Piotr Kotlarz
Instytut Mechaniki Środowiska i Informatyki Stosowanej
Uniwersytet Kazimierza Wielkiego w Bydgoszczy
Zbigniew Kotulski
Instytut Podstawowych Problemów Techniki PAN w Warszawie
Instytut Telekomunikacji PW

Neuronowa realizacja nieliniowych przekształceń szyfrujących

W tej pracy prezentujemy kolejny etap naszych badań dotyczący implementacji symetrycznych algorytmów szyfrujących za pomocą sieci neuronowych. We wcześniejszych naszych pracach [1], [2], [3] przedstawiona została ogólna koncepcja wykorzystania sieci neuronowych do realizacji szyfrów blokowych ze szczególnym uwzględnieniem implementacji permutacji. W tej pracy prezentujemy neuronową realizację nieliniowych funkcji S-blok (S-box).

1. Wprowadzenie

Przekształceniem szyfrującym powszechnie występującym we współczesnych algorytmach szyfrujących jest nieliniowe podstawienie. Podstawienie realizowane jest za pomocą S-bloków. Działanie S-bloku może być interpretowane jako działanie wybierania z tablicy ustalonej wartości przechowywanej w komórce o określonym adresie. S-bloki stosowane w praktycznych implementacjach składają się z tablicy o kilku kolumnach i wierszach. W przypadku algorytmu DES jest to tablica w wymiarach 4 wiersze i 15 kolumn. Adres pojedynczej komórki składa się z 6 bitów, dwa skrajne określają numer wiersza, a 4 środkowe numer kolumny. Jeśli natomiast przyjrzymy się aktualnemu standardowi szyfrowania, jakim jest szyfr AES, to stosowane tam S-bloki stanowią tablicę o szesnastu wierszach i kolumnach. Adres pojedynczej komórki w takim S-bloku składa się z ośmiu bitów. W dalszej części pracy przedstawiona zostanie propozycja realizacji za pomocą sieci neuronowej S-bloku wykorzystywanego w algorytmie DES. Analogicznie można skonstruować S-blok szyfru AES, jednak z powodu jego większego wymiaru takie przekształcenie jest reprezentowane przez znacznie bardziej skomplikowaną sieć neuronową.

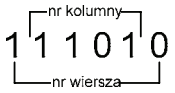
2. Realizacja jednego wiersza z S-bloku algorytmu DES

Chcąc zrealizować za pomocą sieci neuronowej S-blok z algorytmu DES, problem dzielimy na dwa etapy. Pierwszy to realizacja pojedynczego wiersza, drugi to rozbudowa zaproponowanej metody do pełnego S-bloku (4 wiersze). Jeśli rozpatrzymy najpierw problem realizacji jednego wiersza, to sieć neuronowa powinna dać możliwość realizacji wybierania wskazanej wartości z tablicy. W tabeli 1 pokazany jest S-blok numer 1 algorytmu DES; pierwszy wiersz tej tabeli jest przedmiotem rozważań w tym podrozdziale.

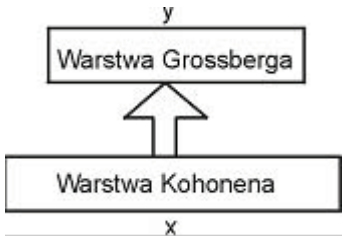
Tabela 1. Zawartość S-blok nr 1 algorytmu DES.

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Ciąg wejściowy S-bloku to 6 bitów:



Tak więc wobec powyższego, adresowanie komórki w obrębie jednego wiersza odbywa się z wykorzystaniem 4 bitów. Pierwszym zadaniem jest rozwiązanie problemu realizacji za pomocą sieci neuronowej wybierania konkretnej wartości z tablicy 1x16. W literaturze dotyczącej sieci neuronowych znana jest koncepcja sieci CP [4], tak zwanej sieci przesyłającej żeton. Ogólna budowa tej sieci przedstawiona jest na rys. 1.

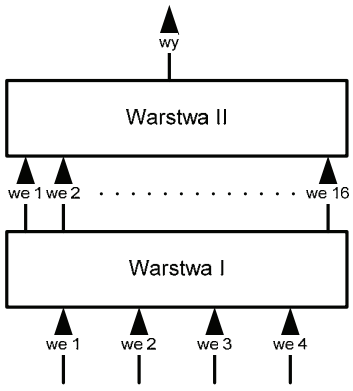


Rys. 1. Koncepcja budowy sieci CP.

Sieć ta składa się z dwóch warstw realizujących odrębne zadania. Warstwa Kohonena działa na zasadzie konkurencyjności. Wartość „1" pojawia się na wyjściu jednego neuronu.

$$y_i = \begin{cases} 1 & dla \max(\varphi_i) \\ 0 & w \quad inny \quad wypadku \end{cases}, \; gdzie \; \varphi_i = x * w^T.$$

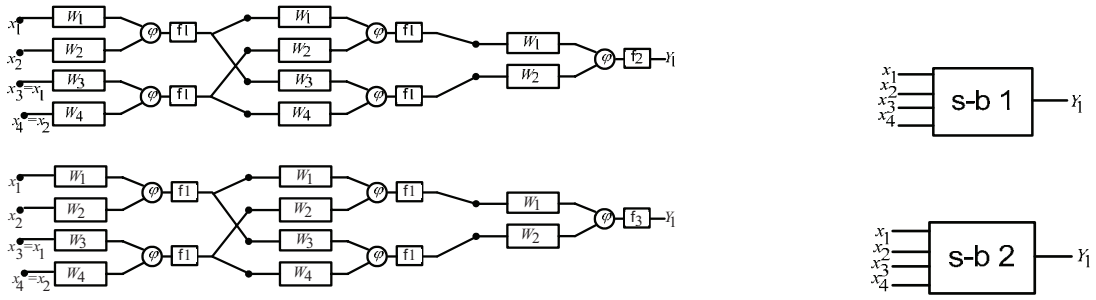
Druga warstwa sieci jest adresowana przez warstwę pierwszą. O wartości na wyjściu sieci CP ostatecznie decyduje warstwa Grossberga. Pomysł na realizację pojedynczego wiersza S-bloku za pomocą sieci neuronowej bazuje na ogólnej koncepcji sieci CP. Opisując proponowaną konstrukcję sieci neuronowej dla przejrzystości opisu założono, że sieć ma udzielać odpowiedzi w postaci dziesiętnej. W toku badań i eksperymentów okazało się, że to, czy sieć ma udzielać odpowiedzi w postaci binarnej czy dziesiętnej, jest kwestią konstrukcji ostatniej warstwy sieci. Wychodząc więc z ogólnej koncepcji sieci CP, strukturę układu neuronowego realizującego jeden wiersz S-bloku można przestawić za pomocą schematu blokowego, pokazanego na rys. 2:



Rys. 2. Schemat budowy sieci realizującej pierwszy wiersz S-bloku.

Warstwa I miałaby realizować klasyfikację obiektów określanych poprzez odpowiednią kombinację 4 bitów na 16 różnych klas (pierwszy wiersz S-bloku zawiera 16 komórek). Skupiając się na pierwszej warstwie, chodzi tu o konstrukcję sieci neuronowej, która dla różnych kombinacji czterech bitów wejściowych dla poszczególnych sygnałów będzie odpowiadała jedynką tylko na jednym wyjściu. Wykorzystane do tego zostaną 4 sieci neuronowe, z których każda będzie realizowała klasyfikację obiektu na 4 klasy przy 4 cechach podawanych na wejściu. Podobnie jak

było to rozwiązane w przypadku realizacji permutacji [1], [2] również i przy neuronowej realizacji S-bloku stosowane są do budowy bloki neuronowe, które można nazwać podsieciami. Rozwiązanie takie zostało przyjęte z dwóch powodów. Pozwala w czytelnej i zwartej formie przedstawiać strukturę sieci, a modułowość ułatwia implementację. Ponadto modułowa budowa ułatwia ewentualną rozbudowę rozmiaru S-bloku. Tak więc sieć pozwalająca na klasyfikację na 4 klasy wykorzystuje dwa bloki elementarne, przedstawione na rys. 3.

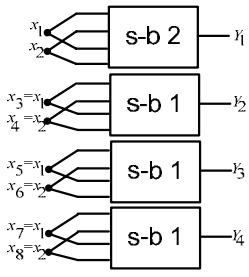


Rys. 3. Neuronowe klasyfikatory elementarne.

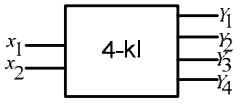
Bloki „s-b 1" oraz „s-b 2" składają się z neuronów z trzema różnymi funkcjami aktywacji:

$$f_1 = \varphi,\ f_2 = \begin{cases} 0, & \varphi \le p_1 \\ 1, & \varphi > p_1 \end{cases}, \text{ oraz } f_3 = \begin{cases} 0, & \varphi \ge p_2 \\ 1, & \varphi < p_2 \end{cases},$$

gdzie: $\varphi = \sum w_i x_i$, $y_j = f_k(\varphi)$, $p_1 = 0{,}5$, $p_2 = 0{,}2$, f – funkcja aktywacji, i – numer wejścia/wagi neuronu, φ_n - potencjał membranowy, p - próg funkcji aktywacji, y_j - wyjście kolejnego neuronu, k - numer funkcji aktywacji. Korzystając z układu powyższych bloków, rys. 3, tworzymy sieć, która działa według tablicy prawdy pokazanej poniżej. Zgodnie z przyjętą zasadą konstrukcji bloków elementarnych sieć realizująca poniższą tablicę prawdy oznaczona zostanie jako „4-kl". Jest ona pokazana na rys. 4.

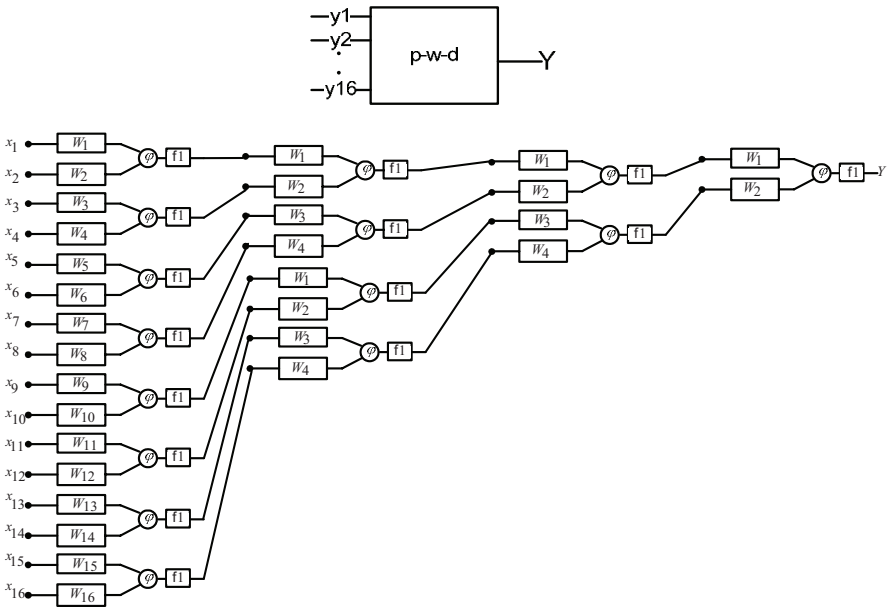


x1	x2	y1	y2	y3	Y4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



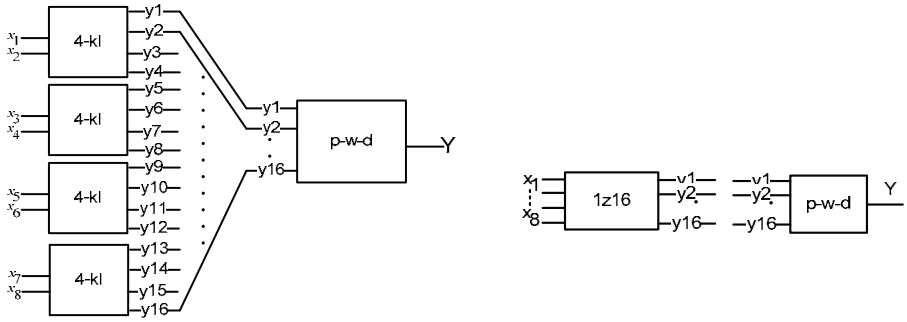
Rys. 4. Realizacja rozpoznawania czterech obiektów.

Można oczywiście dyskutować z faktem złożoności tej sieci realizującej dość proste zadanie. Jednak taki układ wydaje się optymalny pod względem dalszych jego zastosowań i wymogów stawianych przed neuronowym układem szyfrującym do budowy którego zmierzamy. Na tym etapie rozważań druga warstwa sieci ma realizować podawanie konkretnej wartości z zakresu (0-15). Po przeprowadzeniu wielu prób i eksperymentów dobrany został kształt sieci realizujący podawanie 16 różnych wartości dziesiętnych na wyjściu. Sieć ta pokazana jest na rys. 5. Ponieważ na wyjściu sieci nie znajduje się funkcja progowa, to ważne jest takie dobranie struktury sieci, aby uzyskane odpowiedzi sieci były podawane z odpowiednią dokładnością.



Rys. 5. Sieć realizująca podawanie na wyjściu wartości od 0 do 15.

W trakcie dotychczasowych prac został rozwiązany problem realizacji za pomocą sieci neuronowej problemu adresowania komórek wiersza S-bloku oraz rozwiązany problem podawania na wyjściu sieci wartości stanowiącej zawartość komórki o danym adresie. Można teraz przystąpić do złożenia sieci realizującej działanie pojedynczego wiersza S-bloku. Jej schemat ideowy przedstawiono na rys. 6.

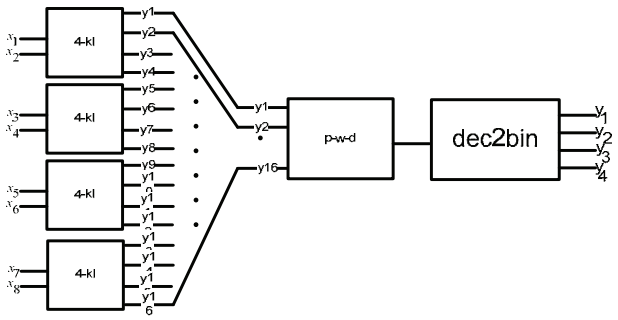


Rys. 6. Sieć realizująca jeden wiersz w S-bloku.

Sieć pokazana na rys. 6 realizuje funkcję pojedynczego wiersza S-bloku algorytmu DES. Można ją jednak przyjąć jako pierwszą wersję wstępną, bardziej jako konstrukcję teoretyczną, ponieważ ma ona 8 wejść. W praktycznym rozwiązaniu komórki w wierszach S-bloku mają adresy 4 bitowe. W rozwiązaniu ostatecznym zrealizowane zostało to poprzez odpowiednie połączenie 4 wejść całego układy (S-bloku) z ośmioma wejściami układu czterech modułów „4-kl”.

3. Neuronowy dekoder wartości dziesiętnych na binarne

Odpowiedź S-bloku jest zawsze w postaci binarnej. Wobec tego koniecznym stało się opracowanie układu neuronowego, który będzie realizował zamianę wartości zmiennoprzecinkowej na binarną. Zakładając, że cały projektowany układ ma być realizowany za pomocą sieci neuronowej, zaistniała więc konieczność konstrukcji neuronowego dekodera. Układ taki na potrzeby przejrzystości dalszego wywodu nazwany został: „dec2bin", rys. 7



Rys. 7. Sieć realizująca jeden wiersz w S-bloku - odpowiedź binarna.

Projektowany układ „dec2bin" posiada jedno wejście, na które podawana będzie odpowiedź pochodząca z układu „p-w-d", realizującego podanie wybranej z wiersza S-bloku wartości w postaci binarnej. „Dec2bin" na wyjściu ma udzielić odpowiedzi w postaci 4 bitowej, binarnej. Tablica prawdy dla projektowanego układu, przedstawiona jest w tabeli 2.

Tabela 2. Tablica prawdy, kodowanie 16 wartości dziesiętnych do postaci binarnej.

x1	y1	y2	y3	y4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Pamiętamy, że głównym celem, do realizacji którego mają doprowadzić badania opisane w ramach tej pracy, jest budowa „neuronowego układu szyfrującego". Funkcje nieliniowe są, obok permutacji, istotnym składnikiem takiego układu. Ważne jest, aby konstrukcja układu szyfrującego była uniwersalna i mogła być w przyszłości rozbudowywana. W tym celu przyjęta została zasada budowy modułowej, dzięki czemu możliwa jest łatwa modyfikacja układu kodującego na przykład ze względu na wielkość zbioru możliwych sygnałów wejściowych, a co za tym idzie długość binarnego sygnału na wyjściu.

Podobnie jak miało to miejsce w poprzednich rozdziałach, pierwszym krokiem w rozważaniach jest sprowadzenie problemu kodowania do realizacji zadania klasyfikacji, typowego dla sieci neuronowych. Jeśli zamiast jednej tablicy prawdy skonstruujemy cztery osobne

tablice zestawiając odpowiednio $x_1 : y_1, x_1 : y_2, x_1 : y_3, x_1 : y_4$, to możemy zaobserwować, że funkcje, które są opisywane przez te tabele realizują podział sygnału wejściowego na dwie klasy. Tak więc zadanie konstrukcji neuronowego układu kodującego polegać będzie na konstrukcji czterech odrębnych sieci, które będą realizowały poszczególne funkcje. Ich charakterystyki zapisane zostały w tabeli 3 oraz na rys. 8.

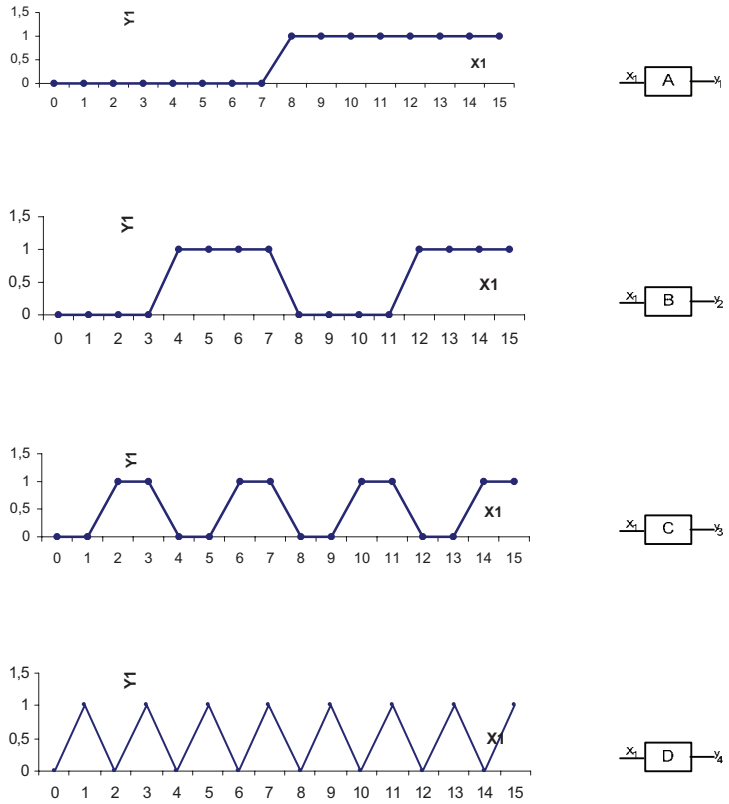
Tabela 3. Tablice prawdy dla układu „dec2bin".

A	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

B	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

C	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

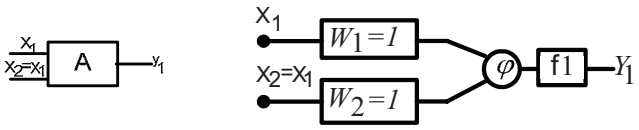
D	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1



Rys. 8. Bloki elementarne sieci „dec2bin".

3.1. Realizacja bloku A

Ten problem jest stosunkowo prostym do realizacji neuronowej, nie wymaga nawet przeprowadzania procesu uczenia i wystarczy pojedynczy neuron, przedstawiony na rys. 9.

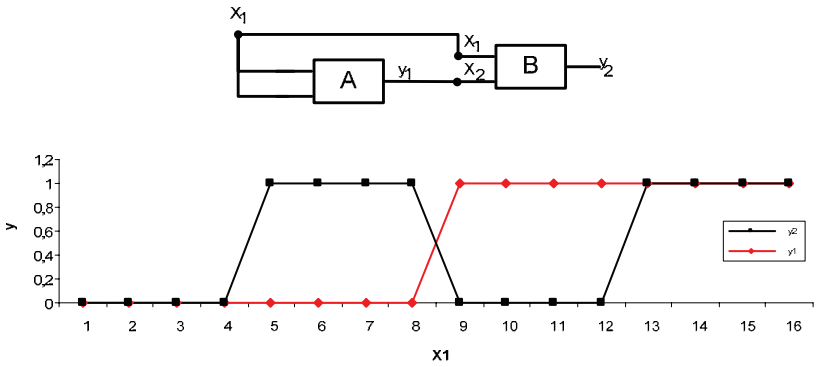


$$f = \begin{cases} 0, & \varphi \le p_1 \\ 1, & \varphi > p_1 \end{cases}, \quad \varphi = \sum w_i x_i, \quad p_1 = 7$$

Rys. 9. Sieć realizująca blok A.

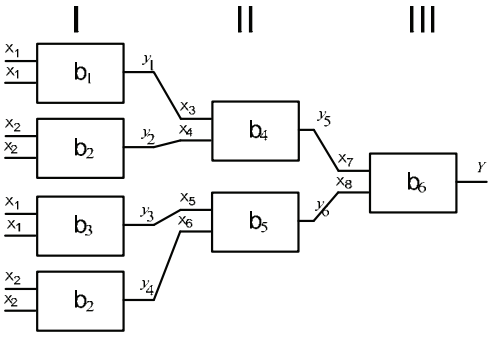
3.2. Realizacja bloku B

W tym przypadku zadanie jest nieco bardziej złożone, jeden neuron już nie wystarczy. Punktem wyjścia do rozważań jest zweryfikowanie, jakie informacje będą podawane na wejście projektowanego układu. Na pewno na wejście sieci „*B*” podawana będzie wartość zmiennoprzecinkowa (ta sama, która jednocześnie znajdzie się na wejściu układu „*A*”). Ponadto do dyspozycji mamy też odpowiedź układu „*A*”. Układ dokonuje podziału wartości zmiennoprzecinkowej na dwie grupy. Sieć, która jest obecnie przedmiotem rozważań, ma za zadanie również dzielenie sygnału wejściowego na grupy. Jak widać na rys. 10, odpowiedź układu „*B*” jest zależna od wartości uzyskanej na wyjściu bloku „*A*”.



Rys. 10. Zależność pomiędzy układami B i A.

Ta zależność właśnie zostanie wykorzystana do budowy układu *B*. Problem można więc rozpatrywać w taki sposób, że mamy podział zbioru wartości sygnału wejściowego na dwie grupy (moduł „*A*”) oraz w każdej grupie podział na dwie podgrupy (moduł „*B*”). W celu realizacji modułu „*B*” skonstruowana została trzywarstwowa siec neuronowa, por. rys. 11 i tabela 4.



Rys. 11. Budowa modułu B.

Tabela 4. Tablice prawdy dla modułu B.

Warstwa I

x1	x2	y1	y2	y3	y4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1	0	0	0
5	0	1	0	0	0
6	0	1	0	0	0
7	0	1	0	0	0
8	1	1	1	0	1
9	1	1	1	0	1
10	1	1	1	0	1
11	1	1	1	0	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1

Warstwa II

x3	X4	x5	x6	Y5	Y6
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1	0	0	0	1	0
1	0	0	0	1	0
1	0	0	0	1	0
1	0	0	0	1	0
1	1	0	1	0	0
1	1	0	1	0	0
1	1	0	1	0	0
1	1	0	1	0	0
1	1	1	1	0	1
1	1	1	1	0	1
1	1	1	1	0	1
1	1	1	1	0	1

Warstwa III

x7	x8	Y
0	0	0
0	0	0
0	0	0
0	0	0
1	0	1
1	0	1
1	0	1
1	0	1
0	0	0
0	0	0
0	0	0
0	0	0
0	1	1
0	1	1
0	1	1
0	1	1

B

x1	x2	Y
0	0	0
1	0	0
2	0	0
3	0	0
4	0	1
5	0	1
6	0	1
7	0	1
8	1	0
9	1	0
10	1	0
11	1	0
12	1	1
13	1	1
14	1	1
15	1	1

Tabela 4 zawiera tablice prawdy dla poszczególnych warstw składających się na sieć neuronową modułu B. Na poszczególne warstwy składają się moduły elementarne, rys. 11, których zasada działania zostanie teraz przestawiona. Moduł b_2 to neuron, którego zadaniem jest przekazanie na wyjście tego samego sygnału, jaki podawany jest na jego wejście. Moduły b_1 i b_3 to pojedyncze neurony realizujące funkcję progową, odpowiednio z progiem $p = 3$ oraz $p = 1$. Elementy b_1, b_2, b_3 nie wymagają procesu uczenia, ich wagi przyjmują wartości 0 i 1. Elementy b_4, b_5, b_6 wymagają już przeprowadzenia procesu uczenia. Do realizacji poszczególnych elementów wystarczy również jeden neuron posiadający dwa wejścia i jedno wyjście. W tabeli 5 przedstawione zostały tablice prawdy, jakie mają realizować poszczególne bloki b_4, b_5, b_6. Jednocześnie są one reprezentacją zbiorów uczących, z tym, że dla polepszenia jakości procesu uczenia w zbiorach uczących wartości „0" zostały zastąpione wartościami „-1".

Tabela 5. Tablice prawdy dla modułów b4, b5, b5.

b4

x1	x2	Y
0	0	0
0	1	0
1	0	1
1	1	0

b5

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

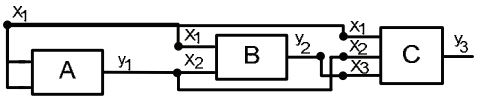
b6

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Tak więc układ składający się z elementów b1 do b6 realizuje funkcję zapisaną jako tabela prawdy układu B.

3.3. Realizacja bloku C

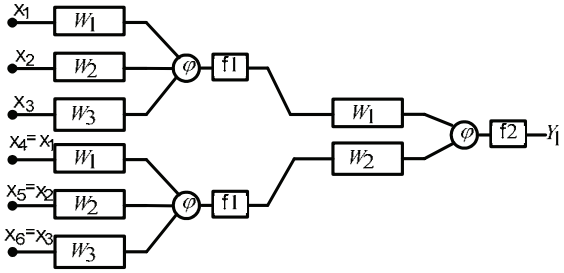
Realizacja kolejnego bloku wymaga użycia sieci neuronowej składającej się z dwóch warstw. Proces uczenia sieci przeprowadzany jest dla całej sieci od razu, bez potrzeby „rozbijania" na podbloki. Na tym etapie realizacji kodera „dec2bin" na wejściu bloku „C" dysponujemy trzema różnymi wartościami. Dostępna jest wartość zmiennoprzecinkowa, wynik działania bloku A i bloku B, co obrazuje rys. 12.



C	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	X2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	y3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

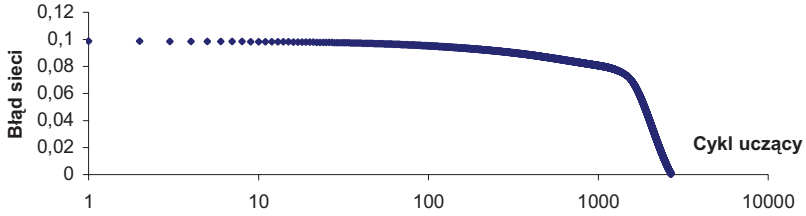
Rys. 12. Schemat działania modułu C.

Szczegółowa konstrukcja sieci pokazana jest na rys. 13. Rys. 14 przedstawia wykres odzwierciedlający prędkość zbieżności w procesie uczenia sieci.



$$f_1 = \varphi,\ f_2 = \begin{cases} 0, & \varphi \geq p_1 \\ 1, & \varphi < p_1 \end{cases}, \text{ gdzie: } \varphi = \sum w_i x_i,\ y_j = f_k(\varphi),\ p_1 = 0{,}5,$$

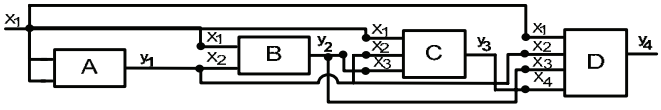
Rys. 13. Sieć neuronowa realizująca moduł C.



Rys. 14. Przebieg procesu uczenia sieci neuronowej realizującej moduł C.

3.4. Realizacja bloku D

Ostatni blok neuronowego kodera wartości dziesiętnych do postaci binarnej zrealizowany został zgodnie z taką samą ideą jak blok C. Różnicą jest tu fakt, że w pierwszej warstwie sieci wykorzystane zostały dwa neurony posiadające cztery wejścia, w celu wykorzystania czterech sygnałów pochodzących z wyjść układów A, B i C oraz wartości dziesiętnej, która poddawana jest kodowaniu.

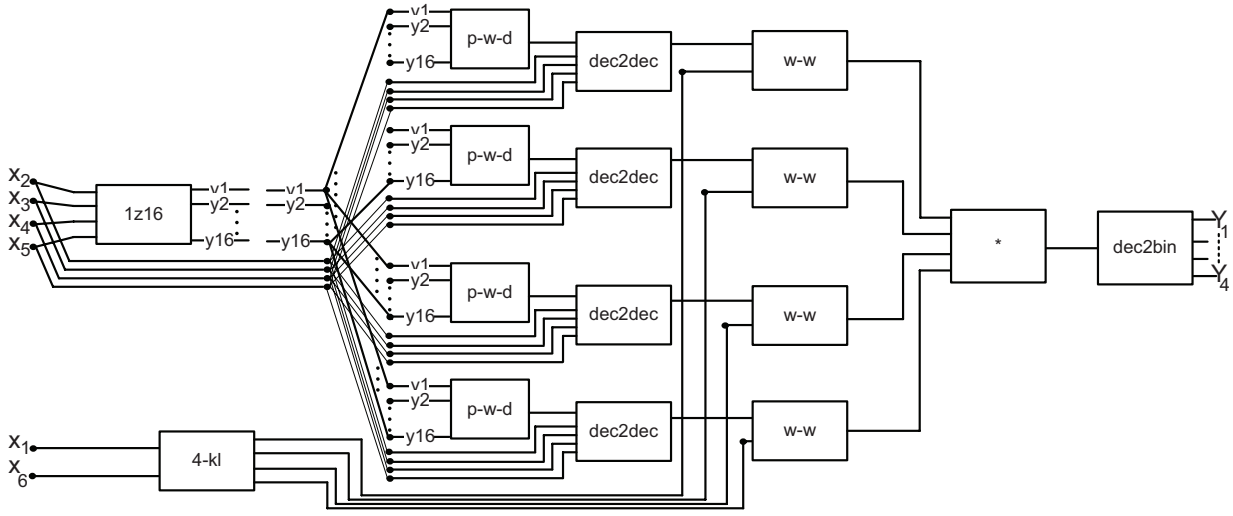


Rys. 15. Schemat układu „dec2bin”.

Tworząc sieć z przygotowanych bloków elementarnych A, B, C i D uzyskujemy sieć neuronową, rys. 14, która realizuje kodowanie wartości zmiennoprzecinkowych do postaci binarnej czterobitowej.

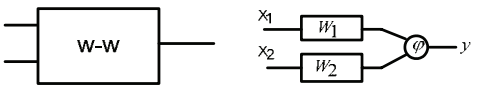
4. Realizacja kompletnego S-bloku

W celu realizacji kompletnego S-bloku niezbędne jest rozwiązanie problemu adresowania wierszy. W 6 bitowym sygnale wejściowych S-bloku dwa skrajne bity wskazują numer wiersza z którego następnie wybierana jest konkretna wartość 1 z 16. Ponieważ w algorytmie DES S-bloki posiadają cztery wiersze, to mamy tu problem wyboru 1 z 4. Problem budowy sieci realizującej podział na cztery klasy został opisany we wcześniejszym rozdziale tej pracy. Neuronowy układ realizujący pełen S-blok pokazany jest na rys. 16.



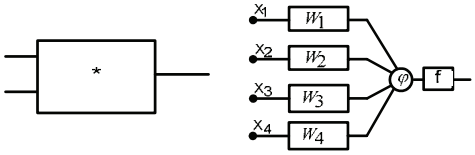
Rys. 16. Neuronowy układ realizujący S-blok algorytmu DES.

Wyjaśnienia wymaga jeszcze kilka szczegółów z powyższego rys. 16. Moduł „w-w" składa się z pojedynczego neuronu, który realizuje funkcję:



$$y = \begin{cases} 0, & x_2 = 0 \\ x_1, & x_2 = 1 \end{cases}$$

Z sygnałów (2 bitowych), jakie mogą się pojawić na wejściu układu „4-kl" wynika, że wartość „≠0" pojawi się tylko na wyjściu jednego z czterech układów „w-w", na pozostałych pojawi się „0". Tą jedną wartością będzie element wskazany przez sygnał wejściowy S-bloku. Kolejnym elementem całego układu jest moduł „*". Jego zadaniem jest przekazanie na wejście modułu „dec2bin" wartości, jaka pojawi się na wyjściu jednego z układów „w-w". Do realizacji tego zadania również wystarczy jeden neuron, tym razem posiadający 4 wejścia, rys. 17.



$$f = \varphi,\ \varphi = \sum w_i x_i$$

Rys. 17 Schemat układu „".*

Kluczowym założeniem, jakie poczynione zostało na początku badań dotyczących neuronowej implementacji szyfrów (a więc i S-bloków), jest aby rozwiązania było uniwersalne w tym sensie, że zmiana szyfru (S-bloku) na inny polega na zmianie wartości wag bez zmieniania struktury układu sieci neuronowej. Cel ten został osiągnięty, proponowany układ realizujący konkretny S-blok wymaga przeprowadzenia procesu uczenia w określonych warunkach, a nie wymaga przebudowy struktury sieci neuronowej.

5. Podsumowanie

Zastosowanie metod sztucznej inteligencji w kryptologii ma już pewna historię. Pierwsze prace dotyczyły wykorzystania algorytmów ewolucyjnych do ataków na proste szyfry podstawieniowe [5], [6]. W pracy [7] zostały porównane metody wykorzystujące algorytmy genetyczne do ataków na szyfry wykorzystujące permutacje, podstawienia i transpozycje. Jedną z ciekawszych cytowanych tam koncepcji jest wykorzystanie algorytmów genetycznych do ekstrakcji klucza z szyfru opartego na koncepcji Feistela [8]. W 1996 roku pojawiła się propozycja użycia algorytmów genetycznych do ataku na kryptosystem Merkle-Hellmana [9]. W 1997 roku [10] zaproponowano procedurę wykorzystania metod genetycznych do przeszukiwania dużych przestrzeni kluczy dla czterowirnikowej maszyny rotorowej. Celem tej pracy było przedstawienie możliwości implementacji szyfrów blokowych za pomocą sieci neuronowych. Pokazano, jak zrealizować skrzynkę S-box algorytmu DES. Założony cel, budowy uniwersalnego układu neuronowego realizującego dowolny S-blok został osiągnięty. Kolejnym etapem badań będzie ocena wydajności proponowanego rozwiązania w procesie uczenia i działania szyfru oraz identyfikacji potencjalnych możliwości praktycznego wykorzystania neuronowej implementacji S-bloku.

Literatura

1. P. Kotlarz, Z. Kotulski, On application of neural networks for S-boxes design, in: P. S. Szczepaniak, J. Kacprzyk, A. Niewiadomski [Eds], *Advances in Web Intelligence: Third International Atlantic Web Intelligence Conference*, AWIC 2005, LNCS 3528, pp. 243-248, Springer-Verlag, Berlin 2005. ISBN: 43-540-26219-9.
2. P. Kotlarz, Z. Kotulski, Neural network as a programmable block cipher, in: J. Pejaś, Kh. Saeed [Eds], *Advances in Information Processing and Protection*, pp. 241-250, Springer-Verlag, Berlin 2007. ISBN: 978-0-387-73136-0
3. P. Kotlarz, Z. Kotulski, "Application of neural networks for implementation of cryptographic functions" in: L. Kiełtyka [Ed.], *Multimedia in Business and Education*, vol.1, pp. 213-218, Fundacja Współczesne Zarządzanie, Białystok 2005, ISBN 83-9182218-7-0.
4. S. Osowski, "Sieci neuronowe do przetwarzania informacji", Warszawa 2000.
5. S. Peleg, A. Rosenfeld, “Breaking substitution ciphers using a relaxation algorithm.” *Communications of the ACM*, 22, pp. 598-605, 1979.
6. D. Hunter, A. McKenzie “Experiments with relaxation algorithms for breaking simple substitution ciphers.”, *The Computer Journal*, Vol. 26, pp. 68-71, 1983.
7. B. Delman, “Genetic Algorithms in Cryptography”, Department of Computer Engineering *Thesis (M.S.)*--Rochester Institute of Technology, 2004.
8. R.A.J. Matthews, “The use of genetic algorithms in cryptanalysis.” *Cryptologia*, Vol. 17, pp. 187-201, 1993.
9. D. Dascalu, C. Vertan, C. Geangala, “Breaking the Merkle-Hellman Cryptosystem by Genetic Algorithms: Locality versus Performance, Real World Applications of Intelligent Technologies”, Romanian Academy, 1996 , pp. 201-208.
10. T. Bagnall, G.P. McKeown, V.J. Rayward-Smith, “The cryptanalysis of a three rotor machine using a genetic algorithm.” In: T. Back [Ed.] *Proceedings of the 7th International Conference on Genetic Algorithms* (ICGA97), San Francisco, CA, Morgan Kaufmann, 1997.