Hash Function Generation by Neural Network

Michal Turčaník, Martin Javurek

Departament of Informatics
Armed Forces Academy of gen. M. R. Štefánik
Liptovský Mikuláš, Slovak Republic
michal.turcanik@aos.sk, martin.javurek @aos.sk

***Abstract*—Effective generation of hash function is very important for an achievement of a security of today networks. A cryptographic hash function is a transformation that takes an input and returns a fixed-size value, which is called the hash value. An artificial neural network (ANN), as a possible approach, could be used for the hash function generation. The performance of the ANN was validated by software implementation of ANN for given network configuration. The principles of artificial neural networks and the possibility of using artificial neural network for hashing are presented in the article. For analysis of ANN were created testing sets on the base of several examples of general texts in English and in Slovak language. In the paper were tested ANNs with one hidden layer. The number of neurons in the hidden layer of the artificial neural network is optimized based on the results in the simulation.**

Keywords—packet filter; firewall; artificial neural network

I. INTRODUCTION

Security of the human communications is still more and more important along with spreading of the Internet connections over the world. Secure communications is a key component of the internet security. Availability of commercial communication systems cause the potential risk of their misusing from adversary side. There are several scenarios how communication can be misused. One of them is to coordination of adversary attacks against friendly units during operations [1]. Security of communication becomes more and more important to achieve success not only in military operations.

II. SECURITY of COMUNICATION

Human kind evolution is tightly connected with evolution of new technologies. One of the most important technological industries is communication in today world. Communication between people has always proceeded. Without it we can't imagine life and perhaps without it even any existed. A way as the world today walks towards a modern age also communication has moved to a higher level. Growing amount of messages and information spreads by the technical means of the transmission and storage. These messages and information disseminated not by word of mouth, but in the form of ones and zeros, either by wire or wireless, are much more vulnerable to abuse and to be captured. This is a reason why it was started to develop area named information security, which addresses the safe transfer, storage, data processing and manipulation. One area that addresses information security is cryptography.

A hash algorithm is a one way function that converts a data string into a numeric string output of fixed length. The output string is generally much smaller than the original data (Fig. 1). Therefore it is also called message digest or message compression algorithm. Hash algorithms are designed to be collision-resistant, meaning that there is a very low probability that the same string would be created for different data. Two of the most common hash algorithms are the MD5 (Message-Digest algorithm 5) and the SHA-1 (Secure Hash Algorithm). MD5 Message Digest checksums are commonly used to validate data integrity when digital files are transferred or stored.

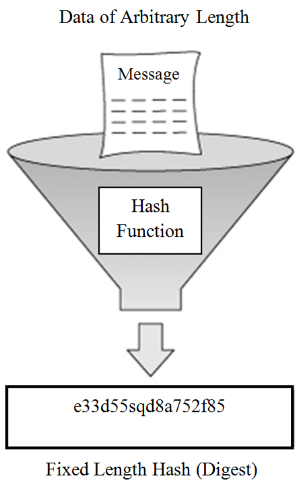

Fig. 1. Hash algorithm.

III. HASH FUNCTION

A function F is a one-way function if its definition is generally known and does not require any secret information for its operation. For any given *x,* it is easy to compute *F(x)*. For given *y*, in the range of F, it is hard to find an *x* such

$$F(x)=y \quad (1)$$

Any efficient algorithm solving a P-problem can succeed in inverting *F* with insignificant probability. The one-way functions existence is not mathematically proven.

A cryptographic hash function is defined as one of the following: a one-way hash function, a collision-free hash function, a trapdoor one-way hash function, or a function from a class of universal hash functions.

One-Way Hash Function is a function *H* that maps an arbitrary length message *M* to a fixed length message digest (*MD*). *MD* is a one-way hash function if it is a one-way function and for given *M* and *H(M)*, it is hard to find a message $M' \neq M$ such that $H(M')=H(M)$.

Collision-Free Hash Function is a function *H* that maps an arbitrary length message *M* to a fixed length message digest (*MD*). The *MD* is a collision-free hash function if it is a one-way hash function. It is hard to find two distinct messages *(M',M)* that hash to the same result $H(M')=H(M)$. Is possible to use efficient algorithm (solving a P-problem) to find such a collision with negligible probability.

Trapdoor One-Way Hash Function is defined

$$F{:}\{0,1\}^{l(n)} \times \{0,1\}^{n} \rightarrow \{0,1\}^{m(n)} \quad (2)$$

If *F* is a trapdoor one-way function and is also a one-way hash function, i.e., if, additionally given *M* and *F(M)*, it is hard to find a message $M' \neq M$ such that $F(M')=F(M)$.

A trapdoor one-way function is a special type of one-way function and it has a secret trapdoor. Computation is easy in one direction and very difficult in the other direction. But, if the secret information is known, it can be easily computed the function in the other direction. It is possible to compute *F(M)* for given *M*, and hard to compute *M* for given *F(M)*. However, there is some secret information, *Y*, such that given *F(M)* and *Y* it is easy to compute *M*.

Length of one-way hash function can influence survivability against an attack. Most practical one-way hash functions produce 128-bit hashes and more.

A slight change in the input string has to cause the hash of the function to change dramatically. Even if 1 bit is changed to 0 in the input string, at least half of the bits of the hash output should flip as a result (avalanche effect). Collision free or collision resistant functions almost do not have same hash value of two different messages. A hash of the document can be used as a cryptographic equivalent of the document. Hash collision happens when different input messages results in the same hash value.

Hashes function can be used for password hashing, file integrity checks and message authentication.

Some users are using one password at multiple sites and getting a stash from one system may give access to others. Storing a hash of the password is a more secure way than the password itself.

Verifying integrity of the file is the most obvious use of hash function. Another application of hash is digital signature.

IV. Artificial Neural Network

Artificial neural networks (ANN) were trained to realize complex functions in different areas (classification, identification, pattern recognition, speech, vision, and automation) [3]. Neural networks are built from simple elementary units (described as a formal neuron) which work in concurrent manner. These units are mimic human nerve systems. Network function is mostly defined by connections (synapses) between these units. Every synapse has defined some weight. Learning of a neural network for specific function is done by setting of the values of the weights between units. Learning of artificial neural networks is realized by presenting a set of inputs and set of target outputs. Adjusting of neural network is based on a comparison of the computed output and the expected target. This process is repeated until the computed outputs of the neural network are matched the expected outputs. Learning set consists of input and expected outputs couples [4].

The most known architecture of neural network is a perceptron. Parameters of the perceptron (weights, biases) are adjusted to generate a proper output vector to the equivalent input vector. A lot of architectures of perceptron were created by Rosenblatt [5]. The simplest one is a single-layer perceptron. The most interesting feature of preceptron is generalization. Application of pattern classification is well suited for perceptron. Trained perceptrons are able to realize suitable problem very quickly and also their reliability is very high. The structure of feed-forward networks can consist of one or several hidden layers of neurons with sigmoid function and one output layer (linear function). Several layers of the neural network which have nonlinear transfer functions enable to train the network for nonlinear dependence between input and output sets. The function fitting can be realized by output layer [6].

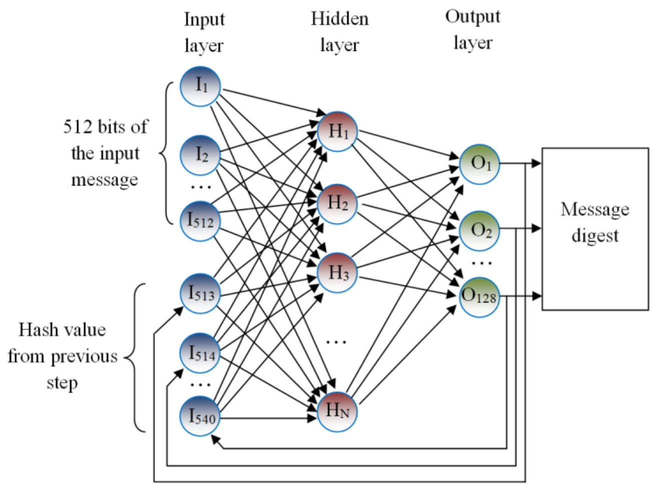

Fig. 2. The structure of the multilayer feed-forward network for hash function.

A multi-layer feed-forward network is composed from neurons and synapses. Synapse is a connection between two neurons. All neurons have an input, activation and output function. Every synapse between two neurons has a value which is called weight. The neurons are divided into layers. There are three different groups of neurons: input neurons, hidden neurons and output neurons. The input neurons accept the input data, and the output neurons generate the output [7].

The final result is calculated by realization of computation which is done layer by layer. At the first, the input values are applied to the input neurons. In the input layer every neuron computes his output function. Computed values are sent to the other layer. The process is finished when the output layer is achieved.

The number of the input layer neurons is 512 and it corresponds to the 512 bits of input message which is hashed by multilayer feed-forward network. The number of the neurons of the hidden layer is a target of the optimization. The number of the neurons of the output layer is 128 which represent the number of bits of the hash function.

The structure of the multilayer feed-forward network for hash function calculation is shown in Fig. 2. The input to the ANN is represented by 512 bits segment of input data from massage. The rest of input neurons are represented by 128 bits from previous hash function calculation and they are used in the next step. The number of hidden layers and the number of neurons in the hidden layers is goal of the optimization. The output of the ANN consists of 128 bits of computed hash function.

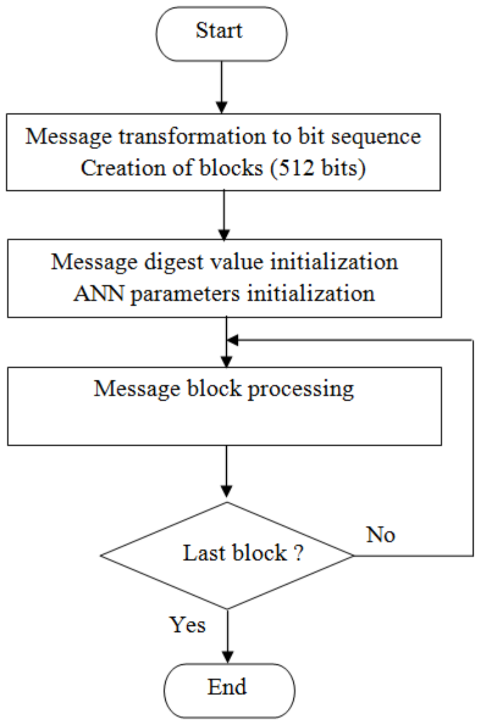
Start
Message transformation to bit sequence
Creation of blocks (512 bits)
Message digest value initialization
ANN parameters initialization
Message block processing
Last block ?
No
Yes
End

Fig. 3. The hash function calculation.

V. Hash Function Calculation Steps

Hash function calculation can be divided into three steps. First step begins with transformation of message into sequence of bits. This sequence of bits is split into 512 bits long blocks. If message has lower number of bits than 512 or after splitting of message a last block is lower than 512 bits padding is added.

The second step starts with generation of initial message digest value which is used as input for hash function calculation (128 bits). This operation is realized only in first step of calculation because we don't have any hash value yet. After realization of first step hash value is used in the next step of calculation together with next 512 bits block of the massage. In Fig. 2 you can see how 128 bits hash value is used as input to the neural network. The artificial neural network parameters are generated in the in the same step. The neural network parameters are values of synapses between network layers and bias values of each neuron. Their number is fixed and it is not changed during using of the neural network.

After that in the third step it is possible to send a block by a block of message to the input of the artificial neural network. Computation of hash function is finished when all blocks of the message is processed. Last value on the output of the ANN is final hash function value. Algorithm for hash function calculation is depicted on Fig. 3.

VI. Selection of ANN for Hash Function Calculation

To optimize neural network structure, an analysis of the number of neurons in hidden layer from point of view of hash function suitable computation [8, 9] must be done. The other parameters of the ANN are not optimized. To find optimal structure of ANN several testing sets and several models of neural network were created.

The number of neurons for input layer and the number of neuron in the output layer of the ANN aren't changing during using of ANN, because the size of input block and also the size of hash function is static. For hash function calculation is the number of neurons in hidden layer variable and it changes in interval from 10 to 130 neurons.

Testing sets were created on the base of several examples of general texts in English and in Slovak language. There are five sets for each language. The number of characters of testing texts is shown in the Table I.

TABLE I. The Number of Characters of Analzyed Texts

Testing text - The number of characters			
English	**The number of characters**	**Slovak**	**The number of characters**
TextEn1	3587	TextSvk1	3912
TextEn2	11,658	TextSvk2	12,104
TextEn3	19,782	TextSvk3	20,025
TextEn4	32,451	TextSvk4	33,147
TextEn5	61,456	TextSvk5	61,259

On the base of these testing sets in English and Slovak language it will be evaluated a quality of hash function generation by a given ANN. In the next step the random change of characters in the analyzed texts was realized (5 and 10 characters were changed). The comparison of given ANNs is done on the base of number of changed characters after generation of hash function for original and modified text. The higher values of changed characters represent better results for a given network. The results for analyzed ANNs are shown in the Table II and Table III.

Parameters of all analyzed networks were generated randomly for specified interval of neurons of hidden layer (from 10 to 130). Parameters of ANN are weights of synapses and biases for whole ANN. No learning algorithm is used to train ANN so no training and validation sets are needed.

TABLE II. COMPARISON of ANN's for 5 CHANGED CHARACTERS in ANALYZED ENGLIS TEXTS

Number of neurons	**Number of changed characters in the hash value**	
	5 changed characters in the text	**10 changed characters in the text**
10	6	7
20	9	9
30	11	10
40	11	11
50	10	9
60	12	11
70	9	8
80	11	10
90	12	13
100	11	12
110	10	9
120	12	12
130	11	12

In the Table II are summarized results of using of ANN for hash function generation for english text. The value in the table represents average value for all English texts which number of characters is shown in the Table I. The value can be from interval 0 to 32 which is maximal number of hexadecimal numbers of hash value computed by ANN. The value is rounded to integer value without any decimal part.

The minimal number of changed characters in the analyzed texts in English language is 6 characters for ANN with 10 neurons in the hidden layer and the maximal number of changed characters in the analyzed texts in English language is 13 characters for ANN with 90 neurons in the hidden layer. Character change corresponds to the hash value change from 18 to 37 %.

TABLE III. COMPARISON of ANN's for 5 CHANGED CHARACTERS in ANALYZED SLOVAK TEXTS

Number of neurons	**Number of changed characters in the hash value**	
	5 changed characters in the text	**10 changed characters in the text**
10	7	7
20	8	7
30	9	10
40	8	9
50	9	11
60	11	11
70	10	11
80	9	11
90	12	12
100	11	12
110	12	12
120	11	12
130	13	12

In the Table III are summarized results of using of ANN for hash function generation for Slovak text. The value in the table represents average value for all Slovak texts which number of characters is shown in the Table I. The value can be from interval 0 to 32 which is maximal number of hexadecimal numbers of hash value computed by ANN. The value is rounded to integer value without any decimal part.

The minimal number of changed characters in the analyzed texts in Slovak language is 7 characters for ANN with 10 neurons in the hidden layer and the maximal number of changed characters in the analyzed texts in English language is 13 characters for ANN with 130 neurons in the hidden layer. Character change corresponds to the hash value change from 21 to 40 %.

VII. CONCLUSION

Application of neural networks in real world is sometimes very difficult thanks their structural complexity. The structural complexity ANN comes from network structure and therefore optimisation of the ANN structure from point of actual application is very important and arising problem to solve. Because structure and model of neural network is dependent on real application the analysis of neural network can be done from point of view of application criteria.

A new approach for hash function generation using neural networks was presented in this paper. The main goal was to build ANN for hash function generation which could be easily used in the modern communication systems. The core of the system was presented but complete system would need some additional modules for using as a real application.

The ANN topology and optimization criterions were designed to solve a problem of hash function generation. The main advantage of using ANN for hash function generation is security. If we use same secret parameters of ANN (synapses and bias values) for sender and for receiver it is very difficult or almost impossible to reconstruct them from transferred messages. Very important is also speed of hash function generation because proposed solution can be realized in parallel manner. Behaviour of ANN also could be easily changed by random generation of new network parameters.

Based on the results of optimization of ANN for hash function generation given proposal has no dependence on language, English or Slovak. Average value of change of hash function for both languages is about 30 % with change of original text for 5 and 10 characters.

REFERENCES

[1] R. Berešík, M. Soták, F. Nebus, and J. Puttera, "Satellite communication system´s detection," Przegląd elektrotechniczny, ISSN 0033-2097, R. 87, Nr. 7, 2011, pp. 249-254.

[2] I. Kenneth, and S. Forrest, "A History and Survey of Network Firewalls," Technical report of University of New Mexico, 2002.

[3] S. Ezzati, H. Naji, A. Chegini, and P. Habibimehr, "Intelligent Firewall on Reconfigurable Hardware," European Journal of Scientific Research, ISSN 1450-216X, Vol.47, No.4, 2010, pp.509-516.

[4] J. L. Elman, "Finding Structure in Time," Cognitive Science, Vol. 14, 1990, pp. 179-211.

[5] F. Rosenblatt, Principles of Neurodynamics, Washington D.C., Spartan Press, 1961.

[6] D. Pattreson, Artificial Neural networks-Theory and Applications, Prentice Hall, 1996.

[7] S. Melacci, L. Sarti, M. Maggini, and M. Bianchini, "A Neural Network Approach to Similarity Learning," Lecture Notes in Computer Science, Artificial Neural Networks in Pattern Recognition, June 30, 2008, pp. 133-136.

[8] I. Mokriš, and M. Turčaník,"Contribution to the analysis of multilayer perceptrons for pattern recognition," Neural Network World, Vol. 10, No. 6, 2000, pp. 969-982.

[9] M. Turčaník, and I. Mokriš,"Possible Approach to Optimization of Neural Network Structure," Proc. of Conf. SECON 97, Warsaw, 1997, pp. 326 - 335.