On Application of Neural Networks for S-Boxes Design

Piotr Kotlarz[1] and Zbigniew Kotulski[2]

[1] Kazimierz Wielki University, Bydgoszcz
[2] Institute of Fundamental Technological Research of the Polish Academy of Sciences, Institute of Telecommunications of WUT, Warsaw

Abstract. In the paper a new schedule of S-boxes design is considered. We start from motivation from block cipher practice. Then, the most popular S-box design criteria are presented, especially a possibility of application of Boolean bent-functions. Finally, we propose integrating neural networks (playing a role of Boolean functions with appropriate properties) in the design process.

1 Foundations

Almost all modern block ciphers have iterative structure. This means that the complicated encryption of a block of text is due to multiple iterations of a simple round function. In the case of Feistel permutation, which was introduced in Lucifer and then widespread in 1977 through American standard DES [1], the input 64-bit block is divided into two 32-bits sub-blocks L_i and R_i. The round transformation, represented as:

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus f\left(R_{i-1}, K_i\right), \end{aligned} \tag{1}$$

substitutes the right subblock R_{i-1} the from previous round as the left subblock L_i at present and calculates bit-wise XOR of the left subblock L_{i-1} from the previous round with the output of the round function f taken on the right subblock R_{i-1} from the previous round and the round key K_i, substituting the result as the new right subblock R_i. Such a structure makes the cipher invertible (if decrypting we put the round keys in inverse order) and secure, provided the round function f has specific properties. These properties (e.g., diffusion and confusion, see [2]) make that during multiple iterations the plaintext bits mix strongly with themselves and bits of the secret key, so they cannot be reconstructed from the ciphertext without knowledge of the key. Usually, the two properties are realized in the round function by two layers of transformations: the diffusion by permutations and the confusion by key-dependent non-linear transformations. Description of permutations is quite simple (it is a table of substitutions), while writing down an arbitrary non-linear transformation mapping n bits block (e.g., $n = 32$ for DES) on n bits block is very complicated.

The problem of non-linear maps was practically solved by the concept of substitution boxes (S-boxes). For example, in DES (and earlier in Lucifer) the confusion layer contains a number of parallel S-boxes. Each S-box transforms 6 bit inputs to 4 bit output. It is a table of 4 rows and 16 columns. The elements of the table are the outputs (elements of the set $\{0, 1, 2, ..., 15\}$), equivalent to 4 bit blocks $\{0000, 0001, ..., 1111\}$. The element of the table is chosen according to the value of the input block: two bits (the first and the last) point the row number while the outstanding four bits give the number of the column. In the whole layer DES uses 8 different S-boxes.

The idea of application of S-boxes of proved to be very fruitful in constructing cryptographic algorithms. For example, in new American encryption standard AES [3] 16×16 S-boxes were applied. The Russian encryption standard GOST [4] also uses a layer of S-boxes, but with content left to the users' invention. S-boxes are used not only in block ciphers but also in certain stream ciphers [5] and hash algorithms.

In the cryptographic practice there is no mathematically precise theory of the S-boxes design. However, there are some intuitive rules and conditions imposed on the S-boxes. Returning to the classical example of DES we can give some properties of all its eight S-boxes:

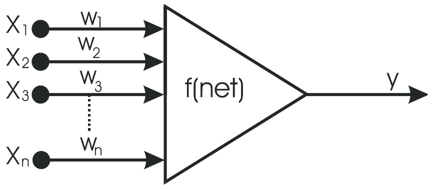
– No S-box is a linear of affine function of its input (they are non-linear functions)
– Every row in any S-box is a permutation (it contains equal number of zeros and ones)
– Changing one bit of input we change at least two bits of the output of S-box (the avalanche property)
– If one input bit is fixed, an S-box minimizes the difference of zeros and ones on the output.

The particular properties of the S-boxes presented above resulted in more general properties of the whole DES cipher: its (maximally possible) resistance to the linear [6] and differential [7] cryptanalysis. This result became a foundation of a practical method of constructing S-boxes: first one should generate the S-box putting its contents at random and then test its statistical properties (as a binary map) to obtain the maximal resistance against the two attacks. The assumed dimension of the S-box and some prior constraints on the S-boxes must be carefully discussed in the context of a cryptographic algorithm the S-box is to be installed. The higher dimension of the S-box, its statistical analysis more complicated both for the algorithm designer and a hostile cryptanalyst.

At present the methods of S-boxes design are based on two parallel methodologies. The first one uses mainly mathematical theories and statistical investigations, while the other is additionally supported by practitioners experience To find a link between the both methodologies we propose the application of the neural networks learning methodology in the S-box design procedure.

2 Neural Networks

Neural networks are a form of multiprocessor matrices with simple processing elements, simple scalar messages and adaptive interaction between elements. The working element of the network is neuron. Its mathematical model is presented at Fig. 1.



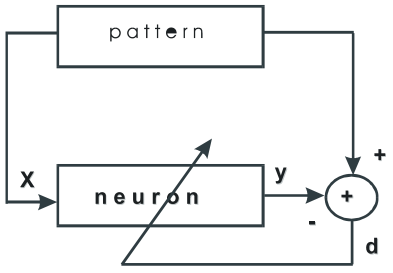
$$net = \sum_{i=1}^{n} x_i w_i,$$
$$x = [x_1, x_2, ..., x_n]^T,$$
$$w = [w_1, w_2, ..., w_n]^T$$

Fig. 1. The mathematical model of neuron

The weighted sum of the inputs x is calculated to compose the activation of the neuron *net*, (also called the post-synaptic potential). The activation signal is passed through an activation (or transfer) function f to generate the output of the neuron. Eq. (2) shows two examples of activation function: the step function (discrete unipolar) f_p and sigmoid (continuous unipolar) f_s ones.

$$f_p(x) = \begin{cases} 1, \sum_{i=1}^{n} x_i w_i \geq p \\ 0, \sum_{i=1}^{n} x_i w_i < p \end{cases}, \; f_s(x) = \frac{1}{1 + \exp\left\{-\alpha\left(\sum_{i=1}^{n} x_i w_i\right)\right\}}. \qquad (2)$$

The training of a neuron or a neural network can be performed with a number of methods. One of them is learning according to the back-propagated delta rule with supervision. The aim of the training is minimizing the sum squared error between the desired output and the actual output.



$$x(i) = [x_1(i), x_2(i), ..., x_m(i)]^T$$
$$y(i) = [y_1(i), y_2(i), ..., y_m(i)]^T$$
$$\bar{y}(i) = [\bar{y}_1(i), \bar{y}_2(i), ..., \bar{y}_m(i)]^T$$
$$d_j(i) = y_j(i) - \bar{y}_j(i)$$
$$w'_j = w_j + \eta d_j(i) x_j, \; j = 1...m$$
$$E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} d_j^2(i)$$

Fig. 2. The neural network training process

Fig. 2 drafts a schedule of training a neural network with $m-$bit inputs and $m-$bit outputs. At the figure, $\bar{x}, \bar{y}$ is the training set (a set of n vector inputs and desired outputs) and i is the number of its element, y is the actual output, $d_j(i)$ is a ith cycle error of the jth coordinate, wj is the weight of jth input, x is the input vector, η is the learning rate and E is the sum squared error. More details about neural networks can be found in the literature (see, e.g. [9]).

3 S-Boxes Design

S-boxes are critical security elements of the Feistel-type ciphers (see [10]) and other cascade algorithms. Well-designed S-box is a strongly non-linear Boolean (vector) map with the avalanche property. In the literature one can find some practical methods of the S-box design [11], [12], but they are rather specific than universal. Among others, one of the most promising seems to be application of Boolean bent-functions (see, e.g., [13], [14]).

The Boolean functions is defined as:

$$f : (Z_2)^n \rightarrow Z_2,\ Z_2 = GF(2). \tag{3}$$

To be the bent-function, the Boolean function must additionally be non-linear and balanced (it should output the equal number of zeros and ones when its argument goes over the whole range). More precisely, a Boolean function is the bent-function (or perfectly nonlinear) if the function $f(x) \oplus f(x \oplus a)$ is balanced for every $a \in (Z_2)^n$ such that $1 \leq hwt(a) \leq n$.

The balance of a Boolean function is a fundamental property from the cryptography point of view (see [15]). In the language of Hamming measures the function f is balanced if $hwt(f) = 2^{n-1}$. The Hamming measure of a binary string is the number of ones in this string. For the Boolean function f, its Hamming measure is the number of ones in its truth table. The Hamming distance of two Boolean functions f and g is defined as $d(f,g) = hwt(f \oplus g) = \sum\limits_{x} f(x) \oplus g(x)$ where the summation is over the whole argument's range.

4 Neural Networks and the Bent-Functions

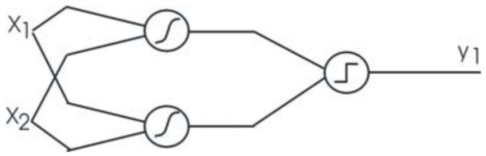
As we remarked, cryptographically secure S-boxes must satisfy certain conditions. Thy critical property is the balance of outputs. This means in fact, that the S-box can be considered as a black box which transforms any input vector into a balanced vector in a non-predictable way (what translates in practice as non-linear). Now we will show that such an security element can be realized by specific neural networks.

To solve the construction problem we start from the 2 × 2 S-box (two bits input and two bits output of a Boolean function) We expect, that the S-box produces a balanced output for an arbitrary input. The sample truth table in such a case can be represented as:

x_1	x_2	y_1	y_2
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

The Hamming measure of this table is 2, by definition of the measure. As it is seen from the table, the output of the vector-valued Boolean function is with such a truth table would be balanced.

To realize the balanced Boolean function we apply certain two-layers neural network. First we start from one block of the network. The first layer of the block is a pair of neurons with the sigmoid activation functions f_s and the other is a single neuron with step activation function f_p (see Fig. 3).



$$y = \begin{cases} 0 \text{ for } f_s(x_i w_i) < 0 \\ 1 \text{ for } f_s(x_i w_i) \geq 0 \end{cases}$$

Fig. 3. Neural network for the one-output bent-function

Combining two such blocks (Fig.4) and training each of them with different training set we obtain the network which outputs the balanced binary vector irrespective of the values of the input binary data.

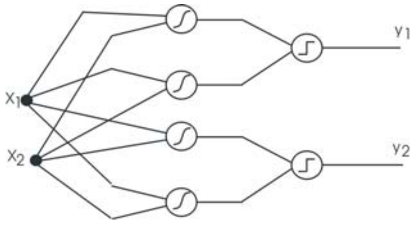


Fig. 4. Neural network for two-outputs bent-function

Now we can take the neural blocks presented at Figs. 3, 4 and combine them into larger structures realizing certain functions. As a simple example we built a network of 189 neurons (Fig.3) to obtain a selector choosing elements from S-box of DES. Compiled with a standard neural network it gave neural representation of the S-box. Certainly, such a network should be optimized to reduce number of "neural cells". This will be the subject of future research.

The balance and avalanche property are fundamental for S-boxes, but they do not suffice their security. The other important property of Boolean functions and S-boxes is the correlation resistance. This property generalizes the balance property. We say that the function f of n arguments has the correlation resistance of order m $(1 \leq m \leq n)$ if the output $f(x)$ is statistically independent of any m chosen elements of x. In the language of information (see [15]) we can write this as

$$H(f(x)) - H(f(x)|x_1, x_2, ..., x_i) = 0. \tag{4}$$

Testing the property (4) can be additional, except of the balance property, easy to implement training condition of the neural network modelling an S-box. Let us remark the training procedure could be performed without exact knowledge of the modelled function, only on the basis of very general requirements lied on it.

5 Summary

In the paper we presented an idea of application of neural networks for constructing S-boxes, the critical security elements of block ciphers and other private key cryptographic algorithms. This element is most difficult to realize by a neural network. The others elements of block ciphers like permutations and linear substitutions are much easier. Thus, realization of the whole algorithm is only a technical problem. The other potentially fruitful possibility of application of neural networks in cryptology is testing cryptographic algorithms. This problem along with improvement of the approach presented in this paper would be the subject of our future research.

References

[1] FIPS 46-3, Data Encryption Standard (DES), NIST 1999.

[2] Z. Kotulski, Construction of block ciphers: new possibilities. (Polish) *Mat. Stosow.* No. 4(45) (2003), pp. 1-24.

[3] FIPS 197, Advanced Encryption Standard (AES), NIST 2001.

[4] GOST 28147-89, Cryptographic Protection for Data Processing Systems. Cryptographic Transformation Algorithm, Government Standard of the U.S.S.R. 1990.

[5] L. Gan, S. Simmons, S. Tavares, A new family of stream ciphers based on cascades small S-boxes, IEEE (1997).

[6] M. Matsui, Linear Cryptanalysis Method for DES Cipher, LNCS Advances in Cryptology, EUROCRYPT'93, pp. 386-397, Springer-Verlag, Berlin 1993.

[7] E. Biham, A. Shamir, Differential Cryptanalysis of Data Encryption Standard, Springer Verlag, Berlin 1993.

[8] C. M. Adams, S. E. Tavares, Designing S-boxes for Ciphers Resistant to Differential Cryptanalysis.

[9] A. Horzyk, R. Tadeusiewicz, Self-Optimizing Neural Networks, in: F. Yin, J. Wang, Ch. Guo [Eds.]: Advances in Neural Networks - ISNN 2004, pp.150-155, LNCS 3173, Springer, Berlin 2004.

[10] L.R. Knudsen, Practically Secure Feistel Ciphers, in: R.Anderson [Ed.], Fast Software Encryption, LNCS 809, pp. 211-222, Springer-Verlag, Berlin 1994.

[11] A F Webster, S E Tavares, On the design of S-boxes, LNCS 218 Advances in Cryptology—CRYPTO'85, pp.523-534, Springer-Verlag, Berlin 1985.

[12] S. Mister, C. Adams, Practical S-Box Design, Workshop on Selected Areas in Cryptography (SAC '96) Workshop Record, Queens University, 1996, pp. 61–76.

[13] C.M. Adams and S.E. Tavares, The Use of Bent Sequences do Achieve Higher Order Strict Criterion in S-Box Design TRTR 90-013 Jan. 1990.

[14] A. Grocholewska-Czurylo, Avalanche and propagation properties of random bent Boolean functions, Proc. RMCIS 2003, p.9.3, Zegrze 2003.

[15] J. Szmidt, Cryptographic properties of the Boolean functions, Proceedings of the 5th National Conference on Cryptography ENIGMA 2001, Warsaw 2001.