Design of Stream Cipher for Text Encryption using Soft Computing based Techniques

S.Santhanalakshmi [1], Sangeeta K [1], G.K.Patra[2]

[1] Dept.of CS&E,AmritaVishwa Vidyapeetham, School of Engineering, Bangalore Campus Bangalore, India

[2]Centre for Mathematical Modelling and Computer Simulations, Council of Scientific and Industrial Research, Bangalore-560037

Abstract:

In cryptography, encryption is the process of transforming information referred to as plaintext using an algorithm (called a cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is information which is, referred to as ciphertext. Stream ciphers are used to encrypt individual bits. This is achieved by adding a bit from a key stream to a plaintext bit.Generating the key stream is the therefore important. In this paper a soft computing based approach is proposed for generating keys to design a stream cipher for text encryption. Optimal weights for the sender and receiver used for the synchronization on the Tree Parity Machine(TPM) neural network, are generated using a Genetic Algorithm(GA).

Keywords:

Key Generation, Stream Cipher, Genetic algorithm, Tree parity machine.

1. Introduction

Neural cryptography creates a shared secret key based on synchronization of Tree Parity Machines (TPM) by mutual learning. Two neural networks trained on their mutual output bits synchronize to a state with identical time dependent weights. This has been used for creation of a secure cryptographic secret key using a public channel. A key stream is a group of characters denoting the keys for text encryption. Once the key stream is generated an XOR operation is performed with the keys and the encoded plain text to obtain the encrypted text.

Stream cipher is a symmetric key encryption where each bit of data is encrypted with each bit of key. The Crypto key used for encryption is changed dynamically so that the cipher text produced is secure and difficult to crack.

In synchronous Stream cipher the key stream is generated independent of plain or cipher text. The simplest form Synchronous stream cipher is called "One-time pad", sometimes called Vernam Cipher [1]. . An One-time pad uses a key stream of completely random digits. Here one bit of the key stream is XOR with a bit of the plain text to get the cipher text for encryption . However, the key stream must be of the same length as the plaintext and generated from a truly random number generator. This makes the system very cumbersome to implement in practice and as a result the One-time pad has not been widely used except for most critical applications.

The modern stream ciphers use much smaller and convenient key (say 128 bits). Based on this key a pseudorandom key stream is generated which can be combined with plaintext digits in a similar fashion to the One-time pad. Only disadvantage here is that the key stream so obtained will be pseudorandom and not truly random.

Biham and Seberry [2] proposed a fast and secure sream cipher for encryption. This method is based on a new kind of primitive, called Rolling Arrays. It also includes variable rotations and permutations. Sreelaja and Pai [3] proposed an Ant Colony Optimization (ACO)[4] based algorithm for key generation. The algorithm is based on the distribution of characters in the plain text.

This paper proposes a soft computing based stream cipher method to encrypt data messages sent in a network. This approach makes use of GA and TPM to generate keys needed for encryption. The paper is organized as follows. Generation of secret keys using optimal weights is described in Section 2. Section 3, describes the implementation of Tree Parity Machine(TPM) network. Section 4 describes the security attack followed by the results in Section 5.

2. Generation of Secret Keys Using Optimal Weights

Neural cryptography is based on synchronization of TPM[4] by mutual learning. Here two identical dynamic systems (Neural Network) starting from different initial weights receive an identical input vector, generate an output bit and are trained based on the output bit till they synchronize to a state with identical time dependent weights [5]. In this case the two parties A & B use their identical weights as secret key needed for encryption. This

Manuscript received December 5, 2012
Manuscript revised December 20, 2012

has been used for creation of a secure cryptographic secret key over a public channel.
In this paper, synchronization of tree parity machines by mutual learning process is achieved using a genetic approach. Here a best fit weight vector is found using a genetic algorithm described in the next section.

2.1. Genetic Algorithm

A Genetic Algorithm (GA) is a population based, probabilistic technique that operates to find a solution to a problem from a population of a possible solutions[6, 7]. In the evolution of the algorithm, the individuals of a given population interchange their genotypes, according to their fitness values and some probabilistic transition rules, in order to produce a new generation. The individuals in the new generation are then evaluated. based on their fitness function provided by the programmer, higher the fitness, higher the chance of being selected. These individuals then "reproduce" to create one or more offspring, after which the offspring are mutated randomly. This continues until a suitable solution has been found or a certain number of generations have passed, depending on the needs of programmer. In this paper this algorithm has been used to generate the input weights for the tree parity machine.
The generation of optimal weights using genetic algorithm comprises of the following steps-- initialization, selection, reproduction and termination as illustrated in Fig 1. To start with the initial population weight vector w_i is taken as set of random numbers in the range [-L, L]. The fitness function f(x) is assumed to be parabolic and is defined as,

$$F(x) = \begin{cases} -L & x < -L \\ x^2 & -L \le x < L \\ L & x \ge L \end{cases}$$

The fitness value for each string in the population is calculated. Based on the fitness value, the most fitted strings from the population are selected using Roulette Wheel selection method. In roulette wheel selection, individuals are given a probability of being selected which is directly proportionate to their fitness. Two individuals are then chosen randomly based on these probabilities and produce offspring. On the selected string crossover and mutation are performed based on the Crossover rate (Pc) and Mutation rate (Pm) to generate new weights, which become the input for a further run of the algorithm. This completes one cycle of GA process. If the termination condition is met, stop the iteration and the new population generated will be considered as the optimal solution. Fig 1 shows the GA Cycle.

Figure 1: GA Cycle

3. Tree Parity Machine Implementation

Synchronization of the tree parity machine Figure 2. using genetic algorithm is achieved as described below:

- Initialize weight values w_i, obtained from GA process in range [-L, L]
- Execute these steps until the full synchronization is achieved
  - Generate random input vector x_i (step 1)
  - Compute the values of the hidden neurons $\sigma_i x_i$
  - Compute the value of the output neuron τ
  - Compare the values τ of both tree parity machines
    - (i) Outputs are different go to step1
    - (ii) Outputs are same: one of the following learning rules is applied to the weights

Hebbian learning rule:

$$w_i^+ = w_i + \sigma_i x_i \;\; \theta(\sigma_i \tau)\theta(\tau^A \tau^B)$$

Random walk:

$$w_i^+ = w_i + x_i \;\; \theta(\sigma_i \tau)\theta(\tau^A \tau^B)$$

In this case, only the hidden unit σ_i which is identical to τ changes its weights If this training step pushes any component W_i out of the interval [–L,+L] the component is replaced by +L or –L correspondingly.

After the full synchronization is achieved , the weights w_i of both tree parity machines are same) . These weights are used by networks A and B as keys.

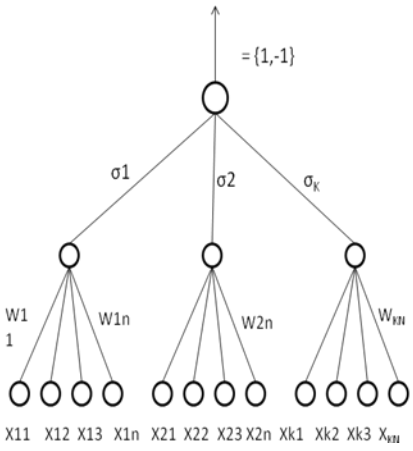

Figure 2: Tree Parity Machine

3.1 Feed Back Mechanism

Feedback mechanism has been designed to vary the length of the keystream generated as per user's desire up to a fixed number of iterations. Here the synchronized weights of the previous iteration would become the input vectors x_i which is defined as follows :

$$x_i = \begin{cases} -1, & w_i < 0 \\ 1, & w_i \geq 0 \end{cases}$$

New set of optimal are now generated as described in section 2. The TPM is synchronized for this new set of inputs and weights and a fresh set of synchronized weights is obtained which is appended to the previous set of synchronized weights to obtain a key of larger length. Thus, the process will terminate after generating the final key stream. This feedback process of generating the stream cipher of required length is depicted in Fig 3.

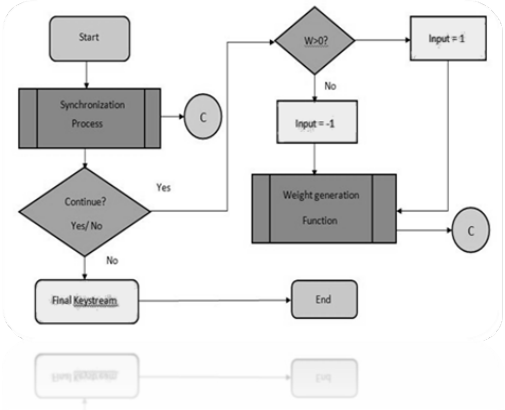

Figure3: Feed Back Mechanism

4. Security Attacks

A Secure Key exchange protocol should have the following property: Any attacker E who knows all of the details of the protocol and all of the information exchanged between A and B should not have the computational power to calculate the secret key. The main problem of the attacker E is that the internal representations of A's and B's Tree Parity Machines are not known .As the movement of the weights depends on σ_i, it is important for a successful attack to guess the state of the hidden units correctly. Of course, most of the known attacks use this approach. More successful is the Flipping Attack strategy[8], in which the attacker imitates one of the parties, but in the step in which his output disagrees with the imitated party's output, he negates ("flips") the sign of one of his hidden units. The unit most likely to be wrong is the one with the minimal absolute value of the local field therefore that is the unit which is flipped.

5. Results

Generation of key and keystreams using the GA algorithm are discussed now.

5.1. Generation of keys

The above algorithm was implemented for a random input vector x_i, of length 12, using random weights chosen with L varying from 3 to 6. An optimal set of weights is obtained from the GA process which leads to a synchronized set of weights. Table 1. Compares the average number of iterations to synchronize the weightssing genetic approach with the random weight approach. A considerable reduction in the number of iterations is observed .[10]

TABLE 1 : Average number of iterations with different weight range using Random and Genetic weights obtained over 100 samples.

No. Of Iterations		
For Weight range	*Using Random weights*	*Using Genetic weights*
3	350	150
4	547	263
5	923	300
6	1185	650

5.2 Generation of key stream

The above algorithm was implemented with the feed back mechanism to generate the key stream of desired length. The final key stream obtained is tested for randomness, which is a probabilistic property.
Once the keystream generated passes all these random tests, [11] it can be used for encryption purposes. If any of the tests fail, process has to be repeated, until a random sequence is obtained.

5.3. Security Analysis:

Here in this paper, it has been shown that flipping attack is not successful when the keystream is generated using a genetic algorithm. The result is shown in Fig 3. Here the attacker's success probability P is plotted as a function of L for the Flipping attack by using Random weights and Genetic weights. In case of weights generated from GA the probability of success becomes very small as L increases.

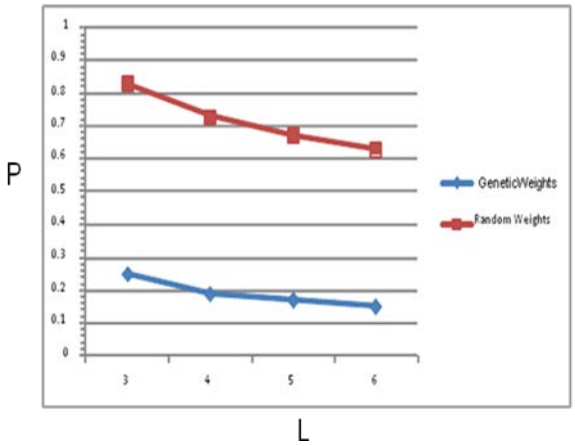

Figure 3. The attacker's success probability P as a function of L, for the Flipping attack using Random weights and Genetic weights averaged over 100 samples.

6. Conclusions

The TPM's were successfully synchronized and the Feedback mechanism generated the required length of key stream, as per user's desire within the limited iterations. Neural cryptography promises to revolutionize secure communication by providing security based on the fundamental laws of physics, instead of the current state of mathematical algorithms or computing technology. Therefore, genetic approach in neural cryptography for generating a key stream may lead to novel applications in the future.

REFERENCES

[1] Charles Pfleeger, Shari Lawrence Pfleeger, Security in computing, Third Edition 2003, pp 48, Prentice Hall of India Pvt Ltd, New Delhi.

[2] Biham E, Seberry, "A Fast and Secure Stream Cipher", EUROCRYPT' 05, Rump Session at the Symmetric the Key Encryption Workshop SKEW 2005), 26-27. May 2005.

[3] Sreelaja.N.K and G.A.Vijayalakshmi Pai," Swarm Intelligence based key generation for Text encryption in Cellular Networks". IEEE Proceedings the Third International Conference on System Software and Middleware and Workshops, 2008.COMSWARE 2008. 6-10. Jan. 2008. pp: 622 – 629.

[4] I.Kanter, W.Kinzel, E.Kanter, "Secure Exchange of Information by Synchronization of Neural Networks", Europhys Lett 57, 141-147 (2002).

[5] Pravin Revankar, Dilip Rathod , Neural Synchronization with queries, International conference on Signal Acquisition and Processing 2010.

[6] Goldberg, D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison- Wesley.

[7] Amit Konar." Computational Intelligence.Principles, Techniques and Applications,Springer-Verlag Berlin Heidelberg 2005

[8] E.Klein, R Mislovathy, I Kanter, A.Ruttor ,W.Kinzel, "Synchronization of Neural Networks by Mutual Learning and its Application to Cryptography", Advances in Neural Information Processing Systems, Volume 17, PP 689-696, MIT Press, Cambridge, MA,2005.

[9] Sreelaja.N.K, G.A.Vijayalakshmi Pai,"Design of Stream cipher for Text Encryption using Particle Swarm Optimization based Key Generation". Journal of Information Assurance and Security 4, 30-41(2009).

[10] S .Santhanalakshmi,TSB Sudarshan,G K Patra , " Neural Synchronization by Mutual Learning Using Genetic Approach for Secure Key Generation", Recent Trends In Computer Networks and Distributed System,CCIS Volume 335, PP 422-430, Springer-Verlag,2012.

[11] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, San Vo, A Statistical test suite for random and pseudorandom number generators for Cryptographic Applications.