XI Konferencja PLOUG
Kościelisko
Październik 2005

Metody sztucznej inteligencji we współczesnej kryptografii

Piotr Kotlarz

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

Zbigniew Kotulski

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

Streszczenie

W praktyce spotykane są różne realizacje (implementacje) algorytmów szyfrujących - ogólnie można je podzielić na realizacje programistyczne oraz sprzętowe. Ogólna koncepcja współczesnego podejścia do realizacji algorytmów szyfrujących polega na „wszyciu" jednego z nich w protokół np. podpisu elektronicznego, w sposób sztywny. Rozumieć przez to należy sytuację taką, że sama implementacja szyfru traktowana jest jak blok zawierający wejście i wyjście. Ewentualnie dodatkowe wejście, które pozwala wpływać na stosowany klucz. Z pozoru jest to sytuacja nie budząca żadnych wątpliwości, o ile został wybrany algorytm bezpieczny, został on właściwie zaimplementowany i spełnione zostało jeszcze kilka warunków, które można na razie pominąć w naszych rozważaniach. Problem pojawia się w momencie, gdy nasze rozwiązanie zostanie rozpowszechnione, może nawet stanie się standardem, po czym bezpieczeństwo wybranego przez nas algorytmu szyfrującego zostanie podważone. Tradycją stało się już w dziedzinie kryptografii, że każdy nowo opublikowany szyfr staje się obiektem licznych ataków. Złamanie algorytmu szyfrującego może spowodować również podważanie bezpieczeństwa protokołu, w którym ów algorytm został wykorzystany. Pociąga to oczywiście za sobą konieczność poprawienia naszego protokołu. Stoimy więc przed problemem, do którego rozwiązania można podejść dwojako: wybór innego algorytmu szyfrującego lub dokonanie zmian w algorytmie szyfrującym zalecanych jako poprawa sytuacji przez jego autorów. Zarówno jedna jak i druga sytuacji pociąga za sobą konieczność dokonania programistycznych zmian w naszym protokole. Najczęściej konieczne jest stworzenie nowej wersji, co oczywiście jest sprawą kosztowną. Jeśli mamy do czynienia z implementacją sprzętową, wtedy koszty oczywiście będą znacznie większe. Mówimy tu o kosztach samego przeprogramowania, dystrybucji nowej wersji oraz o czasie, jaki jest na to potrzebny.

Można by się zastanowić, jaka jest alternatywa. Rozwiązaniem obecnie spotykanym jest zaimplementowanie kilku algorytmów szyfrujących podobnej klasy i danie użytkownikowi możliwości wyboru, który ma być stosowany. Wtedy dezaktualizacja jednego z nich nie powoduje aż tak dużego problemu. Rozwiązanie takie jednak podnosi koszty naszego protokołu. Innym rozwiązaniem, nad którym obecnie prowadzone są badania w różnych ośrodkach, jest zastosowanie układów reprogramowalnych dla rozwiązań sprzętowych.
W naszym referacie zamierzamy zaprezentować propozycję rozwiązania alternatywnego. Proponujemy wykorzystanie do realizacji algorytmów szyfrujących struktur uczących, dokładniej sieci neuronowych. Pomysł nasz dotyczy skonstruowania „bloków", które realizują podstawowe operacje szyfrowania, czyli permutacje oraz podstawienie, włącznie z elementami nieliniowymi (s-boxy). Zastosowanie takiego podejścia do implementacji, w założeniu ma doprowadzić do możliwości dokonania zmian w algorytmie szyfrującym za pomocą odpowiednio skonstruowanego zbioru uczącego. Można by również zaryzykować stwierdzenie, że w przyszłości możliwe stanie się skonstruowanie systemu szyfrowania (nadawca - odbiorca), który będzie sam reagował na wykryte zagrożenia. Problem dokonywania zmian w algorytmach kryptograficznych jest oczywiście bardzo złożony. Nie każda zmiana musi się okazać zmianą na lepsze. Problemem jest więc kontrolowanie dokonywanych zmian oraz ich ocena pod kątem spełniania określonych wymogów. W referacie przedstawimy propozycję wykorzystania sieci neuronowych w procesie oceny implementacji algorytmów szyfrujących.
W referacie pokazane zostaną konkretne przykłady wykorzystania sieci neuronowych do wyżej wymienionych celów. Ponadto przedstawione zostaną znane już i stosowane rozwiązania z zakresu sztucznej inteligencji wspierające narzędzia bezpieczeństwa informatycznego ze szczególnym uwzględnieniem systemów wykrywania intruzów.

Informacja o autorze

Wykształcenie: Akademia Bydgoska, Wydział Matematyki Techniki i Nauk Przyrodniczych, Kier. Technika Komputerowa; Akademia Bydgoska, Wydział Matematyki Techniki i Nauk Przyrodniczych, Studia specjalne z zakresu informatyki.
Doświadczenie zawodowe: Administrator sieci i systemów informatycznych w Zakładzie Energetyki PKP. Współpraca z firmami SEKOM S.A. oraz CSS Kujawy w zakresie sieci informatycznych i okablowania strukturalnego. Prowadzenie i organizacja licznych szkoleń i warsztatów informatycznych (obecnie) Akademia Bydgoska. Pracownik naukowo-dydaktyczny Instytutu Mechaniki Środowiska i Informatyki Stosowanej.
Nauka: Doktorant IPPT Polskiej Akademii Nauk w Warszawie. Udział w konferencjach związanych tematycznie z informatyką stosowaną i teoretyczną. Publikacje w czasopismach i konferencjach naukowych i branżowych. Kierownictwo projektów badawczych z zakresu bezpieczeństwa informatycznego oraz sztucznej inteligencji. Otwarty przewód doktorski w IPPT PAN w Warszawie.

1. Kryptografia

Historia kryptografii sięga czasów starożytnych. Większość już starożytnych cywilizacji wykształciła swoje własne metody tajnego przekazywania informacji. David Kahn w swojej książce [1] prezentuje dzieje kryptografii od jej początków aż po czasy współczesne. Szczególnie intensywny rozwój metod kryptograficznych nastąpił w XX wieku, kiedy to zaczęły powstawać urządzenia szyfrujące, najbardziej znanym z nich jest Enigma. Niebagatelny wpływ na rozwój kryptografii miały również wojny prowadzone w ubiegłym wieku. Okazuje się jednak, że największy rozwój skutecznych i wydajnych metod realizacji usług bezpieczeństwa informacji wymusił postęp technologiczny i wszechogarniająca informatyzacja różnych dziedzin życia. Wraz z nastaniem ery komputerów kryptografowie i kryptoanalitycy otrzymali narzędzie dające potężne możliwości obliczeniowe. Gdy od lat osiemdziesiątych zaczęły się intensywnie rozwijać sieci komputerowe, problem tajności i autentyczności przesyłanych za ich pomocą danych stał się szczególnie istotny.

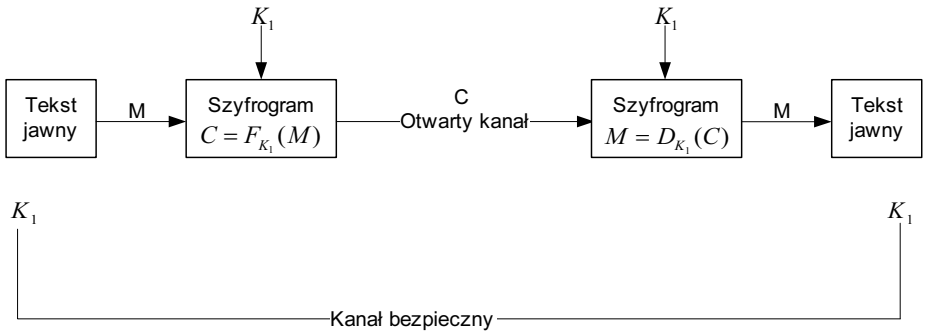
Matematyczne podstawy współczesnej kryptografii określił w 1949 roku Claude Shannon [2][3]. W swoich pracach przedstawił koncepcję zastosowania teorii informacji do konstrukcji i analizy szyfrów. Wprowadzenie teorii informacji oraz komputeryzacja spowodowały zmianę podejścia do problemu szyfrowania. Przed erą komputerów permutacje i podstawienia realizowane była na znakach alfabetu lub grupach takich znaków. W obecnej kryptografii podstawową jednostką operacji kryptograficznych jest bit a nie znak alfanumeryczny. W pracy [3] Shannon wymienia dwie podstawowe operacje elementarne procesu szyfrowania: mieszanie (podstawienie) i rozproszenie (permutacja). We współczesnych szyfrach mieszanie najczęściej realizowane jest poprzez elementy zwane s-blokami, a rozproszenie przez tzw. p-bloki. W latach sześćdziesiątych IBM zainicjował program badawczy w dziedzinie kryptografii komputerowej pod nazwą „Lucyfer". Kierownikiem projektu został Horst Feistel. Efektem prac stał się algorytm szyfrujący pod taką samą nazwą jak nazwa wspomnianego programu badawczego.

15 maja 1972 roku ogłoszony został pierwszy konkurs na standard szyfrowania danych. Wymagania konkursowe oraz inne szczegóły historyczne znaleźć można na stronach internetowych [4] agencji rządowej USA. Jak możemy dowiedzieć się z lektury raportów zamieszczonych na wcześniej wspomnianych stronach WWW, zgłaszanych było wiele algorytmów jednak żaden z nich nie spełniał stawianych wymagań. Dopiero w połowie roku 1974 ośrodek badawczy firmy IBM zgłosił algorytm oparty na wspomnianym wcześniej algorytmie Lucyfer. Po wielu dyskusjach na forum świata naukowego NSA (National Security Agency) ogłosiło 23 listopada 1976 roku algorytm DES jako standard, co zostało opublikowane w [5]. DES doczekał się kilku następców, których konstrukcja oparta była również na permutacjach Feistela oraz s-blokach. Powstały między innymi szyfry FEAL (szybki algorytm szyfrujący) [6] oraz IDEA [7], który stał się międzynarodowym standardem szyfrowania. 2 października 2000 roku rozstrzygnięty został przez Narodowy Instytutu Standardów i Technologii USA (NIST) konkurs AES (Advanced Encryption Standard), czyli konkurs na następcę DES-a. Zwycięzcą został algorytm stworzony przez belgijski zespół algorytm Rijndael. Oparty on został na modelu ogólnym sieci Feistela. Składa się z dwóch algorytmów: szyfrującego i deszyfrującego, wykonujących wzajemnie odwrotne operacje szyfrowania i odszyfrowywania. DES, IDEA, FEAL czy szyfr Rijndael należą do grupy szyfrów symetrycznych, czyli takich, w których do szyfrowania i odszyfrowywania wykorzystywany jest ten sam tajny klucz. Problemem wymagającym rozwiązania jest tu dystrybucja klucza, czyli dostarczenie obu komunikującym się stronom tajnego klucza w sposób bezpieczny. W 1976 roku Diffie i Hellman opublikowali przełomowy artykuł [8] o funkcjach jednokierunkowych z zapadką. Autorzy zaproponowali protokół zdalnego uzgadniania klucza. Tym samym udowodnili, że osoby chcące wymieniać szyfrowaną korespondencję wcale nie muszą się spotykać (lub korzystać z tak zwanego bezpiecznego kanału komunikacyjnego) w celu bezpiecznego uzgodnienia tajnego klucza. W 1978 roku opublikowano pracę, w której przedstawiono kryptosystem z kluczem publicznym, od pierwszych liter nazwisk autorów nazwany RSA [9]. W algorytmie takim nie ma potrzeby

przesyłania tajnego klucza, aby było możliwe zaszyfrowanie, przesłanie i bezpieczne odszyfrowanie wiadomości. Bezpieczeństwo tego algorytmu opiera się na trudności rozkładu dużych liczb na czynniki pierwsze. Kryptografia asymetryczne nie spowodowała wyparcia metod symetrycznych, obecnie doskonale się one uzupełniają w wielu protokołach kryptograficznych.

2. Metody szyfrowania – bardzo krótki przegląd

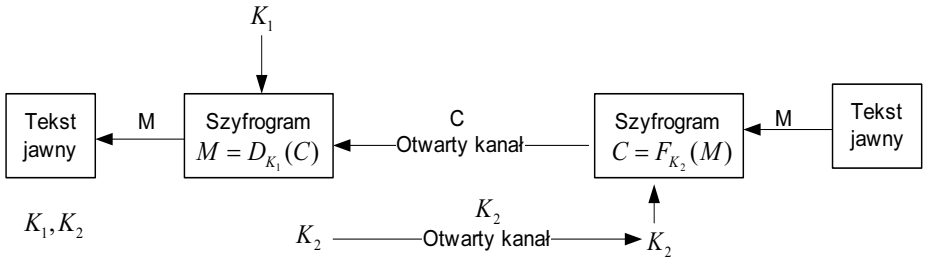
Szyfrowanie symetryczne



Rys. 1. Kryptosystem symetryczny.

Szyfr symetryczny pozwala na przesyłanie informacji pomiędzy nadawcą i odbiorcą za pośrednictwem otwartego (niebezpiecznego) kanału wymiany informacji (Rys. 1). Przed rozpoczęciem transmisji nadawca i odbiorca muszą ustalić wspólny klucz $\boldsymbol{K_1} \in \boldsymbol{K}$, który będzie służył do szyfrowania i odszyfrowywania wiadomości, jak pokazano to na powyższym rysunku. Jeśli jedna ze stron wygeneruje klucz $\boldsymbol{K_1}$ z przestrzeni kluczy $\boldsymbol{K}$, to powinna przekazać go drugiej stronie. Ponieważ transmisja ma się odbywać nie chronionym kanałem wymiany informacji, przekazanie klucza (lub jego uzgodnienie) musi zostać dokonane kanałem bezpiecznym, np. za pośrednictwem zaufanego kuriera. Po zakończeniu procesu dystrybucji klucza nadawca może wiadomość $\boldsymbol{M}$ przekształcić w szyfrogram $\boldsymbol{C}$ posługując się funkcją szyfrującą $C = F_{K_1}(M)$. Tak zaszyfrowaną wiadomość można przesłać do odbiorcy, który posiadając ten sam klucz przekształci szyfrogram do postaci wiadomości $\boldsymbol{M}$ korzystając z funkcji deszyfrującej $M = D_{K_1}(M)$. Warunkiem poprawnego działania całego kryptosystemu jest spełnienie warunku: $M = D_{K_n}(E_{K_n}(M))$.

Szyfrowanie asymetryczne



Rys. 2. Kryptosystem asymetryczny

Jeśli nadawca zamierza przesłać do odbiorcy zaszyfrowaną wiadomość, to powinien posiadać klucz publiczny $\boldsymbol{K_2}$ odbiorcy, który może mu zostać dostarczony otwartym kanałem transmisyjnym, tym samym, którym ma się odbyć przesłanie szyfrogramu (Rys. 2). Klucz publiczny

nie stanowi tajemnicy, ponieważ przy dobrze zaprojektowanym kryptosystemie asymetrycznym jego znajomość nie pozwoli na uzyskanie klucza prywatnego ani na odszyfrowanie wiadomości w „rozsądnie" długim czasie. Nadawca po otrzymaniu klucza publicznego K_2 szyfruje wiadomość M korzystając z funkcji szyfrującej $C = F_{K_2}(M)$. Tak otrzymany szyfrogram przesyłany jest do odbiorcy, który jest w posiadaniu klucza prywatnego K_1 stanowiącego parę z kluczem K_2. Pozwala mu to na przekształcenie otrzymanego szyfrogramu C na tekst jawny M. Kryptosystem działa poprawne, jeżeli spełniony jest warunek $M = D_{K_1}(E_{K_2}(M))$.

3. Szyfrujący układ adaptacyjny

W praktyce spotykane są różne realizacje (implementacje) algorytmów szyfrujących. Ogólnie można je podzielić na realizacje programistyczne oraz sprzętowe. Praktycznie stosowana koncepcja współczesnego podejścia do realizacji algorytmów szyfrujących polega na „wszyciu" wybranego algorytmu, na przykład podpisu elektronicznego czy szyfru, do protokołu w sposób sztywny. Rozumieć przez to należy sytuację taką, że sama implementacja szyfru traktowana jest jak blok zawierający wejście i wyjście. Blok taki może mieć dodatkowe wejście, które pozwala wpływać na stosowany klucz. Z pozoru jest to sytuacja nie budząca, żadnych wątpliwości, o ile został wybrany algorytm bezpieczny, został on właściwie zaimplementowany i spełnione zostało jeszcze kilka warunków, które można na razie pominąć w naszych rozważaniach. Problem pojawia się w momencie, gdy nasze rozwiązanie zostanie rozpowszechnione, może nawet stanie się standardem, po czym bezpieczeństwo wybranego przez nas algorytmu szyfrującego zostanie podważone. Tradycją stało się już w dziedzinie kryptografii, że każdy nowo opublikowany szyfr staje się obiektem licznych ataków. Skompromitowanie algorytmu szyfrującego może wówczas spowodować podważanie bezpieczeństwa protokołu, w którym ów algorytm został wykorzystany. Pociąga to oczywiście za sobą konieczność poprawienia naszego protokołu. Stoimy przed problemem, do którego rozwiązania można podejść dwojako: przez wybór innego algorytmu szyfrującego, lub przez dokonanie zmian w algorytmie szyfrującym zalecanych jako rozwiązanie problemu przez jego autorów. Zarówno jedna jak i druga sytuacji pociąga za sobą konieczność dokonania programistycznych zmian w naszym protokole. Najczęściej konieczne jest stworzenie jego nowej wersji, co oczywiście jest sprawą kosztowną. Jeśli mamy do czynienia z implementacją sprzętową, to koszty będą znacznie większe. Mówimy tu o kosztach samego przeprogramowania, dystrybucji nowej wersji oraz o czasie, jaki jest do tego potrzebny.

Można by się zastanowić, jaka jest alternatywa w istniejącej sytuacji. Rozwiązaniem obecnie spotykanym jest zaimplementowanie kilku algorytmów szyfrujących podobnej klasy i danie użytkownikowi możliwości wyboru, który ma być stosowany. Wtedy dezaktualizacja jednego z nich nie powoduje aż tak dużego problemu. Rozwiązanie takie jednak podnosi koszty naszego protokołu. Innym rozwiązaniem, nad którym obecnie prowadzone są badania w różnych ośrodkach, jest zastosowanie układów reprogramowalnych dla rozwiązań sprzętowych.

Szyfrująca sieć neuronowa

Zastosowanie do realizacji szyfrów układu adaptacyjnego, podatnego na proces uczenia wydaje się umożliwić rozwiązanie problemu, o którym mowa w rozdziale poprzednim. Sieci neuronowe posiadają zdolność „uczenia się", jest to fakt ogólnie znany a metody uczenia bardzo się rozwinęły w ostatnich latach. Pytanie tylko czy sieć neuronową można nauczyć szyfrowania? Problemem przy wykorzystaniu sieci neuronowych jest to, że wyniki ich pracy zwykle są wartościami nieprecyzyjnymi. Znane są zastosowania sieci w procesie rozpoznawani czy klasyfikacji, gdzie akceptowalny jest pewien margines niedokładności. W kryptografii natomiast zależy nam na wysokiej precyzji i całkowitej jednoznaczności.

Istnieje określenie boolowskich sieci neuronowych, które potrafią realizować funkcje boolowskie. Szereg takich sieci prezentowanych jest w [10]. Traktując algorytm szyfrujący jako funkcję przekształcającą ciąg bitów wejściowych, czyli tekst jawny (argumenty funkcji) na ciąg bitów wyjściowych, czyli szyfrogram (wartości funkcji). Można pokusić się o próbę realizacji tej funkcji poprzez sieć neuronową. Inaczej rzecz ujmując zmierzamy do tego, aby skonstruować taką sieć, która da się nauczyć szyfrowania według określonego algorytmu. Tą ideę przedstawia poniższy rysunek (Rys. 3).

Rys. 3. Adaptacyjny układ szyfrujący

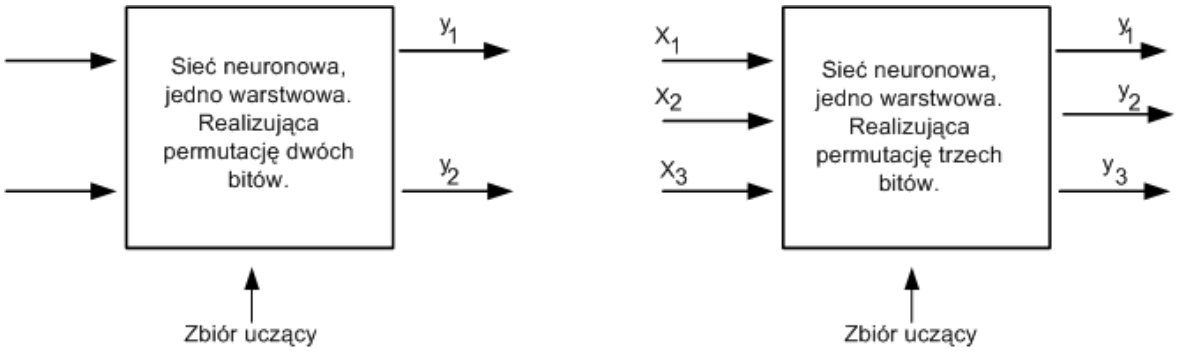
Posługując się odpowiednio skonstruowanym zbiorem uczącym mamy możliwość zmiany („przeprogramowania") układu szyfrującego. Zbiór uczący bierze oczywiście udział tylko w procesie uczenia. Po nauczeniu sieci jej pamięć („wagi" i struktura) są „zamrażane" i taki układ traktowany jest jako zaprogramowany układ szyfrujący.

Nasz propozycja polega na realizacji za pomocą sieci neuronowych poszczególnych przekształceń elementarnych składających się na kompletny algorytm szyfrujący, a następnie na złożeniu kompletnego algorytmu szyfrującego z zaprogramowanych elementów (czyli z nauczonych sieci neuronowych). Najczęściej wykorzystywaną operacją elementarną w różnego rodzaju algorytmach szyfrujących jest permutacja (p-blok). Była o tym mowa w rozdziałach wcześniejszych. W naszej pracy [11] zaprezentowana została koncepcja struktury sieci oraz organizacja procesu uczenia dla potrzeb realizacji permutacji dwóch i trzech bitów. Ze względy na ograniczenia długości artykułu nie zamieszczamy tutaj wyżej wspomnianych informacji, zainteresowanych odsyłamy do prac [11] i [12].

W dalszej części artykułu omówione zostaną wyniki i rezultaty, jakie uzyskane zostały w wyniku implementacji sieci neuronowej jako układu realizującego permutacje.

Implementacja

Cel, jaki został postawiony to zaprojektowanie i zaimplementowanie sieci neuronowej, która będzie realizowała permutacje dla bloku 32 bitów. Wykorzystana została tu koncepcje sieci neuronowych realizujących permutację dwóch i trzech bitów, które zaprezentowane zostały w pracy [11]. W dalszej części, dla zwiększenia przejrzystości, nazywać je będziemy odpowiednio „klockami 2-bitowymi" i „klockami 3-bitowymi". Na Rys. 4 przedstawione zostały schematyczne reprezentacje wspomnianych permutacji, czyli „klocków".

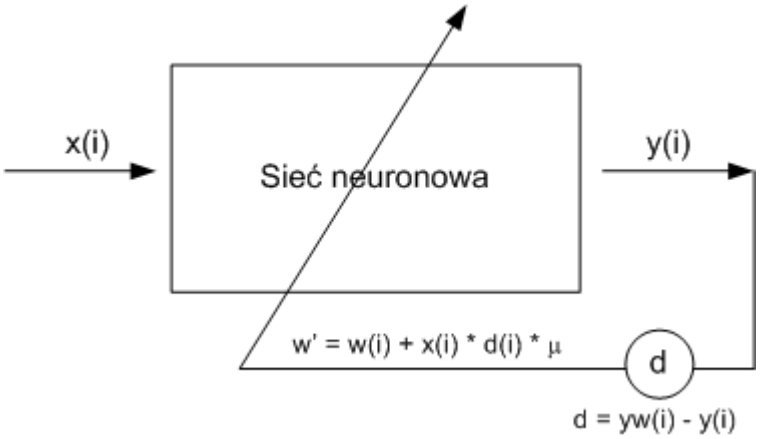


Rys. 4 Adaptacyjne układu szyfrujące, dwu i trzybitowy

Tak jak to przyjęliśmy, do dyspozycji mamy dwa rodzaje „klocków". Aby uzyskać efekt permutacji bloku 32 bitowego można skorzystać z różnych układów (sieci) dwu i trzy-bitowych klocków. Taka możliwość realizacji permutacji została przez nas pokazana w pracy [11]. Celem niniejszej pracy jest pokazanie, na przykładzie implementacji testowej, jak wydajny jest odpowiednio skonfigurowany układ składający się z kilku lub kilkunastu sieci neuronowych (klocków 2 i 3-bitowych). Największym problemem w praktycznej realizacji szyfrów za pomocą sieci neuronowych może się okazać czas potrzebny na zmianę sposobu szyfrowania, czyli w rzeczywistości czas uczenia sieci.

Proces uczenia sieci neuronowej

Na temat uczenia sieci neuronowych można znaleźć wiele opracowań. Dlatego też w tym miejscu przytoczymy jedynie ogólny zarys tego procesu. W dalszej części pracy omówimy wyniki ilościowe uzyskane na podstawie uczącej się implementacji algorytmy realizującego permutację.

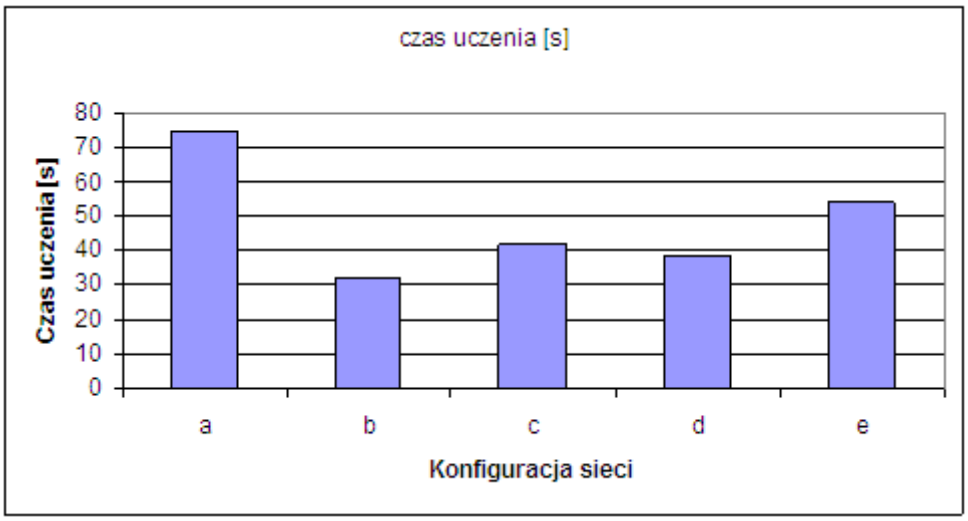


Rys. 5. proces uczenia sieci neuronowej

Rys. 5 przestawia proces uczenia sieci. Wartości x(i) i yw(i) to pary ze zbioru uczącego, wartość d to różnica pomiędzy aktualną odpowiedzią sieci, a wzorcową odpowiedzią pochodzącą ze zbioru uczącego. Uczenie sieci z praktycznego punktu widzenia polega na korygowaniu wartości wag. Wartość nowej wagi, czyli w', obliczana jest według zależności pokazanej na Rys. 5. Istotne znaczenie ma parametr μ. Jest to tak zwany współczynnik uczenia, od doboru jego wartości zależy wielkość korekty wag, a tym samym przebieg procesu uczenia. Dobór właściwej wartości tego współczynnika jest dość kłopotliwy. Gdy współczynnik jest zbyt duży, to sieć może nigdy się nie nauczyć („ominie" właściwą wartość wag); przy zbyt małym współczynniku proces uczenia może trwać nieakceptowanie długo. Oczywiście istotna jest odpowiednia konstrukcja zbioru uczącego i początkowe wartości wag, jednak nie będziemy pisali o tym w tej publikacji.

Wyniki pochodzące z implementacji

Podczas analizy praktycznej realizacji szyfrującej sieci neuronowej skupiliśmy się głównie na procesie uczenia, a dokładniej na czasie potrzebnym na realizację „przeprogramowania" układu tak, aby realizował inną permutację. Wyniki uzyskane dla wybranych eksperymentów zaprezentowano na Rys. 6.



Rys. 6 wyniki z implementacji

Dla potrzeb przeprowadzenia wyżej wspomnianego testu zastosowano różne konfiguracje sieci realizujących permutację 32 bitowego bloku tekstu. Przeanalizowane konfiguracje sieci przedstawiono w tabeli.

Symbol konfiguracji	Składowe sieci
A	16 „klocków 2 bitowych”
B	8 „klocków 3 bitowych” i 4 „klocki 2 bitowe”
C	4 „klocki 3 bitowych” i 10 „klocków 2 bitowych”
D	6 „klocki 3 bitowych” i 7 „klocków 2 bitowych”
E	2 „klocki 3 bitowe” i 13 „klocków 2 bitowych”

Z powyższego wynika, że proces uczenia przebiega szybciej dla konfiguracji sieci zawierających więcej „klocków 3 bitowych”. Są to jednak wyniki dla dobranych wartości współczynnika uczenia. Zaobserwowano, że o ile trzy bitowe klocki wpływają na przyspieszenie procesu uczenia, to dobór dla nich właściwego współczynnika uczenia był nieco trudniejszy. Podobnie rzecz ma się z konstrukcją zbioru uczącego.

Można oczywiście przeprowadzić eksperymenty z blokami większej długości. Jednak na potrzeby tego artykułu został wybrany blok 32 bitowy, ponieważ dla tego przypadku dobrze rysują się różnice pomiędzy sieciami budowanymi z różnych elementów. Widać, że w zależności od sposobu wykorzystania projektowanego systemu kryptograficznego można stosować różne konfiguracje klocków. Dla pojedynczych implementacji lepiej użyć „klocków 2-bitowych”, dla których mamy dłuższy proces uczenia, ale łatwiejszą konstrukcję zbiorów uczących. Dla implementacji powtarzalnych lepsze są „klocki 3-bitowe” z dłuższym czasem przygotowania systemu uczącego, ale szybszym uczeniem bloków.

Inne zastosowania sieci neuronowych w kryptografii

Stosowanym obecnie w praktyce rozwiązaniem jest wykorzystanie sieci neuronowych we wspomaganiu systemów wykrywania intruzów. Systemy IDS (Intrusion Detection System) w ogólnym zarysie działają na zasadzie gromadzenia tak zwanych profili użytkowników w sieci.

Charakteryzują one typowe zachowania dla danego użytkownika. Jeśli nagle charakter ruchu generowane przez danego użytkownika będzie odbiegał od stworzonego profilu, to jest to sygnał alarmowy dla systemu IDS.

W takim przypadku wykorzystanie sieci neuronowych polega na nauczeniu sieci sekwencji ruchu sieciowego generowanego przez określonego użytkownika. W dalszej kolejności taka sieć wykorzystywana jest do monitorowania działalności danego użytkownika. Rolą nauczonej sieci jest przewidzenie najbardziej prawdopodobnej sekwencji zdarzenia, jakie danych użytkownik powinien wykonać. Potem porównywane jest to z faktycznie wygenerowanym ruchem. Nadmienić należy tutaj, że każdy użytkownik posiada własny profil a więc użytkownicy „obsługiwani są" przez osobne sieci.

Podsumowanie

W tak krótkim opracowaniu przedstawiony został oczywiście tylko zarys problemu, oraz ogólna koncepcja naszym rozważań. We wcześniejszej pracy [12] przedstawione została koncepcja realizacji operacji podstawienia za pomocą sieci neuronowej, a konkretnie realizacji s-bloku. Obecnie trwają prace na sprawdzeniem tej koncepcji praktyce.

Bibliografia

[1] D. Kahn. The Codebreakers. Macmillan, New York, 1967

[2] C. E. Shannon, „A mathematical theory of communication" Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[3] C. E. Shannon „ Communication Theory of Secrecy Systems", Bell System Technical Journal, vol.28-4, page 656--715, 1949.

[4] http://nvl.nist.gov/

[5] NBS FIPS PUB 46, "Data Encryption Standard", 1977

[6] A.Shimizu and S.Miyaguchi, Fast Data Encirpherment FEAL, Transactions of IEICE of Japan 1987.

[7] X.Lai and J.Massey, A proposal for a New Block Encryption Standard, EUROCRYPT '90 1991

[8] W. Diffie and M.E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, 1976.

[9] R. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM, 1978.

[10] Piotr S. Szczeaniak „Obliczenia inteligentne, szybkie przekształcenia i klasyfikatory", ISBN 83-87674-77-X

[11] P.Kotlarz, Z.Kotulski, Application of neural networks for implementation of cryptographic functions, in: Leszek Kiełtyka [ed.], *Multimedia in Business and Education*, Vol. 1, pp. 213-218, Fundacja Współczesne Zarzšdzanie, Białystok 2005. ISBN 83-9182218-7-0.

[12] P.Kotlarz, Z.Kotulski, On application of neural networks for S-boxes design, in: P. S. Szczepaniak, J.Kacprzyk, A. Niewiadomski, Advances in Web Intelligence: Third International Atlantic Web Intelligence Conference, AWIC 2005, Lodz, Poland, June 6-9, 2005. *Lecture Notes in Artificial Intelligence*, LNCS 3528, pp. 243-248, Springer, Berlin 2005. ISBN: 43-540-26219-9