BULLETIN OF THE
GREEK MATHEMATICAL SOCIETY
Volume 48, 2003 (61–76)

Cryptography through Interpolation, Approximation and Computational Intelligence Methods

G.C. Meletiou, D.K. Tasoulis and M.N. Vrahatis

Abstract

Recently, numerous techniques and methods have been proposed to address hard and complex algebraic and number theoretical problems related to cryptography. We review several interpolation and approximation techniques. In particular, we discuss techniques related to polynomial interpolation, discrete Fourier transforms, as well as, polynomial approximation. Subsequently, we focus on a recently proposed approach to tackle these problems based on computational intelligence methods. More specifically, a study of the neural network approach to address the discrete logarithm problem, the Diffie-Hellman mapping problem, and the factorization problem related to the RSA cryptosystem, is attempted.

Keywords: Cryptography, interpolation, approximation, computational intelligence, neural networks, data encryption, finite fields, discrete Fourier transforms, discrete logarithm, Diffie-Hellman mapping problem, factorization problem, RSA cryptosystem.

2000 *Mathematics subject classification*: 94A60, 11T71, 68P25, 11T55, 65T50, 41A05, 41A10, 92B20, 82C32.

1. Introduction

A number of hard and complex mathematical problems have been motivated by public key cryptography during the past three decades. These problems are related to computational algebra (finite fields, finite groups, ring theory), computational number theory, probability, logic, Diophantine's complexity, algebraic geometry, etc.

Cryptosystems rely on the assumption that these problems are computationally intractable, in the sense that their computation cannot be completed in polynomial time, e.g. factorization [39], discrete logarithm [1, 32, 33, 37], knapsack problem [24]. Numerous techniques have been proposed to address these problems including algebraic methods, number theory, software oriented methods, interpolation techniques, and generic algorithms (e.g., see [1, 5, 21, 27, 29, 30, 32, 33, 37, 48].

In this paper we focus on the interpolation and approximation techniques [3, 4, 5, 10, 16, 22, 25, 26, 27, 29, 30, 31, 41, 42, 48, 49]. More specifically, we consider polynomial interpolation, discrete Fourier transforms and finally, polynomial approximation.

In a sense Artificial Neural Networks (ANNs) can be considered as generalized approximation techniques. ANNs can be defined as a mathematical model with the ability to learn, adapt, generalize or alternatively, cluster and organize data. In this paper, a study of the ability of ANNs to face the discrete logarithm problem, the Diffie–Hellman mapping problem, and the factorization problem related to the RSA cryptosystem, is attempted.

In the next section we briefly describe the cryptographic problems which are considered in this paper. In Section 3, we briefly expose interpolation and approximation methods that have been applied to address the cryptographic problems under consideration. In Section 4 we describe our approach as well as the experimental results obtained. The paper ends in Section 5 with a short discussion and concluding remarks.

2. Algebraic and Number Theoretical Problems Related to Cryptography

(a) The *Discrete Logarithm Problem (DLP)* [1, 32, 33, 37]. This problem amounts to the creation of an efficient algorithm for the computation of an integer z satisfying the following relation:

$$g^z = h,$$

where g is a primitive element of a finite field, and h a non–zero element. The security of various public–key and private–key cryptosystems [1, 5, 9, 26, 27, 29, 30, 33, 32, 37, 48, 49] is based on the assumption that DLP is computationally intractable. More specifically, we refer to:

(1) the *Diffie–Hellman exchange protocol* [6, 32],
(2) the *El Gamal public key cryptosystem* as well as *the El Gamal digital signature scheme* [9].

In the case of a finite field of prime order p a primitive root g modulo p is fixed, (that is g is a generator of the multiplicative group $\mathbb{Z}_p^*$ of $\mathbb{Z}_p$). We assume that u is the smallest nonnegative integer with $g^u \equiv h (\bmod\, p)$. Then u is called the *index*, or the *discrete logarithm* of h. Gauss [11] referred to the discrete logarithm as the index of a number and pointed out some of its basic properties. He also described methods of computing indices, finding primitive roots and changing bases.

(b) The *Diffie Hellman key Problem (DHP)* [3, 5, 10, 21, 48]. Let, α be a fixed primitive element of F_q; x, y, satisfying, $0 \leqslant x, y \leqslant q-2$, denote the private keys of two users; and $\beta = \alpha^x$, $\gamma = \alpha^y$ stand for the corresponding public keys. Then the problem amounts to computing $\alpha^{x \cdot y}$ from β and γ, where $\alpha^{x \cdot y}$ is the symmetric key for secret communication between the two users.

Consider the special case of the DHP, where $\beta = \gamma$. The term Diffie Hellman mapping refers to the function:

$$\beta = \alpha^x \longmapsto \alpha^{x^2}.$$

The definition of the *Diffie–Hellman Mapping Problem (DHMP)* follows naturally from the previous definition of DHP. Notice that since the following relation holds:

$$\alpha^{x^2} \cdot \alpha^{y^2} \cdot \alpha^{2xy} = \alpha^{(x+y)^2},$$

and the computation of $\alpha^{x \cdot y}$ from α^{2xy} is feasible (square roots over finite fields), the two problems DHMP and DHP are computationally equivalent.

(c) *The factorization problem related to the RSA cryptosystem* [39]. For the RSA cryptosystem to be secure, the factorization of $N = p \cdot q$ must be computationally intractable. The cryptanalyst has to compute a from b. To achieve this it is sufficient to obtain $\phi(N)$ from N, or equivalently to factorize N, since $\phi(N) = (p-1) \cdot (q-1)$ [23, 39].

3. Interpolation and Approximation Methods

In a finite field F_q every function can be represented as a polynomial (Lagrangian interpolation). For every function $f : F_q \to F_q$ there exists a unique polynomial $p(x)$ of degree at most $q-1$ coinciding with f. Interpolation is computationally attractive only in the case of a polynomial with “small” number of non-zero coefficients.

Encryption (and decryption) functions are defined as functions over finite fields, so it is natural to try to express them as polynomials.

Regarding the discrete logarithm function, the well–known formula of Wells [46] exists:

$$\log_a(x) = \sum_{i=1}^{p-2} \frac{x^i}{1-a^i}, \qquad \bigl(x \neq 0, \quad a, x \in \mathbb{Z}_p, \quad a \ \text{ is a generator of the } \ \mathbb{Z}_p^*\bigr).$$

The above mentioned formula can be generalized in the case of a field of prime power order F_q and in the case of a not being a generator of the multiplicative group of the field [26, 27, 29]. A discrete Fourier transform can also be used [27]:

$$\log_a(x) = (1, 2, \ldots, p-1) \cdot \underset{1 \leqslant i,j \leqslant p-1}{(a^{-ij})} \cdot \begin{pmatrix} x \\ x^2 \\ \vdots \\ x^{p-1} \end{pmatrix}.$$

For the Diffie–Hellman mapping the following formula has been found [48]:

$$K(x,y) = -\sum_{1\leqslant i,j\leqslant p-1} x^i y^j a^{-ij}.$$

The above expression can be represented by discrete Fourier transform as follows:

$$K(x,y) = (y, y^2, \ldots, y^{p-1}) \cdot \underset{1\leqslant i,j\leqslant p-1}{(-a^{-ij})} \cdot \begin{pmatrix} x \\ x^2 \\ \vdots \\ x^{p-1} \end{pmatrix}.$$

The Diffie–Hellman key exchange is based on the fact that no easy representation of the Diffie–Hellman mapping $F(g^u, g^v) = g^{u\cdot v}$, $0 < u, v < d$ is known. It can be easily verified that the polynomial [48]:

$$F(x,y) = e \cdot \sum_{i,j=0}^{d-1} g^{-i\cdot j} x^i \cdot y^j, \quad e \cdot d \equiv 1 \bmod p,$$

represents the Diffie–Hellman key function in the F_q field, $q = p^n$. The largest possible degree of F is $2\cdot(d-1)$ and the largest number of non–zero coefficients is d^2. Recently [10, 48] lower bounds for the degree of a two–variable polynomial have been identified. We intend to present further results on this approach in a future correspondence.

Regarding the approximation methods, we consider the notions “near”, “far”, “greater than”, “less than”, “converges to”, “approaches to” etc. These notions have meaning for the elements of real numbers $\mathbb{R}$ or for a function space which consists of real functions of real variables. The field $\mathbb{R}$ is an ordered field, a topological field, etc.

At this point it should be mentioned that it is not possible to find a non trivial distance, norm, order, or topology, which is compatible with the algebraic structure of a finite field. Next, we provide an explanation why it is not possible to find a non trivial topology. Let us assume that F_q is a topological field for a given topology T. Addition and multiplication in F_q have to be continuous operations. Therefore every polynomial is continuous. However every function can be represented as a polynomial, so all functions are continuous. Thus, we conclude that the topology T is trivial.

However, the above mentioned notions are meaningless for the elements of a finite field F_q, or a function space $V \subseteq \{f : F_q \to F_q\}$. Thus, algorithms, techniques and methods from real analysis cannot be used to approximate a solution in the case of finite fields. This is one of the reasons why finite fields are algebraic structures used in cryptography. Their elements can be used in order to represent plaintexts, ciphertexts and keys.

Since the notion “approximation” has no meaning in the classical sense in the finite field case, the need for new definitions arises. In particular, we say that the polynomial

$p(x)$ "approaches" the function $f : F_q \to F_q$ if $p(x) = f(x)$, for all $x \in S \subseteq F_q$ and the cardinality of S is of "reasonable" size. The degree of $p(x)$ is considered as the complexity measure of the problem [5, 41, 42].

In the case of the $\mathbb{Z}_p$ field, $S \subseteq \mathbb{Z}_p^*$ and n the degree of the polynomial whose restriction on S coincides with the restriction of the index function (the discrete logarithm), we have the following bound [5]:

$$n \geqslant \frac{|S|\,(|S|-1)}{2(p-2)}.$$

A probabilistic approach has been given in the case $S \subseteq \mathbb{Z}_p^*$, $|S| = m$, the elements of S are random and uniformly distributed. Trivially there exists a polynomial of degree $m-1$ which represents the restriction of the discrete logarithm on S (Lagrangian interpolation). The probability P to find a polynomial of degree $m-k$ with this property has upper bound:

$$P \leqslant \left(\frac{2m}{p-2}\right)^{\frac{k}{2}}.$$

Generalizations have been given in [5, 41] for a polynomial of two variables $F(x, y)$ satisfying $F(x, indx) = 0$. Also bounds have been given for the approximation of the Diffie–Hellman key mapping.

The polynomials which "approximate" the discrete logarithm function and the Diffie–Hellman mapping coincide with the function on a subset S of the finite field. They do not "come close" to the function. Regarding the complement of S no information can be obtained. Actually, this is a kind of interpolation of the restriction of the functions. For recent results of this approach refer to [42].

Other approaches include the consideration of the domain (or the range) of the the discrete logarithm function and the Diffie–Hellman mapping to be one of $\mathbb{Z}_p$, $\mathbb{Z}_{p-1}$, the k-dimensional Boolean cube, and the real (complex) numbers. In particular, for the case of real numbers the discrete logarithm function and the Diffie–Hellman mapping have been addressed as real functions and have been approximated by real polynomials [42]. The approximating polynomials have been used in various ways. For instance, it is possible to consider the polynomial to coincide with the approximated function on a subset of the domain as before. Another case is to consider the polynomial to come close to the function, since distance between real functions can be defined in various ways. Similar results can be obtained for all the above cases. However, the lower bounds for the computation of the discrete logarithm function and the Diffie–Hellman mapping are not satisfactory. Thus, the discrete logarithm function is indeed a computationally hard function.

Another approach is to consider the notion of linear complexity in order to measure the complexity of a problem. We recall that the linear complexity [7, 8, 15, 22] of a sequence $\{s_i\}$ is the smallest positive integer m such that there are constants

$c_1, c_2, \ldots, c_m$ satisfying the equation:

$$s_i = c_1 \cdot s_{i-1} + c_2 \cdot s_{i-2} + \cdots + c_m \cdot s_{i-m}, \quad \forall\ i \geqslant m.$$

According to this approach, it is well known that the discrete logarithm function and the Diffie–Hellman mapping can be represented as sequences, whose elements are taken from a finite field, and thus the linear complexity can be used as a complexity measure. Also, in these cases lower bounds can be derived indicating the difficulty of computing indices over finite fields.

Furthermore, Boolean functions of n-variables have been used to describe cryptography related decision problems. The Boolean functions permit representations as multivariate polynomials [42]. Interesting results related to the factorization problem inspired from cryptography can been found in [36]. In particular, lower bounds for the degree and the sparsity of polynomials representing the Boolean function, which decides whether a given n-bit integer is square free, have been found. The representation takes place for polynomials over $\mathbb{Z}_n$ as well as for real polynomials. The lower bounds $0.14 \ln n$ and $\frac{n}{5 \ln n}$ have been computed for the degree and the sparsity of the polynomials, respectively.

Finally, we would like to point out here that Shor in [43] has shown that polynomial algorithms for factorization and index computation exist for quantum computers. Provided that quantum computing becomes an available technology in the future, RSA and discrete logarithm based cryptosystems will break.

4. Neural Network Approach

In the past, computing was based on the concept of programmed computing. In this context, for each individual problem, suitable algorithms are designed and implemented. An alternative perspective, inspired by biological systems, was proposed in the form of *Artificial Neural Networks* (ANNs) and *genetic algorithms*. The functionality of biological systems is based on interconnections of specialized physical cells called neurons. ANNs comprise a mathematical model with the ability to learn, adapt, generalize, or to cluster and organize data, by simulating their biological counterparts. All these operations are based on parallel processing of data and can be considerably advantageous and fast, compared to alternative techniques.

In an attempt to provide a strict definition we could say that an ANN is a structure composed of a number of interconnected units (artificial neurons), each unit characterized by an input/output (I/O) relation and implementing a local computation. The output of any unit is determined by its I/O characteristics, its interconnection to other units, and possibly external inputs. Although "hand crafting" an ANN is possible, networks usually develop an overall functionality through training. Evidently, this definition embraces a diverse family of networks. The overall functionality of a network is determined by the topology, the training algorithm applied, as well as, neuron characteristics. One very important type of ANNs is feedforward ANNs. In feedforward networks, all paths lead to one direction. Moreover, the neurons can be disjointedly split in formulations, called layers. In *multilayer feedforward networks*, the inputs form an input layer, while the output neurons form the output layer. All other neurons are assigned to a number of hidden layers. Each neuron in a layer is fully connected to all the neurons of the successive layer. This structure renders it possible to describe networks of this kind with a series of integers that stand for the number of neurons at each layer. For example a network with a topology 4-5-5-1 is a network with 4 inputs at the input layer, two hidden layers with 5 neurons each, and an output layer with a single neuron.

The operation of such networks consists of a series of iterative steps. At the beginning the states of the input layer neurons are assigned to generally real inputs, and the remaining hidden and output layer neurons are passive. In the next step the neurons of the first hidden layer collect and sum their inputs and compute their output. This procedure is propagated to the following layers until the final outputs of the network are computed.

The computational power of neural networks derives from their inherent ability to adapt to specific problems. In [13, 47] the following statement has been proved: "Standard feedforward networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function to any desired degree of accuracy". It has also been proved [35] that a single hidden layer feedforward network with r units in the hidden layer, has a lower bound on the degree of the approximation of any function. The lower bound obstacle can be alleviated if more than one hidden layers are used. Mairov and Pincus in [35] have proved that, on the unit cube in $\mathbb{R}^n$ any continuous function can be uniformly approximated, to within any error by using a two hidden layer network having $2n + 1$ units in the first layer and $4n + 3$ units in the second. This implies that any lack of success in applications can be attributed to inadequate training, an insufficient number of hidden units, or the lack of a deterministic relationship between input and target.

During the training process patterns for which the desired outputs are *a priori* known, are presented to the network. A training set T of P patterns, is defined as:

$$T = \left\{ (x_k, d_k) \;\middle|\; \begin{array}{l} x_k = (x_{k1}, \ldots, x_{kn}) \\ d_k = (d_{k1}, \ldots, d_{km}) \end{array}, \quad k = 1, \ldots, P \right\},$$

where $x_k \in \mathbb{R}^n$ is the input vector of the kth training pattern and $d_k \in \mathbb{R}^m$ is the vector of the desired output for the specific pattern. The ultimate goal of training is to assign to the free parameters of the network W, values such that the output of the network based on that set of weights will be the desired one (at the beginning weights are assigned random values), i.e. it holds that, $y(W, x_k) = d_k$, $k = 1, \ldots, P$. The adaptation process starts by presenting all the patterns to the network and computing a total error function $E = \sum_{k=1}^{P} E_k$, where E_k is the partial network error with respect to the kth training pattern, and is computed by summing the squared discrepancies between the actual network outputs and the desired values of the kth training pattern.

Each full pass of all the patterns that belong to the training set, T, is called a *training epoch*. If the adaptation method succeeds in minimizing the total error function then it is obvious that its aim has been fulfilled. Thus training is a non-trivial minimization problem. The most popular training method is *back propagation* method [12], which is based on the well-known steepest descent method. The back propagation learning process applies small iterative steps which correspond to the training epochs. At each epoch t the method updates the weight values proportionally to the gradient of the error function $E(w)$.

The whole process is repeated until the overall error value drops below a predetermined threshold. At this point we conclude that the network has learned the problem "satisfactorily". The total number of epochs required can be considered as the speed of the method. More sophisticated training techniques can be found in [12, 14, 17, 18, 20, 38, 45].

4.1. *Experimental Results*

The numerical experiments performed address the discrete logarithm problem in modular arithmetic, and the factorization problem, through neural networks. The empirical tests were performed using a C++ neural network interface built under Linux operating system with the gcc compiler. The training methods considered were: Standard Back Propagation (BP) [40], Back Propagation with Variable Stepsize (BPVS) [18], Resilient Back Propagation (RPROP) [38] and On-Line Adaptive Back Propagation (OABP) [17]. All the methods were extensively tested with a wide range of parameters. No single method exhibited significantly different performance relative to the others, with the negative exception of standard back propagation, which encountered grave difficulties in training most of the times. Relative to speed measures RPROP proved to be the fastest. Finally, BPVS managed to train networks in occasions when all other methods failed.

Choosing the "optimal" network architecture for any particular problem is very difficult and remains an open problem up to date. Inevitably, our approach also suffers from this problem. To find a small network that performs the task at hand satisfactorily, we proceed as follows. Starting with a network with a total number of weights smaller than the considered number we reduce its architecture as much as it is permitted by the training method considered. Decreasing the size of the network we observed that all training methods yielded suboptimal solutions, which can be attributed to the local minima effect. To alleviate convergence to local minima we applied the recently proposed *deflection technique* [19] and the *function "stretching"* method [34].

A large variety of network topologies were considered. As expected, different topologies exhibited great differences in the results. After extensive experimentation, we came to the conclusion that networks with 2 hidden layers were much easier to train. A crucial part of the whole procedure is the normalization step of the training process. This step takes place before training, and transforms the data in such a way that the network will find it easier to adapt to. Assuming that the data presented to the network are in $\mathbb{Z}_p$, where p is prime, the space $S = [-1, 1]$, is split in p sub-spaces. Thus, numbers in the data are transformed to analogous ones in the space S. At the same time, the network output is transformed to a number within $\mathbb{Z}_p$ using the inverse operation.

To evaluate network performance two different measures were considered. The first measure, which we call *complete measure* and denote it by μ_0, measures the percentage of the training data, for which the network was able to compute the exact target value. This measure proved insufficient as a performance indicator. The fact that network output was restricted within the range $[-1, 1]$, played a significant role. Very small differences in output, rendered the network unable to compute the exact target but rather to be very close to it. Thus, a second measure, which we call *near measure* and denote it by $\mu_{\pm v}$, was incorporated. This measure reflects the percentage of the data for which the difference between desired and actual output does not exceed $\pm v$ of the real target. The second measure captured the real capability of the network, since when the training procedure was allowed to continue for longer time periods the first measure kept rising to reach the second one.

It is important to mention that the "near" measure $\mu_\pm$ plays different roles for the DLP and the DHMP. The "near" $\mu_\pm$ measure is very important in the case of the DLP. If the size of the $(\pm v)$ interval is $O(\log(p))$ then the "near" measure can replace the "complete" μ_0 one. In general, for some "small" values of v the "near" measure is acceptable since the discrete logarithm computation can be verified i.e. computation of exponents over finite fields [37]. However, the verification of the DH-mapping is an open problem (the DHDMP). Sets of possible values for the DH-mapping can be used to compute sets of possible values for the DH-key. The values of the DH-key can be tested in practice; they are symmetric keys of communication between the two users. The percentage of success for the "near" measure for DHMP can be compared with the corresponding percentage for the DLP. The results of the comparison can be related to the conjecture that the two problems are computationally equivalent.

4.1.1. *Facing the discrete logarithm problem and the Diffie–Hellman mapping problem.* Starting with a small prime number, as for example, the number 19, networks with up to 13 weights were trained with success reaching up to 100% . Results for DHMP and DLP problem are exhibited in Table 1.

A well–known advantage of neural networks is their ability to generalize the acquired knowledge. To test their generalization ability, the training set was reduced by three randomly selected patterns. These patterns were presented to the network after the training process had been completed, as an unknown test data set. In this

Topology	Epochs	μ_0	$\mu_{\pm 2}$	Problem
1-3-3-1	3000	80%	90%	DHMP
1-3-3-1	6000	90%	100%	DHMP
1-3-3-1	6000	40%	70%	DLP
1-5-6-1	13000	100%	100%	DLP

Table 1: Results for the DHMP problem with elements in $\mathbb{Z}_{19}$.

case the networks for the DLP problem managed to achieve better results, as shown in Table 2.

Topology	μ_0	$\mu_{\pm 2}$	Problem
1-5-6-1	33%	66%	DLP
1-5-6-1	0%	33%	DHMP

Table 2: Results for unknown data set with elements in $\mathbb{Z}_{19}$.

Next, trying a larger prime number, the task of training networks becomes harder. Having so many numbers in the range $[-1, 1]$ poses problems for the adaptation process due to the fact that the normalization problem is not fully solved. Consequently, small changes in the network output can cause complete failure. This raises the need for larger architectures. On the other hand, the generalization to unknown data does not fail. Thus, by choosing as prime numbers the numbers 83 and 97 we can get the results exhibited in Tables 3 and 4. The most promising future direction of research seems at this point to be addressing the normalization issue.

Topology	Epochs	μ_0	$\mu_{\pm 2}$	$\mu_{\pm 5}$	$\mu_{\pm 10}$	Problem
1-5-5-1	20000	20%	30%	48%	70%	DLP
1-5-5-1	20000	20%	35%	51%	70%	DHMP

Table 3: Results for networks trained with elements in $\mathbb{Z}_{83}$.

Keeping in mind the generalization problem, it is obvious that for larger prime numbers the training process will encounter the problem of local minima. To alleviate this obstacle two recently proposed techniques have been applied [19, 34]. Incorporating these techniques resulted to faster training most of the times. Furthermore, a rise of 10% for most results was also witnessed. Results for the techniques that were found to be superior for larger prime numbers are exhibited in Table 4. As it is exhibited, very small networks are trained with elements in $\mathbb{Z}_{97}$ and give satisfactory results for the measure $\mu_{\pm}$.

Topology	Epochs	μ_0	$\mu_{\pm 2}$	$\mu_{\pm 5}$	$\mu_{\pm 10}$	Problem
1-5-5-1	25000	20%	30%	48%	70%	DLP
1-5-5-1	20000	20%	35%	51%	70%	DHMP

Table 4: Results for networks trained with elements in $\mathbb{Z}_{97}$.

Using even larger primes the problem seems to become too difficult to tackle. In Table 5, a very poor network performance is illustrated, for the DLP problem and for elements in $\mathbb{Z}_{997}$. Similar results were obtained for the DHMP.

Topology	μ_0	$\mu_{\pm 5}$	$\mu_{\pm 10}$	$\mu_{\pm 20}$
1-5-5-1	3%	8%	20%	30%
1-9-9-1	5%	9%	25%	30%

Table 5: Results for networks trained with elements in $\mathbb{Z}_{997}$.

4.1.2. *Addressing the Factorization Problem.* Finally, the ability of neural networks to break the RSA cryptosystem is investigated. Trying to approximate the $\phi(N)$ mapping $(p \cdot q \to (p-1)(q-1))$, input patterns were numbers $N = p \cdot q$ where p and q are primes, and target patterns where the $\varphi(N) = (p-1)(q-1)$ numbers. In this case results are different. The normalization problem ceases to be an obstacle. What is really intriguing in this case is the generalization performance of the networks, reported in Table 6. Excluding a portion of the data set (33%) from the training set and using it as a Test Set, gives the results exhibited in Table 6. Clearly, the networks are able not only to adapt to the training data, but also to achieve very good results with respect to the test sets.

The next step is to use larger keys. So if we use all the prime numbers between 7000 and 8000, we construct keys that are between 49×10^6 and 7×10^7. A portion of 20% of the dataset is used as a training set and the remaining 80% as a test set. From the results that are exhibited in Table 7, it is evident that small networks have been trained to a very good performance on both training and test datasets. Even when the portion of the training set is small, the networks learn the dynamics behind the data and achieve good generalization results.

Topology	Epochs	μ_0	$\mu_{\pm 2}$	$\mu_{\pm 5}$	$\mu_{\pm 10}$	$\mu_{\pm 20}$	Type
1-5-5-1	80000	3%	15%	35%	65%	90%	Train Set
1-5-5-1	80000	5%	20%	40%	60%	90%	Test Set
1-7-8-1	50000	6%	20%	50%	70%	100%	Train Set
1-7-8-1	50000	5%	20%	50%	70%	90%	Test Set

Table 6: $\phi(N)$ mapping results for networks trained with a 66% of the data set with $N = p \cdot q \leqslant 10^4$.

Topology	$\mu_{\pm 3}$	$\mu_{\pm 50}$	$\mu_{\pm 100}$	Type
1-5-5-1	30%	65%	90%	Train Set
1-5-5-1	30%	60%	90%	Test Set

Table 7: $\phi(N)$ mapping results for networks trained with a 22% of the data set with $49 \times 10^6 \leqslant N = p \cdot q \leqslant 7 \times 10^7$.

5. Discussion and Concluding Remarks

In the present paper, we review some interpolation and approximation techniques and in particular we discuss techniques related to polynomial interpolation, discrete Fourier transforms as well as polynomial approximation. Furthermore, a study of the neural network approach to address the Discrete Logarithm Problem (DLP), the Diffie–Hellman Mapping Problem (DHMP), and the factorization problem related to the RSA cryptosystem has been attempted. It is known that if a method for computing indices over finite fields is available, then the RSA cryptosystem will break. In other words, the DLP is no easier than the factorization problem related to the RSA. Our experimental results conform to this conclusion.

To the best of our knowledge, this is the first attempt to apply neural networks to tackle difficult problems related to Cryptography (see also [28]). Our experience indicates that it is possible to train feedforward neural networks to address this very difficult problem. In particular for small prime numbers we have shown that it is possible to train a Neural Network with a total number of weights smaller than the considered prime number which computes the discrete logarithm and the Diffie–Hellman functions. Naturally, this being our first attempt toward this direction, we strongly believe that numerous issues that remain open have to be addressed in order to obtain a comprehensive view of the ability of neural networks to simulate the suggested functions.

In general, the problem can be considered solved if the measure μ_0 is 100% and the architectural topology is small enough (number of weights $O(\log(p))$. On the other hand, measures like $\mu_{\pm}$ seem promising in the sense that although the exact target might not be accomplished, the output of the network is indeed close to that. Thus, with a predetermined number of trial and error procedures the problem can be resolved. The "near" $\mu_{\pm}$ measure is very important in the case of the DLP. If the size of the $(\pm v)$ interval is $O(\log(p))$ then the "near" measure can replace the "complete" μ_0 one. In general, for "small" values of v the "near" measure is acceptable since the discrete logarithm computation can be verified (computation of exponents over finite fields).

Here we have considered only artificial feedforward neural networks. In a future correspondence we intent to apply various other networks and learning techniques such as Learning Rate Adaptation methods [20, 45], Non-Monotone neural networks [2], Probabilistic neural networks [44], Self-Organized Map algorithm [14], Recurrent networks and Radial Basis function networks [12].

In conclusion our experience indicates that the Neural Network approach on problems related to cryptography is promising. The results we obtained for the RSA related factorization problem are highly supportive of this claim, even though data with larger numbers need to be considered. On the other hand, DLP and DHMP seem to be harder to solve through this approach, despite the fact that numerous issues remain unaddressed and further work is required.

Acknowledgements. We thank the referees for their meticulous comments and suggestions that greatly helped to improve the paper.

References

[1] Adleman L., A subexponential algorithm for discrete logarithm problem with applications to cryptography, *20th FOCS*, 55–60, (1979).

[2] Boutsinas B. and Vrahatis M.N., Artificial nonmonotonic neural networks, *Artif. Intelligence*, **132**, 1–38, (2001).

[3] Canetti R., Friedlander J. and Shparlinski I., On certain exponential sums and the distribution of Diffie-Hellman triples, *J. London Math. Soc. (2)*, **59**, No. 3, 799–812, (1999).

[4] Clausen M., Dress A., Grabmeier J. and Karpinski M., On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields, *Theoret. Comput. Sci.*, **84**, No. 2, 51–164, (1991).

[5] Coppersmith D. and Shparlinski I., On polynomial approximation of the discrete logarithm and the Diffie–Hellman mapping, *J. Cryptology*, **13**, 339–360, (2000).

[6] Diffie W. and Hellman M., New directions in cryptography, *IEEE Trans. Inf. Theory*, **22**, 644–654, (1976).

[7] Ding C. and Helleseth T., On cyclotonic sequences, *Inform. Process. Lett.* **66**, No. 1, 21–25, (1998).

[8] Ding C., Helleseth T. and Shan W., On the linear complexity of Legendre sequences, *IEEE Trans. Inf. Theory* **44** No. 3, 1276–1278, (1998).

[9] El Gamal T., A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, **31**, 469–472, (1985).

[10] El Mahassni E. and Shparlinski I., Polynomial representations of the Diffie-Hellman mapping, *Bull. Austral. Math. Soc.*, **63**, No. 3 467–473, (2001).

[11] Gauss C.F., *Disquisitiones Arithmeticae*, Section III–Residues of Powers, (1801).

[12] Haykin S., *Neural Networks*, Macmillan College Publishing Company, New York, (1999).

[13] Hornik K., Multilayer feedforward networks are universal approximators, *Neural Networks*, **2**, 359–366, (1989).

[14] Kohonen T., *Self–Organized Maps*, Springer, Berlin, (1997).

[15] Konyagin S., Lange T. and Shparlinski I., Linear complexity of the discrete logarithm, *Designs, Codes and Cryptography*, **28**, 135–146, (2002).

[16] Lange T. and Winterhof A., Incomplete character sums over finite fields and their application to the interpolation of the discrete logarithm by Boolean functions, *Acta Arith.*, **101**, No. 3 223–229, (2002).

[17] Magoulas G.D., Plagianakos V.P. and Vrahatis M.N., Adaptive stepsize algorithms for on-line training of neural networks, *Nonlinear Analysis T.M.A.*, **47**, No. 5, 3425–3430, (2001).

[18] Magoulas G.D., Vrahatis M.N. and Androulakis G.S., Effective backpropagation training with variable stepsize, *Neural Networks*, **10**, No. 1, 69–82, (1997).

[19] Magoulas G.D., Vrahatis M.N. and Androulakis G.S., On the alleviation of the problem of local minima in Backpropagation, *Nonlinear Analysis T.M.A.*, **30**, No. 7, 4545–4550, (1997).

[20] Magoulas G.D., Vrahatis M.N. and Androulakis G.S., Increasing the convergence rate of the error backpropagation algorithm by learning rate adaptation methods, *Neural Computation*, **11**, No. 7, 1769–1796, (1999).

[21] Maurer U. and Wolf S., The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms, *SIAM J. Computing*, **28**, 1689–1721, (1999).

[22] Meidl W. and Winterhof A., Lower bounds on the linear complexity of the discrete logarithm in finite fields, *IEEE Trans. Inform. Theory*, **47**, No. 7, 2807–2811, (2001).

[23] Menezes A.J., Van Oorschot C.P. and Vanstone S.A. *Handbook of applied cryptography*, CRC Press, (1996).

[24] Merkle R.C. and Hellman M.E., Hiding information and signatures in trapdoor knapsacks, *IEEE Trans. Inf. Theory*, **24**, 525–530, (1978).

[25] Meletiou G., A polynomial representation for exponents in $\mathbb{Z}_p$, *Bull. Greek Math. Soc.*, **34**, 59–63, (1992).

[26] Meletiou G., Explicit form for the discrete logarithm over the field GF(p, k), *Arch. Math. (Brno)*, **29**, No. 1–2, 25–28, (1993).

[27] Meletiou G. and Mullen G.L., A note on discrete logarithms in finite fields, *Appl. Algebra Engrg. Comm. Comput.*, **3**, No. 1, 75–79, 1992.

[28] Meletiou G., Tasoulis D.K. and Vrahatis M.N., A first study of the neural network approach to the RSA cryptosystem, *IASTEED 2002 Conference on Artificial Inteligence*, 483–488, (2002).

[29] Mullen G.L. and White D., A polynomial representation for logarithms in $GF(q)$, *Acta Arithmetica*, **47**, 255–261, (1986).

[30] Niederreiter H., A short proof for explicit formulas for discrete logarithms in finite fields, *Appl. Algebra Engrg. Comm. Comput.*, **1**, 55–57, (1990).

[31] Niederreiter H. and Winterhof A., Incomplete character sums and polynomial interpolation of the discrete logarithm, *Finite Fields Appl.*, **8**, No. 2, 184–192, (2002).

[32] Odlyzko A.M., Discrete logarithms in finite fields and their cryptographic significance, *EUROCRYPT 84*, (1984).

[33] Odlyzko A.M., Discrete logarithms: the past and the future. *Designs, Codes and Cryptography*, **19**, 129–145, 2000.

[34] Parsopoulos K.E., Plagianakos V.P., Magoulas G.D. and Vrahatis M.N., Objective function "stretching" to alleviate convergence to local minima, *Nonlinear Analysis T.M.A.*, **47**, No. 5, 3419–3424, (2001).

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

[36] Plaku E. and Shparlinski I.E., On polynomial representations of Boolean functions related to some number theoritic problems, *2001 Conference on Foundations of Software Technology and Theoretical Computer Science*, 305–316, (2001)

[37] Pohlig S.C. and Hellman M., An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, *IEEE Trans. Inf. Theory*, **24**, 106–110, (1978).

[38] Riedmiller M. and Braun H., A direct adaptive method for faster backpropagation learning: The RPROP algorithm, *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, 586–591, (1993).

[39] Rivest R., Shamir A. and Adlemann L., A method for obtaining digital signatures and public-key cryptosystems, *Commun. ACM*, **21**, 120–126, (1978).

[40] Rumelhart D.E., Hinton G.E. and Williams R.J., 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).

[41] Shparlinski I., *Number theoretic methods in cryptography: complexity lower bounds*, Progress in Computer Science and Applied Logic, **17**, Birkhäuser Verlag, Basel, (1999).

[42] Shparlinski I., *Cryptographic applications of analytic number theory: complexity lower bounds and pseudorandomness*, Progress in Computer Science and Applied Logic, **22**, Birkhaüser Verlag, Basel, (2003).

[43] Shor P., Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM J. Comput.*, **26**, 1484–1509 (1997).

[44] Specht D.F., Probabilistic neural networks, *Neural Networks*, **3** No. 1, 109–118, (1990).

[45] Vrahatis M.N., Androulakis G.S., Lambrinos J.N. and Magoulas G.D., A class of gradient unconstrained minimization algorithms with adaptive stepsize, *J. Comput. Appl. Math.*, **114**, No. 2, 367–386, (2000).

[46] Wells A.L., A polynomial form for logarithms modulo a prime, IEEE Trans. Inform. Theory, **IT-30**, 845–846, (1985).

[47] White H., Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings, *Neural Networks*, **2**, 359–366, (1989).

[48] Winterhof A., A note on the interpolation of the Diffie-Hellman mapping, *Bull. Australian Math. Soc.*, **64**, No. 3, 475–477, (2001).

[49] Winterhof A., Polynomial interpolation of the discrete logarithm, *Des. Codes Cryptogr.*, **25**, No. 1, 63–72, (2002).

◇ G.C. Meletiou
A.T.E.I. of Epirus,
P.O. Box 110, GR–47100,
Arta, Greece.
gmelet@teiep.gr

◇ D.K. Tasoulis
Department of Mathematics,
University of Patras
Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece.
dtas@math.upatras.gr

◇ M.N. Vrahatis
Department of Mathematics,
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece.
URL:http://www.math.upatras.gr/˜vrahatis
vrahatis@math.upatras.gr