Available online at www.sciencedirect.com

ScienceDirect

Information Sciences 178 (2008) 3118–3132

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Finding the differential characteristics of block ciphers with neural networks

Abbas Ghaemi Bafghi, Reza Safabakhsh [*], Babak Sadeghiyan

Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran 15914, Iran

Received 5 December 2005; received in revised form 27 February 2008; accepted 28 February 2008

Abstract

We have developed a model to represent the differential operation of block ciphers in order to help finding differential characteristics. Through this model, the whole space of differential characteristics for a block cipher is represented by a multi-level weighted directed graph. In this way, the problem of finding the best differential characteristic for a block cipher reduces to the problem of finding the minimum-weight multi-branch path between two known nodes in the proposed graph. In this paper, we use recurrent neural networks to find such a path in the differential operation graph of a block cipher. The path is found through minimization of the network cost function. We use the Hopfield network and the Boltzmann machine with and without chaos to minimize the cost function. Chaos is introduced to assist the network to escape from the local minima of the cost function. Experimental results indicate the usefulness of the approach and comparison of the performance of the used techniques shows that the Boltzmann machine algorithm incorporating simulated annealing produces the best result.

Keywords: Differential cryptanalysis; Block cipher; Differential operation model; Hopfield network; Chaotic Hopfield; Boltzmann machine; Chaotic Boltzmann machine; Simulated annealing

1. Introduction

Differential cryptanalysis was developed by Biham and Shamir in 1990 [4]. This cryptanalysis method is performed in the two stages of “design” and “execution”. In the design stage, a cryptanalyst finds the weaknesses of the cipher algorithm and uses them to find a high probability differential characteristic. In the execution stage, the cryptanalyst gathers sufficient pairs with suitable difference and tries to determine some bits of the last round key through a counting scheme. In order to find high probability differential characteristics in the design stage, a very large space must be searched, the size of which is of an exponential order with respect to the number of rounds of the block cipher. Heuristic techniques can be used

[*] Corresponding author. Tel.: +98 21 64542728; fax: +98 21 66495521.
E-mail addresses: ghaemib@ce.aut.ac.ir (A.G. Bafghi), safa@aut.ac.ir (R. Safabakhsh), basadegh@ce.aut.ac.ir (B. Sadeghiyan).

0020-0255/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2008.02.016

to resolve this problem. Such techniques have been used in different cryptanalysis problems as reported in [13,15,17].

In order to more efficiently find differential characteristics, we have developed a modeling method to represent the differential operation of block ciphers [7–9]. Through this modeling scheme, the whole space of differential characteristics for a block cipher is represented as a multi-level weighted directed graph called the *differential operation graph*. Multi-branch paths are defined in these graphs, and it is proven that each differential characteristic for a block cipher corresponds to a multi-branch path in the differential operation graph of that block cipher [6]. In this way, finding the best differential characteristic for a block cipher is reduced to finding the minimum-weight multi-branch path between two known nodes in the proposed graph.

Neural network techniques have been used in different optimization problems such as VLSI placement, *n*-queen, clustering, packing, graph partitioning, graph coloring, network routing, shortest path finding and TSP. In this paper, we apply several neural network techniques to find the minimum-weight multi-branch path in the differential operation graph. We use the Hopfield network, the Boltzmann machine, the chaotic Hopfield network and the chaotic Boltzmann machine to find the minimum-weight multi-branch paths. We compare these techniques through efficiency and efficacy. We apply each of these techniques to find the 4-round, 5-round, 6-round and 7-round differential characteristics for Serpent block cipher. The optimization procedure is repeated 100 times in each case and the best result is considered as the suitable result.

We describe the modeling and the use of neural networks through applying it to Serpent block cipher. First a recurrent neural network is considered and the weight of each multi-branch path in differential operation graph is incorporated in the network cost function. Then the network algorithm is applied to minimize the cost function. In order to overcome the weakness of the network in converging to local minima, we also introduced simulated annealing and chaos in the learning algorithm.

The paper is organized as follows. Section 2 describes Serpent block cipher briefly. Section 3 describes the differential operation model of block cipher algorithms. In Section 4, we design a recurrent neural network to find the minimum-weight multi-branch path in the differential operation graph. In Sections 5 and 6, the ordinary and chaotic Hopfield network and Boltzmann machine are utilized to achieve the required minimization. Section 7 compares the results produced by the four selected networks and Section 8 concludes the paper.

2. The Serpent algorithm

Serpent [2] is a block cipher algorithm with a block size of 128 and a key size of 256 bits. Its structure is a SP-network, consisting of alternating layers of key mixing, S-boxes and linear transformation. Serpent has 32 rounds with a set of eight 4-bit to 4-bit S-boxes as shown in Table 1. In general, an *m*-bit to *n*-bit S-box is defined as a function $S:\{0,1\}^m \rightarrow \{0,1\}^n$ which maps an *m*-bit value to an *n*-bit value. For example, S-box3 of Serpent maps $8_{\text{hex}} = 1000_{\text{bin}}$ to $D_{\text{hex}} = 1101_{\text{bin}}$.

Table 1
Serpent S-boxes

S-box#	Input															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	3	8	F	1	A	6	5	B	E	D	4	2	7	0	9	C
1	F	C	2	7	9	0	5	A	1	B	E	8	6	D	3	4
2	8	6	7	9	3	C	A	F	D	1	E	4	0	B	5	2
3	0	F	B	8	C	9	6	3	D	1	2	4	A	7	5	E
4	1	F	8	3	C	0	B	6	2	5	4	A	9	E	7	D
5	F	5	2	B	4	A	9	C	0	3	E	8	D	6	7	1
6	7	2	C	5	8	4	6	B	E	9	1	F	D	3	A	0
7	1	D	F	0	E	8	2	B	7	4	C	A	9	3	5	6