Design of Word Based Stream Cipher using Tree Parity Machine

Sowmya D¹ & Priyanka B.R²
¹Assistant Professor Department of CSE, JNNCE, Shivamogga, Karnataka, India
²PG student, Department of CSE, JNNCE, Shivamogga, Karnataka, India

Abstract: Neural Cryptography approaches probabilistic algorithm for key exchange with assured security. This method also helps to overcome the overhead of key distribution that was found in the existing symmetric encryption techniques. The basic arithmetic computation on Tree Parity Machine leads to secret key generation between two ends through the process of synchronization.

1. Introduction

In the area of cryptography, people are only interested in methods that can be used to send the secret messages among the two or more participants through public network. The communication should be carried out in such a way that even though an opponent who listens to it must not be able to retrieve the secret message which is being transmitted. The cryptography methods which generates single key that is used by both sender and receiver for encryption and decryption is called as symmetric cryptography. Later in the year 1976, Diffie and Hellman found that this secret key can be accessible by an opponent therefore existing methods was vulnerable to attacks. The distribution of the secret key among the participants was one of the major drawbacks of the symmetric encryption system. Recently it was found that the involvement of the Artificial Intelligence (AI) concept with the cryptography can also be used to create a symmetric environment without the overhead of key distribution. Usage of AI further includes the neural networks hence this process termed as “Neural cryptography”, is one of the recent method that could aim to resolve the problem of key exchange that was found in the earlier cryptography methods.

Neural cryptography is considered as a non-classical cryptographic protocol used to derive the secret key which further can be used for encryption and decryption process. The security of these neural networks depends on the fact which compares the bidirectional learning of the partners and the unidirectional learning of the attacker and also depends on the attractive and repulsive forces of the networks.

1.1 Artificial Neural Networks

Artificial Neural Network (ANN) is a mathematical mechanism which was originally inspired by the biological model of neurons (neuron structure in the brain). The human brain contains billions of neurons which are interconnected with each other, which accounts for the complex features of human beings, such as memory, creativity, language and consciousness. Computational neural networks exploit the potentially vast and compelled processing power of such a parallel interconnected architecture.

1.2 Tree Parity Machine (TPM)

The Tree Parity Machine (TPM) consists of neurons and structured as feed forward neural network as shown in the figure 1. The network has three layers: input layer, hidden layer and output layer. Input layer (Xij) takes binary input {-1, +1} and an initial random weight vector (Wij) is assigned to the links between input layer and hidden layer. Neurons in the hidden layer called as sigma (σA/B) is computed using sign function as per the equation (1). The hidden layer neurons are connected to an output neuron called tau (τA/B) which can be calculated using equation (2).

\[ σ_i = \text{sign} \left( \sum_{j=1}^{N} W_{ij} * X_{ij} \right) \]  
\[ τ = \prod_{i=1}^{N} σ_i \]  

Two TPMs which share the output bit and input vector on public medium synchronizes to a time dependent weight vector [4]. Mutual learning is a concept behind the synchronization of TPM. If output of TPM A and TPM B are equal, the network trains the perceptron which is having similar output bit in the hidden layer. The perceptrons have been trained using any one of the learning algorithms given in the equation (3) (4) (5). This stochastic process of learning forces attractive and repulsive behaviour on the Tree Parity Machine. When the attractive forces lead the repulsive forces, the two TPMs converge to identical weights which are translated to symmetric keys for encryption and decryption [1].
Figure 1. Structure of tree parity machine.

Hebbian Rule:
\[ W_{ij}^{+} = W_{ij} + \tau X_{ij} \square (\sigma_A \tau \square (\tau^A \tau^B)) \] (3)

Anti-Hebbian Rule:
\[ W_{ij}^{+} = W_{ij} - \tau X_{ij} \square (\sigma_A \tau \square (\tau^A \tau^B)) \] (4)

Random Walk:
\[ W_{ij}^{+} = W_{ij} + X_{ij} \square (\sigma_A \tau \square (\tau^A \tau^B)) \] (5)

2. Synchronization

The communication process between the partners can start when they have the secret key. In order to have a secret key using neural network a process called synchronization had to be followed by all the participants where they can finally get a pseudorandom secret key through which their communication can be made more secure. The synchronization of two or more neural networks is nothing but having the same set of weights [2]. The steps followed to achieve the synchronization state are,

Algorithm:
1. Initialize random weight values.
2. Execute the following steps until the full synchronization is achieved.
   2.1 Generate random input vector x
   2.2 Compute the output values of the hidden neurons.
   2.3 Compute the value of a output neuron
   2.4 Compare the values of both TPMs:
      2.4.1 If the outputs are different then go to Step 2.1.
      2.4.2 If the outputs are same then one of the suitable learning rule is applied in order to update the weight vectors.

A counter value is incremented if \( \tau^A \) and \( \tau^B \) are equal and it is reset to zero if the values are not equal. The counter is used to perceive the amount of synchronization on a network. The flow chart for synchronization of TPM is as shown in figure 2.

3. Feedback Approach

The steps to generate Keystream are:
1. In the first iteration, synchronized weights are used as seed to randomly generate inputs, the sigma value is computed for the new inputs and synchronized weights. Later these sigma values are converted into binary format and xorred with the adjacent bits and stored in a text file (Keystream).
2. The sigma values are converted to decimal and fed back as seed value to generate random inputs to TPM.
3. Weights are modified using the function, where every weight is sum up with their perceptron value (sigma) and updated to the range of synaptic depth (L).
4. Compute sigma,
   4.1 Convert them into binary.
   4.2 Xor the adjacent bits and store it in a text file then go to step 2.

Figure 2. Synchronization process.
The flowchart for generation of keystream is as shown in figure 3, by repeating the steps the keystream of required length can be generated and later its degree of randomness can be tested using NIST’s statistical test suite. The generated keystream bits is as shown in figure 4.

Figure 4. Generated key bits.

4. Randomness Test Suite

The NIST Test Suite is a statistical package consisting of 15 tests (figure 5) that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests [4].