XII Konferencja PLOUG
Zakopane
Październik 2006

Neuronowy układ szyfrujący - analiza bezpieczeństwa

Piotr Kotlarz

Kazimierz Wielki University, Bydgoszcz
piotrk@ukw.edu.pl

Zbigniew Kotulski

Institute of Fundamental Technological Research, PAS and Institute of Telecommunications, WUT

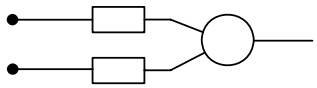
Streszczenie
W pracy tej, która jest kontynuacją kilku wcześniejszych, podejmujemy rozważania na temat układu szyfrującego wykorzystującego sieć neuronową. Nasze prace zmierzają do tego aby skonstruować sieć neuronową, która byłaby w stanie realizować różne algorytmy szyfrujące. Zmiana realizowanego algorytmu ma następować po przeprowadzonym w określonych warunkach procesie uczenia. W naszych wcześniejszych pracach [4],[5],[6], przedstawiliśmy koncepcję realizacji permutacji oraz podstawienia (s-box) za pomocą sieci neuronowej, opartej na neuronie McCulloch-Pittsa [1] oraz regule Hebba. W tej pracy skupimy się na sieci opartej na neuronie logicznym [2]. Główne rozważania na tym etapie pracy dotyczyć będą realizacji permutacji oraz oceny bezpieczeństwa przebiegu procesu uczenia sieci neuronowej.

1. Wprowadzenie

W ostatnim czasie pojawiły się prace na temat wykorzystania układów programowalnych do realizacji algorytmów kryptograficznych. Dla implementacji sprzętowych ma to szczególne znacznie, ponieważ zmiana algorytmu realizowanego sprzętowo jest szczególnie kłopotliwa i kosztowna. Jedną z pierwszych prac była [7] gdzie zaproponowano realizację szyfrowania w oparciu o technologię układów programowalnych jako modułu akceleratora kryptograficznego. Wykorzystując rekonfigurowany procesor RipeRench [8] zrealizowano między innymi takie algorytmy jak CRYPTON [9],[10], czy RC6 [11]. W tej pracy proponujemy wykorzystanie boolowskich sieci neuronowych do realizacji algorytmów szyfrujących.

2. Adaptacyjne sieci logiczne – wprowadzenie

W latach siedemdziesiątych XX w. pojawiła się koncepcja adaptacyjnych sieci logicznych, realizujących funkcje boolowskie [2]. Sieci takie składają się z binarnych neuronów operujących na danych postaci binarnej. Wejścia, wyjście oraz wagi takiego neuronu mogą przyjmować wartości 0 lub 1.



Rys. 1. [3] Model neuron logicznego

Powyższy rysunek przedstawia neuron binarny, tego typu neuron posiada zawsze tylko dwa wejścia a blok f realizuje funkcję [3]:

$$y = \begin{cases} 1, & dla \quad (w_1 + 1)x_1 + (w_2 + 1)x_2 \geq 2 \\ 0, & dla \quad (w_1 + 1)x_1 + (w_2 + 1)x_2 < 2 \end{cases} \tag{1}$$

W sytuacji, gdy wartości wag i wejść neonu mogą przyjmować jedynie wartości 0 lub 1 neuron będzie realizował przy danej kombinacji stanów wag (W) jedną z czterech funkcji boolowskich: AND, LEFT, RIGHT lub OR. Sieci konstruowane z takich neuronów to zwykle sieci wielowarstwowe posiadające jedno wyjście. W sieciach boolowskich każde wejście zwykłe (x_i) jest podwajane przez wejście negowane ($\overline{x_i}$). Wykorzystanie sieci zbudowanej z takich neuronów rozpoczyna się od inicjalizacji, poprzez losowanie wartości wag poszczególnych neuronów. W procesie inicjalizacji sieci, jeszcze przed ustalaniem wartości wag, następuje losowe łączenie każdego neuronu warstwy wstępnej z jednym wejściem prostym i jednym negowanym. Proces uczenia (adaptacji) takiej sieci prowadzony może być według trzech różnych metod: modyfikacji poddawane są wagi neuronów, rozmiar sieci jest redukowany przy nadmiarowości inicjalizowanej sieci lub następuje rozbudowa struktury sieci poprzez dodawanie pojedynczych neuronów lub małych podsieci. Więcej na ten temat można znaleźć w literaturze [2],[3]. Wytrenowana sieć boolowska, ma tę zaletę, że łatwo implementuje się tego rodzaju struktury, również za pomocą rozwiązań sprzętowych. W naszych rozważaniach, o których mowa w dalszych rozdziałach, wykorzystujemy neuron logiczny, jednak odchodzimy nieco od koncepcji sieci boolowskiej z tylko jednym wyjściem oraz z wejściami negowanymi.

3. Realizacja permutacji za pomocą boolowskiej sieci neuronowej

W naszych wcześniejszych pracach przedstawiliśmy propozycję realizacji permutacji z wykorzystaniem sieci neuronowej opartej na neuronie posiadającym wagi o wartościach rzeczywistych

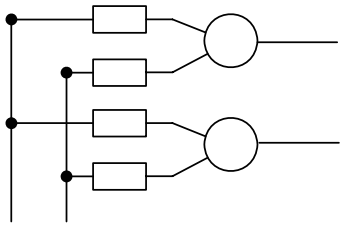
oraz z progową funkcją aktywacji. W tej pracy zamierzamy zaprezentować pomysł na realizację elementarnego przekształcenia szyfrującego, jakim jest permutacja z wykorzystaniem neuronowej sieci boolowskiej. Zastosowanie sieci boolowskiej stwarza nieco inne warunki co do konstrukcji układu permutującego. Mowa tu głównie o tym, iż neurony logiczne posiadają zawsze tylko dwa wejścia i jedno wyjście. Zasadę działania sieci logicznej przedstawiliśmy w wcześniejszym rozdziale. Zakładamy, że skonstruujemy pewną pulę „neuronowych bloków szyfrujących", każdy z nich będzie dawał możliwość realizacji określonej permutacji na danej liczbie bitów. Posiadając takie bloki elementarne (sieci neuronowe) będzie można ich używać do budowy większych sieci, realizujących permutacje dla bloku o określonym rozmiarze tekstu jawnego.

3.1. Permutacja dwóch bitów

Założenie jest takie: skonstruować blok, który będzie realizował dowolną permutację na dwóch bitach tekstu jawnego. W celu konstrukcji takiego bloku należy zbudować sieć neuronową, która będzie posiadała dwa wejścia i dwa wyjścia. Sieć ma umożliwić realizację następujących permutacji:

$$\sigma_A = \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \qquad 2) \qquad , \qquad \sigma_B = \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} \qquad (3)$$

σ_A, σ_B - to permutacja, x,y – odpowiednio: wejście, wyjście układu permutującego. Posługując się odpowiednio skonstruowanym zbiorem uczącym chcemy mieć możliwość decydowania o tym czy sieć będzie realizowała permutacje σ_A czy σ_B. Na poniższym rysunku przedstawiona jest sieć zbudowana z dwóch neuronów boolowskich, realizująca permutację dwóch bitów.



Rys. 2. Sieć realizująca permutację dwóch bitów.

Sieć składa się dwóch takich samych neuronów, których działanie opisane jest przez wzór (1). Zmiana realizowanej permutacji, pomiędzy σ_A i σ_B dokonywana jest z wykorzystanie zbiorów uczących :

permutacja σ_A : $\begin{matrix} x_1 & x_2 & y_1 & y_2 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{matrix}$, permutacja

σ_B : $\begin{matrix} x_1 & x_2 & y_1 & y_2 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$

W wyniku przeprowadzonego procesu uczenia zmianie ulegają wartości wag dla:

σ_A opisanej wzorem (2) wagi sieci: $w_1^1 = 1; w_2^1 = 0; w_1^2 = 0; w_2^2 = 1$ (4)

σ_B opisanej wzorem (3) wagi sieci: $w_1^1 = 0; w_2^1 = 1; w_1^2 = 1; w_2^2 = 0$ (5)

Proces uczenia przebiega według zależności:

d– różnica pomiędzy dopowiedzią wzorcową sieć a rzeczywistą

y_w - odpowiedź wzorcowa sieci

y – odpowiedź rzeczywista sieci $d = y_w - y$

i – nr wejścia neuronu , j – nr neuronu $w_i^j(n+1) = w_i^j(n) + d * X_i^j$

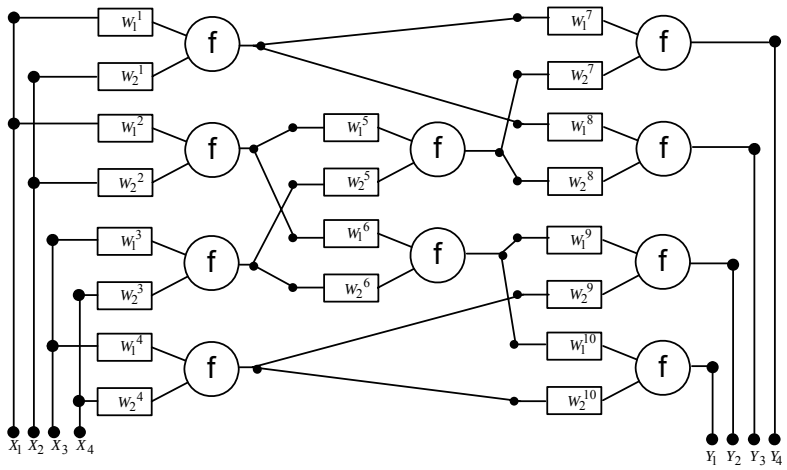
x – wejście neuronu

w – waga neuronu, (n+1) – cykl uczący bieżący a (n) poprzedni

Zmiana realizowanej permutacji odbywa się poprzez przeprowadzenie uczenia sieci z wykorzystaniem odpowiedniego zbioru uczącego.

3.2. Permutacja czterech bitów

W naszych pracach [4], [6], gdzie wykorzystywaliśmy sieci neuronowe oparte na neuronach McCulloch-Pittsa [1] oraz regule Hebba, przedstawiliśmy koncepcję realizacji bloku dokonującego permutacji trzech bitów. W przypadku sieci wykorzystującej neurony logiczne, które mogą posiadać tylko dwa wyjścia, jako drugi proponujemy neuronowy blok permutacji czterech bitów. Poniżej przedstawiona jest neuronowa sieć boolowska, realizująca permutację czterech bitów:



Rys. 3. Sieć realizująca permutację czterech bitów

Z powodów ograniczeń objętościowych tej pracy, przedstawiamy tu tylko jako przykład zbiór uczący dla jednej permutacji na czterech bitach.

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
0	1	0	0	0	0	1	0
0	1	0	1	0	0	1	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1

$$\sigma_A = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix} \tag{7}$$

Po przeprowadzonym procesie uczenia, zbiorem zapisanym powyżej, sieć przyjmie wartości wag dla permutacji σ_A opisanej wzorem (7) :

$$\begin{aligned} &w_1^1 = 0; w_2^1 = 1; w_1^2 = 1; w_2^2 = 0; w_1^3 = 0; w_2^3 = 1; w_1^4 = 1; w_2^4 = 0; w_1^5 = 1; w_2^5 = 0; \\ &w_1^6 = 0; w_2^6 = 1; w_1^7 = 0; w_2^7 = 1; w_1^8 = 1; w_2^8 = 0; w_1^9 = 0; w_2^9 = 1; w_1^{10} = 1; w_2^{10} = 0. \end{aligned} \tag{8}$$

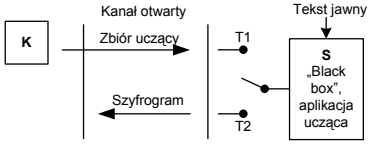
Sieć neuronowa przedstawiona na rys.3, opiera swoje działanie na tych samych neuronach jakie wykorzystano w sieci dla dwóch bitów, (rys.1 zwór (1)).

4. Bezpieczne zmiany realizowanego algorytmu przez sieć neuronową

Zmiana algorytmu – szyfru realizowanego przez sieć dokonuje się poprzez prowadzony w określonych warunkach proces uczenia. Dla uproszczenia, w wszystkich kolejnych rozważaniach, zakładamy, że układ szyfrujący realizuje jedynie permutację bitów, tak jak zostało to opisane powyżej. Przyjmując założenie, że skonstruowaną odpowiednio sieć traktujemy jako czarną skrzynkę umieszczoną na przykład gdzieś na odległym serwerze, konieczna jest możliwość zdalnego wpływania na realizowane przez nią przekształcenie. Przyjmując założenie, że rozważany układ pracuje w układzie klient serwer. Rolę klienta będzie pełnił „właściciel" czarnej skrzynki, który będzie chciał decydować, o tym, jakie przekształcenie szyfrujące ma realizować owa czarna skrzynka.

4.1. Uczenie po stronie serwera

Decydując się na przeprowadzenie procesu uczenia przez serwer (S), niezbędne jest dostarczenie zbioru uczącego skonstruowanego po stronie klienta (K) do aplikacji odpowiedzialnej za przeprowadzenie procesu uczenia. Na rys 4. przedstawiony jest schematycznie, wspomniany wcześniej układ K-S. Układ S może pracować w dwóch trybach. Uczenia (tryb T1) oraz (T2), czyli szyfrowanie tekstu jawnego. Poprzez kanał otwarty rozumiemy sieć, która nie jest chroniona np. Internet.



Rys. 4. Uczenie po stronie serwera

W celu spowodowania zmiany algorytmu realizowanego przez sieć neuronową zbiór uczący musi zostać dostarczony do aplikacji serwera obsługującej proces uczenia sieci. Jeśli przesłany zostanie kompletny zbiór uczący to intruz na podstawie jego analizy będzie w stanie ustalić reali-

zowaną permutację, pod warunkiem znajomości struktury sieci. Nie jest to, więc najlepsze rozwiązanie.

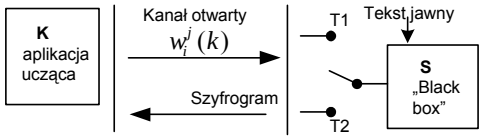
4.2 Uczenie po stronie klienta – przesyłanie wartości wag (funkcja XOR)

Alternatywą dla tego rozwiązania jest przeprowadzenie procesu uczenia po stronie klienta. Do serwerów natomiast przesyłane są nowe parametry sieci. Jeśli przyjmiemy, że pojawienie się w otwartym kanale wymiany informacji nowych wartości wag jest również problematyczne z punktu widzenia bezpieczeństwa to można utrudnić całą operację potencjalnemu intruzowi. Zmuszając go do prowadzenia nasłuchu ciągłego i gromadzenia przechwyconych informacji. Przed wysłaniem nowych wartości wag do serwera poddajemy je działaniu funkcji XOR z wartościami poprzednimi wag.

w_i^j(k) – wartości wysyłane do serwera (n+1) – nowa wartość wagi, (n)wartość poprzednia pozostałe oznaczenia jak we wzorze (6)

$$w_i^j(n+1) \oplus w_i^j(n) = w_i^j(k) \tag{9}$$

$$w_i^j(n+1) = w_i^j(k) \oplus w_i^j(n) \tag{10}$$



Rys. 5. Uczenie po stronie serwera

Tak więc na rysunku nr 5 pokazane jest, że w otwartym kanale wymiany informacji pojawia się jedynie wynik operacji XOR na wartościach poprzednich wag oraz nowych (9). Po stronie (S) przeprowadzona jest operacja odwrotna (10). Na podstawie wykonani funkcji XOR na obecnych wartościach wag oraz otrzymanym ciągu bitów $w_i^j(k)$ otrzymujemy nowe parametry sieci. Tak, więc osoba podsłuchująca transmisję będzie musiała posiadać wiedzę na temat stanu początkowego. Ponadto konieczne jest w takim wypadku prowadzenie nasłuchu ciągłego w celu gromadzenia informacji na temat kolejnych stanów sieci. Jeżeli założymy, że stan początkowy sieci, ustalany będzie w tajemnicy pomiędzy stroną klienta i serwera, to utrzymanie w tajemnicy struktury sieci nie będzie już warunkiem koniecznym.

5. Podsumowanie

Praca ta to rozważania na temat potencjalnych możliwych zaistnienia trudności i zagrożeń dla przeprowadzenia w bezpieczny sposób procesu zmiany realizowanego szyfry przez sieć neuronową. Ograniczyliśmy się tutaj jedynie do realizacji operacji permutacji z wykorzystanie sieci neuronowej boolowskiej. Jak zostało to przedstawione w powyższych rozdziałach przesyłanie kanałem otwartym zbioru uczącego nie jest rozwiązaniem najlepszym. Rozwiązanie zaproponowane w rozdziale 4.2 wydaje się lepszym z punktu widzenia bezpieczeństwa. Kolejnym etapem prac, będzie przeprowadzenie, rzeczywistych ataków, dla różnych konfiguracji sieci neuronowej.

Bibliografia

[1] pW.S.Mc Culloch and W.Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, No 5, 1943

[2] W.Armstrong and J.Gecsei, "Adaptation Algorithms for Binary Tree Networks", IEEE Trans. on Systems, Man and Cybernetics, Vol. 9, pp. 276-285, 1979.

[3] P.S.Szczepaniak, „Obliczenia inteligentne szybkie przekształcenia i klasyfikatory”, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2004.

[4] P.Kotlarz, Z.Kotulski, "Application of neural networks for implementation of cryptographic functions", "Multimedia w biznesie i edukacji", Częstochowa 2005

[5] P.Kotlarz, Z.Kotulski, “On application of neural networks for S-boxes design”, Lecture Notes in Artificial Intelligence, LNCS 3528, pp. 243-248, Berlin 2005.

[6] P.Kotlarz, Z.Kotulski, „Metody sztucznej inteligencji we współczesnej kryptografii, Materiały konferencyjne Ploug’05, Zakopane październik 2005.

[7] Emeka Mosanya, Christof Teuscher, “A Reconfigurable and Modular Cryptographic Coprocessor”, Lecture Notes in Computer Science , Berlin 1999

[8] S.Goldstein, “A High-Perfomance Flexible Architecture for Cryptography, Proceedings” of the Workshop on Cryptographic Hardware, August 1999

[9] Lim C.H, “CRYPTON: A New 128-bit Block Cipher”, Proceedings of the First Advanced Encryption Standard Candidate Conference, California, August 1998

[10] W.Laskowski, Układy programowalne jako narzędzia wspomagające kryptograficzną ochronę danych, Przegląd telekomunikacyjny, nr 3/2001

[11] Lars R. Knudsen Department of Informatics, University of Bergen, N-5020 Bergen : Correlations in RC6, July 29, 1999