Pavel URBANOVICH[1,2], Marcin PLONKOWSKI[2], Konstantsin CHURIKOV[1]

Belarusian State Technological University, Minsk, Belarus (1), Catholic University of Lublin, Lublin, Poland (2)

The appearance of conflict when using the chaos function for calculating the hash code

Abstract. *The paper considers the hash function built on artificial neural network (ANN). The data about the process of synchronization of an ANN, obtained by experiment, are presented. The fact, that the obtained vector of weight coefficients for the networks after the synchronization is different for each new session, is determined.*

Streszczenie. *W artykule opisane zostały funkcje skrótu zbudowane w oparciu o architekturę sztucznej sieci neuronowej (SSN). Przedstawiono wyniki eksperymentalne procesu synchronizacji z SSN. Udowodniono, że uzyskane współczynniki wektora wag sieci po synchronizacji są inne dla każdej nowej sesji. (**Wygląd konfliktu podczas wykorzystywania funkcji chaosu do obliczeń skrótowych**).*

Keyword: neural network, hash function.
Słowa kluczowe: sieci neuronowe, funkcja skrótu.

Introduction

The technology of artificial neural networks (ANN) usage for modeling cryptographic communication systems is a new, promising direction in the field of information security.

There are three basic learning algorithms of ANN: with a teacher, without a teacher, with reinforcements. ANN interaction model of "learning without a teacher" has properties such as peer education, self-study, and stochastic behavior, low sensitivity to noise, inaccuracies (data corruption, weight coefficients, errors in the program). This can be used for solving of crypto conversion tasks in systems with a public key, key distribution, messages hashing and generating random numbers. Neural network architecture for messages hashing (convolution) is proposed in [1]. This idea is developed by using algebra of complex numbers in [2]. At the same time neural network model suggests applying dependencies known from chaos theory as a transition function.

This article presents the new experimental data of synchronization process investigating of neural networks and calculating of hash function based on them.

Architecture and the neural network model for message hashing

Artificial neural network allows to combine, disperse, and compress the input sequence of bits. This allows to generate the hash function based on the use of neural networks technology. The network, operating according to the hash function criteria, consists of multiple of neurons, interconnected to layers arbitrarily, thus realizing the conversion and compression of input data. It is also necessary to note, that in contrast to neural networks, that perform information processing, network, that serves as a hash function, may have one-way transition function, for which there is no need to know the opposite value. Consequently, different hashing variants are provided by connection coefficients between neurons, type of transition function, and architecture of the network.

Conventionally, the network can be divided into three layers: input, hidden and output. In the input layer X, the sequence of bit is projected onto the set of complex numbers. Hidden layer Z realizes a hash function itself. The output of each neuron of the layer Z is the transition function of weighted sum of its inputs. The output layer Y converts the values of the hidden layer Z to bit sequence (Fig. 1).

The architecture under consideration consists of one output neuron, K hidden neurons and K × N input neurons (Fig. 1).

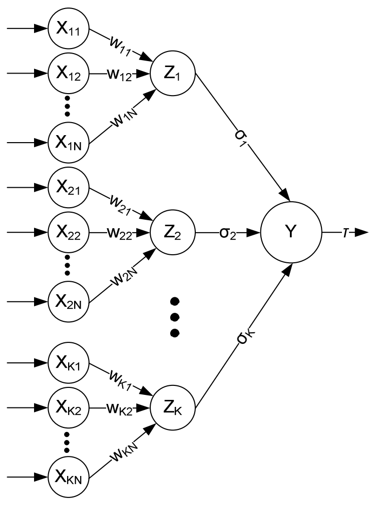


Fig.1. A neural network that performs hashing

Input neurons take binary values:

(1) $$x_{ij} \in \{-1;+1\}\,.$$

The weights (weight coefficients) affecting the input neurons value (between input and hidden neurons) are the following:

(2) $$w_{ij} \in \{-L,...,0,...,+L\}\,.$$

The value of each hidden neuron is the sum of the input value and weight coefficient products:

(3) $$\sigma_i = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} x_{ij}\right), \qquad \operatorname{sgn}(x) = \begin{cases} -1 & if \quad x \le 0, \\ 1 & if \quad x > 0. \end{cases}$$

The value of the output neuron is the product of all hidden neurons:

(4) $$\tau = \prod_{i=1}^{K} \sigma_i\,.$$

The output value is also a binary value.

In Figure 1 x_{ij} are values of the input neurons, respectively (1) obtained from the input values vector X_{KN}, where $i \in [1;K]$, $j \in [1,N]$, w_{ij} are the values of weight coefficients (2), which are set at random order at the first initialization of the neural network.

The algorithm of information exchange based on TPM, TPCM.

We consider two networks (A and B), each of which is based on the architecture of TPM, TPCM (Fig. 1). Networks exchange output values via open channels. Networks synchronization occurs in accordance with the following algorithm.

1. We set the random values of weight coefficients $w_{11}, w_{12}, \ldots, w_{KN}$.
2. We perform the following steps, until it is synchronized:
   2.1. We generate a random input vector $X(x_{11}, \ldots, x_{1N}, \ldots, x_{KN})$.
   2.2. We calculate the values of hidden neurons according to the formula (3).
   2.3. We calculate the value of the output neuron according to the formula (4).
3. We compare the outputs of two TPM (TPCM):
   3.1. If the outputs are different, then we make a transition to item 2.1.
   3.2. If the outputs are identical, then we apply the selected rule to weight coefficients.

After full synchronization (the weights w_{ij} of both TPM are identical) the users (network) A and B can use weights as the key.

During the neural networks operation it was noted that after full synchronization each following step results in synchronous change of the weights vectors. This property can be used as a recursive function (5), which provides irreversible property of hash function based on the neural network.

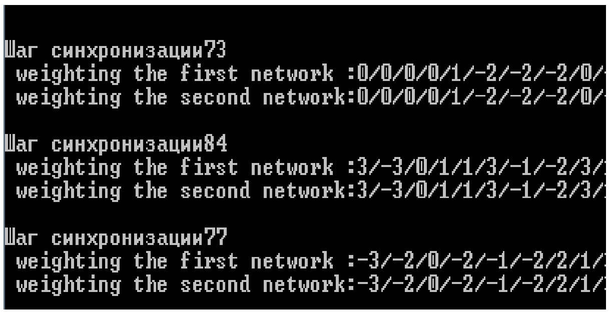
(3) $$w_{ij} = f(w_{ij} - 1, x_i),$$

where $w_{ij} - 1$ is a previous value of the weight coefficient.

This means that for every new input vector-message the convolution function with a new key, which provides a variety of hash code values, even when the input receives the same blocks of vector-message will be used.

Figure 2 shows the information about multiple synchronization sessions, namely:

- step after which there was a complete networks synchronization
- vector of weight coefficients obtained after full networks synchronization.



```
Шаг синхронизации73
 weighting the first network :0/0/0/0/1/-2/-2/-2/0/
 weighting the second network:0/0/0/0/1/-2/-2/-2/0/

Шаг синхронизации84
 weighting the first network :3/-3/0/1/1/3/-1/-2/3/
 weighting the second network:3/-3/0/1/1/3/-1/-2/3/

Шаг синхронизации77
 weighting the first network :-3/-2/0/-2/-1/-2/2/1/
 weighting the second network:-3/-2/0/-2/-1/-2/2/1/
```

Fig.2. Steps of neural networks synchronization

Figure 2 shows, that with each new process of neural networks synchronization, a step, after which comes the synchronization, is different. The figure also shows that at each new synchronization the values of weight coefficients $w_{11}, \ldots, w_{ij}$ are different from the same values obtained at different synchronization session. Since the values of weight coefficients would be used as a key for the convolution function, we can say that for the same message M hash code value H(M) at various sessions will be different. This property makes the system resistant to collisions.

Let us suppose that there is a message M for which it is necessary to obtain the hash code. H1 and H2 are the convolution functions for the first and the second synchronization sessions of neural networks respectively, As it was mentioned above, the vectors of weight coefficients that are used as key for the convolution function obtained in the first and second synchronization processes are different, then hash code value after convolution of the message M will be as different, i.e., H1 (M) ≠ H2 (M).

Operation of such a neural network can be compared with the scheme "one-time pad." As in the above mentioned scheme, the vector of weight coefficients is applied once and then is discarded. But here the message of the same length as the hashable message is not applied, because the message will be hashed block by block, and the new key for encrypting the next block will be calculated using a recursive function.

According to the description given above this hashing system can be classified as the resistant system [3] because:

1. Key is generated for each message (each key is used once);
2. The key is statistically reliable (i.e., the probability of each possible symbols occurrence are equal, the symbols in the key sequence are independent and random)
3. Key length is equal to or greater than the message length (in our case, the message block)

Network analysis was performed with the parameters $N = 8$, $K = 33$, where K is the number of hidden layer perceptrons, and N is the number of inputs which have one perceptron.

Conclusions

The fact that after full synchronization of two neural networks using the method of peer learning, each step in the direction of synchronization would lead to the same change in the values of the weights vectors is defined. This feature and one-sidedness property of neural networks can be used for generating the hash code, if the vector-message is submitted to the inputs of the neural network.

In such a way when calculating the hash code, vector-message, submitted to the input, as well as in neural networks learning, can be used for changing the values of weights vectors that are the key for the messages convolution. Thus, for each new message convolution function will have a new key.

REFERENCES

[1] Kinzel W., Kanter I., Interacting neural networks and cryptography, *Advances in Solid State Physics*, 42 (2005), 383-392

[2] Płonkowski M., Analiza funkcji chaosu w funkcjach skrótu opartych na sieciach neuronowych, *Przegląd Elektrotechniczny*, 84 (2008), n.3, 102-104

[3] Shannon C., The works on information theory and cybernetics, M., *Inostr. Literatura*, (1963), 333-369 (in Russian)

Authors: *prof. P. Urbanovich, dr M. Plonkowski, Katholic University of Lublin (Poland), E-mail: plomyk@kul.lublin.pl;*
graduate K. Churikov, Belarusian University of Technology, E-mail: upp@rambler.ru