УДК 003.26.09, 004.032.26
DOI: https://doi.org/10.26642/tn-2018-2(82)-195-203

С.І. Бондарчук, магістрант
В.Н. Ковальчук, к.пед.н.
А.М. Ковальчук, к.т.н., доц.
А.А. Єфіменко, к.т.н.
Житомирський державний технологічний університет

Реалізація та дослідження алгоритму криптографічного захисту даних з відкритим ключем на основі нейронної мережі

Реалізовано алгоритм криптографічного захисту даних з відкритим ключем з використанням багатошарового перцептону. Здійснено об'єктно-орієнтоване проектування ієрархії класів програмного рішення. Виконана декомпозиція дозволяє легку модифікацію системи, зокрема можлива заміна нейронної мережі або порогової функції на інші подібні компоненти. Проведено експериментальне дослідження програмної реалізації системи для порівняння ефективності різних порогових функцій, а також дослідження швидкодії розпаралелюного алгоритму навчання.

***Ключові слова**: криптографічний захист; шифрування даних; перцептрон; нейронні мережі; розпаралелювання алгоритму навчання.*

Постановка проблеми. Реалізація і дослідження алгоритмів асинхронного шифрування з відкритим ключем з використанням нейронних мереж є актуальною задачею нейрокриптографії.

Аналіз останніх досліджень і публікацій. В наш час захист інформації в телекомунікаційних мережах поєднує різні напрямки досліджень, як то крипто аналіз, дослідження і створення алгоритмів шифрування та дешифрування, протоколів обміну інформацією, аналіз і генерація штучних атак [3, 5]. Нейрокриптографія – перспективний напрямок криптографії, що вивчає застосування стохастичних алгоритмів, зокрема нейронних мереж в різних аспектах захисту інформації.

Штучна нейронна мережі (ШНМ) як спрощеною моделлю біологічного мозку була вперше запропонована Френком Розенбладом в 1957 році. Подальший розвиток концепції призвів до створення різних типів нейромереж. Зокрема, за способом поширення сигналу розрізняють нейромережі прямого поширення (Feed Forward) та рекурентні мережі. Такі властивості нейронних мереж як взаємне навчання, самонавчання і стохастична поведінка дозволяє застосовувати ШНМ в різних напрямках криптографії і захисту інформації [9, 6, 4]. Є теоретичні дослідження для використання штучної нейронної мережі для створення симетричного шифра на основі AES [1]. В ряді публікацій [8, 1] розглядаються аспекти застосування ШНМ до систем шифрування з відкритим ключем.

Мета дослідження. Метою дослідження є реалізація ШНМ прямого поширення та експериментальна перевірка впливу різних алгоритмів роботи системи на ефективність і швидкість процесу генерації закритих та відкритих ключів та шифрування і дешифрування повідомлень в цілому.

Викладення основного матеріалу. Аналіз існуючих класичних алгоритмів шифрування виявив такі недоліки, як ймовірність появи в зашифрованому і розшифрованому тексті спотворень, до недоліків нейромережних алгоритмів можна віднести ускладнення алгоритму шифрування порівняно з алгоритмом AES і необхідність в створенні і підтримки більш складної системи на етапі впровадження [9].

Для реалізації методу шифрування в роботі використано багатошаровий перцептрон, який дозволяє двом користувачам обмінюватись інформацією і отримувати відкритий ключ. Два користувача використовують цей відкритий ключ, для того щоб зашифрувати повідомлення. Таке рішення є подібним до системи Діффі-Хелмана, яка надає користувачам надійний доступ для отримання ключа для шифрування даних шляхом обміну інформацією.

Відомий алгоритм RSA базується на задачі розкладення на множники великих чисел, в той час як безпека методу Діффі-Хелмана базується на складній задачі обрахування дискретних логарифмів [7]. Проте, для забезпечення безпеки обох систем необхідними є великі прості числа, довжиною від 300 десяткових цифр. Задача ідентифікації простих чисел є обчислювально складною, як використовувана разом з нею задачею розкладення на множники.

Як відомо, в моделі багатошарового перцептрона один шар містить 256 сигмоїдальних нейронів. Отже, число входів в структуру дорівнює 1, а число синаптичних вагів, до яких застосовується навчання, дорівнює 256.

В такому випадку користувацький ключ складається з 257 дійсних чисел. Одне дійсне число використовується як вхід до багатошарового перцептрона, а інші 256 дійсних чисел використовуються як вихід мережі. Саме на множині цих даних здійснюється навчання мережі.

Навчання мережі здійснюється на основі регулювання кожного синаптичного вагового коефіцієнту за наступною формулою (1):

$$w_z' = w_z + 0.001\,(y_z - \varphi_z)(1 - \varphi_z)(1 + \varphi_z)x, \quad (1)$$

де w_z – значення, пов'язане з нейроном z, тобто, його вага; φ_z – вихід нейрона z; y_z – бажаний вихід нейрона z і φ_z представлено у вигляді (2):

$$\varphi_z = \tanh(x\,w_z). \quad (2)$$

Кількість проходів навчання визначається оновлюваністю розрахованих за (1) значень ваг.

Лью і Де Сілва [10] – науковці, що детально вивчали подібну схему, емпіричним шляхом довели, що обмеження користувацького ключа в межах від -1 до 1, ініціалізація початкових вагових коефіцієнтів з середнім значенням 0 і дисперсією 1, і обмеження кількості епох навчання до 5, дає можливість отримати індивідуально навчені мережі з відповідними значеннями, які дають середньоквадратичну похибку, яка дозволяє достатньо точно узгодити всі вагові коефіцієнти перцептрона зі значеннями бажаного виходу [9, с. 111]. Також вони встановили, що оптимальне значення середньоквадратичної помилки між вагами двох користувацьких ключів становить близько 3×10^{-4} [9, с. 117].

В рамках дослідження було проаналізовано комбінацію мережі з наступними характеристиками: $|x| \leq 0.75$, $|y| \leq 0.1$. Для перевірки середньоквадратичної помилки спочатку треба обчислити значення φ.

Оскільки процес навчання базується на основі функції тангенса, властивості цієї функції звужують коло оптимальних числових даних в якості входу і виходів в діапазоні від -1 до 1. До речі, більшість поширених функцій, що використовуються як порогові в нейронних мережах мають схожі рамки, що дозволяє легко протестувати й порівняти альтернативні функції в якості порогових в штучній нейронній мережі, що досліджується.

Проаналізуємо наступне рівняння:

$$\Delta w = 0.001 \times (y - \varphi) \times (1 - \varphi^2) \times x. \quad (3)$$

Для визначення максимального значення φ, при якому рівняння досягає максимального значення, потрібно знайти першу похідну по φ:

$$3\varphi^2 - 2\varphi - 1. \quad (4)$$

Прирівнюємо рівняння до 0 і знаходимо корені рівняння - $\varphi_1 = -1/3$ і $\varphi_2 = 1$. Останнє значення відкидається як значення у точці повороту.

Підставивши всі значення отримаємо:

$$\Delta w = 0.001 \times (0.1 - (-1/3)) \times (1 - (-1/3)^2) \times 0.75 \approx 1.5 \times 10^{-4}; \quad (5)$$

Отриманий в (5) результат доводить, що середньоквадратичне відхилення знаходиться в межах оптимального.

В запропонованій моделі багатошарового перцептрона міститься 256 ваг, які навчаються. Після навчання ваг, відповідні проміжні вагові коефіцієнти або лежать на різних сторонах від початкових вагових коефіцієнтів, або в одній і тій же стороні від початкових значень вагових коефіцієнтів. Коли обидва проміжні вагові коефіцієнти лежать в одній і тій же стороні, два користувача навчають їх однаково, або збільшуючи, або зменшуючи значення ваг.

Припустимо, що два проміжних значення лежать в протилежній стороні від початкового вагового коефіцієнта. Отже, кінцеве значення вагового коефіцієнту повинно знаходитись десь між проміжними ваговими коефіцієнтами. Ймовірність отримання кінцевих вагових коефіцієнтів із проміжних буде дорівнювати 2^{-256}, оскільки буде існувати 256 комбінацій перевірки. Ймовірність всіх 256 вагових коефіцієнтів не становить 100 %, оскільки існує ймовірність того, що проміжні значення ваг лежать по одну сторону від початкового.

Доведено [9, с. 120], що ймовірність отримання вагових коефіцієнтів по одну сторону дорівнює ймовірності отримання ваг по різні сторони і становить 0.5. Тоді очікуване число вагових коефіцієнтів, що знаходяться в протилежних сторонах від початкових значень складає $0.5 \times 256 = 128$.

Складність отримання значення кінцевих вагових коефіцієнтів у випадку перебору буде потребувати 2^{128} комбінацій.

Для того, щоб ключі абонентів були безпечними, потрібно виконати умову складності отримання ключів із доступної інформації. Криптосистема багатошарового перцептрона потребує наявності наступних значень:

– початкові вагові коефіцієнти;
– проміжні вагові коефіцієнти моделей багатошарового перцептрона, які були навчені двома різними користувачами;
– алгоритм навчання вагових коефіцієнтів.

З відомими початковими і проміжними ваговими коефіцієнтами може бути отримано 256 систем нелінійних рівнянь виду:

$$w_{\text{пром}} - w_{\text{поч}} = 0.001 \times (y - \varphi) \times (1 - \varphi^2) \times x.$$

Рішення системи нелінійних рівнянь потребує використання трудомістких числових методів, а це забезпечує високий рівень безпеки криптосистеми, побудованої на основі багатошарового перцептрона [9, с. 121].

Користувач U обирає 257 дійсних вхідних чисел x, на основі яких мають бути отримані бажані виходи від y_1до y_{256}. На ці числа накладаються наступні обмеження: $-0.75 < x < 0.75$ та $-0.1 < y_i < 0.1$, де і – номер бажаного виходу, загальна кількість яких складає 256. Користувач U генерує початкові ваги $w_{\text{поч}}$ та використовує x і y_1до y_{256} для навчання $w_{\text{поч}}$ і отримує w_{xy}.

Відкритий ключ становить вектор значень $w_{\text{поч}}$ і w_{xy}.

Секретний ключ містить масив вхідних значень x і відповідний масив значень від y_1до y_{256}.

Алгоритм шифрування поляає в наступноме: нехай користувач A хоче відправити 64 біта даних. Тоді користувач A обирає u і від t_1до t_{256} такі, що $-0.75 < u < 0.75$ та $-0.1 < t_i < 0.1$, де і – номер бажаного виходу, загальна кількість яких складає 256. Далі користувач навчає шар початкових значень $w_{\text{поч}}$ для отримання w_{ut} і навчає шар w_{xy} для отримання масиву w_{xyut}. Отримані ваги w_{xyut} округлюються до третьої цифри після коми по принципу, якщо 4 цифра після коми дорівнює 8 чи 9, то заокруглення відбувається в більшу сторону, якщо четверта цифра після коми 1 чи 2 то в найменшу. Потім користувач обирає перші 64 елемента w_{xyut}, що відовідають позиціям, в яких w_{ut} не дорівнює w_{xy}. Порівняння ведеться тільки до третьої цифри після коми по принципу, описанному вище. При цьому користувач A модифікує w_{xyut} так, що $w_{xyut} = w_z$, якщо w_{xyut} не дорівнює w_z, де z – це ut або xy. Модифікація проводиться у випадку, якщо відповідний біт має бути 1. Тоді w_{xyut} і w_{ut} відсилаються як шифрограма.

Відповідний алгоритм дешифрування полягає в наступному: після отримання шифрограми ваг w_{xyut} і w_{ut} користувач U навчає вихідні значення w_{ut} вираховувати w_{utxy}, округлюючи елементи w_{utxy} до третьої цифри після коми за вже визначеним принципом. Визначаючи положення, де w_{ut} не дорівнює w_{xy}, користувач U відновлює біти, порівнюючи w_{xyut} і w_{utxy}. Де величини w_{xyut} і w_{utxy} не дорівнюють одна одній, зашифрована 1, а де дорівнюють – 0.

Схема цифрового підпису на базі багатошарового перцептрона має багато схожого з алгоритмом шифрування.

Користувач U вибирає два набори дійсних чисел, перший набір складається з x і від y_1до y_{256}, а другий ряд – з uі відt_1доt_{256}, таких, що $-0.75 < x, u < 0.75$ і $-0.1 < y_i, t_i < 0.1$, де і – номер бажаноговиходу, загальнакількістьякихскладає 256. КористувачU генерує початкові ваги $w_{\text{поч}}$, на основі x і від y_1до y_{256} отримує w_{xy} і на основі u і від t_1до t_{256} отримує w_{ut}. Користувач U також навчає w_{ut} з допомогою x і від y_1до y_{256} і отримує w_{utxy}, округлюючи елементи w_{utxy} до третьої цифри після коми за принципом викладеним вище.

Відкритий ключ містить шари $w_{\text{поч}}$,w_{xy}, w_{ut}, w_{utxy}, u і масиви значень від t_1доt_{256}.

Секретний ключ містить масив вхідних значень x і відповіних масив значень y_1до y_{256}.

Алгоритм підпису: нехай користувач U хоче підписати 64 біта даних. Користувач U обирає перші 64 елемента w_{utxy}, що відовідають позиціям, в яких w_{ut} не дорівнює w_{xy}. Порівняння ведеться тільки до третьої цифри після коми по принципу, описанному вище. При цьому користувач A модифікує w_{utxy} так, що $w_{utxy} = w_z$, якщо w_{utxy} не дорівнює w_z, де z – це u t або xy. Модифікація проводиться у випадку, якщо відповідний біт має бути 1. Тоді $w_{\text{поч}}$, w_{xy}, w_{ut}, w_{utxy}, u і від t_1 до t_{256} і 64 біта даних відсилаються користувачу A.

Алгоритм перевірки: щоб перевірити підпис, користувач A використовує u і від t_1до t_{256} для навчання w_{xy} і отримання w_{utxy}, округлюючи w_{utxy} до третьої цифри після коми по описанному вище правилу. В позиції, де w_{xy} і w_{ut} не дорівнюють один одному, користувач A порівнює значення w_{xyut} і w_{utxy} і отримує 1, де вони відрізняються, та 0, де значення співпадають. Таким чином закодоване повідомлення можна легко відновити зі значень вагових коефіцієнтів і для перевірки його можна порівняти з оригнальним повідомленням.

Алгоритм роботи розробленої моделі наведено на рис. 1 і він складається з таких пунктів:

1. Генерація випадкових величин x, y та u, t першим та другим користувачем відповідно. Дані величини являються секретними ключами;
2. Перший користувач генерує випадкові значення для початкових вагових коефіцієнтів – величини, що характеризують вагові коефіцієнти нейронів в мережі;
3. Проводиться навчання проміжних вагових коефіцієнтів на основі початкових з використанням власного секретного ключа;
4. Далі початкові ваги і проміжні ваги, обраховані першим користувачем, передаються другому. Дані величини є відкритим ключем;
5. Другий користувач проводить навчання початкових вагових коефіцієнтів з використанням власного секретного ключа і отримує значення проміжних вагових коефіцієнтів;

6. Наступним кроком є навчання проміжних вагових коефіцієнтів, отриманих від першого користувача своїм секретним ключем. Отримані вагові коефіцієнти становлять основу шифротексту;

7. Далі отримані вагові коефіцієнти модифікуються таким чином, щоб вмістити в себе повідомлення, що треба передати іншому користувачеві;

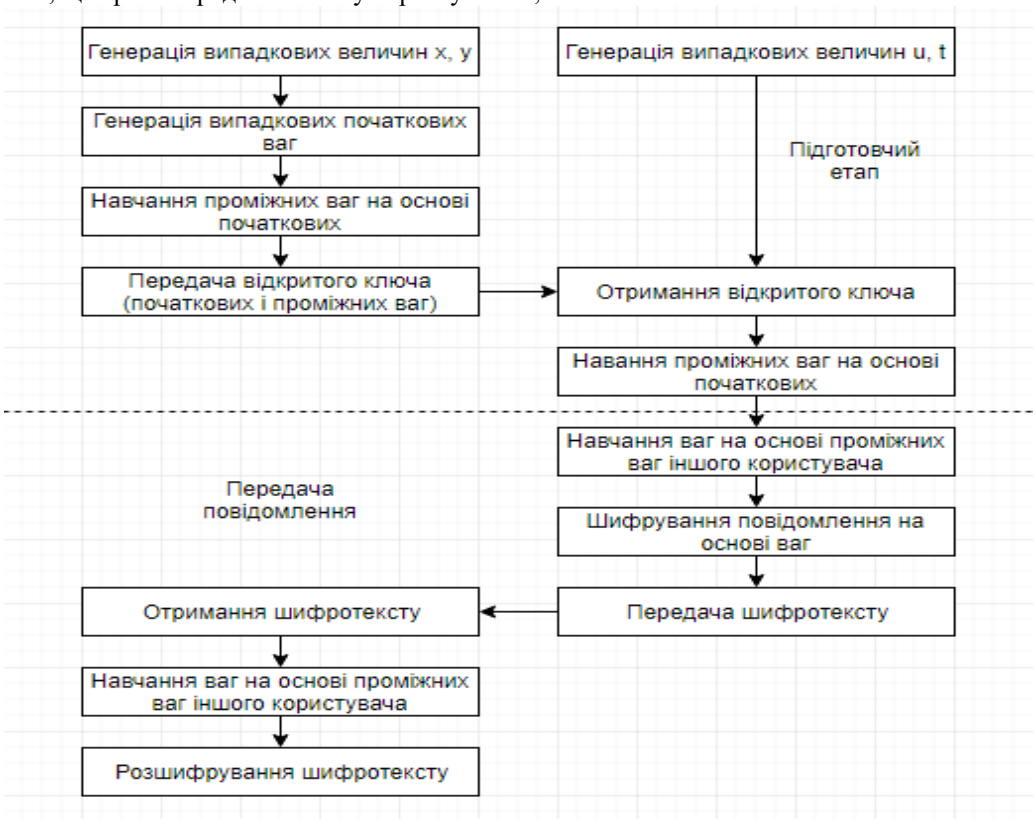

Рис. 1. Алгоритм роботи моделі

8. Потім відбувається передача шифротексту відкритими каналами;

9. Проводиться обрахування вагових коефіцієнтфв на основі проміжних вагових коефіцієнтів іншого користувача з використанням свого секретного ключа;

10. Нарешті, відбувається розшифрування повідомлення на основі порівняння шифротексту і вагових коефіцієнтів, отриманих в попередньому пункті.

Також варто відмітити наступне:

– пункти 1–5 виконуються один раз перед початком активної передачі повідомлень між користувачами;

– пункти 6–10 виконуються кожний раз при передачі одного повідомлення від першого користувача другому.

Створена бібліотека, що містить реалізацію нейроної мережі складається з ряду класів. Ряд ключових класів та зв'язки між ними показано на рисунку 2.

Клас Neuron містить weights – набір вагових коефіцієнтів та size – кількість вагових коефіцієнтів. Також, він містить метод, що нормалізує вектор вагових коефіцієнтів (ділить кожний елемент на довжину), метод, що повертає скалярну суму вектора вагових коефіцієнтів і вхідного вектора, а також метод, що коректує значення вагових коефіцієнтів на основі параметрів.

Інтерфейс Function описує порогову функцію і містить два абстрактних метода для обрахування значення функції і похідної.

Класи TanhFunctioni Sigmoid реалізують інтерфейс Function і описують однойменні математичні функції (рис. 2).

Абстрактний клас NeuronNet описує шар нейронів. Він має ряд властивостей: neurons – масив класів Neuron, neuronCount– кількість нейронів в мережі, function – екземпляр типу Function, який описує порогову функцію, що використовується в даній нейронній мережі. Даний клас містить ряд методів: абстрактний метод Study описує алгоритм навчання мережі, метод Calculate обраховує вихід мережі на основі входу з використанням вагових коефіцієнтів нейронів, а метод CalculateError обраховує помилку на основі очікуваного і реального виходів.

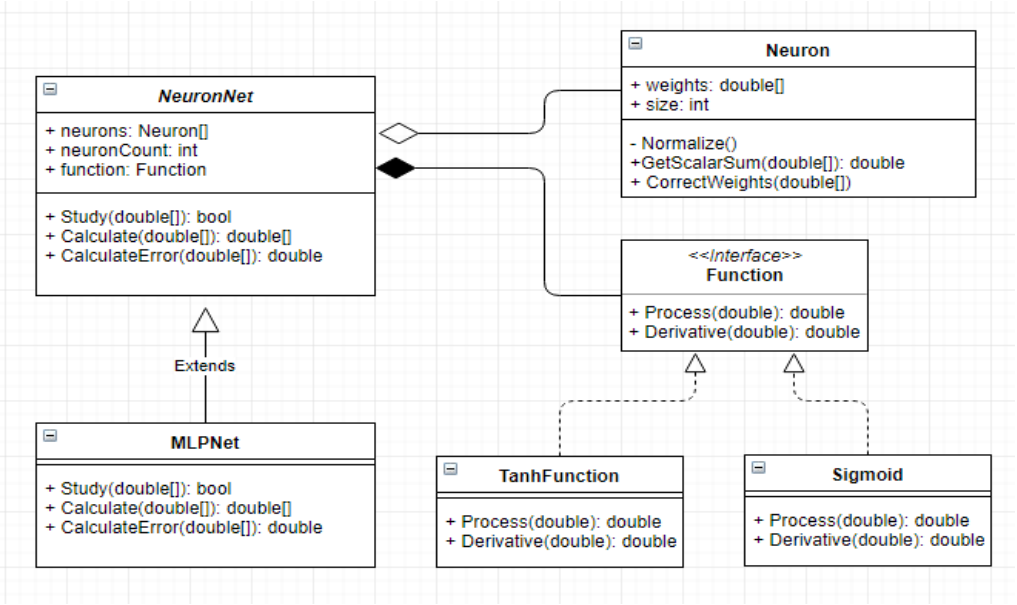

Рис. 2. Структура класів

Клас MLPNet реалізує клас NeuronNet. Цей клас описує нейронну мережу типу багатошаровий перцептрон.

Для тестування характеристик розробленої моделі криптозахисту на основі штучної нейронної мережі була створена спеціальна консольна програма. За допомогою цієї програми було проведено тестування наступних показників:

– тестування швидкості генерації ключів;
– тестування швидкості шифрування повідомлення та розшифрування шифротексту;
– тестування швидкості функціонування штучної нейронної мережі при роботі в один потік та з використанням декількох паралельних потоків;
– тестування конфігурацій з використанням інших порогових функцій.

Перший експеримент полягав в зменшенні кількості епох навчання для лінійного зменшення кількості обчислювальних операцій, що виконується (рис. 3, рис.4).

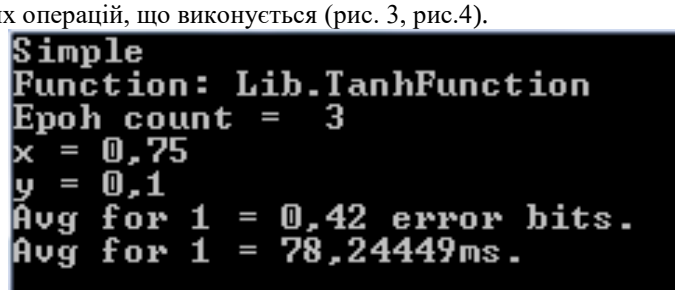

Рис. 3. Експеримент номер 2 з використанням тангенціальної функції

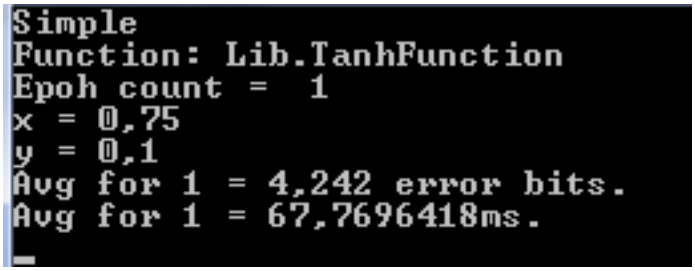

Рис. 4. Експеримент номер 3 з використанням тангенціальної функції

Дослідження показало, що зменшення кількості епох веде до збільшення похибки – кількості неправильно розшифрованих біт. Також виявлено кількість епох, достатню для забезпечення властивості асоціативності – сталості результату не залежно від порядку виконаних операцій.

Також було проведено тестування штучної нейронної мережі з використанням сигмоїдальної порогової функції (рис. 5).

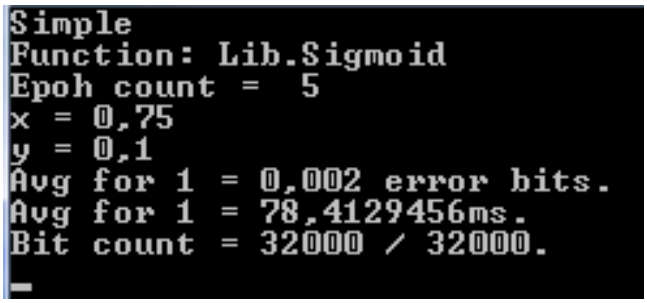

Рис. 5. Експеримент номер 7 з використанням сигмоїдальної функції

Використання сигмоїдальної порогової функції показало чудові результати. Підкоректувавши межі генерації секретного ключа, вдалось добитись ймовірності помилки менше ніж одна помилка на одне шифроповідомлення довжиною в 64 біти (рис. 6).

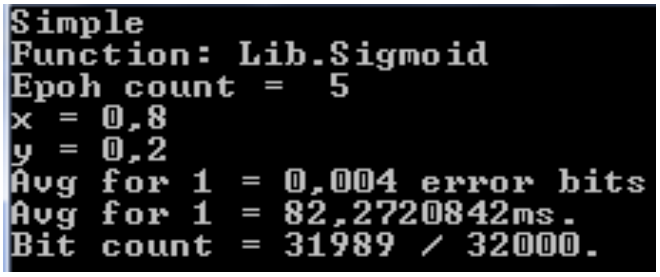

Рис. 6. Експеримент номер 2 з використанням тангенціальної функції

Провівши ряд досліджень, кращою конфігурацією виявилась конфігурація під номером 9. Саме при цій конфігурації було досягнуто цілі отримати якнайменшу кількість неправильно розшифрованих байт і ця кількість прямує до 0. Однак, для такої конфігурації існує ймовірність того, що така досить об'ємна нейронна мережа не зможе зашифрувати блок інформації довжиною 64 біта, оскільки при такій конфігурації не буде досягнуто достатнього розподілення проміжних ваг для шифрування повідомлення (рис. 7).

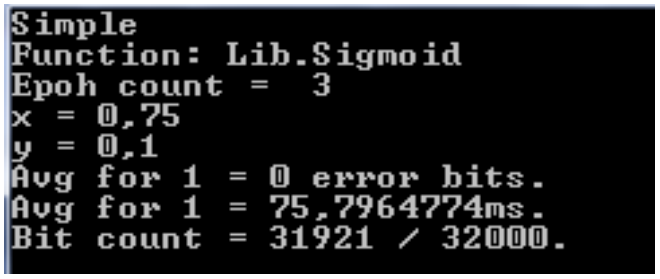

Рис. 7. Експеримент номер 9 з використанням тангенціальної функції

Результати замірів для сигмоїдальної і тангенціальної порогової функцій при різних вхідних параметрах можна представити у вигляді наступної таблиці (табл. 1).

Таблиця 1

Таблиця залежності показників роботи алгоритму від конфігурації

Номер експерименту	**Порогова функція**	**Верхня межа для *x***	**Верхня межа для *y***	**Кількість епох**	**Кількість помилкових біт**	**Втрачені біти**
1	Тангенс	0,75	0,1	5	0,064	257
2	Тангенс	0,75	0,1	3	0,026	466
3	Тангенс	0,75	0,1	1	0,002	3990
4	Тангенс	0,8	0,2	5	0,106	0
5	Тангенс	0,8	0,2	3	0,032	67
6	Тангенс	0,8	0,2	1	0,004	970
7	Сигмоїда	0,75	0,1	5	0,002	0
8	Сигмоїда	0,75	0,1	3	0	79
9	Сигмоїда	0,75	0,1	1	0,002	961
10	Сигмоїда	0,8	0,2	5	0,006	10
11	Сигмоїда	0,8	0,2	3	0,002	20
12	Сигмоїда	0,8	0,2	1	0	219

Також було проведено тестування швидкості генерації відкритого і секретного ключів. Багатопотокове виконання коду в цілому пришвидшує процес генерації.

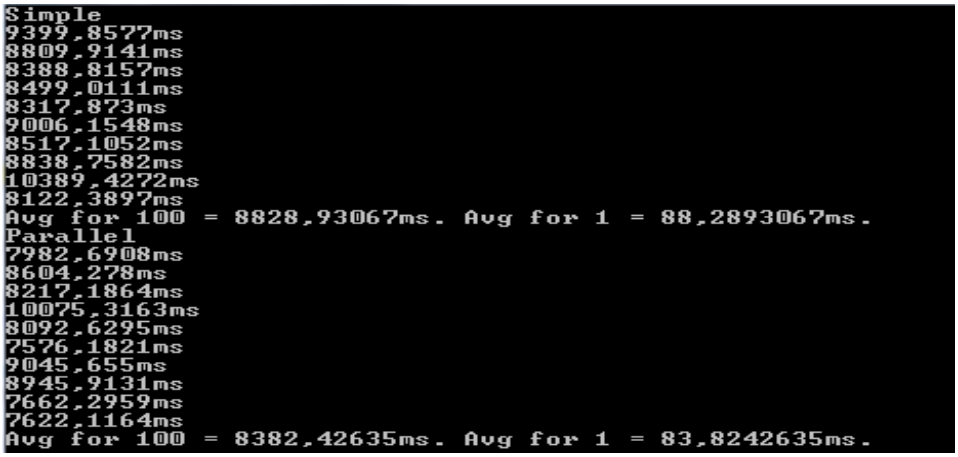

Рис. 8. Порівняння швидкості генерації ключів з використанням одного та декількох потоків

Дані замірів можна подати у вигляді наступної таблиці. Як ми бачимо, середнє значення підвищення швидкодії для ініціалізації секретного і відкритого ключів складає близько 5,5 %.

Таблиця 2

Переваги та недоліки базових рекомендаційних алгоритмів

Дані, що порівнюються	Один потік, мс	Паралельні обрахування, мс	Різниця, %
Експеримент 1	9399,86	7982,69	17,8
Експеримент 2	8809,91	8604,28	2,4
Експеримент 3	8388,82	8217,19	2,1
Експеримент 4	8499,01	10075,32	-15,6
Експеримент 5	8317,87	8092,63	2,8
Експеримент 6	9006,15	7576,12	18,9
Експеримент 7	8517,11	9045,66	-5,8
Експеримент 8	8838,76	8945,91	-1,2
Експеримент 9	10389,43	7662,3	35,6
Експеримент 10	8122,39	7622,12	6,6
Середній показник за 100 замірів	8828,93	8382,43	5,3
Середній показник за 1 замір	88,29	83,82	5,3

Для покращення швидкодії роботи бібліотеки, проведення обчислень виходу нейронів і обрахування дельти на основі порогової функції були розпаралелені. Для того, щоб забезпечити можливість паралельних обчислень вагових коефіцієнтів та дельт був використаний клас Parallel, що входить до списку стандартних бібліотек NET.

Як видно з рисунка 10, використання паралельних обчислень для обчислення значень порогової функції дає приріст швидкодії.

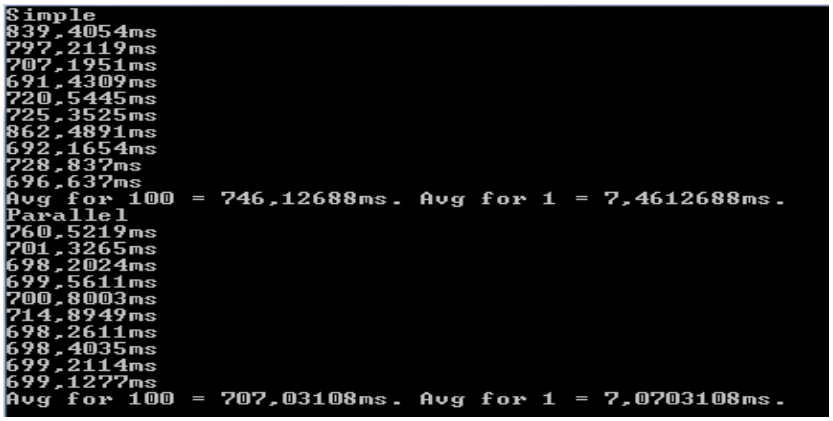

Рис. 9. Порівняння швидкості шифрування/дешифрування повідомлення з використанням одного та декількох потоків

Дані замірів можна подати у вигляді наступної таблиці. Як ми бачимо, середнє значення підвищення швидкодії для пари операцій за шифрування/розшифрування складає близько 5,5 %.

Таблиця 3

Переваги та недоліки базових рекомендаційних алгоритмів

Дані, що порівнюються	**Один потік, мс**	**Паралельні обрахування, мс**	**Різниця, %**
Експеримент 1	839,4	760,52	10,4
Експеримент 2	797,21	701,33	13,7
Експеримент 3	707,2	698,2	1,3
Експеримент 4	691,43	699,56	–1,2
Експеримент 5	720,54	700,8	2,8
Експеримент 6	725,35	714,89	1,5
Експеримент 7	862,49	698,26	23,5
Експеримент 8	692,17	698,4	–0,9
Експеримент 9	728,84	699,21	4,2
Експеримент 10	696,64	699,13	–0,4
Середній показник за 100 замірів	746,13	707,03	5,5
Середній показник за 1 замір	7,46	7,07	5,5

Висновки. Розроблений алгоритм шифрування даних на основі реалізованої моделі нейронної мережі інтегровано в бібліотеку, що надає можливість гнучкої інтеграції з додатками для шифрування користувацьких даних, і також надає можливість легко корегувати модель для отримання кращих результатів базуючись на специфіці конкретної задачі. На основі вбудованого в .NET класу Parallel було виконано розпаралелювання обчислень з метою підвищення швидкодії системи.

Для навчання нейронів була обрана тангенціальна порогова функція, оскільки ряд проведених досліджень показали кращі результати даної функції в порівнянні з популярною сигмоїдальною функцією.

В майбутньому, функціонал даної системи можна суттєво розширити: використати нові порогові функції чи їх комбінації, використати за основу інший тип нейронної мережі. Також можна ускладнити структуру мережі додатковими шарами нейронів, в комбінації з методом зворотного поширення помилки.

Список використаної літератури:

1. *Liu N.* A new public-key encryption scheme based on neural networks and its security analysis / *N.Liu, D.Guo* // International Conference on Computational and Information Science. Computational Intelligence and Security. – 2006. – Pp. 443–450.
2. *Osowski S.* Sieci neuronowe w ujęciu algoritmicznym / *S.Osowski*. – Warszawa : Wydawnictwa Naukowo-Techniczne, 1996. – 349 p.
3. *Rojas R.* Neural networks – a systematic introduction / *R.Rojas*. – Springer-Verlag, Berlin, New-York, 1996. – 502 p.
4. *Анин Б.Ю.* Защита компьютерной информации / *Б.Ю. Анин*. – СПб. : БХВ, 2000. – 384 с.
5. *Брюхомицкий Ю.А.* Нейросетевые модели для систем информационной безопасности / *Ю.А. Брюхомицкий.* – Таганрог : Изд-во ТРТУ, 2005. – 160 с.
6. *Грушо А.А.* Теоретические основы защиты информации / *А.А. Грушо, Е.Е. Тимонина.* – М. : Изд-во агентства «Яхтсмен», 1996. – 130 с.
7. *Евдокимов А.А.* Интеллектуальные информационные системы защиты информаци : монография / *А.А. Евдокимов, Э.Е. Тихонов* / Невинномысский ин-т экономики, управления и права. – Невинномысск : НИЭУП, 2012. – 119 с.
8. *Саломаа А.* Криптография с открытым ключом / *А.Саломаа*. – 1995. – 318 с.
9. *Томашевський О.М.* Криптографічний захист інформаційних систем на базі штучної нейронної мережі / *О.М. Томашевський* // Збірник «Труды Одесского политехнического университета». – Одеса, 2001. – Т. 4 (16).– С. 74–77.
10. Применение искусственных нейронных сетей и системы остаточных классов в криптографии / *Н.Червяков, А.Лавриненко, И.Лавриненко, А.Галушкин, А.Евдокимов.* – Издательство : Физматлит, 2012. – 280 с.

References:

1. Liu, D.Guo (2006), «A new public-key encryption scheme based on neural networks and its security analysis», International Conference on Computational and Information Science. Computational Intelligence and Security, Pp. 443–450.

2. Osowski, S. (1996), *Sieci neuronowe w ujęciu algoritmicznym*, Wydawnictwa Naukowo-Techniczne, Warszawa, 349 p.
3. Rojas, R. (1996), *Neural networks – a systematic introduction*, Berlin, New-York, Springer-Verlag, 502 p.
4. Anin, B.Yu. (2000), *Zashchita komp'yuternoy informatsii*, BKhV, SPb., 384 p.
5. Bryukhomitskiy, Yu.A. (2005), *Neyrosetevye modeli dlya sistem informatsionnoy bezopasnosti*, Izd-vo TRTU, Taganrog, 160 p.
6. Grusho, A.A. and Timonina, E.E. (1996), *Teoreticheskie osnovy zashchity informatsii*, Izd-vo agentstva «Yakhtsmen», M., 130 p.
7. Evdokimov, A.A. and Tikhonov, E.E. (2012), *Intellektual'nye informatsionnye sistemy zashchity informatsii*, monografiya, Nevinnomysskiy in-t ekonomiki, upravleniya i prava, NIEUP, Nevinnomyssk, 119 p.
8. Salomaa, A. (1995), *Kriptografiya s otkrytym klyuchom*, 318 p.
9. Tomashevs'kiy, O.M. (2001), «Kriptografichniy zakhist informatsiynikh sistem na bazi shtuchnoї neyronnoї merezhi», *Zbirnik «Trudy Odesskogo politekhnicheskogo universiteta»*, Odesa, Vol. 4 (16), Pp. 74–77.
10. Chervyakov, N., Lavrinenko, A., Galushkin, A. and Evdokimov, A. (2012), *Primenenie iskusstvennykh neyronnykh setey i sistemy ostatochnykh klassov v kriptografii*, Izdatel'stvo, Fizmatlit, 280 p.

Бондарчук Сергій Іванович – магістрант за освітньою програмною «Інженерія програмного забезпечення».

Наукові інтереси:
- безпека інформації в складних програмних системах;
- конструювання інтерфейсів.

E-mail: pi46_bondarchuksi@ ztu.edu.ua.

Ковальчук Вікторія Наумівна – кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення Житомирського державного технологічного університету.

Наукові інтереси:
- інформаційна безпека особистості;
- методика викладання профільних дисциплін спеціальності ПІ;
- автоматизовані системи перевірки завдань з програмування.

E-mail: bukachuk1@gmail.com.

Ковальчук Андрій Михайлович – кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення Житомирського державного технологічного університету.

Наукові інтереси:
- комп’ютерне та математичне моделювання технічних та екологічних систем;
- конструювання інтерфейсів ПЗ;
- функціональні мови програмування.

E-mail: sekitochuk@gmail.com.

Єфіменко Андрій Анатолійович – кандидат технічних наук, завідувач кафедри комп’ютерної інженерії та кібербезпеки Житомирського державного технологічного університету.

Наукові інтереси:
- проектування, побудова та експлуатація комп’ютерних мереж;
- захист інформації в комп’ютерних мережах;
- технології та засоби віртуалізації комп’ютерних мереж.

E-mail: yefimenko.andrii@gmail.com.

Стаття надійшла до редакції 16.08.2018.