algorithms have been applied to Model-Driven Engineering (MDE) to protect artefacts in industrial environments. For equivalent input data, these algorithms compute comparable outputs. For this reason, they have been used to create a key constructing block to protect intellectual property, assess authenticity, as well as to make comparison and retrieve solutions for various application domains [21].

Various hash algorithms have been examined to enhance the security and efficiency of transmitted data, mainly by increasing the message digest length they produce, which raised another concern related to the increased bandwidth utilization and low output [22]. This issue was examined in particular in [23] where the authors have presented a hashing method called Efficient Hash Algorithm (EHA), which is mostly based on SHA-160 architecture. EHA consists on increasing the algorithm's complexity, keeping the length of the message digest as smaller as possible. This technique allows to improve the overall data security.

In [23], authors have evaluated the performance of their algorithm compared to existing techniques, such as MD2, MD5, SHA-160, SHA-256, SHA-384 and SHA-512. This comparison was performed using NIST statistical test suite for random numbers and avalanche criteria. The results showed that EHA is efficient in terms of unpredictability and throughput; and accordingly, it may be applied for any situation dealing with sensitive data. The weaknesses of MD5 were also discussed in [1], where the authors have suggested to vary the length of this algorithm and to use a hash key in order to produce the cipher form related to the original data. To increase the security, integrity and effectiveness of data, they have proposed a mixture algorithm based on flexible output length, instead of the fixed output generated by the original MD5 algorithm. They have also introduced a hash key to transform the plain data text into its cipher version to maintain data security and integrity. They have implemented their enhanced MD5 with web technologies such as PHP, JavaScript, HTML5 and CSS3 with PHP as server. It is to be noted that the execution time of this enhanced version of MD5 is slower than the original one, as a cryptographic technique, used to maintain data integrity and security, that needs to process data [1].

Different hashing algorithms were also reviewed in [24] and the authors conclude that the security and the length of a hash value are interrelated; when the length increases, the security becomes higher. They have also found that MD5 algorithm is faster than the SHA-512 procedure, which is more secure than MD5. As a conclusion, the authors stated that today with the advanced technology, it becomes easier to crack the hashes generated by SHA-512. For this reason, they recommend considering adding various techniques that make attacks become slower, such as PBDKF2, BCrypt and SCrypt [24].

Based on the previous introduction, our proposed technique offers an original approach that relies more on computations rather than symbolic and logical processing of bits and bytes. The contribution of the paper can be summarized in the following points.

1) The RNN, a new hashing technique that utilizes artificial intelligence paradigms, is used. The RNN is parallel and compact in nature. The parallelism is an advantage that can be utilized with parallel programing and multithreading for faster execution of code.
2) The RNN method provides flexibility in message length as increasing the number of neurons will provide more hashing capabilities. This flexibility is advantageous over other methods and can be tuned to compromise between the needed level of integrity and the speed of execution.
3) The weights of the RNN that are used in the hashing are not unique and can vary with different initializations. This is an advantage that adds more security to this method. The weights could be updated frequently which makes it harder for hackers to crack the hashing.
4) For more varieties and for the option of adding more elusiveness in the hashing process, two types of RNN can be used: one with single feedbacks and the other one with full feedbacks. Both could be alternatively selected in hashing in a random order.

The organization of the rest of the paper can be summarized as follows:

1) Section 2: Literature review and comparisons between the recent techniques of hashing.
2) Section 3: The proposed RNN method for hashing.
3) Section 4: Discussions and conclusions.

2. Literature Review.

In this section, different hashing techniques with more technical and historical details are presented. Various cryptographic hash functions, such as MD5, SHA-1, SHA-2, and SHA-3, have been used to apply privacy, confidentiality, security, and integrity of transmitted data [1]. Using mathematical formulas, they convert a text having some data of a random-length to a fixed-length hash output. The input data, which can be a password or a simple key that requests to be protected, is almost difficult to regenerate from its hash value alone [2].

MD5 is a message digest algorithm that was created in 1992 to be a safe replacement of its antecedent MD4 [3]. It takes an input data of random-length and produces an output of 128-bit hash code. The input message is divided into portions of 512-bit blocks (sixteen 32-bit words). If the input data is not an integer multiple of 512-bit blocks, it is expanded so that its size is divisible by 512 [4]. MD5 consists of four rounds of 16 equivalent calculations, or 64 operations. With MD5, it is easy to produce a hash collision that occurs when two different inputs generate the same hash output. For this reason, MD5 is considered unsecure, as there are a significant number of collision attacks. MD5 is generally useful to process data of short length such as passwords, credit card numbers, or any field values of data to be stored in databases systems [4]. SHA-1 is equivalent to MD5 and generates a 160-bit hash value. It takes an input data of random length and produces an output of 160-bit message digest. The input data is broken up into portions of 512-bit blocks (sixteen 32-bit words). In case the input data is not an integer multiple of 512-bit blocks, it is expanded so that its length is divisible by 512 [4]. SHA-1 function works on a 160-bit state, divided into five 32-bit words. An input data requires 80 similar operations to be processed. SHA-1, or SHA-160, was being used in various applications. It was known to be much more secure than the previous hashing methods. Limited attacks were identified on SHA-1, which are much less severe than those recognized on MD5 [4]. However, in 2005, different security issues and cryptographic weaknesses were identified with this algorithm, which arises the need to apply a stronger hash algorithm [2]. SHA-2 was developed to produce a more secure and robust hash function than SHA-1. Its algorithm is based on SHA-1 and has three comparable hash methods with different block sizes, known as SHA-256, SHA-384, and SHA-512, which differ in the word and block sizes. They all take an input of random size. SHA-256 produces an output of 256-bit hash code, using 32-bit words, 512-block size, and 64 similar rounds. SHA-384 makes an output of 384-bit hash code, using 64-bit words, 1024-block size, and 80 similar rounds, and SHA-512 generates an output of 512-bit hash value with 64-bit words, 1024-block size, and 80 similar rounds [4]. These three SHA-2 functions use unrelated number of rounds, unrelated shift values, and additive coefficients, but their structures are basically the same. Regarding security issues, no successful reports of any security-related attacks were reported [2]. SHA-3 was published in 2015, its structure is different from MD5, SHA-1, and SHA-2 structure, but supports the same hash lengths as SHA-2. The primary purpose behind the creation of SHA-3 algorithm is to be resistant to assaults like length expansion, in opposition to MD5 and SHA-1 that were decided to be unprotected to theoretical attacks [5]. In their paper [5], the authors discussed some limitations related to the application of different hash functions. They recommended not using the MD5 algorithm to encrypt passwords. This is mainly because this algorithm is fast in processing small size of data and that an attacker can try billions of candidate passwords per second. The authors have also found out that, SHA-1 requires a lot of computing power and resources, and that SHA-256 and SHA-512 have an increased resistance to the collision because they produce more extended outputs (256-bit and 512-bit respectively) than SHA-1 (160-bit).

In [2], the authors analyzed the security level of hash functions based on the randomness and uncorrelated output generated by the function for a correlated or random data. They have found out that all SHA algorithms generate arbitrary results because they have succeeded in the randomness tests that were applied. The authors have also compared the acceptance of SHA-1 and SHA-2 in the market. They found out that SHA-2 functions were not adopted quickly in comparison to SHA-1 because there is no support for SHA-2 on systems running Windows XP SP2 or older. In 2017, Microsoft declared that Internet Explorer and Edge would stop using SHA-1. In 2015, the Google Chrome group broadcasted a plan where they have stopped progressively the use of SHA-1 in their web browser [2].

A comparison among SHA-1, SHA-256 and MD5 was made in [6] regarding encryption time, power consumption, and latency for different input sizes. The authors have concluded that there is slight difference in the encryption times of the three methods for an input length of 600kb. Conversely, this difference which increases with the input length becomes more important. Moreover, for an input length of one Mb, SHA-256 spent more time to be executed than MD5. Similarly, SHA-256 consumed five times more than MD5 to encrypt an input data of length 10MB. Concerning both power consumption and latency, it was noticed that MD5 used the least amount of power and produced the greatest latency, pursued by SHA-256, then SHA-1.

In conclusion, the authors declared that MD5 is the best algorithm with regards to power consumption and encryption time. SHA-1 spent the greatest power among these three functions. MD5 has the highest latency, tailed by SHA-256, then SHA-1 [24]. Finally, SHA-256 is more secure than SHA-1 and MD5. However, it is considered lengthy and time consuming compared to the other members of its family [2] for data input of long size. However, it is considered faster than SHA-512 for hashing small data strings.

In our work, we are introducing a new method that technically differs from all previous techniques. We demonstrate the use of artificial intelligence paradigms in performing a highly nonlinear task of generating hashing output. The approach is flexible and provides many possible solutions for the same input text depending on the initial weights. It will be difficult to crack this hashing generated by the RNN as it is possible to re-initialize

Table 1. Comparison between various cryptographic algorithms

Parameters	Algorithms					
	MD5	SHA-1	SHA-256	SHA-384	SHA-512	SHA-3
Block size (bits)	512	512	512	1024	1024	1600-2∗bits
Message digest size (bits)	128	160	256	384	512	256
Word size (bits)	32	32	32	64	64	64
Maximum message size	2^{64-1}	2^{64-1}	2^{64-1}	2^{128-1}	2^{128-1}	∞
Rounds	$4 * 16 = 64$	80	64	80	80	24
Collision found	Yes	Theoretical attack	None	None	None	None
Performance (Cycle per byte cpb)	4.99	3.47	7.63	5.12	5.06	8.59
Operations	and, or, xor, rot	and, or, xor, rot	and, or, xor, rot, shr	and, or, xor, rot, shr	and, or, xor, rot, shr	and, xor, not, rot
First published	1992	1995	2001	2012	2012	2015

the RNN at different random times for different groups of inputs. In the next section we will describe this approach and show some example for two types of RNNs.

3. **The Recurrent Neural Network Approach.**

The hashing or content-based signature is generated for every block of data during its subjection to the RNN. The stream of data sourcing from the block is fed to the RNN as binary values based on the ASCII code of the plain text characters. Every 10 integer values, generated from the plain text input, are fed to the RNN. Recurrent Neural Networks (RNNs) have feedbacks from its own outputs to its inputs. The dynamics are temporal and are affected by current and previous inputs [25-28]. The idea here is to incorporate both the sequence of characters and the order of the characters within the inputs of the RNN. Different orders of the same characters will generate different RNN outputs. Therefore, the final state of the outputs of the RNN is not only a function of its current input but also a function of the history of inputs. With this mechanism, the final output of the RNN, which is a real number, will definitely reflect the texture and the structure of the plaintext input block. In that sense the final output vector of the RNN can serve as hashing or content-based signature of the current block of plaintext. The RNN is initialized with random values of weights based on some random seed number. The RNN will end up generating wide range of possible values of signatures generated from the different blocks. The RNN uses a real value activation function which makes it almost impossible for two different blocks to have the same signature.

If a block of length of 1000 characters is used, then 100 generation cycles of the RNN will be executed considering 10 integers (ASCII characters) are fed sequentially to the RNN, 10 characters at a time. At the end of every block, signature (hashing value) of real values with a number equal to the number of neurons of the RNN is generated. The signatures are converted to binary and transmitted with the block to the receiving end. Figure 1 shows the architecture of the NN and the RNN. The outputs of the neurons are real values that are represented with 12 bits each (5-bit base, 6-bit exponents, and 1-bit

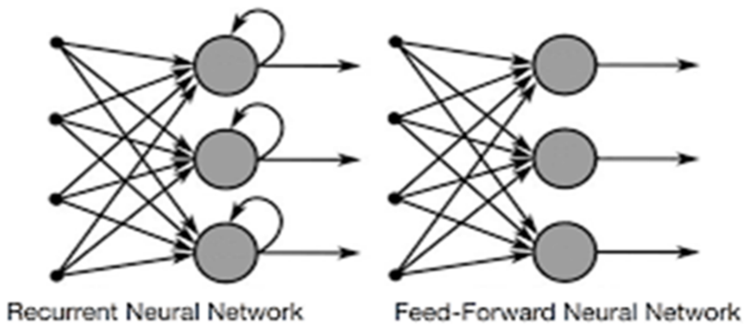

Figure 1. Architectures of neural networks

sign). For N neurons in the RNN, we have $N \times 12$ bits to be transmitted as signatures. On the other hand, for the 10 initial weights (weight for every input) of a single neuron in the RNN we have 120 bits for every neuron. For the N neurons, we have $N \times 120$ bits length key used in generating the hashing value of the block, and this is in addition to the bits of the signatures. A total of $N \times 132$ bits are needed to be packed with the load data (1000 characters) and transmitted.

More than 1000 simulations using 8 neurons were implemented. No collisions were detected. Two cases were demonstrated: one self feedback RNNs and fully connected feedback RNNs. If an RNN reaches a stage of stagnation during hashing generation, the RNN weights are re-initialized and the process of calculating the hashes continues until the block is finished. The real values generated by the neurons at the last input vector of the block are the signature (hash) of that block. Table 4 shows comparable results for the two types of RNNs: RNN1 and RNN2 (single self feedback, and full feedback). The results are almost identical and the performance is comparable to the other standard techniques. The fact that there was one self-feedback or full feedback architecture did have large effect on the results. Figure 2 summarizes the hashing values generation using RNNs. Note that linear activations are used in the RNN to generate real value outputs for the signatures. This provides more versatility and variance in the output values. Below are some examples of the hashing processes: RNN1 is RNN with one self feedback, and RNN2 is RNN with complete feedback to all neurons.

***Text1* example:**

"Forensic science, also known as criminalistics, is the application of science to criminal and civil laws, mainly on the criminal side during criminal investigation, as governed by the legal standards of admissible evidence and criminal procedure. Forensic scientists collect, preserve, and analyze scientific evidence during the course of an investigation. While some forensic scientists travel to the scene of the crime to collect the evidence themselves, others occupy a laboratory role, performing analysis on objects brought to them by other individuals".

Hashing codes: 8 neurons output, each with 12 bits, total is 96 bits, which is 24 possible hex values for every hash. Please see Table 2 below.

Table 2. Samples of the signature values (1)

RNN1	RNN2
'13821A'	'0775E7'
'0188C4'	'06B05F'
'04F057'	'02E54A'
'036136'	'05156E'

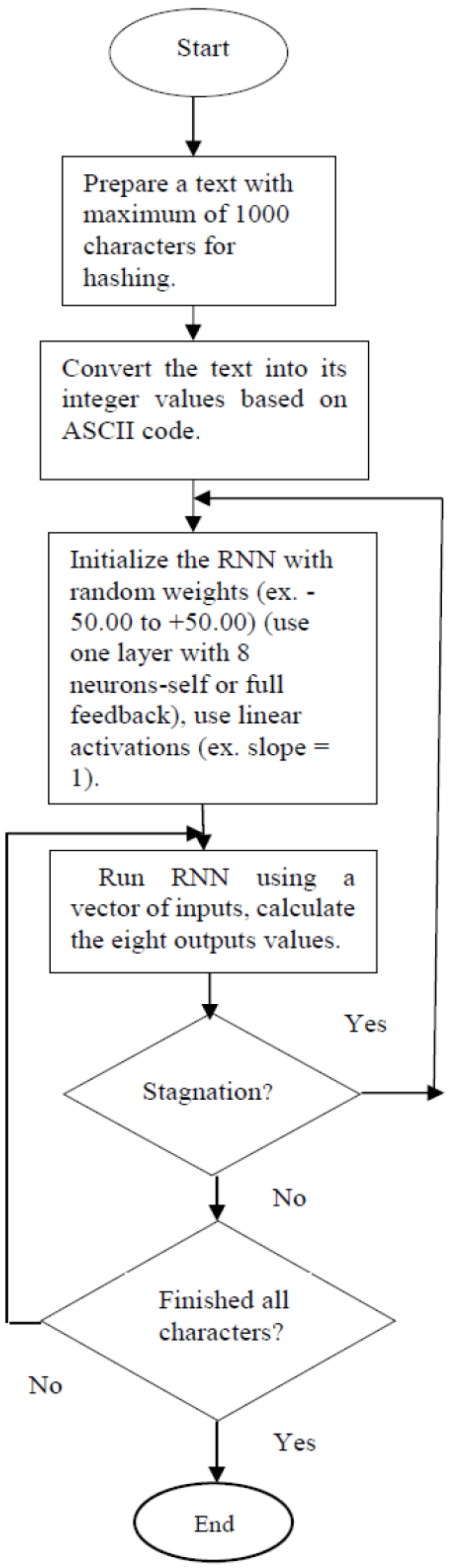
Start
Prepare a text with maximum of 1000 characters for hashing.
Convert the text into its integer values based on ASCII code.
Initialize the RNN with random weights (ex. -50.00 to +50.00) (use one layer with 8 neurons-self or full feedback), use linear activations (ex. slope = 1).
Run RNN using a vector of inputs, calculate the eight outputs values.
Stagnation?
Yes
No
Finished all characters?
No
Yes
End

FIGURE 2. The RNN hashing generation process

***Text2* example:**

"Pharmacy is the science and technique of preparing, dispensing, and reviewing drugs and providing additional clinical services. It is a health profession that links health sciences with pharmaceutical sciences and aims to ensure the safe, effective, and affordable use of drugs. The professional practice is becoming more clinically oriented as most of the drugs

are now manufactured by pharmaceutical industries. Based on the setting, the pharmacy is classified as a community or institutional pharmacy. Providing direct patient care in the community of institutional pharmacies are considered clinical pharmacy".

Hashing codes: 8 neurons output, each with 12 bits, total is 96 bits, which is 24 hex values for every hash. Please see Table 3 below.

Table 3. Samples of the signature values (2)

RNN1	RNN2
'07F07C'	'036331'
'0AC5C1'	'123BE6'
'01B4D6'	'0C0D2B'
'060C00'	'0D1216'

Table 4. The recurrent NN performance

Parameters	RNN1	RNN2
Block size (characters)	1000	1000
Message digest size (characters)	10	10
Word size (characters)	10	10
Collision found	None	None
Performance (Cycle per byte cpb)	9.9	10.8
Operations	and, or, xor, rot, shr	and, or, xor, rot, shr

4. Discussions and Conclusions.

Sufficient literature review has been introduced for different hashing techniques in the previous sections. The RNN approach was introduced with two different architectures: first case was single self-feedback and the other case was full feedback to all neurons. The RNN was used to generate the hashing values with random initializations for the weights. There was no training at all for the RNNs. The RNNs were used only in generating the codes for every block of 1000 characters. A number of eight neurons were used in every case. The fact that we have fast computers nowadays and a complexity not exceeding $O(n^2)$ resulted in execution time that is very acceptable. The contribution of this work is mainly introducing an artificial intelligence based paradigm that can be competitive in the generation of hashing values. The comparable results with other techniques were encouraging to proceed further in investigating this method. The message digest and word sizes were smaller than other techniques; however, that did not affect the performance of the RNN as no collisions were found at all in the simulations. The biggest advantage with using the RNN is that it is impossible to crack the hashing. There are infinite possible initial weights that can accomplish the hashing and those could be updated frequently to increase the security level to the highest. However, those weights need to be encrypted themselves when sending from the transmitter to the receiver. Those weights could be appended to the data and encrypted with it. If the data itself was breached, then the hashing itself will be irrelevant. Of course, this is a very unlikely case especially if an encryption mechanism with appropriate keys was used. If eight weights were used in the RNN with 12 bits representation for each weight, then the hashing key used is of $10 \times 12 \times 8 = 960$ bits length. Add to that the length of the signature reached after finishing the block of 1000 characters which is equal to $8 \times 12 = 96$ bits, we will end up with 1056 bits which need to be appended to the 1000 characters of data. Assuming each character is 8 bits, this implies 13.2% increase in the data block size. Based on the no collision cases we achieved in the 1000 experiments we made, this is a very good result in terms of security and data integrity.

Future work will include further testing and simulations by increasing the block size to more than 1000 characters and monitoring the collision cases. We would like to know to what extent we can increase the block size before any collisions are reached and the same time achieving fast execution for the hashing operations. We would like to apply our hashing on different types data other than the plain text and evaluate its efficiency.

REFERENCES

[1] A. Bhandari, M. Bhuiyan and P. W. C. Prasad, Enhancement of MD5 algorithm for secured web development, *Journal of Software*, vol.12, no.4, pp.240-252, 2017.

[2] S. Saraireh, J. Al-Saraireh and M. Saraireh, Integration of hash-crypto-steganography for efficient security technique, *International Journal of Circuits, Systems and Signals Processing*, vol.12, pp.274-278, 2018.

[3] H.-H. Chang, Y.-C. Chou, C.-C. Tseng and T. K. Shih, A high payload steganography scheme for color images based on BTC and hybrid strategy, *Journal of Computers*, vol.26, no.2, pp.46-55, 2015.

[4] E. P. Singh and E. P. S. Saini, A novel approach to robust and secure image steganography based on hash and encryption, *International Journal of Engineering Sciences and Research Technology*, vol.5, no.3, pp.194-201, 2016.

[5] Y. Sasaki and L. Wang, Improved single-key distinguisher on HMAC-MD5 and key recovery attacks on Sandwich-MAC-MD5, *International Conference on Selected Areas in Cryptography*, pp.493-512, 2014.

[6] X. Lu, Y. Chen and X. Li, Discrete deep hashing with ranking optimization for image retrieval, *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[7] R. Rahim et al., Hashing variable length application for message security communication, *ARPN Journal of Engineering and Applied Sciences*, vol.14, no.1, pp.259-264, 2019.

[8] L. Xiang, X. Shen, J. Qin and W. Hao, Discrete multi-graph hashing for large-scale visual search, *Neural Processing Letters*, vol.49, no.3, pp.1055-1069, 2019.

[9] Z. Tang, F. Yang, L. Huang and M. Wei, DCT and DWT based image hashing for copy detection, *ICIC Express Letters*, vol.7, no.11, pp.2961-2967, 2013.

[10] P. P. Pittalia, A comparative study of hash algorithms in cryptography, *International Journal of Computer Science and Mobile Computing*, vol.8, no.6, pp.147-152, 2019.

[11] N. G. Kumar and K. P. K. Rao, Hash based approach for providing privacy and integrity in cloud data storage using digital signatures, *International Journal of Computer Science and Information Technologies*, vol.5, no.6, pp.8074-8078, 2014.

[12] M. A. Shah, R. Swaminathan and M. Baker, *Privacy-Preserving Audit and Extraction of Digital Contents*, HPL Technical Report No. HPL-2008-32, 2008.

[13] S. Singh, P. Sharma and D. Arora, Data integrity check in cloud computing using hash function, *International Journal of Advanced Research in Computer Science*, vol.8, no.5, pp.1974-1978, 2017.

[14] V. M. Lomte, D. R. Ingle and B. B. Meshram, A secure web application: E-tracking system, *International Journal of UbiComp*, vol.3, no.4, pp.1-18, 2012.

[15] J. Maan, Mobile web – Strategy for enterprise success, *International Journal on Web Service Computing*, vol.3, no.1, pp.45-53, 2012.

[16] S. Rafique, M. Humayun, Z. Gul, A. Abbas and H. Javed, Systematic review of web application security vulnerabilities detection methods, *Journal of Computer and Communications*, vol.3, no.9, pp.28-40, 2015.

[17] M. J. Wang and Y. Z. Li, Hash function with variable output length, *Proc. of the 2015 International Conference on Network and Information Systems for Computers*, 2015.

[18] P. Ora and P. R. Pal, Data security and integrity in cloud computing based on RSA partial homomorphic and MD5 cryptography, *Proc. of the 2015 International Conference on Computer, Communication and Control*, 2015.

[19] Z. Hu and Y. Lu, A method based on MD5 and time for preventing deception in electronic commerce, *Proc. of the International Conference on Cyberspace Technology*, 2014.

[20] G. Di Crescenzo and F. Vakil, Cryptographic hashing for virus localization, *Proc. of the 4th ACM Workshop on Recurring Malcode (WORM)*, 2006.

[21] S. Martinez, S. Gerard and J. Cabot, Robust hashing for models, *The 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS'18)*, 2018.

[22] H. Hayouni, M. Hamdi and T.-H. Kim, A novel efficient approach for protecting integrity of data aggregation in wireless sensor networks, *Proc. of 2015 International Wireless Communications and Mobile Computing Conference*, pp.1193-1198, 2015.

[23] G. K. Sodhi and G. S. Gaba, An efficient hash algorithm to preserve data integrity, *Journal of Engineering Science and Technology*, vol.13, no.3, pp.778-789, 2018.

[24] P. V. Rao, S. G. Rao, P. C. Reddy, G. R. Sakthidharan and Y. M. Kumar, Improve the integrity of data using hashing algorithms, *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol.8, no.7, 2019.

[25] R. Chandra and M. Zhang, Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction, *Neurocomputing*, vol.86, pp.116-123, 2012.

[26] T. Gao, X. Gong, K. Zhang, F. Lin, J. Wang, T. Huang and J. M. Zurada, Recalling-enhanced recurrent neural network: Conjugate gradient learning algorithm and its convergence analysis, *Information Sciences*, vol.519, pp.273-288, 2020.

[27] D. Kreuter, H. Takahashi, Y. Omae, T. Akiduki and Z. Zhang, Classification of human gait acceleration data using convolutional neural networks, *International Journal of Innovative Computing, Information and Control*, vol.16, no.2, pp.609-619, 2020.

[28] R. A. Zitar, Capturing the Brachistochrone: Neural network supervised and reinforcement approaches, *International Journal of Innovative Computing, Information and Control*, vol.15, no.5, pp.1747-1761, 2019.