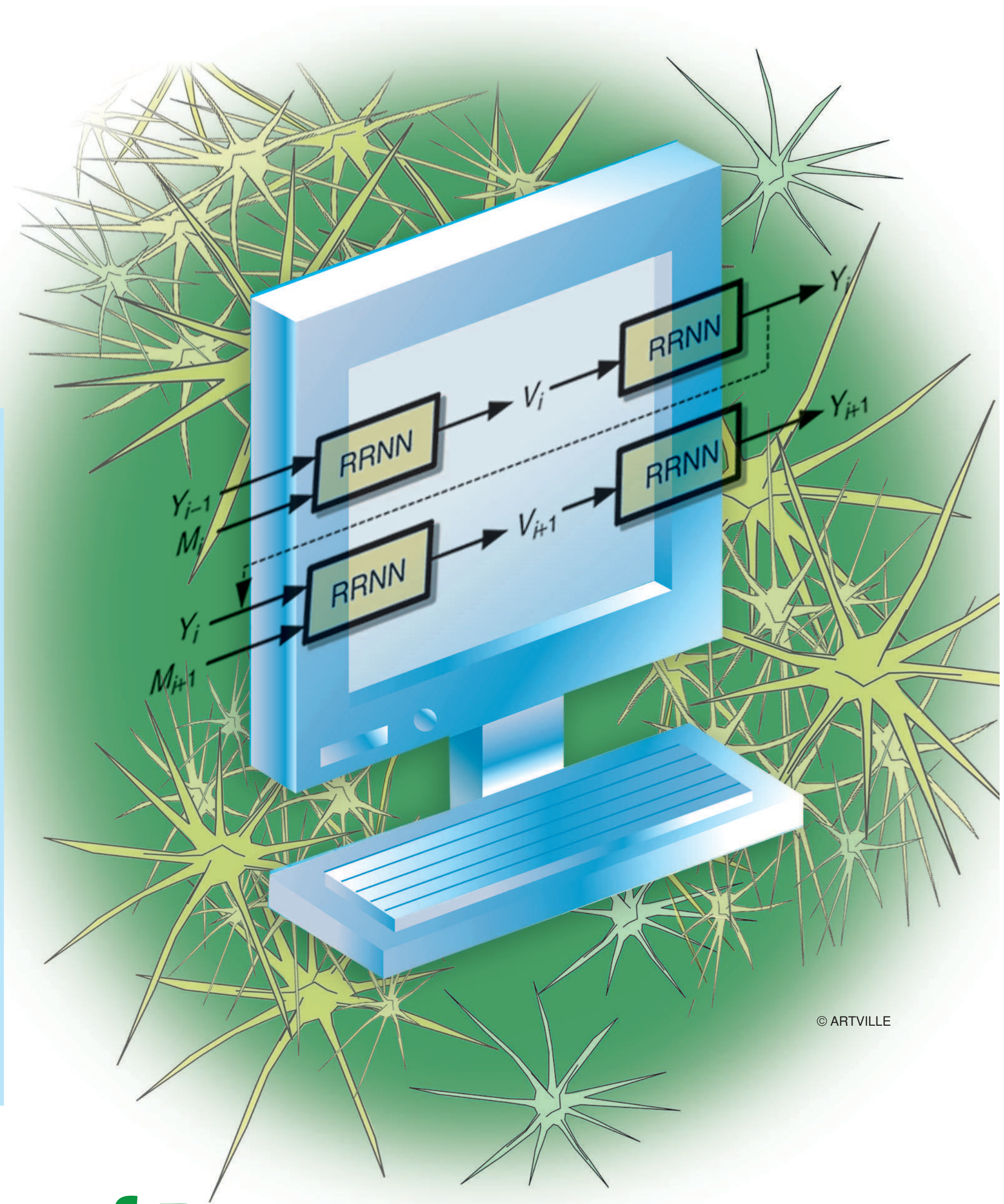
M. Arvandi, S. Wu, and A. Sadeghian
Ryerson University, CANADA



Abstract: In this article, we describe an innovative form of cipher design based on the use of recurrent neural networks. The well-known characteristics of neural networks, such as parallel distributed structure, high computational power, ability to learn and represent knowledge as a black box, are successfully applied to cryptography. The proposed cipher has a relatively simple architecture and, by incorporating neural networks, it releases the constraint on the length of the secret key. The design of the symmetric cipher is described in detail and its security is analyzed. The cipher is robust in resisting different cryptanalysis attacks and provides efficient data integrity and authentication services. Simulation results are presented to validate the effectiveness of the proposed cipher design.

On the Use of Recurrent Neural Networks to Design Symmetric Ciphers

1. Introduction

Information security is generally accepted to be essential to modern business and technology, both for privacy of transactions and communications, as well as for defense against malicious intruders. Cryptography is the study of information security and the feasibility of communication over an insecure channel while preserving the secrecy of the information transmitted [1], [2]. Cryptographic techniques should offer at least the following three security features concerning data transmission: confidentiality, authentication and integrity. Confidentiality is fundamental—third parties are expected to see the encrypted data but should not be able to decipher it. Authentication methods allow the receiver to verify that the sender is legitimate. Lack of authentication makes systems vulnerable to fraudulent transactions and denial of service attacks. Integrity of the transmitted data must be verifiable, i.e., the receiver should be able to check that no part of the message

Digital Object Identifier 10.1109/MCI.2008.919075

1556-603X/08/$25.00©2008IEEE

was lost or altered during transmission. As the sophistication of cryptanalytic attacks increases and their cost decreases, there is constant pressure to improve cryptographic methods on all three of these fronts [3], [4].

In this research, we intend to investigate the use of neural networks in data encryption/decryption. Artificial neural networks have been motivated from their inception by the recognition that the brain computes in an entirely different way from the conventional digital computer. The brain contains billions of neurons with massive interconnections. Similarly, neural networks are massively parallel-distributed processors that are made up of artificial neurons with interconnections. These are nonlinear dynamic machines which expand the expression of input data as a linear combination of inputs to synapses and then perform a nonlinear transformation to compute the output [5].

In this article, we investigate new directions in the design of cryptographic systems by means of neural networks. We propose a symmetric cipher design using recurrent neural networks [6]–[9]. A neural network (NN) based approach offers an attractive solution to this problem in that it provides a suitable framework within which data can be readily coded. We propose that the computational power of NNs, deriving from their dynamic and parallel distributed structure [10], their inherent ability to learn [11], their power to generalize and approximate [12], [13], and their knowledge representation potentials [14], and their black box attribute [15], can be advantageous when used for cryptography purposes. The cipher is implemented in two steps: (i) a neural network backpropagation learning procedure is used for key extension. The secret key is presented to the network as the training set and the network synaptic weights are initialized according to the secret key through a supervised learning procedure. The resulting network with the secret initial state is the extended secret key to be later used for data encryption, and (ii) the learning procedure is controlled by neural network parameters to generate the chaotic cipher text and synchronize the weight of states during bulk data encryption/decryption. The feedforward data manipulation process through multiple layers of the neural networks has functional resemblance to multiple round operations in a symmetric cipher, i.e., linear permutations, followed by nonlinear transformations. In the case of neural networks, without the knowledge of the weight matrix of the network, the analysis of the data only based on the output would be very difficult. If the learning procedure can be controlled, it is possible to change the neural network structure to be an infinite state machine. As a result, the pattern is revealed but never perfectly repeated and the neural system becomes a dynamic system generating complex variations.

We also discuss security considerations for the proposed symmetric cipher design and possible cryptanalysis attacks against it. In this regard, a self-adaptive procedure, i.e., to update the network learning rate, is introduced to maintain a satisfactory level of security for the symmetric cipher. This procedure will require very little tuning of the cipher parameters and can help to accommodate different applications requirements.

As the sophistication of cryptanalytic attacks increases and their cost decreases, there is constant pressure to improve cryptographic methods.

In what follows, a brief review of previous research pertinent to use of neural networks in cryptography is provided in Section 2. In Section 3, the cipher design methodology is proposed. Section 4 discusses, in detail, the security analysis of the cipher. Section 5 presents simulation results. Finally, in Section 6, the findings and conclusion of this work are presented.

2. Related Literature

A thorough literature survey indicates that there has been an increasing interest in the application of different classes of neural networks to problems related to cryptography in the past few years [16]. The relationship between cryptography and machine learning in general and neural networks in particular has been studied in [17], [18]. Recent works have examined the use of neural networks in different layers of cryptosystems. Typical examples include key management, generation and exchange protocols; design of pseudo random generators; prime factorization; hash functions; symmetric ciphers; authentication; and authorization. A considerable number of studies have presented the innovative use of the Tree Parity Machine (TPM) in cryptography [19]–[25]. Kinzel and Kanter proposed and analytically studied a neural cryptography scheme which was based on the TPM and the mutual learning process between two parity feed-forward neural networks with discrete and continuous weights. The synchronization process is non-self-averaging and the analytical solution is based on random auxiliary variables. The main advantage of their approach is the generation of symmetric key exchange over a public channel using chaos synchronization of Tree Parity Machines [26], [27]. The learning time of an attacker that is trying to imitate one of the networks has been examined analytically and is reported to be much longer than the synchronization time. Kilmov et al. [28] have shown that Kinzel's protocol can be broken by geometric, probabilistic, and genetic attacks, and as such is not entirely secure. J. Zhou et al. [29] have also demonstrated that this protocol is not completely secure under regular and majority flipping attacks, and have proposed a scheme to improve the security against flipping attacks by splitting the mutual information and the training process. Mislovaty et al. [30] reported a new attack strategy involving a large number of cooperating attackers that succeeds in revealing the encryption key. Attempts to restore the security against cooperating attacks are presented in [31].

A number of research works have sought a link between chaotic neural networks (CNN) and increased security in cryptosystems. Su et al. [32] proposed to use unpredictable outputs of CNN together with dedicated hardware to encrypt digital signals. The randomness of the output of the system, built using a specific VLSI architecture, determines whether the encrypted data is predictable or not. Xiao and Liao [33] have reported a combined hash and encryption scheme by CNN where the weights of the neural network are distributed with random chaotic sequences. Li et al. [34] have evaluated the security of the CNN-based encryption scheme, and have shown that it is not secure against known/chosen-plaintext attacks. It is also argued that security against the brute-force attack was over-estimated, and methods to improve the encryption scheme have been suggested. Bose [35] proposed a public key encryption using multiple chaotic systems and a set of linear functions for key exchange over an insecure channel. Wang et al. [36] have shown the weaknesses of such cryptosystems where given the public key and the initial vector, the secret key can be calculated based on the Parsevala theorem. Another interesting approach is the use of clipped Hopfield neural networks (CHNN). Zhou et al. [37] have discussed the merits of the application of CHNN to cryptography [38].

Yee and De Silva [39] proposed another method for block cipher design in which a NN is used for key scheduling. The network is trained to store the information about the secret key and then use it to generate keys for the multiple rounds of data encryption. Meletiou et al. have studied a neural network approach in the RSA cryptosystem [40]. They have also investigated the use of computational intelligence techniques in cryptography [41]. Karras and Zorkadis have used neural networks for secure management of communication systems by improving pseudorandom stream generators [42]. Table I presents a chronological listing of research related to the use of neural networks in cryptography [43]–[49]. We will work on a novel cipher design and a key extension scheme with no limitation on the length of the key, using hardware-independent, real-time recurrent neural networks. This is in contrast to previous works, where the application of hardware specific chaotic neural networks, the use of NNs for key scheduling with fixed key length, and the use of synchronizing feedforward NNs and TPM for encryption is suggested.

3. Neural Network-Based Symmetric Cipher Design—Methodology

This section describes the proposed application of neural networks methodology in cryptography. These include the proposed structure, the key extension approach, encryption, ciphertext generation, on-epoch training, decryption, and related design.

3.1 Proposed Structure

The proposed symmetric cipher is designed based on real-time recurrent neural networks (RRNN) as shown in Fig. 1. The RRNN has a multilayer structure where the dimension of the input vector X is twice of that of the output vector Y, and one of the hidden layers has only one neuron with an output denoted by ξ. The symmetric cipher operates in two stages: key extension and data encryption/decryption. The proposed network is required to be of robust size with the capability to learn the training data fairly well and still to maintain an accurate generalization capability for the tested data. A robust size is identified by pruning [50], [51] the weights of the neural network. That is, at the beginning of the learning process a large-size neural network is used then the irrelevant weights and

TABLE 1 Chronological list of research related to use of neural networks in cryptography.

1991	• A STUDY OF THE RELATIONSHIP BETWEEN CRYPTOGRAPHY AND MACHINE LEARNING [17]
1994	• A STUDY OF NEURAL NETWORKS AND THEIR CRYPTOGRAPHIC APPLICATIONS [18]
1999	• DESIGN OF SYMMETRIC ENCRYPTION SCHEME BASED ON CHAOTIC ATTRACTORS OF NEURAL NETWORKS [48]
2000	• DESIGN CHAOTIC NEURAL NETWORKS WITH DEDICATED HARDWARE TO ENCRYPT/DECRYPT SIGNALS [32]
	• APPLICATION OF MULTILAYER PERCEPTRON NEURAL NETWORKS FOR PUBLIC KEY SCHEDULING [39]
2001	• APPLICATION OF CLIPPED HOPFIELD NEURAL NETWORKS IN THE DESIGN OF KEYSTREAM GENERATOR [38]
2002	• A STUDY OF NEURAL NETWORK METHODOLOGY IN THE RSA CRYPTOSYSTEM [40]
	• DESIGN OF A CRYPTOGRAPHY SCHEME BASED ON MUTUAL LEARNING PROCESS BETWEEN TWO PARITY FEED-FORWARD NEURAL NETWORKS [19]
	• REPORT ON VULNERABILITY OF CRYPTOGRAPHY BASED ON MUTUAL LEARNING PROCESS [28]
	• DESIGN OF ONE-WAY HASH FUNCTION USING MULTILAYER PERCEPTRON NEURAL NETWORKS [44]
	• DESIGN OF SYMMETRIC BLOCK CIPHERS USING MULTILAYER PERCEPTRON NETWORKS [45]
2003	• USE OF COMPUTATIONAL INTELLIGENCE TECHNIQUES IN CRYPTOGRAPHY [41]
	• IMPROVEMENT OF PSEUDORANDOM STREAM GENERATORS USING NEURAL NETWORKS [42]
	• DESIGN OF PSEUDORANDOM BIT SEQUENCE GENERATORS AND EVALUATION FOR SECURE INTERNET COMMUNICATIONS USING NEURAL NETWORK TECHNIQUES [47]
2004	• DESIGN OF COMBINED HASH AND ENCRYPTION SCHEME USING CHAOTIC NEURAL NETWORKS [33]
	• REPORT ON VULNERABILITY OF CHAOTIC NEURAL NETWORK BASED ENCRYPTION AGAINST KNOWN/CHOSEN-PLAINTEXT AND THE BRUTE-FORCE ATTACKS [34]
	• DESIGN OF A SYMMETRIC CRYPTOGRAPHY APPROACH BASED ON CLIPPED HOPFIELD NEURAL NETWORKS [37]
2005	• DESIGN OF A PUBLIC KEY ENCRYPTION TECHNIQUE BASED ON MULTIPLE CHAOTIC SYSTEMS [35]
	• REPORT ON WEAKNESS OF PUBLIC KEY ENCRYPTION TECHNIQUE BASED ON MULTIPLE CHAOTIC SYSTEMS [36]
	• APPLICATION OF NEURAL NETWORKS TO THE INTEGER PRIME-FACTORIZATION PROBLEM [46]
	• REPORT ON CHOSEN-PLAINTEXT CRYPTANALYSIS OF A CLIPPED HOPFIELD NEURAL-NETWORK-BASED CHAOTIC CIPHER [49]
	• REPORT ON IMPROVING THE ABILITY OF PASSIVE ATTACKS OF CHAOTIC ENCRYPTION BY USING NEURAL NETWORK [43]
2006	• DESIGN OF A SYMMETRIC CIPHER USING RECURRENT NEURAL NETWORKS [8]
2007	• REPORT ON VULNERABILITY OF NEURAL NETWORK-BASED SYMMETRIC CIPHER AGAINST CHOSEN PLAINTEXT ATTACK [56]

nodes of the network are removed in an iterative process until a smaller size network is derived.

This simple architecture satisfies the confusion and diffusion properties of the cipher that are two basic techniques for obscuring the redundancies in a plaintext message. From the input layer up to the hidden layer II, the layer with only one neuron ξ, confusion is achieved, which is similar to the effect of substitution. Then by applying the simple non-linear function (sigmoid function in this case) to the inputs, diffusion is performed, which is similar to transposition. The simplicity of this architecture facilitates its analysis.

3.2 Key Extension

Suppose there are two users at different ends of a communication line with an identical symmetric cipher based on a neural network similar to that in Fig. 1. They will exchange a secret key S that contains the following three parts of information: (i) the input vector X; (ii) the training target Y; and (iii) the critical value of the self-adaptive procedure α. Vectors X and Y will then be presented to the neural network for training. The purpose of the training process is to make the neural network detect, store or remember the secret key information. The trained neural network parameters will be kept unrevealed and become the extended secret key for subsequent encryption and decryption procedures. The last actual output of the network during the key extension will be the initial vector, M_0, used for the encryption (Fig. 2). It is commonly assumed that the weight distribution of the hidden layers is chaotic and unpredictable without the knowledge of the training data (i.e., the original secret key). Therefore, it is not feasible for a cryptanalyst to analyze the extended key. By changing the length of the secret key and the dimension or the hierarchy of the hidden layers, the user can adjust the security level accordingly. A major advantage of the proposed cipher design is its capability to release the constraints imposed on the length of the secret key.

3.3 Encryption

The structure of the symmetric cipher design (Fig. 1) ensures that among the hidden layers of the neural network, there exists at least one that has only one neuron (denoted as neuron ξ). This feature is used to decompose the feedforward operation of the neural network into two functions F_1 and F_2. In this decomposition, F_1 is the feedforward operation over the weight and bias matrices performed from the input layer to neuron ξ, and F_2 is the similar type of operation performed from neuron ξ to the output layer. These functions are then used in the encryption process that consists of two steps: (i) ciphertext generation; and (ii) iterative one-epoch training of neural network that already has the secret key information.

3.4 Ciphertext Generation

The plain text should first be mapped to vectors $M_{i(i=1,\ldots,n)} = \{M_1, M_2, M_3, \ldots M_n\}$ according to the dimension of input vectors. The first vector of the message is combined with the initial vector M_0 from the key extension procedure to build the following initial input vector

$$X_1 = (M_0 || M_1)$$

where $||$ denotes a vector concatenation operator, i.e., two $(n \times 1)$ vectors M_0 and M_1 are concatenated to form a $(2n \times 1)$ vector. Next, X_1 is presented to the neural network to produce both the intermediary neuron output V_1 in the hidden layer and the output Y_1 (Fig. 3).

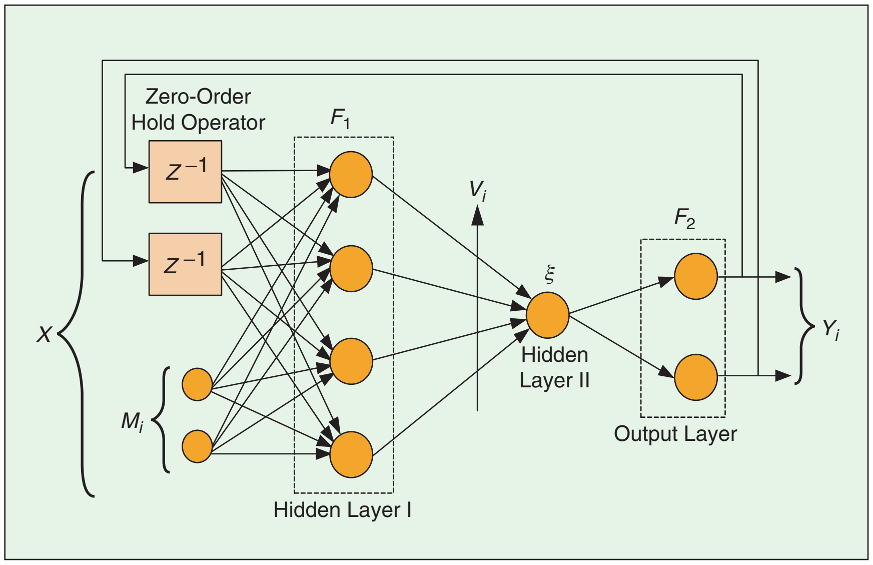


FIGURE 1 Proposed recurrent neural network for cipher design.

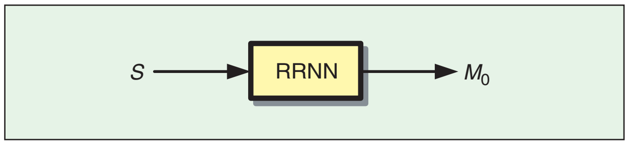


FIGURE 2 Key extension process.

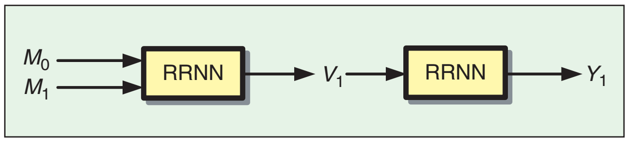


FIGURE 3 First stage of ciphertext generation.

A neural network based approach offers an attractive solution to this problem in that it provides a suitable framework within which data can be readily coded.

The error signal is calculated as $E_1 = M_1 - Y_1$, where M_1 is the target of the identity mapping. Finally, E_1 and V_1 are considered as the first block of the ciphertext referred to as $C_1\{V_1, E_1\}$.

3.5 One Epoch Training

After the construction of the first ciphertext blocks C_1, the existing neural network with a priori knowledge of the secret key is retrained using X_1 as the input vector and M_1 as the training target. Based on the proposed recurrent neural network structure (Fig. 1), for other ciphertext blocks, $C_i(i = 2, \ldots, n)$, the input vectors are built according to:

$$X_i = (Y_{i-1}||M_i), \qquad i = 2, \ldots, n$$

The above two steps of encryption are repeated to generate values for V_i and Y_i, and process the neural network for one-epoch at a time (Fig. 4). In fact, the above encryption procedure will result in a symmetric cipher working in the cipher block-chaining (CBC) mode as implicitly shown in Fig. 5.

To summarize the above procedure, the ciphertext blocks C_i are constructed as follows:

$$V_i = F_1(X_i) \qquad (1)$$
$$Y_i = F_2(V_i) \qquad (2)$$
$$E_i = M_i - Y_i \qquad (3)$$
$$ST_i = C\{V_i, E_i\} \qquad (4)$$

where ST_i refers to ith ciphertext. The recurrent neural network structure in Fig. 2 is a schematic representation of (1) to (4). The first hidden layer defines F_1 in (1). The second hidden layer has one neuron ξ. The output layer implements the function F_2 that computes the output Y_i as shown in (2). Finally, the output at time instant i is fed back through a zero order operator hold to construct the input to the network at the following time instant.

3.6 Decryption

The decryption procedure, Fig. 6, works in a similar fashion as that of the encryption procedure. When the symmetric cipher receives the ciphertext $C_i\{V_i, E_i\}$, the output Y_i is computed as

$$Y_i = F_2(V_i) \qquad (5)$$

Next, the original plaintext block can be restored using

$$M_i = Y_i + E_i \qquad (6)$$

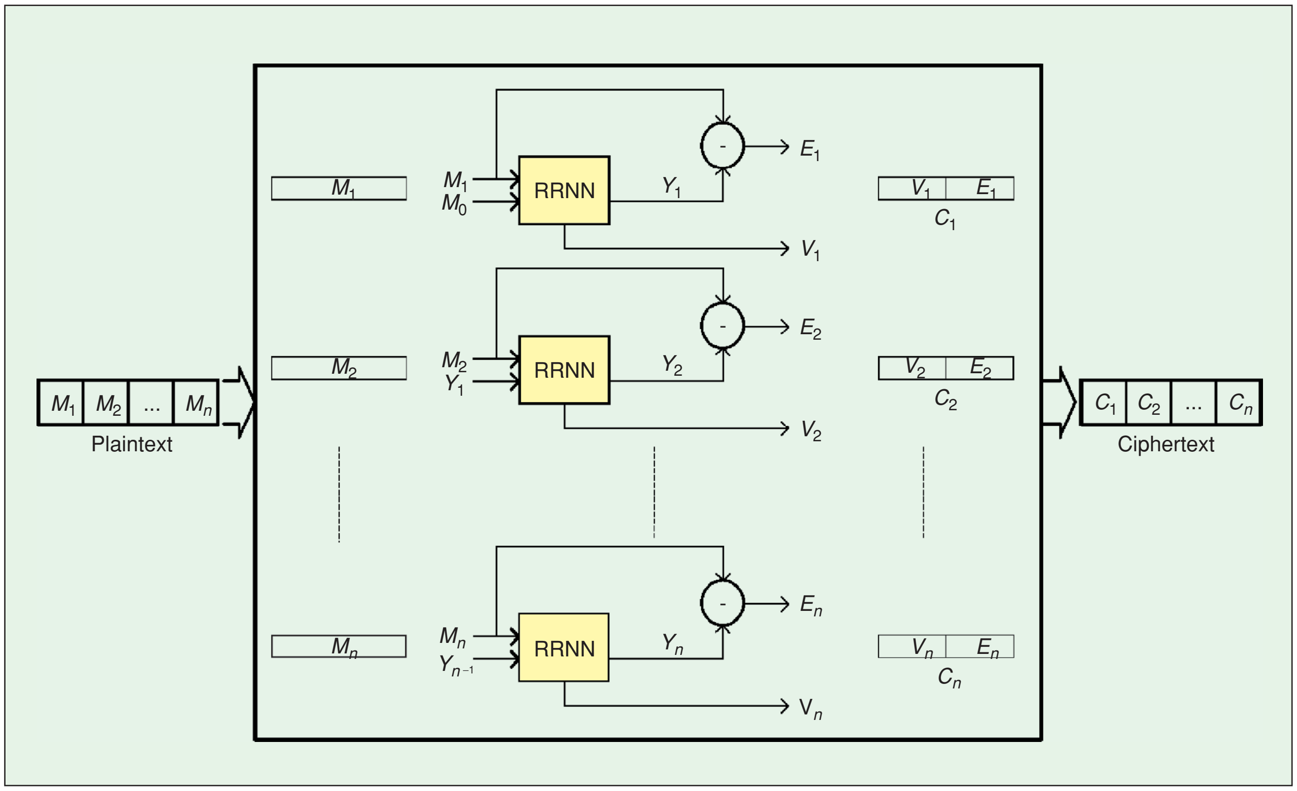


FIGURE 4 Encryption process.

After the message block M_i is restored, the one-epoch training step is performed using $X_i = (Y_{i-1}||M_i)$ as the input vector M_i as the training target. The output V_i of the final block can be used as the message authentication code (MAC) for the whole ciphertext. After calculating Y_i from V_i during decryption, M_i can be produced, and hence $X_i = (M_i||Y_i)$ is once again reconstructed. Then, V_i' is computed

$$V_i' = F_1\,(X_i) \tag{7}$$

Next, V_i' is compared with V_i to verify data integrity and authentication. In general, at the end of the data encryption/decryption stages, the cipher block chaining-message authentication code (CBC-MAC) [4] is prepared (or examined if it already exists) to ensure data integrity. The CBC mode encryption and decryption is illustrated in Fig. 7, where the P_i's are plaintext blocks and the C_i's are ciphertext blocks. CBC-MAC is a simple method that uses the last encrypted block as the MAC for the ciphertext chain.

The well-known characteristics of neural networks, such as parallel distributed structure, high computational power, ability to learn and represent knowledge as a black box, are successfully applied to cryptography.

3.7 RRNN and Cipher Design

By means of the real-time recurrent neural networks (Fig. 1), the symmetric cipher uses the forward dynamics (1) to (4) to generate the ciphertext and the MAC. The output of the network forward dynamics is computed as:

$$Y_j(n+1) = \varphi\left(\sum_{i=A\cup B} w_{ji}(n)U_i(n)\right), \qquad j \in B \tag{8}$$

where φ is a nonlinear activation function and the variable w_{ji} represents the synaptic weight between the neurons i and j, at discrete time n. In (8), $U_i(n)$ is the input vector to the RRNN, defined as in [5],

$$U_i(n) = \begin{cases} X_i(n) & \text{if} \quad i \in A \\ Y_i(n) & \text{if} \quad i \in B \end{cases} \tag{9}$$

where A denotes the set of indices i for which $X_i(n)$ is an external input, and B denotes the set of indices i for which $U_i(n)$ is the output of the neuron. Furthermore, the term representing the argument of the linear activation function in (8) is the neuron internal activity function V_i defined in (1). To define the initial values for the weight $w_{ji}(0)$, a set of uniformly distributed random numbers is chosen. Next, the dynamic process for updating the network weights in real time is defined by means of the following triple index

$$\vartheta^j_{kl}(n+1) = \varphi\left(V_j(n)\right) \times \left[\sum_{i\in B} w_{ji}(n)\vartheta^j_{kl}(n) + \delta_{kl}U_l(n)\right] \tag{10}$$

where $j \in B$, $k \in B$, $l \in A \cup B$, and $\varphi(.)$ is the derivative of the nonlinear activation function. In (10), δ_{kl} is the Krönecker delta, which equals to one when $k = l$ and zero otherwise. The triple index is initialized such that $\vartheta^j_{kl}(0) = 0$. The index in (10) is used to update the RRNN weights as follows:

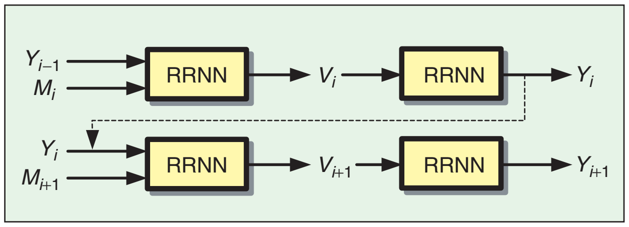


FIGURE 5 The symmetric cipher in CBC mode.

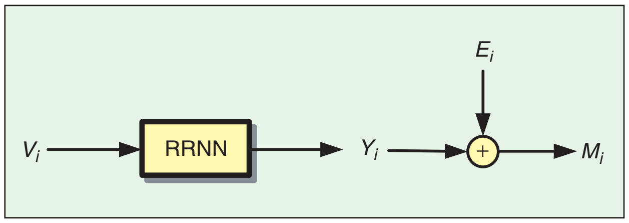


FIGURE 6 Decryption process.

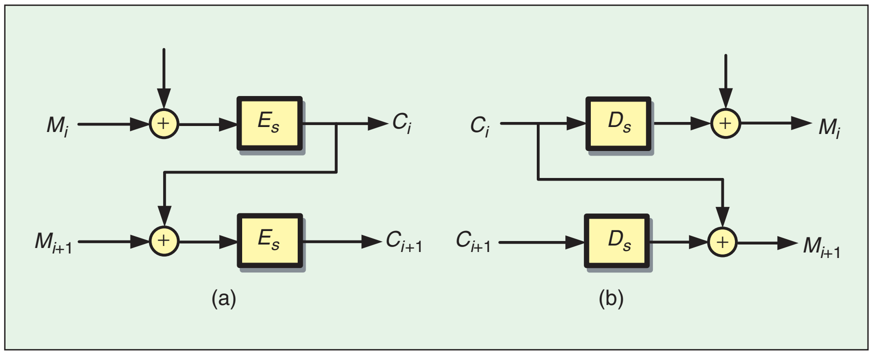


FIGURE 7 CBC mode—(a) encryption and (b) decryption.

Recent works have examined the use of neural networks in different layers of cryptosystems. Typical examples include key management, generation and exchange protocols; design of pseudo random generators; prime factorization; hash functions; symmetric ciphers; authentication; and authorization.

$$\Delta w_{ki}(n) = \eta \sum_{j} E_j(n) \vartheta^{j}_{kl}(n) \tag{11}$$

where Δw_{kl} denotes the update to the weight w_{kl}, and the parameter η refers to the learning rate of the network. In (11), the error function E_{j} at time instant n is computed as

$$E_j(n) = M_j(n) - Y_j(n)$$

Finally, the weight w_{kl} is updated according to the following:

$$w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n)$$

Both forward and backward dynamics vary in time to ensure that the learning procedure of the RRNN has the capability to detect temporal patterns of the training data. Consequently, the cipher can prepare the MAC to maintain both data integrity and data authentication.

4. Security Considerations

In this section, a number of attacks against the proposed symmetric cipher are examined. These include: (i) attack against the data encryption scheme itself; and (ii) attack against data confidentiality. Issues related to stability of neural networks during the learning process and their self adaptive learning procedure are also discussed.

4.1 Attacks against the Data Encryption Scheme

The encryption procedure of the proposed symmetric cipher can be viewed as a nonlinear mapping in which the ciphertext is the nonlinear transformation of the plaintext. If this function is static, the nonlinear equations can possibly be solved when the cryptanalyst has large volumes of plaintext with the corresponding ciphertext available. In comparison to other existing algorithms such as the data encryption standard [3], the extended key length k of the proposed symmetric cipher is much longer. Because the symmetric cipher makes use of the learning procedure of the neural network to encrypt data, it is assumed that a key stream encrypts the plaintext blocks. As a result, the extended key length k should be the total sum of all these keys within the same key stream period. The longer the key stream period is, the longer the extended key length k will be. This will result in a stronger symmetric cipher. If it can be guaranteed that the learning procedure will not converge quickly, the symmetric cipher can then generate a long period key stream. Consequently, the nonlinear transform function should be dynamic when it is applied for data encryption. The feedforward dynamics of the RRNN must keep varying in time to provide security protection for the plaintext. Furthermore, since the learning procedure usually tends to be convergent, cryptanalysis attacks based on the stability of the neural networks during learning may be an issue of importance. This is studied in the next subsections.

4.1.1 Attack against Data Confidentiality

Let G denote the set of plaintexts, Z the set of local and global minima, and L the largest invariant set in Z. L will contain all of the possible points at which the solution might converge and the trajectory can be trapped. Assume L contains only one fixed-point γ. A cryptanalyst will train the symmetric cipher with the known plaintext repeatedly until it converges to L. After the cipher is stabilized, all the secret plaintexts input that belong to G will converge to this fixed point. Although the cryptanalyst has no knowledge of the weight matrix and the initial state of the cipher, she can obtain the convergent point γ in L by means of the known plaintext. Then the cryptanalyst can restore the following secret plaintext M via the error signal E using $M = Y + E$. It shows that the stability of the neural networks will eventually help the cryptanalyst to break the cipher without knowledge of the weight matrix. To resist such an attack, the learning procedure needs to guarantee that convergence will not drift towards an invariant set L after the training of a large volume of plaintexts. This consideration is directly related to the stability problem of neural networks to be discussed in the next section.

4.1.2 Stability Problem of Neural Networks During Learning Process

Based on the above cryptanalysis discussion, the learning procedure has a tendency to be stable. During the recurrent back-propagation procedure, the Euclidean distance between the fixed point and the desired pattern is progressively reduced. As a result, the error signals will have smaller values. That will cause the dynamic of the backward propagation to decrease. The stability of the backward propagation will result in a stable weight matrix.

A RRNN can be modeled as nonlinear dynamic system, and the direct Lyapunov function [5] can be used to analyze the stability of neural networks—providing that such a function be found and used for the back-propagation algorithm processing. This can prove to be difficult. Alternatively, through a local analysis of the learning procedure of neural networks, it can be assumed that the local stability of the forward propagation is a sufficient condition for the local stability of the backward propagation and vice versa [52], [53]. Consequently, there is only a need to guarantee the instability of the backward propagation in (9), so that the forward propagation (used to generate the ciphertext) is ensured to be chaotic and unpredictable. According to (9), the instability of the backward propagation depends on both the error signal and the weight matrix. An estimate of the gradient has been used to approximate the true gradient curve of the cost function in order to perform real-time learning. If the learning rate η is set to a large value, a small mismatch between the output and the learning target will have a dramatic effect on the weight update process; hence this will cause the forward propagation to be unstable, i.e., chaotic. This chaotic oscillation of the learning behavior can then be generated in order to provide the desired data security.

4.2 Self Adaptive Learning Procedure

The self-adaptive function of the symmetric cipher is a necessary component to resist possible cryptanalysis attacks. This algorithm is required to implement such a function and it needs to detect the trend of the learning procedure via monitoring the mean squared error performance function (MSE), and then adjusts the learning rate by a Multiplicative-Increase Gradual-Decrease (MIGD) method, e.g., the TCP Vegas congestion control protocol [54]. At first, a low-pass filter for the MSE learns the trend detection as follows [5]:

$$T(k) = \delta T(k-1) + (1-\delta)^* MSE(k)$$

where δ is often selected between 0 and 1, $T(k)$ is the output of the low-pass filter of MSE at instance k and the initial state $T(0)$ is set to be zero. The learning stop condition MSE^{stop} is defined as:

$$MSE^{\text{stop}} \leq \alpha$$

where α is the critical value of $T(k)$. The learning rate will adapt itself according to the MIGD method based on one of the following three cases:

Case 1: $T(k) \leq \alpha$. The condition shows that the learning procedure tends to be convergent to the learning goal. To avoid the stability of the learning and restore the chaotic behavior, the learning rate η is increased aggressively by a factor λ, for example $\lambda = 2$. In this case: $\eta = \lambda \cdot \eta$.

Case 2: $T(k) > \alpha$ and $T(k) > T(k-1)$. The condition shows that the learning procedure tends to be oscillating. Hence, to maintain the learning rate close to the maximum allowable value, it should be gradually decreased by a factor θ, for example $\theta = 0.9$. In this case: $\eta = \theta \cdot \eta$.

Case 3: $T(k) > \alpha$ and $T(k) \leq T(k-1)$. In this case, the learning rate keeps the same value.

The above self-adaptive procedure can be performed at the conclusion of each epoch of training in both the encryption and decryption procedures. The critical value α can guarantee that the learning procedure will not settle at a stable point. At the same time, it helps maintain the learning rate close to the maximum allowable value so that the learning trajectory is closely related to the training data. More precisely, it will make the learning trajectory behave more randomly, which in turns makes the analysis of the learning procedure more difficult without the knowledge of the initial state of the network.

5. Simulation Results

A simulation software program in a MATLAB environment is developed to validate the proposed cipher. The simulation contains an encoder (Fig. 8) to encrypt the plaintext and a decoder

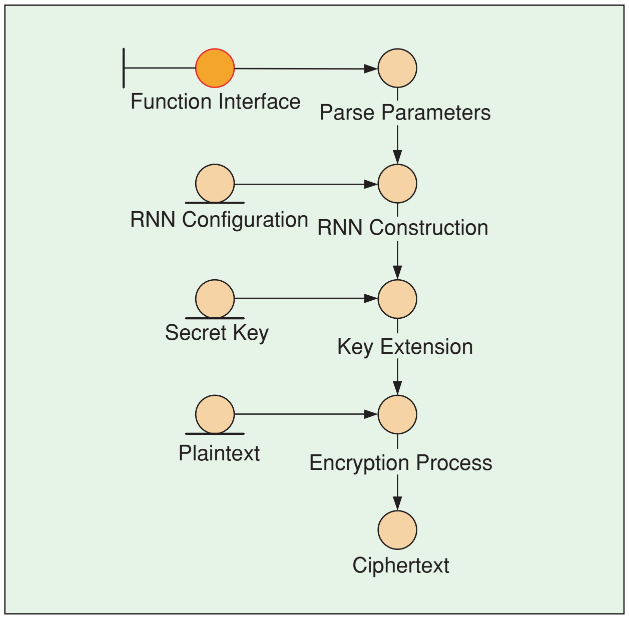


FIGURE 8 Encoder model.

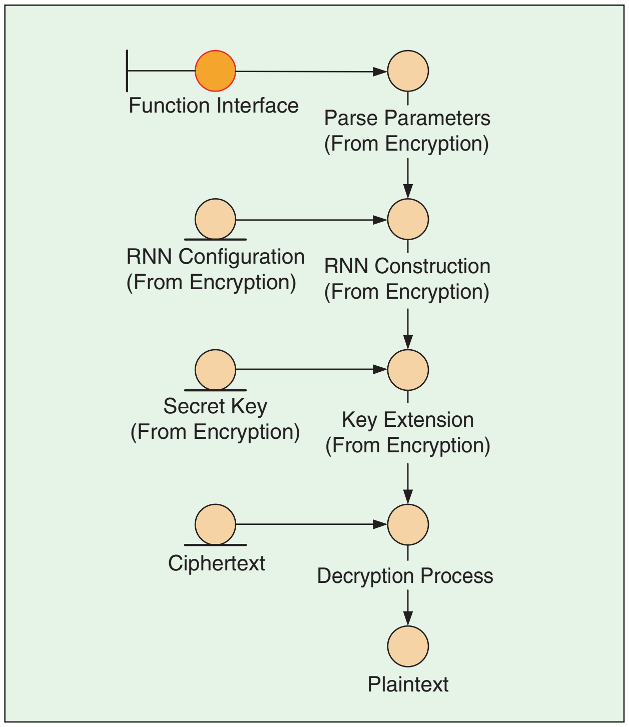

Function Interface
Parse Parameters (From Encryption)
RNN Configuration (From Encryption)
RNN Construction (From Encryption)
Secret Key (From Encryption)
Key Extension (From Encryption)
Ciphertext
Decryption Process
Plaintext

FIGURE 9 Decoder model.

The feedforward data manipulation process through multiple layers of the neural networks has functional resemblance to multiple round operations in a symmetric cipher, i.e., linear permutations, followed by nonlinear transformations.

(Fig. 9) to restore the plaintext from ciphertext. The encryption and decryption flowcharts are given in Figs. 10 and 11.

A script file is used to control the configuration of the cipher. This file defines some crucial parameters such as the dimension of the input/output vectors, the dimension of the hidden layer, the learning stop condition, the initial values for the weights and bias, etc. The symmetric cipher is constructed using a multi layer perceptron network. In order to perform the recurrent real-time learning, the output of the symmetric cipher is fed back as part of the input vector. The dimensions of the input and output vectors are respectively four and two. The hidden layer has only one neuron and its output represents the first part of the ciphertext. A sample plaintext is used as the external input signal for the symmetric cipher. The plaintext contains a long string of character "a", followed by a short string of character "z", then followed again by a long string of character "a". The plaintext is first translated to the corresponding ASCII code, and then scaled between 0 and 1. These values are permuted and padded by 0 (if necessary) to form several (4×1) vectors as input data for the symmetric cipher. The first two simulation experiments are carried out to analyze the effect of the learning rate on the network learning performance, while in the third experiment, the effect of the self-adaptive algorithm for updating the network's learning rate is investigated. The configuration parameters for all the experiments are given in Table II.

For the first two experiments, the ciphertext output is illustrated in Fig. 12 in terms of V_i and E_i. Fig. 12(a) is the first part of the ciphertext V_i as described in (6). Since V_i is actually the output of the neural network, it is a value in the interval $(0, 1)$. In Fig. 12(b), the second part of the ciphertext in E_i (as described in (6)) is presented. Since E_i

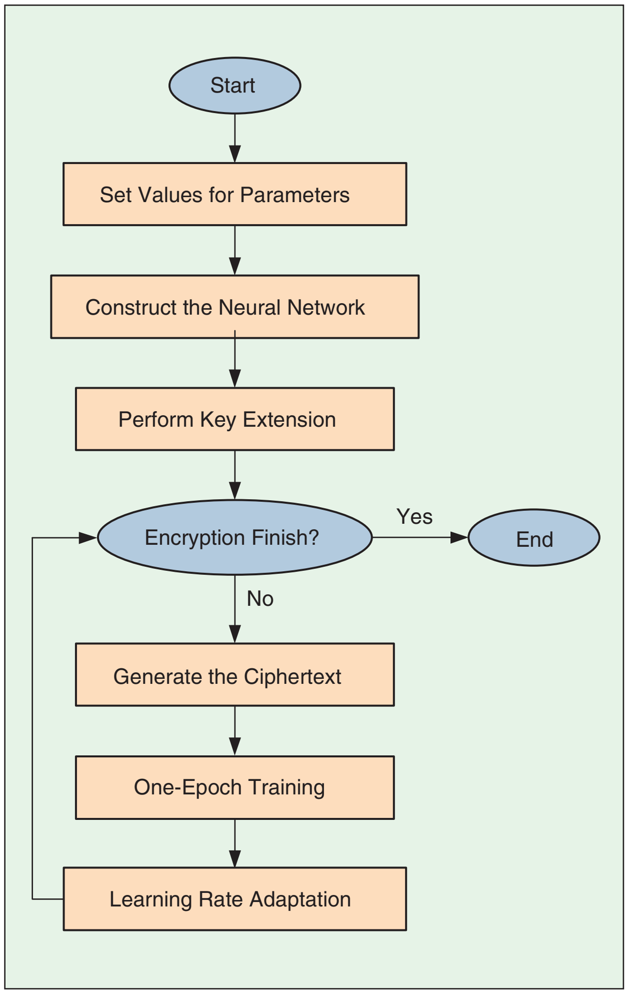


FIGURE 10 Encryption flowchart.

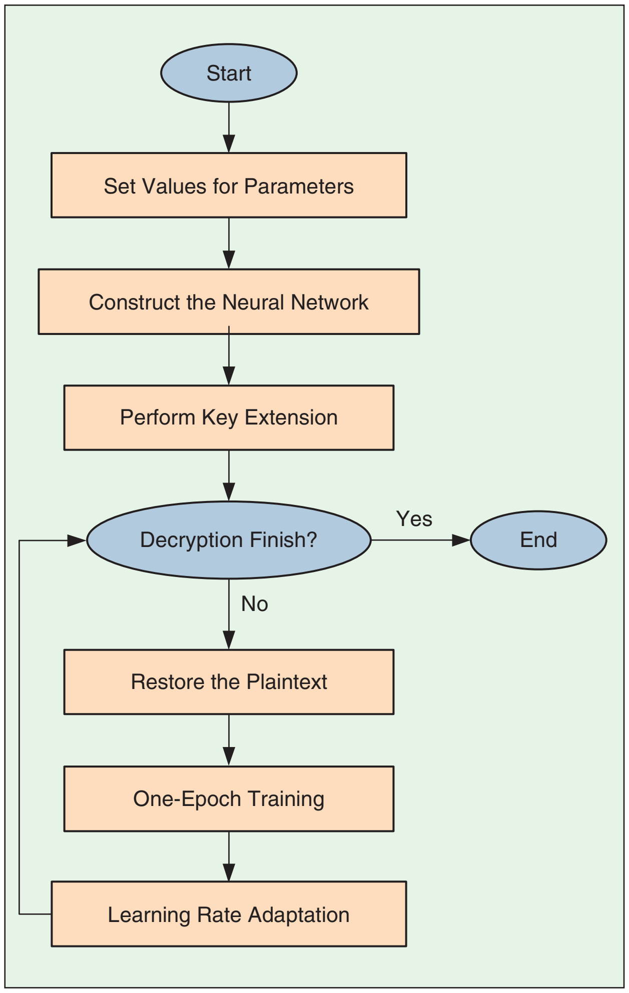


FIGURE 11 Decryption flowchart.

TABLE 2 Configuration parameters for the first, second, and third experiments.

EXPERIMENT	WEIGHT INITIAL VALUE	EPOCHS FOR KEY EXTENSION	DIMENSION OF INPUT VECTOR	DIMENSION OF OUTPUT VECTOR	LEARNING RATE FOR ENCRYPTION	LEARNING STOP CONDITION	LEARNING RATE ADAPTATION
1	0.5	600	4	2	0.05	1e-50	DISABLED
2	0.5	600	4	2	35	1e-50	DISABLED
3	0.5	600	4	2	1	1*e*-50	CRITICAL VALUE α: 0.04 INCREASE FACTOR λ: 2 DECREASE FACTOR θ: 0.9

is the two-dimensional error signal between the input and the output, it may assume negative values. The only difference between the first and second experiments is in their learning rate. The ciphertext output for the second experiment is illustrated in Fig. 13.

Comparing the results of these two experiments, it is observed that the second part of the ciphertext, E_i, has weaker protection than the first part when the learning rate is small. Hence, the first part of the ciphertext output can be the MAC for the corresponding plaintext blocks. Therefore, it will be much difficult for a cryptanalyst to perform an attack based on the first part of the ciphertext. Since the weakest point on the text can be used to examine the security of the symmetric cipher, the focus should be on the analysis of the second part of the ciphertext.

Based on experiments, when the learning rate is small, the second part of the ciphertext, E_i, will expose the temporal pattern of the plain text. The cryptanalyst can then perform the attacks discussed earlier to guess the new character "*z*". But when the learning rate is set to a large value, the learning procedure can be prevented from convergence and the temporal structure of the plain text input can be protected because the ciphertext is chaotic. Even though the ciphertext looks chaotic, it is difficult to determine whether it has a limited number of states. It is often desirable for the error signal to have an unpredictable number of states so that the further cryptanalysis is impossible. This can be achieved by introducing more random factors into the ciphertext generation process (i.e., the learning process of the neural network). A suitable source for random factors would be the plaintext itself. This is illustrated in the third experiment. The two parts of the ciphertext output for this experiment are shown in Fig. 14.

When the learning error reaches the critical value α, the self-adaptive procedure will be triggered. In that case, the learning rate value is multiplied by an increase factor λ. Afterwards, if the learning procedure oscillates according to the case 2 of the MIGD method studied previously, the learning rate value is multiplied by the decrease factor θ. On the other hand, the learning rate value is sustained if the learning procedure is in accordance with Case 3 of the MIGD method. In the third experiment, when the learning rate is set to a large value, the error signal will diverge away from the critical value and learning goal. Consequently, the parameter α can be used as a knob to control the learning and make it unpredictable, thus guaranteeing a desirable instability. A large learning rate can help to hide the temporal structure of the plaintext input data and force the symmetric cipher to generate chaotic ciphertext.

6. Challenges, Considerations and Future Research Directions

In this article, the use of neural networks to design symmetric ciphers has been investigated. This work offers the promise of a new source for private key cryptographic schemes that are not based on number theoretic functions, and which have small time and memory complexities. In the long term, it is hoped to better understand the potential relation between the machine learning and cryptography, and to design, develop and to evaluate protocols to create neural network based cryptosystems.

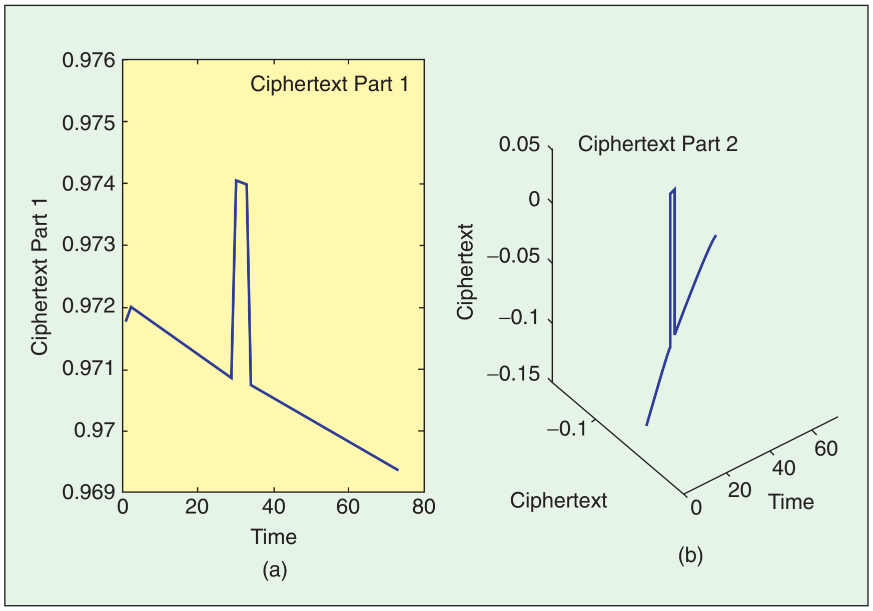


FIGURE 12 Small fixed learning rate effect. (Learning rate = 0.05, learning rate adaptation disabled).

Additional research is required to further this work and to fully highlight the strength and limitations of neural networks application to cryptography. These include:

1. The proposed symmetric cipher is based on an error-correcting learning algorithm (i.e., the self-adaptive procedure). Because of the nature of this algorithm, the reliability of the symmetric cipher is not as strong as it should be. This might be due to the authors' observation that the speed of the change of the gradient of the learning trajectory is sometimes much faster than the change of the error. Therefore, when the precision limit is met before the threshold, the learning procedure is broken, and stops suddenly. In general, it is not possible for the algorithm to control or predict the gradient, and the state of the system is difficult to determine when the symmetric cipher stops suddenly. There are two possible ways to avoid this problem: (i) place some constraints on the selection of values for the threshold; (ii) investigate the possibility of using other learning algorithms than the one proposed in this article. This latter alternative requires a comparative study of existing learning and pruning algorithms to determine the best possible choice.
2. It is desired to benchmark the results in relation to classic algorithms such as the Advanced Data Encryption Standard-Rijndael (AES) [55].
3. A comprehensive statistical analysis must be completed. Statistical analysis plays an important role in evaluation of stream ciphers and randomness properties of the ciphers.
4. A rigorous cryptanalysis of the proposed cipher is to be performed. The proposed cipher should be resistant to cryptanalytic attacks. To this end, common attacks are to be investigated. These include distinguishing attacks, exhaustive key search, statistical attacks, high order correlation attacks, divide and conquer attacks, and re-keying attacks, differential (chosen-plaintext) and linear (known-plaintext) attacks. For example, a possible chosen-plaintext attack has already been proposed in [56].
5. Ciphers are primarily evaluated for both security and practicality. Practical ciphers must be extremely efficient, must perform encryptions and decryptions very quickly, and must be implementable in dedicated hardware, ASICs or FPGAs.
6. The neural network computations mostly include matrix manipulations and nonlinear transformations. The computational complexity versus existing methods where calculations are mostly integer and bitwise primitive operations have to be studied.
7. The continued development and refinement of the proposed symmetric cipher design and its self-adaptive learning procedure should remain an important area of research into the foreseeable future.

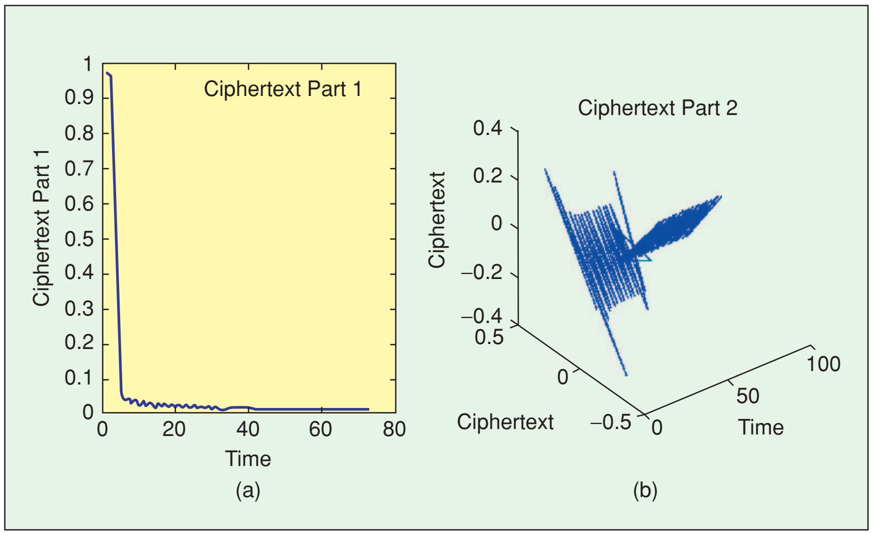


FIGURE 13 Large fixed learning rate effect. (Learning rate = 35, learning rate adaptation disabled).

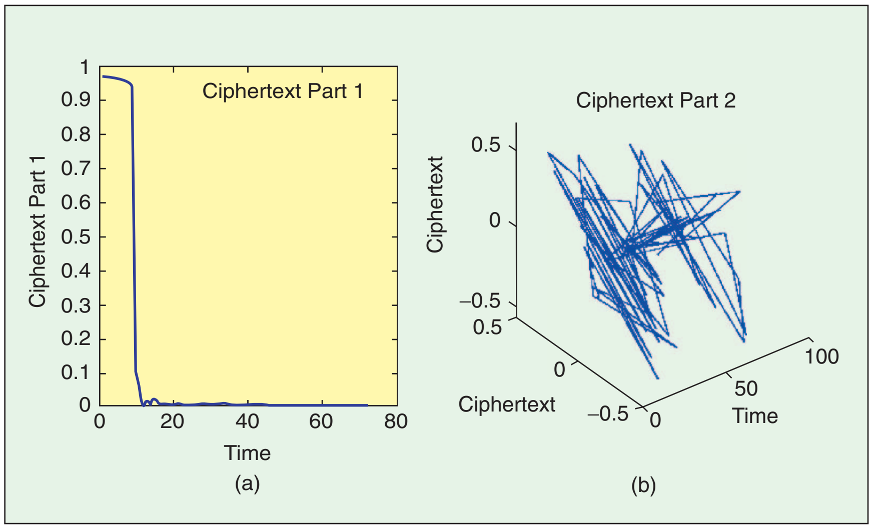


FIGURE 14 The two parts of the ciphertext output for the third experiment.

7. Conclusion

In this article, a symmetric cipher design based on the application of recurrent neural networks in cryptography was described. The proposed design has several advantages due to use of RRNN for symmetric ciphers. The security of the proposed cipher is based on the assumption that the weight distribution of the hidden layers is unpredictable without knowledge of the original key. The proposed cipher fulfills a number of requirements including: support for variable key length, support for variable block length, and support for improved security. Different cryptographic services are provided by an integrity scheme with a relatively simple architecture. Furthermore, the inherent parallel computing capability of the cipher can accommodate high performance data encryption requirements such as secure point-to-point file transfer between gateways. Simulation results show that the learning procedure of the recurrent neural network can be controlled to provide secure protection for data by adapting the learning rate.

Acknowledgments

The authors would like to acknowledge the insightful comments and helpful suggestions by the anonymous reviewers and the editors which contributed to the improvement of the final version of the article. This work greatly benefited from valuable discussions, suggestions, and the assistance of Dr. I. Woungang and Dr. W. Melek. The authors also acknowledge the following people, who read the draft of this article and made useful suggestions: Dr. E. Harley, M. Zamani, V. Akbarzadeh, and R. Salmon. A portion of this paper appears in [9], Copyright 2007, IGI Global, and is used by permission of the publisher.

References

[1] R. Stinson. *Cryptography, Theory and Practice*, 2nd edition, CRC Press, 2002.

[2] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. Handbook of Applied Cryptography, CRC Press, 1997.

[3] W. Stallings. *Cryptography and Network Security: Principles and Practices*. Prentice Hall, 2003.

[4] B. Schneier. *Applied Cryptography*. John Wiley & Sons Inc., New York, 1996.

[5] S. Haykin. *Neural Networks: a Comprehensive Foundation*. 2nd edition, Prentice Hall, 1998.

[6] S. Wu. *A Block Cipher Design Using Recurrent Neural Networks*. M. Eng. dissertation, Ryerson University, Toronto, Canada, 2003.

[7] M. Arvandi. *Analysis of Neural Network Based Ciphers*. M.A.Sc. dissertation, Ryerson University, Toronto, Canada, 2005.

[8] M. Arvandi, S. Wu, A. Sadeghian, W. Melek, and I. Woungang, "Symmetric Cipher Design Using Recurrent Neural Networks," in *Proc. of the International Joint Conference on Neural Networks*, pp. 2039–2046, 2006.

[9] I. Woungang, A. Sadeghian, S. Wu, S. Misra, M. Arvandi, "Wireless Web Security Using a Neural Network-Based Cipher," Chapter II in "*Web Services Security and E-Business*," G. Radhammani and G. S.V. Radha Krishna Rao (Eds.), Idea Group Publishing Inc., USA, ISBN: 1-59904-168-5, pp. 32–56, 2006.

[10] D.E. Rumelhart, G.E. Hinton, and J.L. McClelland, "A general framework for parallel distributed processing Source," *Parallel distributed processing: explorations in the microstructure of cognition, Computational Models of Cognition and Perception Series*, vol. 1, pp. 45–76, MIT Press, 1986.

[11] H. White, "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks,* vol. 3, no. 5, pp. 535–549, 1990.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multi-layer feedforward networks are universal approximator," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[13] A. Pincus, "Approximation theory of the MLP model in neural networks," *Acta Numerica*, pp. 143–195, Cambridge University Press, 1999.

[14] J.A. Hertz, R.G. Palmer, and A. Krogh, *Introduction to the Theory of Neural Computation*. Westview Press, 1991.

[15] J. Benitez, and J.L. Castro, and I. Requena, "Are artificial neural networks black boxes?" *IEEE Trans. Neural Networks*, vol. 8, no. 5, pp.1156–1164, Sept. 1997.

[16] A. Sadeghian, M. Zamani, and V. Akbarzadeh, "Applications of Artificial Neural Networks in Cryptography—A Survey," in progress.

[17] R.L. Rivest, "Cryptography and Machine Learning," Advances in Cryptology, Lecture Notes In Computer Science, vol. 739, pp. 427–439, 1991.

[18] D. Pointcheval, "Neural Networks and Their Cryptographic Applications," in *Proc. of Eurocode*, pp. 183–193, 1994.

[19] W. Kinzel, and I. Kanter, "Neural Cryptography," *Proc. of the 9th Int'l Conf. on Neural Information Processing (ICONIP'02)*, vol. 3, pp. 1351–1354, 2002.

[20] A. Ruttor, W. Kinzel, L. Shacham and I. Kanter, "Neural cryptography with feedback," *Phys. Rev. E.*, (69): 046110—1-7, 2004.

[21] R. Mislovaty, E. Klein, I. Kanter, and W. Kinzel, "Public Channel Cryptography by Synchronization of Neural Networks and Chaotic Maps," *Phys. Rev. Lett., (91), 118701-1-4*, 2003.

[22] L.N. Shacham, E. Klein, R. Mislovaty, I. Kanter, and W. Kinzel, "Cooperating attackers in neural cryptography," *Phys. Rev. E.*, (69), 066137-1-4, 2004.

[23] R. Mislovaty, Y. Perchenok, I. Kanter, and W. Kinzel, "Secure key-exchange protocol with an absence of injective functions," *Phys. Rev. E.*, (66): 066102-1-5, 2002.

[24] W. Kinzel and I. Kanter, "Interacting Neural Networks and Cryptography," *Advances in Solid State Physics*. B. Kramer (ed.), vol. 42, pp. 383–391, Berlin: Springer, 2002.

[25] M. Rosen-Zvi, I. Kanter and W. Kinzel, "Cryptography based on neural networks—analytical results," *J. Phys. A: Math. Gen.*, vol. 35, no. 47, pp. 707–713, 2002.

[26] M. Volkmer and S. Wallner, "Tree Parity Machine Rekeying Architectures," *IEEE Trans. Computers*, vol. 54, no. 4, pp. 421–427, April 2005.

[27] E. Klein, R. Mislovaty, I. Kanter, A. Ruttor, and W. Kinzel, "Synchronization of neural networks by mutual learning & its application to cryptography," *NIPS*, pp. 689–696, 2004.

[28] A. Klimov, A. Mityaguine, and A. Shamir, "Analysis of Neural Cryptography," *AsiaCrypt 2002*, pp. 288–298, 2002.

[29] J. Zhou, Q. Xu, W. Pei, Z. He, and H. Szu, "Step to improve neural cryptography against flipping attacks," *Int. J. Neural Syst.*, vol. 14, no. 6, pp. 393–405, Dec. 2004.

[30] R. Mislovaty, E. Klein, I. Kanter, and W. Kinzel, "Security of neural cryptography," *11th IEEE Int'l Conf. on Electronics, Circuits and Systems*, pp. 219–221, pp. 13–15, 2004.

[31] A. Ruttor, W. Kinzel and I. Kanter, "Neural cryptography with queries," *J. Stat. Mech.*, P01009, 2005.

[32] S. Su, A. Lin, and J. Yen, "Design and realization of a new chaotic neural encryption/decryption network," *IEEE Asia-Pacific Conf. Cir. & Syst.*, pp. 335–338, 2000.

[33] D. Xiao and X. Liao, "A Combined Hash and Encryption Scheme by Chaotic Neural Network," *Proc. Int'l Sym. Neural Nets 2004*, vol. 2, pp. 633–638, 2004.

[34] C. Li, S. Li, D. Zhang and G. Chen, "Cryptanalysis of a Chaotic Neural Network Based Multimedia Encryption Scheme," *5th Pacific Rim Conf. Multimedia*, pp. 418–425, 2004.

[35] R. Bose, "Novel public key encryption technique based on multiple chaotic systems," *Phys. Rew. Lett. 95*, 2005.

[36] K. Wang, W.J. Pei, Z.L. Hua, Z.Y. He, "Comment on Novel Public Key Encryption Technique Based on Multiple Chaotic Systems," *Phys. Rew. Lett. 95*, 2005.

[37] T. Zhou, X. Liao, Y. Chen, "A novel symmetric cryptography based on chaotic signal generator and a clipped neural network," *Lecture notes in Computer Science, Advances in Neural Networks*, vol. 3174, pp. 639–644, 2004.

[38] C. Chi-Kwong, L.M. Cheng, "The convergence properties of a clipped Hopfield network and its application in the design of keystream generator," *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 340–348, March 2001.

[39] L.P. Yee and L.C. De Silva, "Application of Multilayer Perceptron Networks in Public Key Cryptography," in *Proc. of the 2002 Int'l Joint Conf. on Neural Networks*, vol. 2, pp. 1439–1443, 2000.

[40] C. Meletiou, D.K. Tasoulis and M.N. Vrahatis, "A First Study of the Neural Network Approach in the RSA Cryptosystem," in *Proc. of the Int'l Conf. on Artificial Intelligence and Soft Computing for Information Security—ASC 2002.*

[41] C.G. Meletiou, D.K. Tasoulis, and M.N. Vrahatis, "Cryptography through Interpolation, Approximation and Computational Intelligence methods," *Bulletin of the Greek Mathematical Society*, vol. 48, pp. 61–75, 2003.

[42] D.A. Karras and V. Zorkadis "On neural network techniques in the secure management of communication systems through improving and quality assessing pseudorandom stream generators," *Neural Networks*, vol. 16, no. 5–6, pp. 899–905, 2003.

[43] X. Yang, X. Huang and H. Huang, "Improving Ability of Passive Attacks of Chaotic Encryption by Using Neural Network," *Lecture Notes in Computer Science*, pp. 624–629, 2005.

[44] L.P. Yee, and L.C. De Silva, "Application of multilayer perceptron network as a one-way hash function," in *Proc. of the International Joint Conference on Neural Network*, vol. 2, pp. 1459–1462, 2002.

[45] L.P. Yee, and L.C. De Silva, "Application of multilayer perceptron networks in symmetric block ciphers," in *Proc. of the International Joint Conference on Neural Network*, vol. 2, pp. 1455–1458, 2002.

[46] B. Jansen, and K. Nakayama, "Neural Networks Following a Binary Approach Applied to the Integer Prime-Factorization Problem," in *Proc. of the International Joint Conference on Neural Network*, vol. 4, pp. 2577–2582, 2005.

[47] D.A. Karras, and V. Zorkadis, "Improving pseudorandom bit sequence generation and evaluation for secure Internet communications using neural network techniques," in *Proc. of the International Joint Conference on Neural Network*, vol. 2, pp. 1367–1372, 2003.

[48] D. Guo, L.M. Cheng and L.L. Cheng, "A New Symmetric Probabilistic Encryption Scheme Based on Chaotic Attractors of Neural Networks," *Applied Intelligence*, vol. 10, no. 1, pp. 71–84, 1999.

[49] C. Li, S. Li, D. Zhang, and G. Chen, "Chosen-plaintext cryptanalysis of a clipped-neural-network-based chaotic cipher," *Lecture Notes in Computer Science, Advances in Neural Networks*, vol. 3497, pp. 630–636, 2005.

[50] R. Reed, "Pruning Algorithms: a Survey," *IEEE Trans. Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[51] E. Cantú-Paz, "Pruning Neural Networks with Distribution Estimation Algorithms," in *Proc. of the Genetic and Evolutionary Computation Conf.*, pp. 790–800, 2003.

[52] S. Townley, A. Iichmann, M.G. Weiss, W. McClements, A.C. Ruiz, D.H. Owens, and D. Pratzel-Wolters, "Existence and Learning of Oscillations in Recurrent Neural Networks," *IEEE Trans. Neural Networks*, vol. 11, no. 1, pp. 205–214, 2000.

[53] L. Almeida, "Learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Proc. of the 1st IEEE International Conf. on Neural Networks*, vol. 2, pp. 105–110, 1987.

[54] U. Hengartner, J. Bolliger, and T. Gross, "TCP Vegas revisited," in *Proc. of the INFOCOM, 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1546–1555, 2000.

[55] ADVANCED ENCRYPTION STANDARD (AES), Federal Information Processing Standards Publication 197, Nov. 26, 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[56] M. Arvandi and A. Sadeghian, "Chosen Plaintext Attack against Neural Network-Based Symmetric Cipher," in *Proc. of the International Joint Conf. on Neural Networks*, pp. 847–851, 2007.