Improving Pseudorandom Bit Sequence Generation and Evaluation for Secure Internet Communications Using Neural Network Techniques

D.A. Karras¹ and V. Zorkadis²

¹Hellenic Aerospace Industry, University of Hertfordshire (UK) and Hellenic Open University, Rodu2, Ano Iliupolis, Athens 16342, Greece, e-mails: dakarras@hol.gr, dkarras@haicorp.com, dakarras@usa.net
²Data Protection Authority, Omirou 8, 10564 Athens, Greece, e-mail: zorkadis@dpa.gr

Abstract. Random components play an especially important role in secure electronic commerce and Internet communications. For this reason, the existence of strong pseudo random number generators is highly required. This paper presents novel techniques, which rely on artificial neural network architectures, to strengthen traditional generators such as ANSI X9 based on DES and IDEA. Additionally, this paper proposes a test method for evaluating the required non-predictability property, which also relies on neural networks. This non-predictability test method along with commonly used statistical and non-linearity tests are proposed as methodology for the evaluation of strong pseudo random number generators. By means of this methodology, traditional and proposed generators are evaluated. The results show that the proposed generators behave significantly better than the traditional, in particular, in terms of non-predictability.

I. INTRODUCTION

Cryptographic protocols for electronic payment systems, authentication, integrity, confidentiality, non-repudiation or key management may have random components, which require methods to obtaining numbers that are random in some sense. For instance, authentication mechanisms in electronic commerce applications may use nonces, i.e., random numbers to protect against replay attacks [1] like the corresponding mechanisms in ITU X.509 [2]. Symmetric and asymmetric cryptographic systems like DES [3], IDEA, RSA [4] that are employed for confidentiality purposes as well as a basic element of other security protocols in electronic commerce require random cryptographic keys, should the cryptoanalysis remain a hard problem.

Furthermore, integrity mechanisms such as ISO 8731-2 [5] or cryptographic key exchange mechanisms such as the Diffie-Hellman Protocol [6] or the construction of digital signatures like the ElGamal or Digital Signature Scheme (DSS) [4] need the generation and use of random numbers. In addition, random numbers are used for the generation of pseudonyms and of traffic and message padding, in order to protect against traffic analysis attacks and for the computation of strong and efficient stream ciphers [4].

Random bit sequences of good quality, i.e., of good behavior in statistical and non-predictability terms, are desired. Otherwise it would be possible for a cryptoanalyst, given a segment of this bit sequence and reasonable computer resources, to calculate the next bits or more about them [7]. In the last two decades considerable work has been made in the design and analysis of pseudo random number or bit generators [7-16]. In this paper, we briefly survey some of these generators, propose a methodology to strengthen them and to evaluate their behavior and strength.

The level of randomness of a sequence can be defined in terms of statistical tests, which emulate computations encountered in practice, and check that the related properties of the sequence under investigation agree with those predicted if every bit (or number) was drawn from a uniform probability distribution [13]. The generators we consider in this paper are those used by the system-theoretic approach to the construction of stream ciphers. Secure keystream generators have to satisfy design criteria, such as long period, ideal k-tuple distributions, large linear complexity, confusion, diffusion and nonlinearity criteria [13]. Most of them are contained in the proposed evaluation methodology.

In the next section a method is outlined to strengthen these generators by means of neural network based mechanisms. The third section is dedicated to the proposed evaluation methodology for random number generators, namely to the non-predictability test and to the appropriate statistical and non-linearity tests. In section IV, we present evaluation results obtained by applying the proposed methodology on various traditional and strengthened generators. Finally, we conclude our paper and outline future work on this subject.

II. STRONG PSEUDORANDOM NUMBER GENERATORS

II.1 Traditional Generators

The great majority of random number generators used for traditional applications such as simulations are linear congruential generators, which behave statistically very well, except in terms of non-predictability, since there exists a linear functional relation connecting the numbers of the sequence. A sequence of random numbers produced by these generators is defined as follows: \[ Z_i = (aZ_{i-1} + c) \mod m \], where \( m \), \( a \) and \( c \) are the coefficients, i.e., the modulus, the multiplier and the increment, correspondingly. \( Z_0 \) is the seed or initialization value. All are nonnegative integers. Such generators are inappropriate for security mechanisms, since the disclosure of one of them could very easily lead to the computation of the others.

In random components of secure electronic commerce systems like electronic payment systems, authentication and key generation and exchange the primary concern of the used...
The above desired abilities, however, are acquired in MLPs by training them well with known input vectors provided overfitting has not occurred [17]. In the case of overfitting it is well known [17] that the network, on the one hand, learns very well the training samples but, on the other hand, it is unable to generalize when fed with unknown input patterns. This happens, because the network draws the fitting surface of the training samples in a much more complex way, that is its fitting surface is of much higher degree, than needed to map the actual pattern population distribution. In such a case MLP response is not predictable since there is no analytic formula for describing the previously mentioned complex fitting surface. Even for unknown data with small distances from the training samples (similar inputs) network outputs will be very different from the ones obtained with the corresponding training data. Although in pattern recognition/control/prediction applications such a situation is highly undesirable, it could be exploited, however, in the case of random number generation. We demonstrate, in this paper, that overfitting in MLPs could be exploited as a mechanism for the generation of strong pseudorandom bit sequences as follows

1. Train an MLP of topology e.g. 4-6-6-1 in the parity-4 problem so as to learn it more than it is required, i.e., for instance for 2500 epochs even when 500 epochs are enough. The goal is that overfitting should occur in this MLP training process. Of course other binary such benchmarks as well as much more complex MLP topologies could have been involved.

2. Test the already trained MLP using test input vectors with components produced by a traditional (pseudo)random number generator, like the ones involved in the experimental section of the paper, whose values are in the interval [0,1].

3. Form a complex function of the internal representations of such an MLP and compute its value when a test input vector as previously defined is presented to the network.

In this way a sequence of (pseudo)random numbers is produced whose quality is quantitatively evaluated by utilizing the statistical tests presented in the next section. The complex function of MLP internal representations used in the present paper has the following analytic formula.

\[ y = \text{modf} \left( 1000 \cdot \sum \left| \frac{\tanh(o_k) - \tanh(o_{k+1})}{\tanh(o_{k+1}) - \tanh(o_{k+2})} \right| \right) \]

Where \( y \) is the random number obtained and 

\[ o_k, o_{k+1}, o_{k+2} \]

are the activations of the processing elements of the MLP layer preceding its output one. \text{modf} is a Unix-function that extracts the fractional part of a real number.
The previous discussion determines all the steps of the approach adopted here for designing strong (pseudo)random bit sequences generators employing the overfitting MLP recall properties.

Apart from MLPs we consider how Hopfield type neural networks could produce strong pseudo-random numbers for electronic commerce systems. The methodology for transforming Hopfield type recurrent ANNs into strong (pseudo)random number generators is herein depicted by exploiting their properties to minimize a cost function involving their weights and neuron activations under certain conditions concerning their weight matrix [17]. More specifically, a Hopfield network possesses the following important characteristics [17], which are next summarized.

a) If the weight matrix of a Hopfield recurrent ANN is symmetric with zero valued diagonals and furthermore, only one neuron is activated per iteration of the recurrent recall scheme then, there exists a Liapunov type cost function involving its weights and neuron activations, which decreases after each iteration until a local optimum of this objective function is found.

b) The final output vector of the Hopfield network, after the convergence of the above mentioned recurrent recall scheme, has minimum distance or is exactly equal to one prototype stored in the network during its weight matrix definition (learning phase) provided that the prototypes stored are orthogonal to one another and their number $M \leq 0.15 N$, where N is the number of neurons in the network.

c) If the prototypes stored in the Hopfield ANN are not orthogonal or their number $M > 0.15 N$ then, the recurrent recall scheme converges to a linear combination of the prototypes stored when it is fed with a variation of one of these prototype vectors, provided that the weight matrix has the properties discussed in (a) above.

d) Hopfield net outputs are given by the following formula, which is precisely the update formula for the single neuron activated during the iterations of the recurrent recall scheme mentioned in (a) above.

$$O_k = \tanh (\sum W_{ki} O_i)$$

The $g = \tanh ()$ nonlinearity is considered in the following.

These properties lead us intuitively to the principles of the proposed random number generation methodology involving such recurrent ANNs, summarized as follows.

1) If we impose a perturbation to the recurrent network weight matrix so that its symmetry is broken and its diagonal units obtain large positive values then, the convergence property of the recurrent recall scheme will be lost. This can be achieved, for instance, by adding a positive parameter $\delta$ to every unit in the upper triangle of the matrix, including diagonal units, and subtracting the negative quantity $-\delta$ from every unit in the lower triangle of the matrix.

2) Moreover, if we let a large number of neurons (in our experiments $N/2$ neurons) update their activations by following the formula of (d) above, then, the recurrent recall scheme will lose its convergence property to a local optimum of the suitable Liapunov function associated to the network.

3) If the recurrent recall scheme is not guaranteed to converge to a network output that corresponds to the local optimum of a cost function then, the behavior of the network becomes unpredictable.

4) If the network is large and the patterns stored in it are orthogonal and thus, uncorrelated (that is, they have maximum distances from one another) then, the possibility of obtaining predictable outputs after several iterations of the recurrent recall scheme is minimum compared to the one associated with storing non-orthogonal prototypes, which are correlated to one another. In our experiments we use binary valued orthogonal patterns.

5) If the history of the network outputs during its recall phase is considered for $T$ iterations of the recurrent recall scheme then, predicting the sequence of these output vectors is much harder than trying to predict a single output vector.

The above principles lead us to use the following function of network outputs over $T$ iterations of the recurrent recall scheme as a pseudorandom number generator. To obtain better quality pseudorandom numbers, we have considered the Unix-function $\text{modf}$, which eliminates the non-integral part of a real number, as the required mechanism for aiding Hopfield net output to acquire the desired properties, since the first digits of its decimal part are predictable, due to the fact that the sigmoidal nonlinearity $g$ is a mapping on the $(0,1)$ interval. Consequently, the formula of the Hopfield recurrent ANN proposed random number generator is as follows.

$$O = \text{modf}(1000^*(1/TN) \sum_{t=1}^{T} \sum_{i=1}^{N} (g(\sum W_{ki} O_i(t)))^2)$$

The previous discussion determines all the steps of the approach adopted here for designing strong (pseudo)random bit sequences generators employing the recurrent recall scheme of Hopfield networks.

Additionally, the above two presented neural network based methodologies for constructing strong pseudo-random bit sequences could be involved in strengthening traditional generators as follows. The initial weights of MLP are drawn from the traditional random number generator produced bit sequence and then, the MLP is employed as previously described as a random number generator mechanism. Thus, a two-stage generator is created, which, as illustrated in section 4 enhances the quality characteristics of the
corresponding traditional one. On the other hand, Hopfield recurrent ANN initial input values are drawn in the same way from the traditional random number generator produced bit sequence and then, this Hopfield net is involved as a random bit generator. Thus, again, a two-stage generator with enhanced properties is produced, which is evaluated in section 4.

III. EVALUATION METHODOLOGY FOR RANDOM NUMBER GENERATORS IN COMMUNICATION SYSTEMS

Two criteria are used for the evaluation of the quality of random numbers obtained by using some generator in traditional applications such as simulation studies: uniform distribution and independence. The most important requirement imposed on random number generators is their capability to produce random numbers uniformly distributed in [0,1]; otherwise the application’s results may be completely invalid. The independence requires that the numbers should not exhibit any correlation with each other. Additionally, random number generators should possess further properties: to be fast in computing the random numbers, to have the possibility to reproduce a given sequence of random numbers and to be able of producing several separate sequences of random numbers. However, for random number generators involved in the implementation of security mechanisms such as authentication, key generation and exchange in electronic commerce systems the most important property might be to produce unpredictable numbers. True random numbers possess this property. It is well known that pseudorandom number generators that are used for simulations such as the linear congruential generators have not this property since each number they produce can be expressed as a function of the initialization value or of its predecessor value and the coefficients of the generator.

III.1 The Non-predictability Test

In addition to these traditional tests we introduce a predictability test for random bit sequences based on the MLP capabilities to approximate functions without any kind of assumption about their model, either linear or nonlinear ([17-19]). To this end, if we consider a random bit sequence as a time series, then, by scanning it with a sliding window of length M we could form from it a series of patterns suitable for defining a training task for an MLP. Thus, M such samples comprise its inputs while their corresponding next one comprises its desired output. The training task for such an MLP is to perfectly learn these predictability patterns. If the random bit sequence has N samples then, there exist N-M such patterns. The rationale underlying the suggested test is that if such a task is learnable then, obviously, there exists possibility that future numbers of the sequence under consideration can be inferred from their present and past values in the sequence. Therefore, by applying the above discussed learning task to an MLP and estimating the corresponding Minimum Average Sum of Squared Errors (SSE) per pattern, during the whole MLP training session, we could have a view of how difficult is to predict the given random bit sequence. This SSE based measure of non-predictability, however, provides a hint for the average performance of the generator only. There might exist portions of the sequence that could be more predictable than others. A measure, suitable to account for such a fact, is the Maximum Approximation Probability (MAP), which counts the maximum number of correctly predicted patterns, with respect to a predefined approximation error ε per pattern, within the total number of patterns. It is obtained during the above specified MLP training session. Therefore,

MAP = max [predicted patterns (with SSE < ε)] / [total number of patterns (N-M)] during the whole MLP training session. In our simulations the quantities above described take on the values ε = 0.001, N=5000 and M=2 respectively.

III.2 Statistical and Non-linearity Tests

Statistical tests are applied to examine whether the pseudorandom number sequences are sufficiently random [11]. In the following we shortly discuss the empirical tests we use to evaluate the quality of the pseudorandom numbers obtained by the generators involved in this paper, i.e., how well they resemble true random numbers.

The first empirical test we apply is the most basic technique in the suite of the methods used for evaluating pseudorandom numbers quality, namely, the chi-square test ($x^2$ test). According to this method the interval $[0,1]$ is divided into $k$ subintervals of equal length. The $k$ should be at least 100, and $n_k$ should be at least 5, where $n$ is the length of the sequence [21]. In our examples $n = 5000$ and $k = 101$. We build $x^2 = \sum_{i=1}^{k} \frac{(n_i - \bar{n})^2}{\bar{n}}$, where $n_i$ is the number of random quantities that fall in the $j$th subinterval. For large $n$, $X^2$ will have an approximate chi-square distribution with $k-1 \text{ df}$ (degrees of freedom) under the null hypothesis that the obtained pseudorandom numbers are identically, independently and uniformly distributed in the interval $[0,1]$ [20, 21]. We reject this hypothesis at level of confidence $\beta$ if $x^2 > x^2_{\beta-1,k-1}$, where $x^2_{\alpha-1,k-1}$ is the upper $1-\alpha$ critical point of the chi-square distribution with $k-1 \text{ df}$ [20, 21]. In this paper, we make use of an approximate value of the
suggested in [20,21], where \( z_{1-s} \) is the upper \( 1-s \) critical point of the \( N(0,1) \) distribution.

The second empirical test we use is the run test. The run tests look for independence and therefore, as discussed in the introduction of this work, are the most important tests for electronic commerce applications. With these tests we examine the length of monotone (increasing) portions of the pseudorandom number sequence. The test statistic is

\[
V = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} (R_i - NB_i) (R_j - NB_j),
\]

where \( N \) is the sequence length, \( R_i \) is the number of monotone portions of length \( i \) with \( i < 6 \) and \( R_6 \) is the number of the rest portions of length \( > 5 \). The matrices of coefficients \( A_{ij} \) and \( B_i \) are approximate values taken from [20]. The statistic \( V \) should have the chi-square distribution with six degrees of freedom, when \( N \) is large, i.e., greater than 4000. If \( V \) is greater than the critical point of the \( x^2 \) distribution with 6 df at confidence level \( 1 - \alpha \), we reject the hypothesis of independence.

Finally, concerning the classical empirical tests, the sample means and variances of the pseudorandom number sequences obtained by the generators herein employed have been computed and compared with their expected values associated to the uniform distribution in the range \([0,1)\), i.e. 0.5 and \( (1/12) \), respectively.

**IV. EVALUATION AND DISCUSSION**

An experimental study has been carried out in order to demonstrate the efficiency of the suggested, in section 2, procedures for designing pseudorandom number generators. The following experiments have been conducted by applying the empirical tests depicted in section II, on

1. A random sequence produced by the IDEA algorithm and another one produced by the ANSI-X.9 based on the IDEA algorithm
2. A random sequence produced by the Hopfield recurrent ANN using the methodology described in section II.
3. A random sequence produced by an overfitting MLP based pseudorandom number generator, whose initial weights and all its inputs have been computed from a random sequence resulted by running a linear congruential number generator.
4. Two two-stage generators involving in their second stage an MLP generator while, at their first stage, this MLP’s weights are initialized from: (a) the IDEA sequence of (1) above, (b) the ANSI-X.9 sequence of (1) above.

The Hopfield ANN herein employed has \( N = 100 \) neurons connected following the conventional feedback architecture. All the sequences herein produced and compared have 5000 points. All the results obtained from the above specified experiments concerning the statistical and the non-predictability tests are presented in Tables 1 and 2, respectively. From these tables we can derive the following:

1. Indeed, it is possible to obtain strong pseudorandom numbers using the complex recurrent recall scheme of Hopfield type ANNs.
2. It is possible to obtain strong pseudorandom numbers using the complex mapping properties of overfitting feedforward ANN of the MLP type to respond unpredictably in unknown input vectors even when they are similar to the training patterns.
3. Additionally, the proposed pseudo-RNG can enhance the quality of random numbers obtained by traditional generators when they are involved in a two stage procedure.
4. These pseudorandom numbers are of good quality, passing several critical evaluation tests.
5. Although it has been demonstrated that there exists at least one function of MLP weights, or Hopfield network parameters, which can be designed to have the required properties, it is not obvious how to construct it. Building such functions relies on intuition and experimentation.
6. The proposed nonlinear non-predictability test can reveal hints when analyzing a given pseudorandom number sequence. To be more specific, it can provide indications about average and maximum predictability properties of such a sequence.

<table>
<thead>
<tr>
<th>Generator</th>
<th>( \chi^2 )_test</th>
<th>Run_test</th>
<th>Sample mean</th>
<th>Sample Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEA</td>
<td>109.186</td>
<td>2.464</td>
<td>0.503</td>
<td>0.0836</td>
</tr>
<tr>
<td>ANSI-X.9</td>
<td>98.26</td>
<td>4.861</td>
<td>0.500</td>
<td>0.0828</td>
</tr>
<tr>
<td>Hopfield-recurrent ANN</td>
<td>79.837</td>
<td>3.745</td>
<td>0.498</td>
<td>0.0835</td>
</tr>
<tr>
<td>MLP (4-20-20-1) Off-line BP</td>
<td>57.874</td>
<td>0.268</td>
<td>0.500</td>
<td>0.0832</td>
</tr>
<tr>
<td>MLP (4-6-6-1) Off-line BP, parity-4 (n=1.0)</td>
<td>80.989</td>
<td>1.323</td>
<td>0.498</td>
<td>0.0819</td>
</tr>
<tr>
<td>IDEA-MLP (4-6-6-1) Off-line BP, parity-4</td>
<td>92.63</td>
<td>1.351</td>
<td>0.502</td>
<td>0.0830</td>
</tr>
</tbody>
</table>

1371
Table 1. The classical empirical test results

<table>
<thead>
<tr>
<th>Generator</th>
<th>Minimum SSE per pattern (= total Min SSE/ number of patterns=4998) in a 2-35-35-1 MLP, Online BP (n=0.2, η=0.3)</th>
<th>MAP (ε=0.001)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEA</td>
<td>420.72/4998</td>
<td>66.07%0</td>
</tr>
<tr>
<td>ANSI-X.9</td>
<td>417.44/4998</td>
<td>53.82%0</td>
</tr>
<tr>
<td>Hopfield- recurrent ANN</td>
<td>425.59/4998</td>
<td>53.04%0</td>
</tr>
<tr>
<td>MLP (4-20-20-1) Off-line BP</td>
<td>422.74/4998</td>
<td>51.78%0</td>
</tr>
<tr>
<td>MLP (4-6-6-1) Off-line BP, parity-4 (n=1.0)</td>
<td>422.20/4998</td>
<td>51.50%0</td>
</tr>
<tr>
<td>IDEA-MLP (4-6-6-1) Off-line BP, parity-4 (n=0.4)</td>
<td>426.38/4998</td>
<td>45.02%0</td>
</tr>
<tr>
<td>ANSI-X.9-MLP (4-6-6-1) Off-line BP, parity-4 (n=1.0)</td>
<td>425.34/4998</td>
<td>43.14%0</td>
</tr>
</tbody>
</table>

Table 2. Non-predictability test results. SSEs are considered on all the N-M patterns of each random sequence, where N=5000 is the sequence length and M=2 is the sliding window length.

V. CONCLUSIONS

We have studied the use of recurrent ANN of the Hopfield type with the tanh sigmoidal, output function and the use of Overfitting MLP neural network training properties as generators of pseudorandom numbers and as strengthening elements of traditional generators to be integrated in the protocols of secure communications. The same characteristics of the neural network architectures are exploited by their role as strengthening elements of traditional generators. The proposed non-predictability test, which is based on the time series forecasting properties of MLP, along with empirical and theoretical tests comprise an evaluation methodology of pseudorandom number generators. The neural-network-based and the strengthened generators have been shown to behave better than the traditional ones in terms of statistical and non-predictability tests. Future work aims to extend the role of further neural network architectures as generators or as strengthening elements of generators. The evaluation of often used generators by means of the proposed methodology is, also, a pursuit of our current work. But the most practical aspect of our work is to integrate such algorithms in the protocols of multimedia communications for secure transactions in the delivery of multimedia content, which is under way and will be presented in the near future.

REFERENCES