Oriental Journal of Computer Science & Technology Vol. **2**(1), 11-17 (2009)

Approaches in RSA Cryptosystem using Artificial Neural Network

SIAMAK HAGHIPOUR and BABAK SOKOUTI*

Faculty of Engineering, Islamic Azad University - Tabriz Branch, Tabriz, (Iran).

(Received: March 01, 2009; Accepted: April 13, 2009)

ABSTRACT

RSA cryptosystem, the first public key cryptography presented in 1977. Neural networks are often used as a powerful discriminating estimator for tasks in function approximation. This paper describes a neural-network-based method relies on Radial Basis Function and Levenberg-Marquardt Backpropagation for estimating the behavior of the function used in RSA cryptosystem to calculate the function. The difficulty of the RSA cryptosystem relies on the difficulty of the factorization, to have the RSA cryptosystem broken, it suffices to factorize N which is the product of two prime numbers p, q ($N = p.q$). This will be equivalent to calculate the Euler function $\varphi(N) = (p-1)(q-1)$.

Keywords: Artificial Neural Network, Factorization, Radial Basis Function, Back Propagation, RSA Cryptosystem.

INTRODUCTION

The science of encrypting and decrypting messages by using mathematic to achieve information secrecy is Cryptography which enables us to store or transmit the sensitive information via an insecure channel (i.e. internet) that can not be read and understood by anyone except the intended recipient. A simple cryptosystem consists of a plaintext, cipher text, an encryption key, encryption algorithm, a decryption key and decryption algorithm as shown in Fig. 1.

There are two general forms of cryptosystems: symmetric (secret-key) and asymmetric (public-key) cryptosystems.

In the symmetric cryptosystem, the encryption and decryption keys are the same and the entity who generates the key must share the encryption key with the entity that will be able to decrypt the encrypted message; the security of the cryptosystem relies on keeping the encryption key secret. Suppose x as plain text, y as cipher text and k as the encrypting and decrypting key (a symmetric key), the sender will produce the cryptogram $y = E_k(x)$ that is transmitted through the insecure channel to the receiver that decrypts the message and deduce the original plaintext $x = D_k(y) = D_k (E_k(x))$. The disadvantage of this cryptosystem is the requirement for providing a secure channel for transmitting the shared key. In public-key cryptosystem, the encryption and decryption use different keys and there is no need for encryption key to be kept secret and the entity who will receive the cipher text is able to deduce the message by corresponding decryption key; there is no computational way of obtaining decryption key from the encryption key.

The RSA

The most popular public-key cryptosystem is the RSA, named for its creators: Rivest, Shamir and Adleman[1] proposed in 1977

that can be used for both encryption and digital signatures in the literature. Since then several public-key systems have been proposed[2,6] that their security relies on the hard computational problems[3,6]. The RSA cryptography is based on the notion of one-way trapdoor function[2,7], gets its security from the difficulty of factoring integer numbers. For implementing an RSA cryptosystem, the generation of two keys are needed; first of all two large prime numbers p, q are chosen[8, 9]. (This can be done by Monte-Carlo prime number finding algorithm), then their product $N = p.q$ and the Euler function $\varphi(N) = (p-1)(q-1)$ will be computed. The next step is to find e and d. A random integer number e< $\varphi(N)$ is chosen such that gcd (e, $\varphi(N)$)=1, then the Extended Euclid Algorithm is used to find the integer decryption key d from $e.d \equiv 1 \bmod(\varphi(N))$. For the encryption process the sender will encrypt the confidential message with the public key e in the following form of a modular exponentiation:

$$C \leftarrow M^e \bmod(N)$$

And for the decryption process the receiver will decrypt the encrypted message by the decryption key d again in the form of a modular exponentiation:

$$M \leftarrow C^d \bmod(N) \equiv (M^e)^d \bmod(N) \equiv M \bmod(N)$$

Any adversary knows the public keys pairs (N, e) and the structure of the system and is familiar with a modular exponentiation equation. The most well known attacks against the RSA cryptosystem is to factor N to its prime integer numbers that computationally is impossible; it is also possible to attack the RSA by guessing the value of Euler

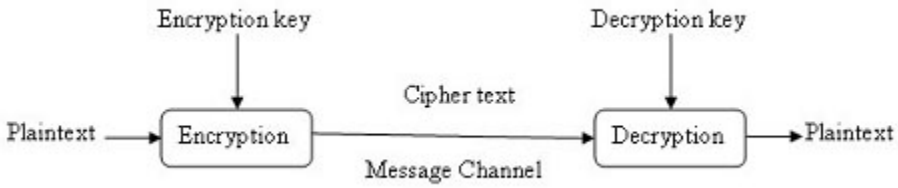

Fig. 1: A simple cryptosystem diagram

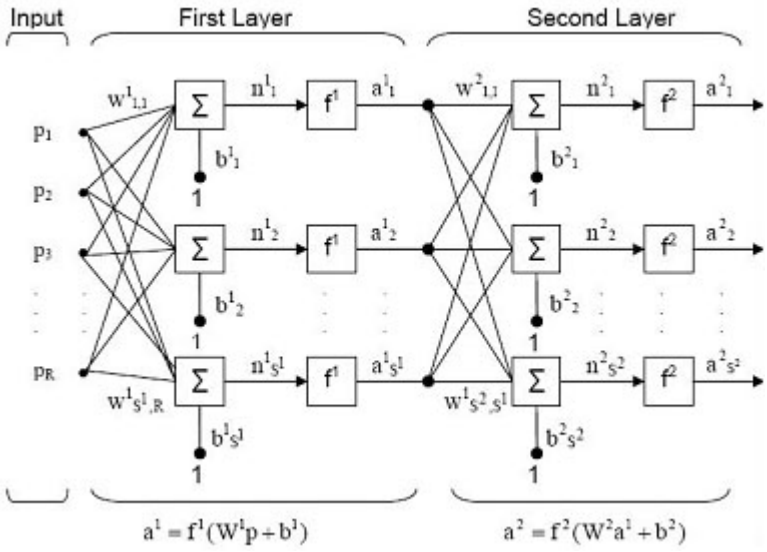

Fig. 2: Feedforward Neural Network Scheme

function $\varphi(N)=(p-1)(q-1)$ that is not easier than factoring N[10,11] Though it is also possible for the adversary to use the Brute Force attack, means to try all the decryption keys which are less than $\varphi(N)$ to obtain the intended message, it is less efficient than factoring modulus N. Neural networks are used to determine the Euler function $\varphi(N)=(p-1)(q-1)$ from inputting $N=p.q$ where p and q are prime numbers and the neural networks output will be $\varphi(N)=(p-1)(q-1)$.

Neural Network Techniques

The RBF Algorithm

The Radial Basis Function Networks are similar to the biological networks behavior; primarily the hidden layers contain almost sensing units and the output layer contains linear units. Usually for a common RBF network the transfer function in the hidden layer is a Gaussian function that is introduced in equation (1) as below:

$$\varphi(x) = \exp\left(-\left(\frac{|x-\mu_i|}{h}\right)^2\right) \quad ...(1)$$

where as, $\mu_i \in R^d$ is called the center vector, and $h \in R$ is called the kernel width (or smoothing parameter). The basic RBF network provides a nonlinear transformation of a pattern $x \in R^d \rightarrow R^c$ according to equation (2):

$$f_j(x) = b_j + \sum_{i=1}^{m} w_{ji}\varphi\left(\frac{|x-\mu_i|}{h}\right) \quad ...(2)$$

where m is the number of basis functions, wji is a weight, bj is a bias[12,15].

The Levenberg-Marquardt (LM) Backpropagation Algorithm

A variation of the standard backpropagation algorithm is Levenberg-Marquardt that is used for training the multilayer networks and is preferably used over the standard backpropagation; this algorithm is the fastest for multilayer neural networks[16,17]. When the network is a single layer one it is easy to find the optimum solution in selecting the learning rate but it has more than one hidden layer, it gets complex. This needs for the learning rate being able of changed between the steepest descent and the Newton algorithm depending on weather the value of the function is near the optimum solution or not; The Levenberg-Marquardt algorithm has this property. The disadvantage of the Levenberg-Marquardt over the standard backpropagation is the storage requirement for approximate Hessian matrix ($r_x r$ matrix) where r is the number of parameters (weights and biases) in the network. A Feedforward Multilayer Neural Network Scheme is presented in Fig. 2. Suppose that pattern inputs (p_q) and corresponding target outputs (t_q) are given as $\{p_1,t_1\},\{p_2,t_2\},\ldots,\{p_Q,t_Q\}$ the iteration of the Levenberg-Marquardt algorithm will be as follows:

1. All inputs are given to the network and the corresponding network outputs and error will be computed:

$a^0=p_1$ (3)

$a^{m+1}=f^{m+1}(W^{m+1}a^m+b^{m+1})$; m=0,2,…,M-1 (4)

$a=a^M$ (5)

Where a^0 is the output for input layer p_1 , a^{m+1} is the output for layer m+_1, f^{m+1} the transfer function for layer m+1, W^{m+1} is the weights matrix

Table 1: Results for networks trained with RBF algorithm and data set $N = p.q \leq 1000$

Topology	Epochs	Correlation rate
1-5-1	5	0.928822
1-10-1	10	0.948572
1-20-1	20	0.986274
1-30-1	30	0.997146

Table 2: Results for networks trained with RBF algorithm and data set $N = p.q \leq 1000$

Topology	Epochs	Correlation rate
1-50-1	50	0.92706
1-60-1	60	0.930847
1-100-1	100	0.943982
1-200-1	200	0.98831

and b^{m+1} is the bias vector, a^M is the last layer's output.

2. The squared errors over all inputs will be computed from the following equation:

$$F(x) = \sum_{q=1}^{Q} (t_q - a_q)^T (t_q - a_q) \quad ...(6)$$

3. Jacobean matrix (J(x)) and sensitivities (S^{-M}) with the recurrence relations will be computed as follows:

$$J(x) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_{1,1}^1}, \frac{\partial e_{1,1}}{\partial w_{1,2}^1}, \ldots, \frac{\partial e_{1,1}}{\partial w_{S^1,R}^1}, \frac{\partial e_{1,1}}{\partial b_1^1}, \ldots \\ \frac{\partial e_{2,1}}{\partial w_{1,1}^1}, \frac{\partial e_{2,1}}{\partial w_{1,2}^1}, \ldots, \frac{\partial e_{2,1}}{\partial w_{S^1,R}^1}, \frac{\partial e_{2,1}}{\partial b_1^1}, \ldots \\ \vdots, \\ \frac{\partial e_{S^M,1}}{\partial w_{1,1}^1}, \frac{\partial e_{S^M,1}}{\partial w_{1,2}^1}, \ldots, \frac{\partial e_{S^M,1}}{\partial w_{S^1,R}^1}, \frac{\partial e_{S^M,1}}{\partial b_1^1}, \ldots \\ \frac{\partial e_{1,2}}{\partial w_{1,1}^1}, \frac{\partial e_{1,2}}{\partial w_{1,2}^1}, \ldots, \frac{\partial e_{1,2}}{\partial w_{S^1,R}^1}, \frac{\partial e_{1,2}}{\partial b_1^1}, \ldots \\ \vdots \end{bmatrix} \quad ...(7)$$

$$S_q^{-M} = -F^M(n_q^M) \quad ...(8)$$

$$S_q^{-M} = -F^M(n_q^M)(W^{m+1})^T S^{-m+1} \quad ...(9)$$

$$S^{-m} = \left| S_1^{-m} \mid S_2^{-m} \mid \ldots \mid S_Q^{-m} \right| \quad ...(10)$$

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial w_{i,j}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} * \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = s_{i,h}^{-m} * \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = s_{i,h}^{-m} * a_{j,q}^{m-1}$$

...(11)

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial b_i^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} * \frac{\partial n_{i,q}^m}{\partial b_i^m} = s_{i,h}^{-m} * \frac{\partial n_{i,q}^m}{\partial b_i^m} = s_{i,h}^{-m}$$

...(12)

where S^{-m} is the Marquardt sensitivities matrix and $[J]_{h,l}$ is the element of the Jacobean matrix that corresponds to the h^{th} row and l^{th} column.

4. Δx_k will be obtained from the below equation:

$$\Delta x_k = -[J^T(x_k)J(x_k) + \mu_k I]^{-1} J^T(x_k) v(x_k) \quad ...(13)$$

5. $x_k + \Delta x_k$ will be recalculated to obtain the sum of squared errors, if it is smaller than the on in step 2 then divide μ(i.e. 0.01) by ϑ (larger than 1, i.e. 10) then let $x_{k+1} = x_k + \Delta x_k$ and start from step 1; otherwise if the sum is not reduced then multiply μ(i.e. 0.01) by ϑ (larger than 1, i.e. 10) and start from step 4.

6. This process will be continued until the difference between the network response and the target reduced to the error goal that means the algorithm has been converged.

EXPERIMENTAL RESULTS

The experimental results are made by using Neural Networks in MATLAB interface in Windows XP operating system SP2. The Neural Networks used are:

- Radial Basis Function Neural Network (RBF).[18]
- Levenberg- Marquardt Backpropagation Neural Network.[16]
- Standard Backpropagation Neural Network. [19]

All the above methods were tested and their correlation rates for their performance were calculated wit ha large range of parameter. How ever the results showed some significant differences between the Neural Network models. The standard Backpropagation has big difficulties in the training stage most of the times; Levenberg-Marquardt Backpropagation managed to be the fastest RBF Neural Network is good in function approximation but it has the limitations of a one-hidden layer Neural Networks.

According to the network structure, choosing optimal network architecture for the related problem is very difficult and is a problem in recent years.

Table 3: Results for networks trained with RBF algorithm and data set $N = p.q \leq 2000$

Topology	Epochs	Correlation rate
1-50-1	50	0.870128
1-60-1	60	0.887932
1-100-1	100	0.921691
1-200-1	200	0.955836

Table 4: Results for networks trained with Levenberg-Marquardt Backpropagation algorithm and data set $N = p.q \leq 2000$

Topology	Epochs	Correlation rate
1-10-1	1000	0.972984
1-10-10-1	1000	0.999718
1-20-20-1	1000	0.999998

Table 5: Results for networks trained with Levenberg-Marquardt Backpropagation algorithm and data set $N = p.q \leq 1000$

Topology	Epochs	Correlation rate
1-10-1	1000	0.919369
1-10-10-1	1000	0.950031
1-20-20-1	1000	0.960536

Table 6: Results for networks trained with Levenberg-Marquardt Backpropagation algorithm and data set $N = p.q \leq 1000$

Topology	Epochs	Correlation rate
1-10-1	1000	0.916092
1-10-10-1	1000	0.928758
1-20-20-1	1000	0.941809

Table 7: Results for networks trained with Levenberg-Marquardt Backpropagation algorithm and RBF algorithm and 66% of data set $N = p.q \leq 1000$

Algorithm	Topology	Epochs	Correlation rate	Type
RBF	1-200-1	200	1	Train set
RBF	1-200-1	200	0.032932	Test set
LM BP	1-20-20-1	1000	0.982042	Train set
LM BP	1-20-20-1	1000	0.546685	Test set

Table 8: Results for networks trained with Levenberg-Marquardt Backpropagation algorithm and RBF algorithm and 66% of data set $N = p.q \leq 2000$

Algorithm	Topology	Epochs	Correlation rate	Type
RBF	1-200-1	200	0.97881	Train set
RBF	1-200-1	200	-0.12899	Test set
LM BP	1-20-20-1	1000	0.961804	Train set
LM BP	1-20-20-1	1000	0.724941	Test set

A large variety of network topologies were used in t his approach. By using different network topologies, the obtained results were considerable; and after extensive testing, a conclusion can be made that 2 hidden layers networks were much easier in train and much accurate in performance. Here the input patterns were and the target outputs were $\varphi(N) = (p-1)(q-1)$ in which the network tries to learn the function that transfers N to $\varphi(N)$ (i.e. $N = pq \rightarrow \varphi(N) = (p-1)(q-1)$) and predicts the output $\varphi(N)$ by taking the input N to overcome the RSA problem.

To test the network performance here, the correlation rate value is considered as where the value is near the integer number 1 or -1 the there is a good prediction of the output and where the value is near zero there is not a good prediction in that network. It is important that the good correlation rate value plays an outstanding role as the computation of the function can be solved by a second degree polynomial equation since $\varphi(N)$ =N-p-q+1. In this test the limited number of data set ($N = pq \leq 100$, $N = pq \leq 1000$, $N = pq \leq 2000$) is considered to find out the related neural networks' performance.

In table 1 we can see the results of the network trained by RBF algorithm with a data set $N = pq \leq 100$. In Table 2, 3 the larger prime numbers are tried with a significant change in the network topology. These test are also performed for the Levenberg-Marquardt Backpropagation that can be seen in Table 4,5,6 that can be seen that a 2 hidden layer performance in prediction is better that the one hidden layer network. Finally, network performance is done by the data set 33% for Test set and 66% of the data set as Train set, gives us the results exhibited in Table 7, 8. As it is clear the Levenberg-Marquardt Backpropagation Network is able not only to adapt to train data but to reach good results to test sets in comparison with the RBF Neural Network

CONCLUSION

In this paper, the approaches of one-hidden layer (RBF) and 2-hidden layer (Levenberg-Marquardt Backpropagation) Neural network against the RSA cryptosystem were employed and from the illustrated results it is clear that though the RB neural network is very good at function estimation, in the RSA problem its usage is not reached in good results. Additionally the 2-hidden layer neural network can be employed for this purpose to reach better results according to the network topology and the training of Levenberg-Marquardt algorithm parameters. In general, the RSA problem could be solved if the correlation rate value reaches the number 1. There is still a long way to employ the real power of the neural networks.

In conclusion, the approaches of neural networks in the RSA problem are promising though many works is needed to be done.

REFERENCES

1. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM,* **21**(2):120–126, (1978).
2. W. Diffie and M. Hellman, New Directions in Cryptography, *IEEE Trans. I.T.*, **22,** 644-654 (1976).
3. T. El Gamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Trans. I.T.*, **31,** 469–472 (1985).
4. N. Koblitz, Elliptic Curve Cryptosystems, *Math. Comp.*, **48,** 203–209 (1997).
5. R.C. Merkle and M.E. Hellman, Hiding Information and Signatures in Trapdoor Knapsacks, *IEEE Trans. I.T.*, **24,** 525–530 (1978).
6. S.C. Pohlig and M. Hellman, An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance,

IEEE Trans. I.T., **24,** 106–110 (1978).

7. W. Diffie and M. Hellman. Multiuser cryptographic techniques. In Proceedings of AFIPS 1976 NCC, pages 109–112. AFIPS Press, Montvale, N.J., (1976).
8. R. Solovay and V. Strassen, A Fast Monte Carlo Test for Primality, *SIAM J. Comput.*, **6**(1), 84–85 (1977).
9. A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC, (1996).
10. C.K. Wu and X.M. Wang, "Determination of the True Value of the Euler Totient Function in the RSA Cryptosystem from a Set of Possibilities," *Electronics Letters*, **29**(1), 84–85 (1993).
11. G.L Miller, Rieman's Hypothesis and Tests for Primality, *Journal of Computer and System Sciences*, **13,** 300–317 (1976).
12. Cybenko, G., Approximation by super-positions of a sigmoidal function. *Math. Controls, Signals, Syst.*, **2,** 303-314 (1992).
13. Hartman, K., Keeler, J. D. and Kowalski, J. M., Layered Neural Networks with Gaussian hidden units as universal approximation. *Neural Computation*, **2,** 210-215 (1990).
14. Hornik, K., Sitnchcombe, M. and White, H., Multilayer feedforward networks are universal approximators. *Neural Networks*, **2,** 359-366 (1989).
15. Schalkoff Robertj, Artificial Neural Networks, Mcgraw Hall, (1997).
16. Hagan, M., and M. Menhaj,: Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks,* **5**(6):989-993, (1994).
17. Hagan, M., H. Demuth and M. Beale,: Neural network design. PWS Publishing Co., Boston, (1996).
18. S. Haykin, *Neural Networks*, (New York: Macmillan College Publishing Company, (1999).
19. D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning Internal Representations by Error Propagation, In D.E. Rumelhart and J.L. McClelland (Eds.) *Parallel distributed processing: Explorations in the microstructure of cognition*, Cambridge, MA, MIT Press, 1, 318–362 (1986).