Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему: «Метод криптографічного перетворення на основі нейронних мереж»
08-20.МКР.014.00.000 ПЗ

Виконав: студент 2 курсу, групи 1 БС-18м
спеціальності 125 – Кібербезпека (Безпека інформаційних та комунікаційних систем)

_____ Татарчук А. Є.

Керівник: к. т. н., доцент каф. ЗІ

_____ Куперштейн Л. М.

Рецензент
к. т. н., доц., доц. кафедри ОТ

_____ Крупельницький Л. В.

Вінниця - 2019 року

Вінницький національний технічний університет

Факультет Інформаційних технологій та комп'ютерної інженерії
Кафедра Захисту інформації

Спеціальність – 125 Кібербезпека
Спеціалізація – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д.т.н., проф.
_____В. А. Лужецький
"____" _____2019 року

З А В Д А Н Н Я

НА КОМПЛЕКСНУ МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Татарчук Артему Євгеновичу

1. Тема роботи: «Метод криптографічного перетворення на основі нейронних мереж»
керівник роботи: Куперштейн Леонід Михайлович, к. т. н., доцент,
затверджені наказом №254 ректора ВНТУ від 02 вересня 2019 року № 2.
2. Строк подання студентом роботи 12 грудня 2019 р.
3. Вихідні дані до роботи:
   - операційна система – Windows 10;
   - для 32-х та 64-х бітних систем;
   - початкові дані для створення ключа;
   - обробка нейронними мережами;
   - середовище розробки – PyCharm.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. 1. Аналіз предметної області. 2. Розробка технічного проекту. 3. Розробка робочого проекту. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Схема криптосистеми з використанням нейронних мереж (плакат, А4); Схема нейронної мережі (плакат, А4); Схема процесів інформаційної технології (плакат, А4); Загальна схема двох нейромереж (плакат, А4); Схема алгоритму обміну ключами (плакат, А4); Алгоритм роботи криптографічного модулю (плакат, А4); фрагмент інтерфейсу додатку (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Куперштейн Л.М., к.т.н., доц. каф.ЗІ		
2	Куперштейн Л.М., к.т.н., доц. каф.ЗІ		
3	Куперштейн Л.М., к.т.н., доц. каф.ЗІ		
4	Мацкевічус С. С. ст. вик. каф. ЕПВМ		

7. Дата видачі завдання _____ 2019 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів комплексної магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2019 – 04.09.2019	
2	Розробка технічного завдання	05.09.2019 – 15.09.2019	
3	Аналіз літературних джерел за напрямком комплексної магістерської кваліфікаційної роботи	16.09.2019 – 22.09.2019	
4	Розробка рішень, моделей, алгоритмів	23.09.2019 – 29.09.2019	
5	Практична реалізація, моделювання, експериментування, результати	30.09.2019 – 12.10.2019	
6	Розробка економічного розділу	14.10.2019 – 10.11.2019	
7	Аналіз виконання ТЗ, висновки	11.11.2019 – 17.11.2019	
8	Оформлення пояснювальної записки, підготовка ілюстративного матеріалу	18.11.2019 – 24.11.2019	
9	Попередній захист МКР	25.11.2019 – 30.11.2019	
9	Виправлення зауважень, перевірка МКР на плагіат	28.11.2019 – 01.12.2019	
10	Представлення МКР до захисту, рецензування	02.12.2019 – 10.12.2019	
11	Захист МКР	11.12.2019 – 14.12.2019	

Студент _____ Татарчук А. Є.
(підпис)

Керівник роботи _____ Куперштейн Л.М.
(підпис)

АНОТАЦІЯ

У роботі було досліджено базові принципи роботи деревовидних машин парності і їх синхронізації. Було проведено аналіз літературних джерел у напрямку криптографії, передачі секретного ключа, нейрокриптографії. На основі цього аналізу було визначено необхідність розробки методу передачі ключа за допомогою нейронних мереж та виконано постановку задачі дослідження. Досліджено існуючі роботи у цьому напрямку. Метод створено у вигляді програмного опису мовою Python.. Визначено термін окупності програмного засобу.

ABSTRACT

The robot base has the basic principle of the robot of tree-shaped machines in particular and synchronization. Bulo conducted an analysis of literary dzherel in direct cryptography, transmission of the secret key, neurocryptography. On the basis of the whole analysis of bulo, the need for working out the method of transmitting the key for another neural measure and the task setting is completed. Doslidzheno isnuyuchi robots have tsyomu straightforwardly. The method has been tweaked for the software view. I will describe my Python. The term for the payload of the software is marked.

ЗМІСТ

ВСТУП..... 5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ..... 8
1.1 Науково-технічне обґрунтування 8
1.2 Аналіз криптографічних перетворень..... 10
1.3 Аналіз застосувань нейронних мереж в криптографії 18
1.4 Формалізація вимог та постановка задачі 27
2 МЕТОД ГЕНЕРАЦІЇ ТА ПЕРЕДАЧІ СИМЕТРИЧНОГО КЛЮЧА ШИФРУВАННЯ.....29
2.1 Метод передачі симетричного ключа шифрування за допомогою нейронних мереж 29
2.2 Архітектура нейронних мереж 30
2.3 Можливі атаки на перехоплення ключа..... 35
3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ.....41
3.1 Обґрунтування вибору засобів програмної реалізації методу 41
3.2 Розробка програмного засобу 42
3.3 Тестування програмного засобу 47
4 ЕКОНОМІЧНА ЧАСТИНА.....41
4.1 Оцінювання економічного потенціалу розробки.....48
4.2 Розрахунок кошторису витрат на розробку.....51
4.3 Прогнозування прибутків від застосування результатів розробки.....55
4.4 Розрахунок ціни реалізації виробу та чистого прибутку.....57
ВИСНОВКИ.....62
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....66
ДОДАТОК А.....69
ДОДАТОК Б.....74

ВСТУП

У сучасному світі із розвитком інформаційних технологій, дуже важливим є питання вибору самого надійного та ефективного способу забезпечення інформаційної безпеки підприємства. Криптографічні методи захисту інформації важливі для забезпечення безпеки загальних даних будь-якої компанії, нюансів фінансової діяльності, інформації про постачальників і клієнтів і багатьох інших відомостей, конфіденційність і цілісність яких безпосередньо впливає на комерційний успіх і душевну рівновагу. Широке застосування комп'ютерних технологій і постійне збільшення обсягу інформаційних потоків викликає постійне зростання інтересу до криптографії.

Проблема обміну ключами є однією з ключових проблем в інформаційній безпеці. На даний момент, один з найвідоміших протоколів обміну ключами - протокол Діффі-Хеллмана. Даний протокол базується на теорії чисел, а конкретно на складності дискретного логарифма. Але в сучасному світі, при постійно зростаючих потужностях обчислювальних машин, і ймовірне появі квантових комп'ютерів в найближчому майбутньому, дана обчислювальна проблема може зникнути, що поставить під удар безпеку протоколів обміну ключами, заснованих на теорії чисел.

Існує і альтернативні рішення проблеми обміну ключами, які не засновані на теорії чисел. Одним з таких, рішень є нейрокриптографічний протокол обміну ключами. За допомогою синхронізації двох деревовидних машин парності, можна забезпечити безпечну передачу даних між двома абонентами через публічний канал.

Актуальність Криптографія з відкритим ключем заснована на проблемі великих обчислень, або в якій невідомий ефективний алгоритм вирішення. Коли ефективний алгоритм вирішення не відомий, розвиток обчислювальної потужності та розвиток технологій зростає, наприклад, хмарні обчислення або використання великих ботнетів, дозволяють проводити жорстокі атаки. Як відповідь на це ми спостерігаємо постійне збільшення довжини ключів, що використовуються в

криптографічному алгоритмі, до безпечного розміру, але це також збільшує час, необхідний для шифрування та дешифрування повідомлень. Альтернативний підхід - це пошук нових методів криптографії, який не покладається на обчислювально важкі проблеми. Цікавими є спроби застосування методів штучного інтелекту в криптографії. Використання штучних нейронних мереж для обміну криптографічними ключами для подальшого спілкування партнера за допомогою відкритого каналу зв'язку є перспективним напрямком.

Об’єктом є процес передачі секретного ключа між сторонами.

Предметом є методи криптографічних перетворень.

Метою магістерської кваліфікаційної роботи є створення методу формування та передачі секретного ключа за допомогою нейронних мереж.

Для досягнення мети необхідно розв’язати такі задачі:

– проаналізувати використання нейронних мереж в криптографії;

– розробити метод формування та передачі ключа шифрування за допомогою нейронних мереж;

– розробити програмний засіб;

– виконати тестування розробленого програмного засобу;

– виконати економічну доцільність обґрунтування методу;

Наукова новизна. Запропоновано метод передачі симетричного ключа шифрування, що полягає у використанні двох взаємно синхронізованих нейронних мереж прямого поширення для його формування та використання регістру зсуву з лінійним зворотнім зв’язком для формування помилкових бітів, для заплутування зловмисника. Це дозволяє забезпечити можливість передачі ключа по відкритому каналу, що унеможливлює формування ключа зловмисником.

Практична цінність. Розроблено програмний засіб на основі нейронних мереж, що дозволяє організувати захищену передачу даних по відкритому каналу.

Результати наукової роботи доповідалися на XLVIII науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії

ВНТУ (Вінниця, ВНТУ, 2019) та конференції Молодь в науці: дослідження, проблеми, перспективи (Вінниця, ВНТУ, 2019).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Науково-технічне обґрунтування

У комп'ютерній безпеці існує наступна проблема: два партнера А і В хочуть обмінятися секретним повідомленням через канал. Для забезпечення захисту вмісту повідомлення від опонента Е, А шифрує своє повідомлення, використовуючи швидкий алгоритм симетричного шифрування. Але тепер В необхідно знати ключ, який належить А для того, щоб прочитати повідомлення. Існують системи з секретним і відкритим ключем Система з секретним ключем зображена на рисунку 1.1.

Рисунок 1.1 – Система з секретним ключем

Основним недоліком симетричного шифрування є загальний ключ для шифрування і дешифрування, що ускладнює використання симетричного шифрування в розподілених системах. Проблема викликана тим, що ключ передається через загальнодоступний канал зв'язку, через який зловмисник може викрасти як саму інформацію, так і ключ для її дешифрування.

Для вирішення даної проблеми обміну ключами існує кілька рішень. Перше, А і В можуть використовувати другий закритий канал для обміну ключами, також

вони можуть обмінятися ключами при особистій зустрічі. Але зазвичай це вкрай складно або навіть неможливо здійснити. В якості альтернативи, ми можемо використовувати криптографію з відкритим ключем. На рисунку 1.2 зображено систему з відкритими ключами.

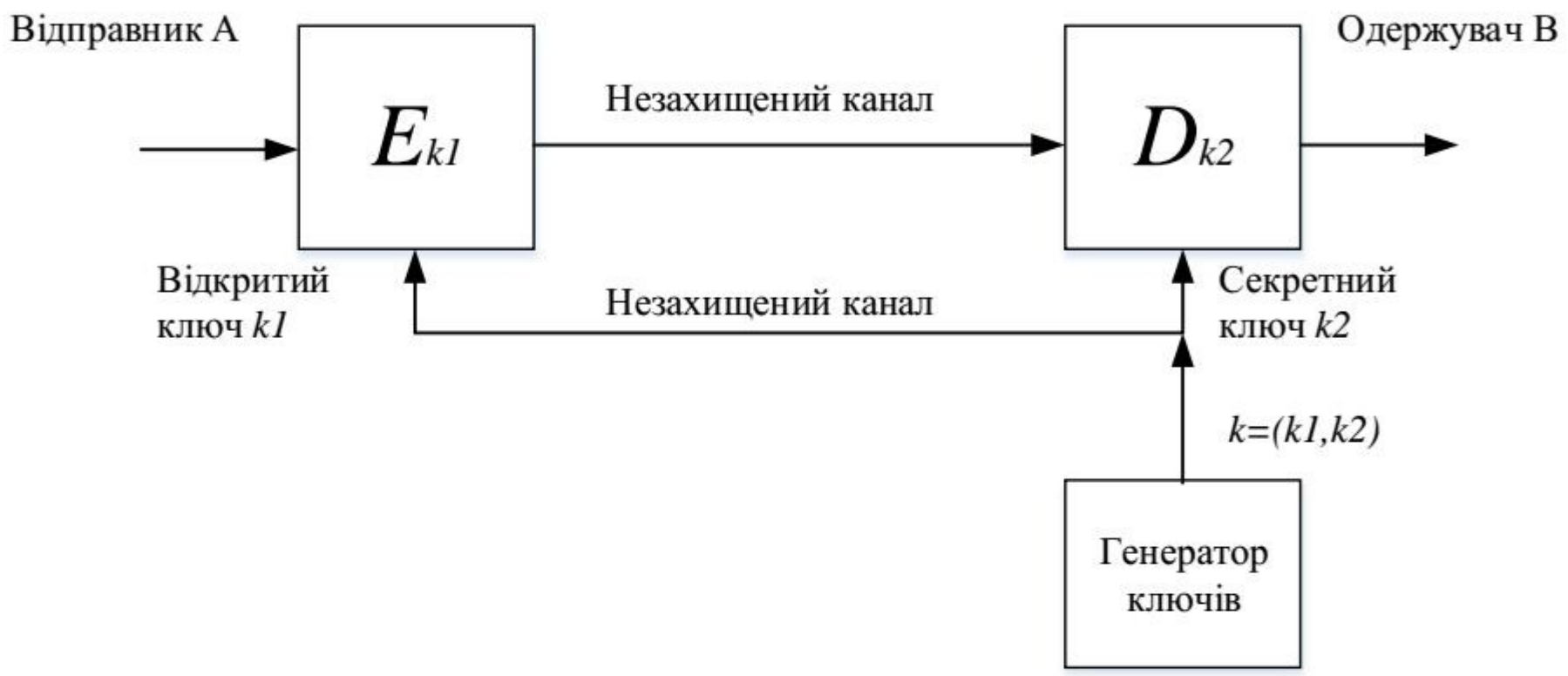


Рисунок 1.2 – Система з відкритим ключем

В даному випадку використовується алгоритм асиметричного шифрування, таким чином пара публічних ключів A і B може бути передана між партнерами без необхідності тримати їх в секреті. Але асиметричний алгоритм шифрування набагато повільніший, ніж симетричний. Однак, можна досягти такого ж результату, використовуючи протокол обміну ключами. У цьому випадку повідомлення передається через публічний канал після чого A і B генерують закритий ключ, грунтуючись на інформації, що передається. Але E не може отримати ключ, тому що спостереження за інформацією переданої по каналу недостатньо.

Крім того, процеси, що беруть участь у створенні відкритого ключа дуже складні та трудомісткі. Тому актуальним є питання розробки нових методів передачі ключа.

1.2 Аналіз криптографічних перетворень

Криптографія – наука про способи перетворення інформації з метою її захисту від несанкціонованого доступу. Одним із видів такого перетворення є шифрування, яке забезпечує практичну неможливість читання або модифікації інформації зловмисниками. Існує цілий ряд алгоритмів шифрування даних, які можна розбити на дві великі групи, що показано на рисунку 1.2.

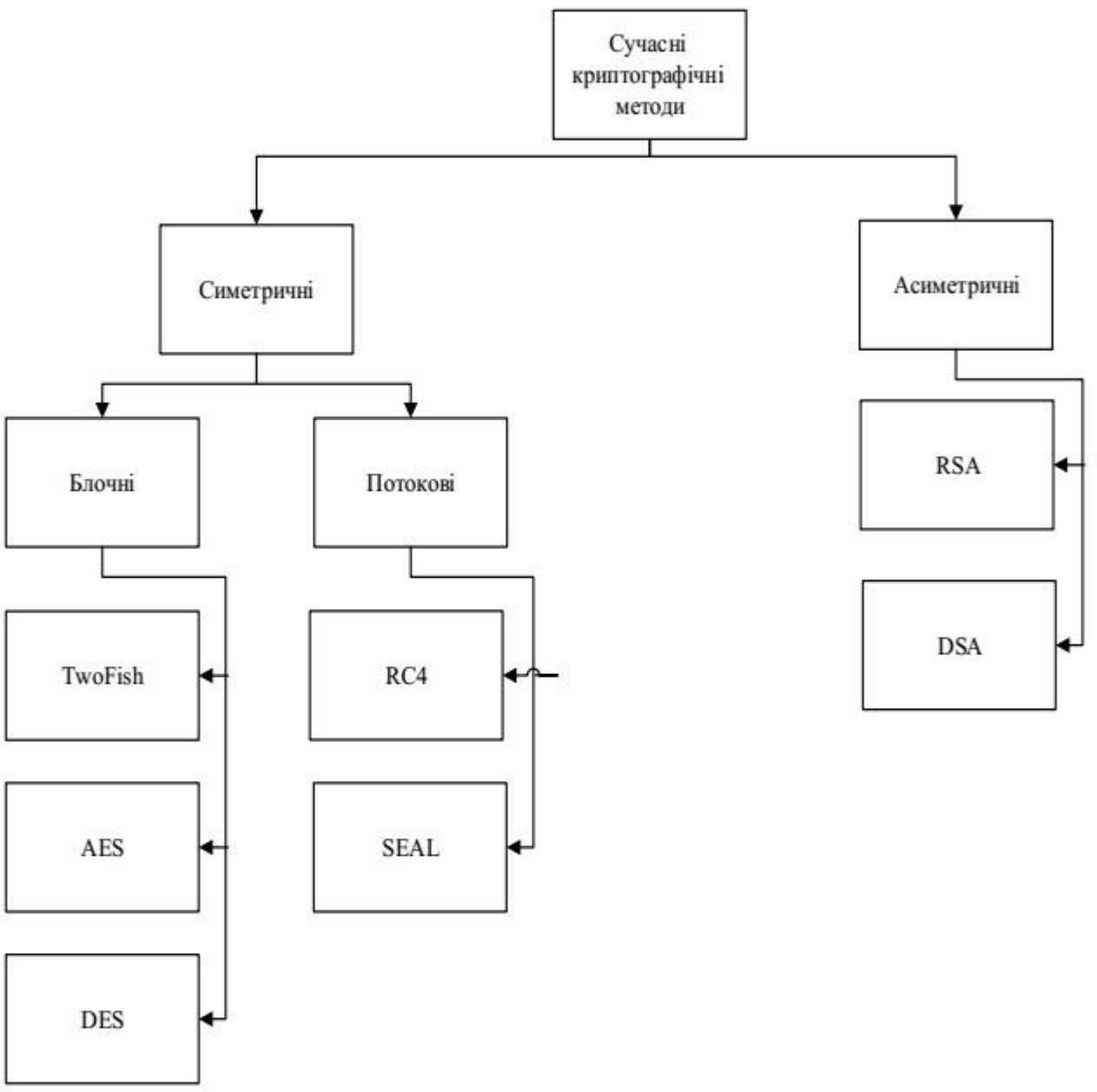


Рисунок 1.2 – Алгоритми криптографії

Проведемо порівняльний аналіз симетричної та асиметричної криптографії. Характерною рисою симетричної системи є використання одного і того ж ключа для виконання операції шифрування і розшифрування, яке можливе лише тоді, якщо існує безпечний канал передачі інформації. Це не стосується військової справи, розвідки, фінансово-кредитних операції тощо. У таких ситуаціях використовують асиметричні алгоритми шифрування. Дані системи криптографічних перетворень характеризуються тим, що для шифрування даних використовується один ключ (відкритий, тобто доступний користувачам), а для

розшифрування – інший (секретний) ключ. Ця властивість дозволяє в певній мірі вирішити проблему розподілу ключів між користувачами, яка є основним недоліком симетричних систем. Для гарантії захисту даних, до асиметричних систем шифрування ставляться дві найважливіші умови [2] :

– перетворення відкритого тексту повинно бути незворотнім і виключати його відновлення на основі відкритого ключа;

– визначення секретного ключа на основі відкритого має бути неможливим для сучасного рівня розвитку обчислювальних засобів.

Вирішення задач автентифікації, розповсюдження ключів відкритими каналами зв'язку, застосування електронного підпису реалізується лише засобами асиметричної криптографії. Однак слід зазначити, що алгоритми асиметричних криптосистем настільки трудомісткі в порівнянні зі звичайними симетричними алгоритмами, що на практиці раціонально їх використовувати там, де обсяг шифрованого інформації незначний, але дуже важливий. Практичний досвід показує, що застосування асиметричних алгоритмів шифрування не дозволяє забезпечити інтерактивний режим роботи сучасних інформаційно-телекомунікаційних систем. Таким чином, очевидна необхідність використання в таких системах пристроїв шифрування, які побудовані на симетричних криптографічних алгоритмах. Симетричні алгоритми шифрування можна розділити на потокові та блочні. Потокові алгоритми шифрування послідовно обробляють текст повідомлення, блочні алгоритми, в свою чергу, працюють з блоками фіксованого розміру. Як правило, довжина блоку дорівнює 64 бітам, але, в алгоритмі AES використовуються блоки довжиною 128 біт. Симетричні алгоритми шифрування не завжди використовуються самостійно. Блокові шифри є основою, на якій реалізовані практично усі криптосистеми. Методика створення ланцюжків із зашифрованих блоковими алгоритмами байт дозволяє шифрувати ними пакети інформації необмеженої довжини. Така властивість блокових шифрів, як швидкість роботи, використовується асиметричними криптоалгоритмами, повільними за своєю природою. Відсутність статистичної кореляції між бітами

вихідного потоку блокового шифру використовується для обчислення контрольних сум пакетів даних і в хешуванні паролів.

IDEA, CAST128, BlowFish, DES, ГОСТ є всесвітньо визнаними стійкими алгоритмами і публікацій про універсальні методи їх злому в засобах масової інформації на момент створення матеріалу не зустрічалося.

Віддалення статистичних залежних у відкритому тексті можливо з допомогою попередня архівування, но воно не вирішує завдання повністю, тому що у файлі залишається службова інформація архіватора, і не завжди технічно припустимо.

Класичними прикладам симетричних криптографічних алгоритмів є:

- проста перестановка;
- одиночна перестановка по ключу;
- подвійна перестановка;
- перестановка «Магічний квадрат».

Проста перестановка без ключа - один з найпростіших методів шифрування. Повідомлення записується в таблицю по стовпцях. Після того, як відкритий текст записаний колонками, для освіти шифртекста він зчитується по рядкам. Для використання цього шифру відправнику і одержувачу потрібно домовитися про спільний ключі в вигляді розміру таблиці. Об'єднання букв в групи не входить в ключ шифру і використовується лише для зручності запису несмислового тексту.

Більш практичний метод шифрування, званий одиночній перестановкою по ключу, дуже схожий на попередній. Він відрізняється лише тим, що колонки таблиці переставляються за ключовим словом, фразою або набору чисел довжиною в рядок таблиці.

Для додаткової секретності можна повторно шифрувати повідомлення, яке вже було зашифровано. Цей спосіб відомий під назвою подвійна перестановка. Для цього розмір другої таблиці підбирають так, щоб довжини її рядків і стовпців відрізнялися від довжин в першій таблиці. Найкраще, якщо вони будуть взаємно

простими. Крім того, в першій таблиці можна переставляти стовпці, а в другій рядка. Нарешті, можна заповнювати таблицю зигзагом, змійкою, по спіралі або якимось іншим способом. Такі способи заповнення таблиці якщо і не посилюють стійкість шифру, то роблять процес шифрування набагато більш цікавим.

Магічними квадратами називаються квадратні таблиці з вписаними в їх клітини послідовними натуральними числами від 1, які дають в сумі по кожному стовпцю, кожному рядку і кожній діагоналі одне і те ж число. Подібні квадрати широко застосовувалися для вписування шіфруемоготексту за наведеною в них нумерації. Якщо потім виписати вміст таблиці по рядках, то виходила шифровка перестановкою букв. На перший погляд здається, ніби магічних квадратів дуже мало. Проте, їх число дуже швидко зростає зі збільшенням розміру квадрата. Так, існує лише один магічний квадрат розміром 3 х 3, якщо не брати до уваги його повороти. Магічних квадратів 4 х 4 налічується вже 880, а число магічних квадратів розміром 5 х 5 близько 250000. Тому магічні квадрати великих розмірів могли бути хорошою основою для надійної системи шифрування того часу, тому що ручний перебір всіх варіантів ключа для цього шифру був немислимий.

В сучасних криптоситемах, використовуються комбінації симетричних та асиметричних алгоритмів, для того, аби отримати переваги обох схем. До таких систем належить SSL, PGP та GPG. Асиметричні алгоритми використовуються для розповсюдження ключів швидших симетричних алгоритмів. До деяких відомих, поширених алгоритмів з гарною репутацією належать: Twofish, Serpent, AES, Blowfish, CAST5, RC4 та IDEA [3]. В цілому ряді задач для повноцінного забезпечення інформаційної безпеки використання лише криптографічних методів є недостатнім, оскільки вони не дозволяють приховати власне факт передачі й зберігання конфіденційної інформації. Подібні задачі можливо вирішувати з застосуванням методів крипто-стеганографічних алгоритмів. Крипто-стеганографічна система захисту інформації – це складний інформаційний комплекс методів та засобів, загальна стійкість якого залежить від правильного узгодження криптографічної і стеганографічної складових системи. Ключову роль

при цьому відіграють алгоритми узгодження, які дають змогу перетворити рівномірно розподілені бітові послідовності, отримані на виході криптографічних алгоритмів, на бітові послідовності, аналогічні тим, що використовуються для вкраплення стеганографічними алгоритмами у пусті контейнери. Вимогою коректного використання цих алгоритмів є точна статистична відповідність вхідних і вихідних даних. Інтеграція крипто-стеганографічних алгоритмів дає можливість позбутися вразливих сторін відомих методів захисту інформації та розробити ефективніші з позицій обчислювальної складності і стійкості до зламу нові методи розв'язання задач інформаційної безпеки як програмного додатку так і інформаційних потоків даних.

У 1976 р У. Діффі і М. Хеллмана дали опис криптографічних систем з відкритим ключем (public key cryptosystem), в основу яких покладені методи класичної та сучасної алгебри. Пропонується розглядати таку систему шифрування і / або електронного підпису, при якій відкритий ключ передається по незахищеному (відкритого) каналу і надалі використовується для перевірки електронного підпису і для шифрування повідомлення. При цьому для створення електронного підпису і для розшифровки повідомлення використовується закритий ключ [2]. В даній схемі шифрування використовує відкритий ключ, розшифрування - закритий.

Основними видами асиметричних шифрів є:

1). RSA (Rivest-Shamir-Adleman) - криптографічний алгоритм з відкритим ключем, який базується на складності завдання факторизації великих цілих чисел;

2). DSA (Digital Signature Algorithm) - криптографічний алгоритм з відкритим ключем тільки для створення електронного підпису. Алгоритм заснований на обчислювальної складності взяття логарифмів в кінцевих полях;

3). Elgamal (шифросистемамі Ель-Гамаля) - асиметричні алгоритми шифрування, заснована на обчислювальної складності дискретних логарифмів в кінцевому полі. Криптосистема включає в себе як алгоритм шифрування, так і алгоритм цифрового підпису;

4). Diffie-Hellman (Обмін ключами Діффі - Хелмана) – криптографічний протокол, дозволяє декільком (двом і більше) абонентам отримати загальний секретний ключ, використовуючи незахищений канал зв'язку. отриманий ключ використовується для шифрування подальшого обміну за допомогою алгоритмів симетричного шифрування;

5). ECDSA (Elliptic Curve Digital Signature Algorithm) - алгоритм з відкритим ключем для створення цифрового підпису та інші.

Одним з головних переваг асиметричних шифрів є відсутність необхідності попередньої передачі секретного ключа по захищеному каналу зв'язку. В даному випадку використовується пара «закритий ключ – відкритий ключ», значення якої з одного боку пов'язані, а з іншого бок обчислення закритого ключа з відкритого ключу практично неможливо [2].

На практиці асиметричні криптосистеми використовуються в поєднанні з іншими алгоритмами. Це пов'язано, перш за все, з тим, що в чистому вигляді вони вимагають істотних обчислювальних ресурсів

Криптографічні системи з відкритим ключем широко застосовуються в різних стандартах цифрового підпису та мережевих протоколах. Такі криптосистеми будуються шляхом вибору класу задач, для якого не відомий ефективний алгоритм рішення (в якому випадку) і в ньому виділяється підзадача, для якої такий алгоритм існує. Далі вибране завдання маскують під задачу загального вигляду і вибирають ключ шифрування. При цьому в Як секретного ключа використовується інформація, що дозволяє перевести вибране завдання в початковий вигляд.

Моделі атаки або типи атаки у криптоаналізі є класифікацією криптографічних атак, що визначають вид доступу, який криптоаналітик має до системи, що піддається атаці, при спробі «зламати» зашифроване повідомлення (також відоме як шифротекст). Чим більший доступ може отримати криптоаналітик, тим більше корисної інформації може бути вилучено та використано для порушення роботи системи [3].

Деякі загальні моделі атак:

Атака на основі шифротексту (COA) — в даному виді атаки передбачається, що криптоаналітик має доступ лише до шифрованого тексту та не має доступу до відкритого тексту. Цей тип атаки є найбільш поширеним випадком, який зустрічається в криптоаналізі в реальному житті, але є найслабшою атакою через відсутність достатньої інформації для криптоаналітиків. Сучасні шифри повинні бути дуже стійкими до такого типу атаки. Насправді, успішний криптоаналіз у моделі COA зазвичай вимагає, щоб криптоаналітик мав певну інформацію про відкритий текст, таку, як мова, на якій написаний відкритий текст, стандартні дані протоколу або фрейми, які є частиною відкритого тексту, і т. д. [2]

Атака грубої сили або вичерпний пошук ключа — в цій атаці всі можливі ключі проходять випробування, доки не знайдеться правильний. Кожен шифр, крім нерозривних методів теоретико-інформаційної безпеки[en] як, наприклад, шифр Вернама, є вразливим до цього методу, оскільки його складність не залежить від шифру, а лише від довжини ключа — це не вважається справжнім «криптоаналізом» шифру. Якщо ключ має N бітів, існує 2N можливих ключів для перевірки, тому атака грубої сили може вразити шифрування у найгіршому випадку, пропорційному 2N і за середній час 2N-1. Це часто використовується як стандарт порівняння для інших атак. Атаку грубої сили силу можна застосовувати лише в налаштуваннях шифрування, але криптоаналітик повинен мати достатньо інформації про відкритий текст (принаймні N бітів), щоб дозволити ідентифікацію правильного ключа після її спроби.

Атака з відомим відкритим текстом (KPA) — в цьому типі атаки передбачається, що криптоаналітик має доступ, принаймні, до обмеженої кількості пар відкритого тексту та відповідного шифрованого тексту. Цікавий приклад можна навести з історії Другої світової війни, під час якої Антигітлерівська коаліція використовувала відомі відкриті тексти у своїй успішній операції криптоаналізу Enigma. Зразки простих текстів називаються «шпаргалки»; цей

термін походить від Блечлі-Парку та британської операції розшифровки у Другій світовій війні.

Атака на основі підібраного відкритого тексту (CPA) — в цій атаці криптоаналітик може вибрати ряд простих текстів, які будуть шифруватися, і мати доступ до результуючого шифрованого тексту. Це дозволяє йому досліджувати будь-які області простору станів відкритого тексту, і може дозволити йому використовувати вразливості та випадкові наслідки, які з'являються лише з певними відкритими текстами. У широко використовуваній асиметричній криптосистемі ключ, що застосовується для шифрування відкритого тексту, публічно розповсюджується, і кожен може використовувати його, дозволяючи криптоаналітику створювати зашифрований текст будь-якого відкритого тексту, який він хоче. Отже, алгоритми публічного ключа повинні бути стійкими до всіх подібних атак.

Атака на основі підібраного шифротексту (CCA) — в цій атаці аналітик може вибрати довільний зашифрований текст і отримати доступ до розшифрованого тексту на основі попереднього. У реальному житті це вимагатиме, щоб аналітик мав доступ до каналу зв'язку та повідомлень одержувача.

Атаки з моделлю відкритого ключа — тип атаки, де атакуючий має певні знання про ключі шифру.[3]

Атака сторонніми каналами — Це, насправді, не криптоаналітична атака, і не залежить від якості шифру. Це стосується використання інших даних про процес шифрування або дешифрування для отримання інформації про повідомлення, наприклад, електронний шум, вироблений шифрувальними машинами, звук, вироблений натисканням клавіш, або вимірювання часу, необхідного для виконання різних обчислень.

Різні моделі атаки використовуються для інших криптографічних примітивів або, більш загально, для всіх типів систем безпеки. Прикладами таких моделей атак є: адаптивна вибрана атака повідомлень для цифрового підпису.

1.3 Аналіз застосувань нейронних мереж в криптографії

1.3.1 Нейронні мережі

Штучні нейронні мережі (ШНМ) - це програмна імплементація нейронних структур нашого мозку. Вони можуть змінювати тип переданих сигналів в залежності від електричних або хімічних сигналів, які в них передаються. Нейронна мережа у людському мозку - величезна взаємопов'язана система нейронів, де сигнал, який передається одним нейроном, може передаватися у тисячі інших нейронів. Навчання відбувається через повторну активацію деяких нейронних з'єднань. Через це збільшується імовірність виведення потрібного результату при відповідній вхідній інформації (сигналах). Такий вид навчання використовує зворотний зв'язок - при правильному результаті нейронні зв'язки, які виводять його, стають більш щільними.

Штучні нейронні мережі імітують поведінку мозку у простішому вигляді. Вони можуть бути навчені контрольованим та неконтрольованим шляхами. У контрольованій ШНМ, мережа навчається шляхом передавання відповідної вхідної інформації та прикладів вихідної інформації. Наприклад, спам-фільтр у електронній поштовій скриньці: вхідною інформацією може бути список слів, які зазвичай містяться у спам-повідомленнях, а вихідною інформацією - класифікація для відповідного повідомлення (спам, чи не спам). Такий вид навчання додає ваги зв'язкам ШНМ, але це буде обговорено пізніше.

Неконтрольоване навчання у ШНМ намагається "змусити" ШНМ "зрозуміти" структуру переданої вхідної інформації "самостійно". Ми не будемо розглядати це у даному пості [4].

Біологічний нейрон імітується у ШНМ через активаційну функцію. У задачах класифікації (наприклад визначення спам-повідомлень) активаційна функція повинна мати характеристику "вмикача". Іншими словами, якщо вхід більше, ніж деяке значення, то вихід повинен змінювати стан, наприклад з 0 на 1 або -1 на 1. Це

імітує "включення" біологічного нейрону. У якості активаційної функції зазвичай використовують сигмоїдну функцію:

Біологічні нейрони ієрархічно з'єднані в мережамі, де вихід одних нейронів є входом для інших нейронів. Ми можемо представити такі мережі у вигляді з'єднаних шарів з вузлами. Кожен вузол приймає зважений вхід, активує активаційну функцію для суми входів, та генерує вихід. На рисунку 1.4 зображено вузли з входами

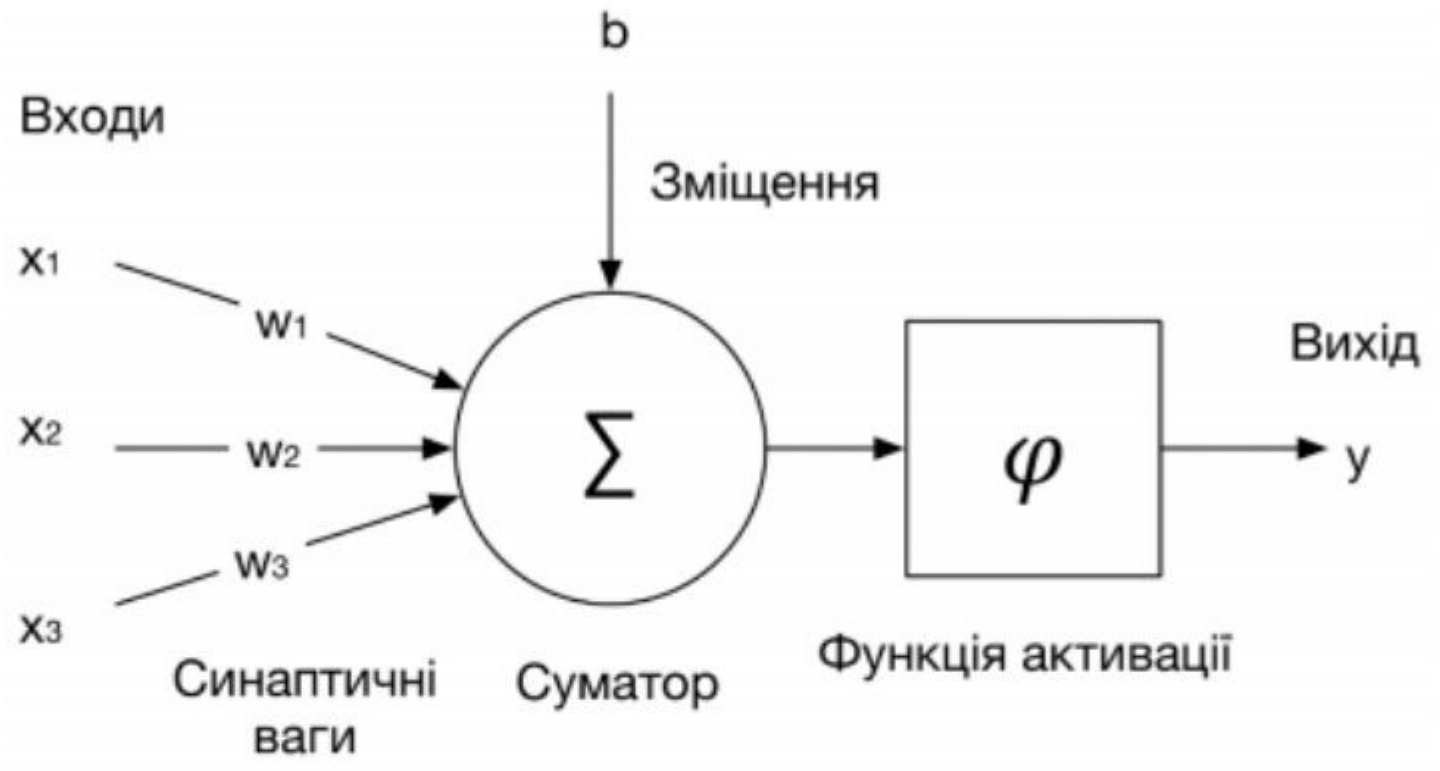


Рисунок 1.4 – Вузли з входами

Коло на картинці зображує вузол. Вузол є "місцерозташуванням" активаційної функції, він приймає зважені входи, сумує їх, а потім вводить їх в активаційну функцію. Вивід активаційної функції представлений через h. Примітка: у деякій літературі вузол також називають персептроном.

За вагу беруться числа (не бінарні), які потім множаться на вході і сумуються у вузлі. Іншими словами, зважений вхід у вузол має вигляд:

$$x_1w_1+x_2w_2+x_3w_3+b$$

Де w_i - числові значення ваги (b ми будемо обговорювати пізніше). Ваги нам потрібні, вони є значеннями, які будуть змінюватись протягом процесу навчання. b є вагою елемента зміщення на +1, включення ваги b робить вузол гнучкішим.

У нейронних мережах, вони представляють з'єднання між нейронами і здатностями, що підсилюють або послабляють нейронний сигнал, наприклад, вмикають їх на сигналах, в той час модифікуючи їх. Зміна сигналу нейронної мережі, радіаційна нейтронна мережа здатна вказувати на вплив виходу нейрона, слідуючи, активація нейрона залежить від свого входу і вагів. Входи можуть бути отримані з виходів інших нейронів, або з зовнішнього світу. Таким чином, як і колись використовується внутрішня презентація нейтронних мереж та впливає на її вихід, ми можемо розглянути їх як знання нейронної мережі та виміряти вісов, що ввели в зміну

Вводом для активаційної функції у цьому вузлі є просто $x_1 w_1$.

Природні нейрони організовані шарами, кожен з яких забезпечує певний рівень обробки. Наприклад, вхідний шар забезпечує напівпроцедуру із зовнішнього світу, в той час, як вихідний шар може вказати вплив на зовнішній світ. Між тими шарами може бути наявне неточне кількість скритих шарів, які не взаємодіють із зовнішнім світом наперед. Усі нейрони на рівні мають загальний векторний вихідний вектор вхідної та функціональної активності.

Нейронні мережі можуть складатися з декількох шарів, формуючи, так звані, багатошарові нейронні мережі. Шари можуть бути розділені на три класи:

1) Вхідний шар.

2) Прихований шар.

3) Вихідний шар.

На практиці, додатковий рівень додає ще один рівень абстракції для зовнішніх стимулів, тим самим підвищуючи здатність нейронної мережі представляти більш складні знання.

Нейронні мережі можуть мати різну архітектуру, в залежності від того, як нейрони або нейронні шари пов'язані один з одним [4]. Архітектура нейронної розробляється під певні потреби. Нейронні мережі можуть бути використані для вирішення великої кількості проблем і, в залежності від природи даної проблеми, нейронна мережа повинна бути розроблена в з урахуванням того, щоб вирішити

дану проблему максимально ефективно. Існують такі різновиди архітектур нейронних мереж:

- По нейронних зв'язках:
  - одношарові.
  - багатошарові.
- По поширенню сигналу:
  - прямого поширення.
  - зі зворотним зв'язком.

В одношаровій архітектурі всі нейрони лежать на одному і тому ж рівні, формуючи один єдиний шар. Нейронна мережа отримує вхідний сигнал і направляє його нейронам, які обчислюють вихідний сигнал. Нейрони можуть бути сильно пов'язані один з одним, з рекурентними зв'язками або без. На рисунку 1.5 зображено схему одношарової нейронної мережі.

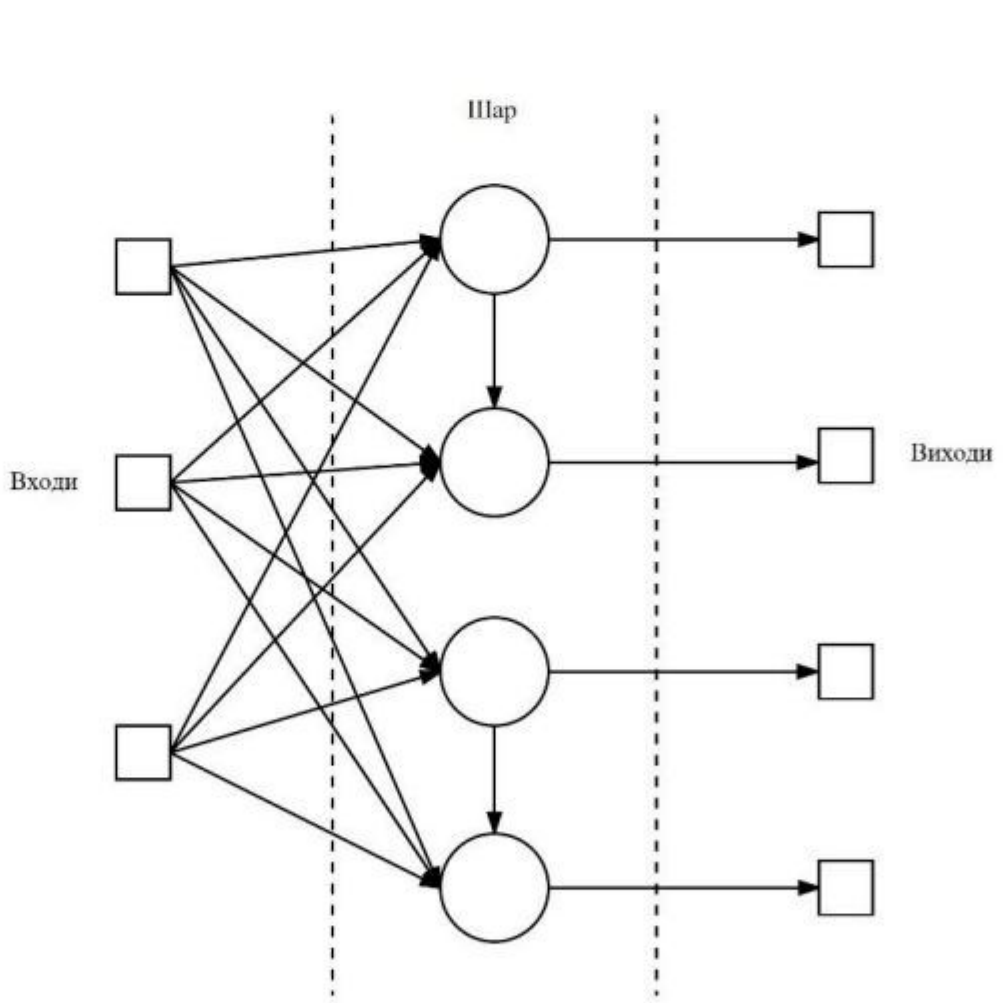


Рисунок 1.5 – Схема одношарової нейронної мережі

У багатошаровій архітектурі, нейрони розділені на кілька шарів. В рамках рівня, нейрони мають загальний вхідний сигнал і одну функцію активації. Свої

вихідні сигнали кожен шар повідомляє наступного на вхід. На рисунку 1.6 зображено схему багатошарової нейронної мережі.

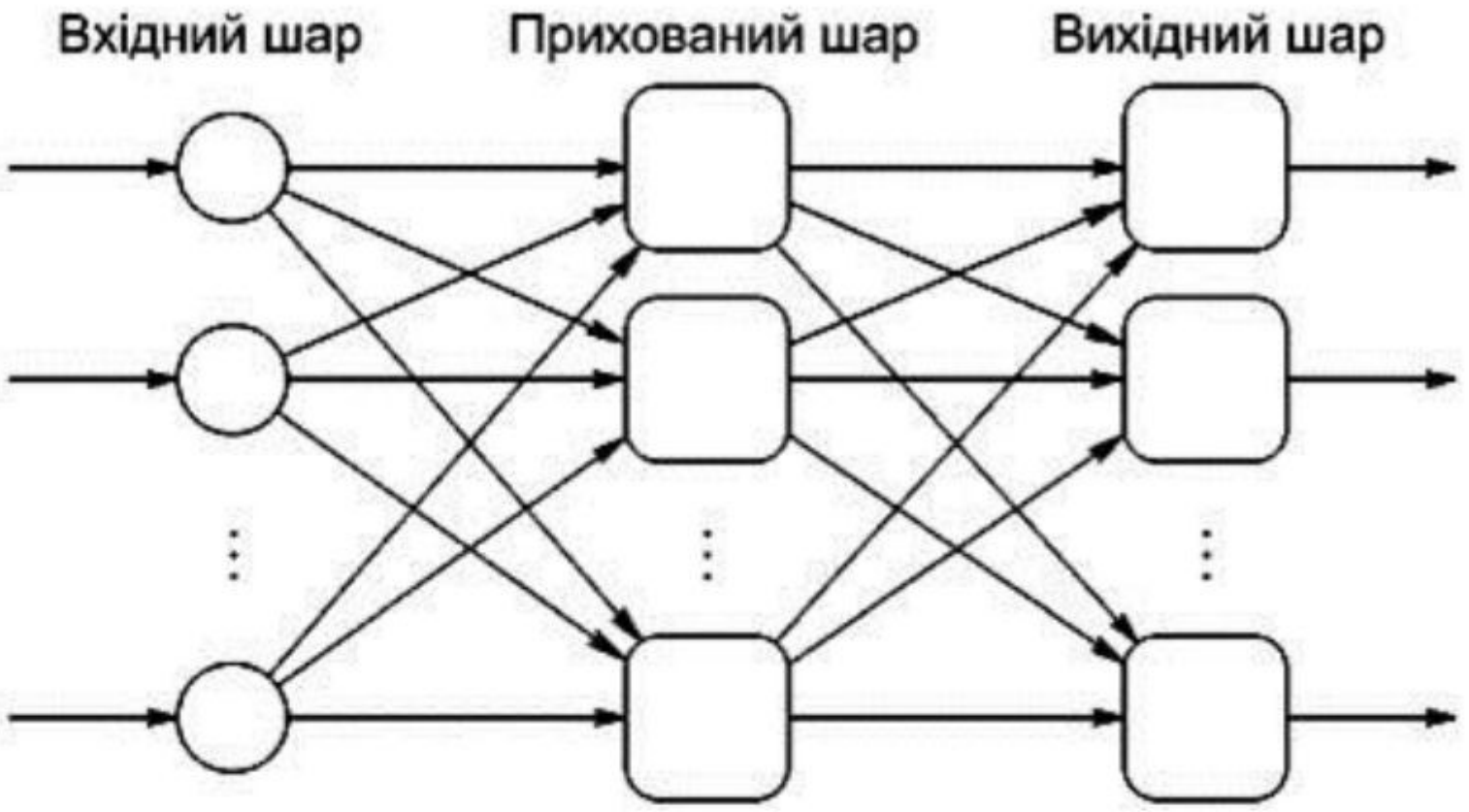


Рисунок 1.6 – Схема багатошарової нейронної мережі

Радіальні базисні функції і багатошаровий персептрон, є хорошими прикладами цієї архітектури. Так як ці мережі мають багатошарову архітектуру обробки, ці мережі адаптуються до навчання за нелінійним даними[4].

Сигнал нейронної мережі прямого поширення може розповсюджуватись або тільки в одному напрямку, або мати рекурентні залежності. Першу архітектуру ми називають нейронною мережею прямого поширення. Спочатку сигнал подається на вхідний рівень. Потім, після того як він був оброблений, він передається на наступний шар. Багатошаровий персептрон і радіально базисні функції так само є хорошим прикладом нейронних мереж прямого розповсюдження. Сигнал нейронної сіті може поширюватися або тільки в одному напрямку, або мати рекурентні залежності. Першу архітектуру ми називаємо нейронною мережею прямого поширення. Спочатку сигнал подається на вхідний рівень. Потім, після того як він був оброблений, він передається на наступний шар. Багатошаровий персептрон і радіально базисні функції так само є хорошим прикладом нейронних мереж прямого поширення.

Коли нейронна мережа має свого роду внутрішні рекурентні зв'язки, це означає, що сигнал подається назад нейрону або шару, який вже отримував і

обробляв цей сигнал. Такі нейронні мережі називаються нейронними мережами зі зворотним зв'язком.

Особлива причина додавання зворотних залежностей в нейронну мережу - це отримання динамічного поведінки. На практиці, нейронна мережа, призначена для проблем, що включають тимчасові ряди або розпізнавання патернів, вимагають рекурентні залежності, для підкріплення навчання. Однак, таку мережу складно навчати. Більшість нейронних мереж зі зворотним зв'язком мають один шар, такі як, мережа Елмана і Хопфилда, але можливе створення багаторівневих рекурентних мереж, такі як ехо і рекурентний багаторівневий персептрон. На рисунку 1.7 зображено нейронну мережу зі зворотним зв’язком.

У нейронних мережах, всі зв'язки між нейронами регулюються нейронами. Як було сказано раніше, ваги представляють знання нейронної мережі. Різні ваги є причиною того, що нейронні мережі виробляють різний результат для одних і тих

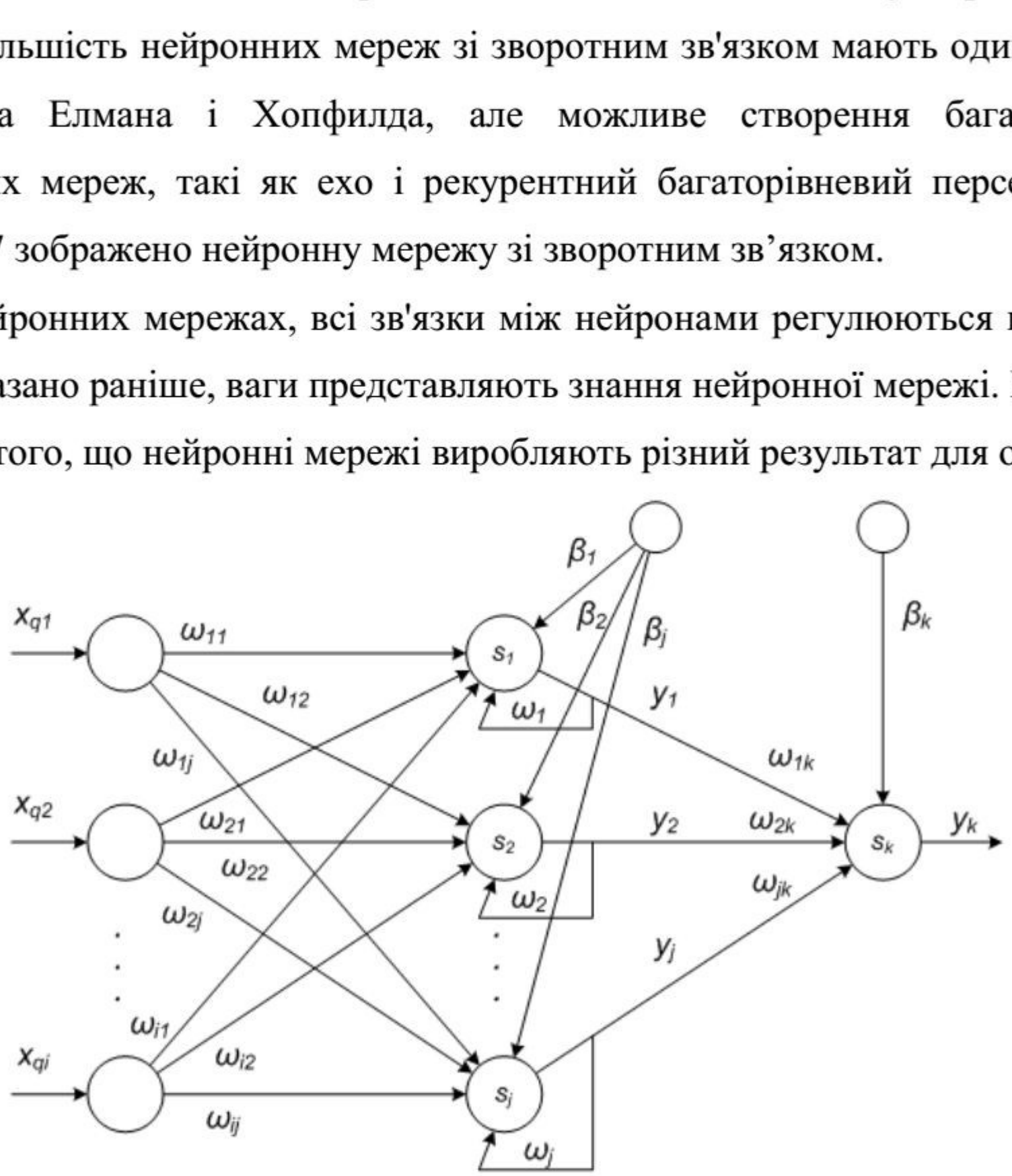


Рисунок 1.7 – Нейронна мережа зі зворотним зв’язком

же входів. Таким чином, нейронні мережі можуть покращувати свої результати регулюванням своїх ваг згідно з правилом навчання.

1.3.2 Аналіз застосувань нейронних мереж в криптографії

Нейрокриптографія - це область криптографії, призначена для аналізу застосування стохас-тичних алгоритмів, особливо нейромережевих алгоритмів, для використання в шифруванні і криптоаналізі.

В криптоаналізі використовується здатність нейронних мереж досліджувати простір рішень. Також є можливість створювати нові типи атак на існуючі алгоритми шифрування, засновані на тому, що будь-яка функції може бути представлена нейронною мережею. Зламавши алгоритм, можна знайти рішення, принаймні, теоретично. При цьому використовуються такі властивості нейронних мереж, як взаємне навчання, самонавчання, і стохастична поведінка, а також низька чутливість до шуму, неточностей (спотворення даних, вагових коефіцієнтів, помилки в програмі). Вони дозволяють вирішувати проблеми криптографії з відкритим ключем, розподілу ключів, хешування і генерації псевдовипадкових чисел, візуальної криптографії, стеганографії, шифруванні [5].

Сама модель штучних нейронних мереж добре підходить для побудови хеш-функцій на її основі. Штучний нейрон є базовим блоком при побудові ІНС та має n входів, n вагових характери-стик по одній на кожен вхід, зміщення, яке подається на вхід функції активації і одне вихідне значення. Таким чином, знаючи вхідний вектор обчислити вихідне значення легко, проте завдання отримання вхідного вектора нейрона за відомим значенню представляється важкою задачею. На підставі цих міркувань можна зробити висновок про те, що штучні нейронні мережі мають властивість односпрямованності. В роботі [6] показано побудову алгоритму хешування з використанням штучних нейронних мереж, яка має властивість однобічності, високої чутливості вихідного значення до вхідних даних і ключа користувача.

Також модель штучної нейронної мережі підходить для задач шифрування. Нейронні мережі використовуються для класифікації та апроксимації функцій, виділення завдань, які стійкі до деяких неточностей, для яких є багато доступних

даних для навчання, але до яких не можуть бути застосовані жорсткі правила. В статті [7] автор спробував реалізувати Rijndael-криптосистему за допомогою ІНС. Ця криптосистема має менш складну будову, ніж AES і не лінійна в експлуатації. Нелінійної повинна бути нейронна мережа зі зворотним зв'язком, що до-зволило б виконати шифрування / розшифрування відкритого тексту / зашифрованого тексту з високою продуктивністю і дуже низьким рівнем помилок. Ідея автора полягала в тому, щоб роз-робити таку нелінійну ІНС. Нелінійність необхідна для зменшення ймовірності злому алгоритму. Зменшення ймовірності злому досягається за допомогою нелінійної функції активації, також властивість нелінійної апроксимації мережі є корисним для практичного застосування.

Для обміну ключами між двома абонентами найбільш часто використовується алгоритм Діффі-Хеллмана. Його більш безпечна заміна заснована на синхронізації двох деревовидних машин парності (TPM, tree parity machines). Синхронізація цих машин схожа на синхронізацію двох хаотичних осциляторів в теорії хаотичних зв'язків (chaos communications).

Динаміка двох мереж і їх вагових коефіцієнтів знайшла застосування в явищі, де мережі синхронізують стану з ідентичними ваговими коефіцієнтами, залежними від часу. Ця концепція швидкої синхронізації за взаємною навчання може бути застосована до протоколу обміну секретним ключем через публічний канал. А згенерований ключ може бути використаний для шифрування і дешифрування переданого повідомлення. Алгоритм не оперує великими числами і методами з теорії чисел, отже, призводить до швидкої синхронізації відкритого ключа. Безпека нейрокриптографії в процесі обговорення, ведеться через те, що метод заснований на стохастичною процесі, є невеликий шанс, що зловмисник синхронізується з ключем.

Також, було встановлено, що захищеність звичайних криптографічних систем можна поліп-шити, збільшивши довжину ключа. У нейрокриптографії замість ключа збільшується синаптична довжина L. Це збільшує складність атаки

експоненціально, в той час як витрати абонентів на дешифрування ростуть поліноміально. Таким чином, злом подібної системи є NP-складною задачею [8].

Нейронні мережі використовуються як форма м'якої обчислювальної техніки для вирішення обчислювально складних задач, наприклад. розпізнавання мови та обличчя, прогнозування генів тощо. Ці мережі складають форму динамічних систем, що демонструють різноманітну складну поведінку, включаючи явище хаосу та синхронізації [9]. Хаотичні системи - це нелінійні динамічні системи, які чутливі до початкових умов.

У цих системах хвилинні відмінності в початкових умовах створюють сильно розходяться результати, що робить довгострокове прогнозування загалом дуже складним. Це трапляється, навіть якщо ці системи є детермінованими, тобто такий тип поведінки існує навіть тоді, коли їх майбутня динаміка повністю визначається їх початковими умовами, без випадкових елементів. Однією популярною моделлю хаотичних систем є хаотичні карти [10].

Хаотичні карти - це математичні функції еволюції, які демонструють хаотичну поведінку, і можуть бути параметризовані дискретним параметром часу або безперервного часу. Хаотичні системи іноді виявляють властивість синхронізації. Синхронізація - це координація подій для керування системою в унісон, або досягнення еквівалентних станів у системах під час взаємодії один з одним. Більшість систем стохастичні можуть бути лише приблизно синхронізованими. Фазова синхронізація, одна з широко застосовуваних концепцій синхронізації, - це процес, за допомогою якого два або більше циклічних сигналів мають тенденцію коливатися з повторюваною послідовністю відносних фазових кутів [11]. Однією з форм синхронізації, яка вже відома в системах шифрування, є перевірка того, що приймаючі шифри в потрібний час декодують правильні біти.

Таким чином, здавалося, що використання синхронізації та хаотичних характеристик динамічних систем може дати перспективний напрямок проектування нових та ефективних криптосистем.

У 2002 році Кантер та ін. опублікував серію пов'язаних робіт, де вони показали прекрасний зв'язок між нейронними мережами та криптографією [12, 13, 14]. Вони продемонстрували, що синхронізація нейронних мереж може призвести до способу обміну секретними повідомленнями або ключами . Наприклад, використання правила Хеба, щодо їх взаємних результатів, дозволяє мережам розвивати еквівалентні стани своїх внутрішніх синаптичних ваг, тобто мережі синхронізуються до стану з однаковими залежними від часу вагами. Ці синхронізовані ваги потім використовуються для побудови протоколу обміну ключами. Вони виявили, що не можна розшифрувати секретне повідомлення навіть для опонента, який знав протокол та всі деталі будь-якої передачі даних. Вони також довели, що це було насамперед тому, що відстеження ваг нейронних мереж під час синхронізації було важкою проблемою. Але, з іншого боку, складність генерації захищеного каналу лінійна за розмірами мережі. Ці результати відкрили нові шляхи в сучасній криптографії та показали, як синхронізація шляхом взаємного навчання в нейронних мережах може бути застосована до обміну секретними ключами через публічні канали. Цей та подібні результати пізніше породили сферу нейронної криптографії. Після цього дослідники спробували кілька різних методів криптографії з використанням нейронних мереж у різних формах.

1.4 Формалізація вимог та постановка задачі

Криптографія з відкритим ключем заснована на проблемі великих обчислень, або в якій невідомий ефективний алгоритм вирішення. Коли ефективний алгоритм вирішення не відомий, розвиток обчислювальної потужності та розвиток технологій зростає, наприклад, хмарні обчислення або використання великих ботнетів, дозволяють проводити жорстокі атаки. Як відповідь на це ми спостерігаємо постійне збільшення довжини ключів, що використовуються в криптографічному алгоритмі, до безпечного розміру, але це також збільшує час,

необхідний для шифрування та дешифрування повідомлень. Альтернативний підхід - це пошук нових методів криптографії, який не покладається на обчислювально важкі проблеми. Цікавими є спроби застосування методів штучного інтелекту в криптографії. Використання штучних нейронних мереж для обміну криптографічними ключами для подальшого спілкування партнера за допомогою відкритого каналу зв'язку є перспективним напрямком.

Дослідження вивчення штучних нейронних мереж у варіанті з учителем призвело до виявлення цікавих явищ, якими є синхронізація цих мереж. Мережевий процес навчання вимагає використання навчального набору, який складається з пар вхідних імпульсів - вектора x та очікуваної реакції на мережу, яка може бути одиничним числом або вектором y. Мережеве навчання полягає у виборі ваг, для яких відповідь мережі буде такою ж, як і очікувалася викладачем, іншими словами процес навчання складається з мінімізації помилок мережі. Особливо цікавою є можливість модифікувати навчальний набір під час навчання мережі. Зокрема, навчальний набір може генеруватися іншою нейронною мережею. У цьому випадку навчена мережа імітує вчителів, стаючи наприкінці навчання копією мережі вчителя. Однак, якщо мережі обміняються своїми ролями і вони будуть один для одного викладачем та учнем, ми спостерігатимемо синхронізацію між ними, після чого обидві мережі матимуть однакові вектори ваги. Синхронізуючі мережі впливають один на одного, змінюючи навчальний набір, завдяки чому процес синхронізації проходить швидше, ніж пасивне навчання мережі.

Ця різниця у часі, необхідна для досягнення сумісного вектору ваги третьою стороною, є основою для використання мережевої синхронізації в криптографії. У застосуванні використовуються невеликі нейронні мережі з одним прихованим шаром, який називається ДМП (деревовидна машина парності). Тому пропонується метод передачі секретного ключа за допомогою нейронних мереж з використанням помилкових бітів при передачі по мережі.

2 МЕТОД ГЕНЕРАЦІЇ ТА ПЕРЕДАЧІ СИМЕТРИЧНОГО КЛЮЧА ШИФРУВАННЯ

2.1 Метод передачі симетричного ключа шифрування за допомогою нейронних мереж

Механізм роботи метода можна представити у вигляді криптосистеми:

- процес ініціалізації– подання на кожну сторону згенерованих початкових даних за допомогою генератора псевдовипадкових чисел;
- процес синхронізації – ініціалізація початкових параметрів нейронних мереж і створення ключа шифрування;
- процес зашифрування тексту – використання ключа для зашифрування інформації.

На рисунку 2.1, схему криптосистеми з використанням нейронних мереж.

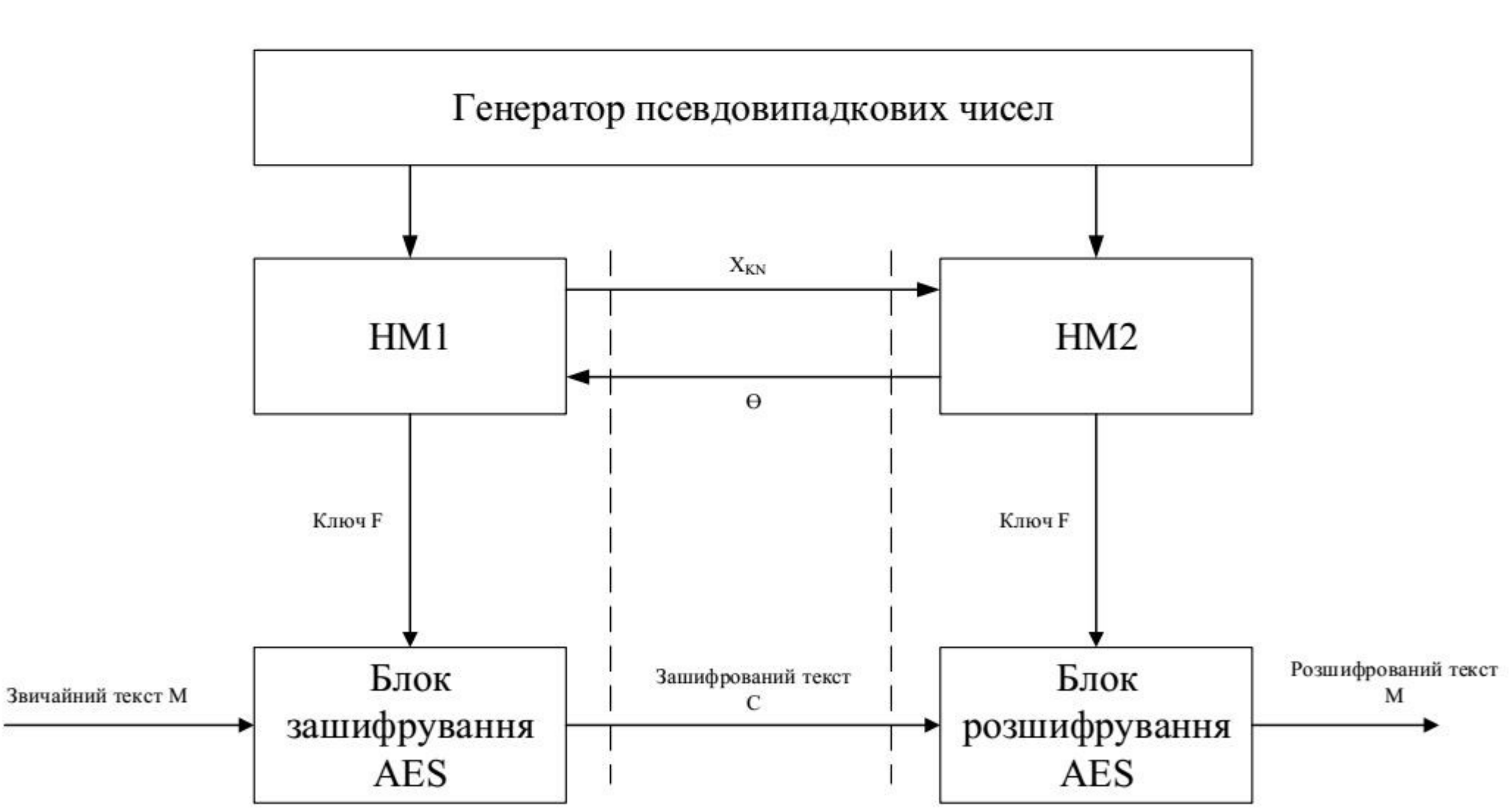


Рисунок 2.1 – Схема криптосистеми з використанням нейронних мереж

Криптосистема представляє проблему обміну ключами між двома нейронними мережами з використанням концепції взаємного навчання. Дві нейронні мережі, які навчаються на їх взаємному виході, синхронізуються з однаковим залежним від часу ваговим вектором. Дві мережі обмінюються своїми результатами (в бітах), і ключ між двома комунікаційними сторонами в кінцевому підсумку представляється в остаточному вивченому зважуванні, коли, як кажуть, дві мережі синхронізовані.

Така система може бути визначена через сукупність $S = (M, F, C, E, D)$ введених множин, де M, F, C - кінцеві множини можливих відкритих текстів, ключів і зашифрованих текстів. Через E позначено множину $\{E_f : f \in F\}$, що складається з правил шифрування $E_f : M \to C$ на ключі $f \in F$. Тоді $D_f : E_f(M) \to M$ - це правило дешифрування на ключі $f \in F$, і D - множина $\{D_f : f \in F\}$. Ключ F можна визначити як $F = \{W_{ij}\}$, де W – це ваги нейронної мережі, які складаються з $W_{ij} = \{K, N, L, X, A_{learn}\}$.

2.2 Архітектура нейронних мереж

Нейронна мережа представлена архітектурою деревовидної машини парності (ДМП), що представляє собою особливий вид багаторівневої нейронної мережі прямого поширення, яку зображено на рис. 2.2.

Вже згадана архітектура складається з одного вихідного нейрона, K прихованих нейронів та N вхідних нейронів. Входи в мережу двійкові: $X_{KN} = \{-1, 1\}$ Ваги між вхідними та прихованими нейронами приймають значення: $W_{KN} = \{-L,. . ., 0,. . ., L\}$ Значення виходу кожного прихованого нейрона обчислюється як сума всіх множень вхідних нейронів і цих ваг:

$$\sigma = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} x_{ij}\right) = \{-1, 1\} \qquad (2.1)$$

sgn - це проста порогова функція, яка повертає -1 або 1:

$$\text{sgn}(\ x) = \begin{cases} -1, x \le 0, \\ 1, x > 0 \end{cases} \tag{2.2}$$

Якщо скалярний добуток дорівнює 0, вихід прихованого нейрона відображається на -1, щоб забезпечити бінарне вихідне значення. Вихід нейронної мережі потім обчислюється як множення всіх значень, вироблених прихованими елементами:

$$\tau = \prod_{i=1}^{K} \sigma_i = \{1, -1\} \tag{2.3}$$

Вихід машини парного дерева є двійковим. Іншою широко використовуваною моделлю таких кінцевих точкових мереж є двійкова зміна .

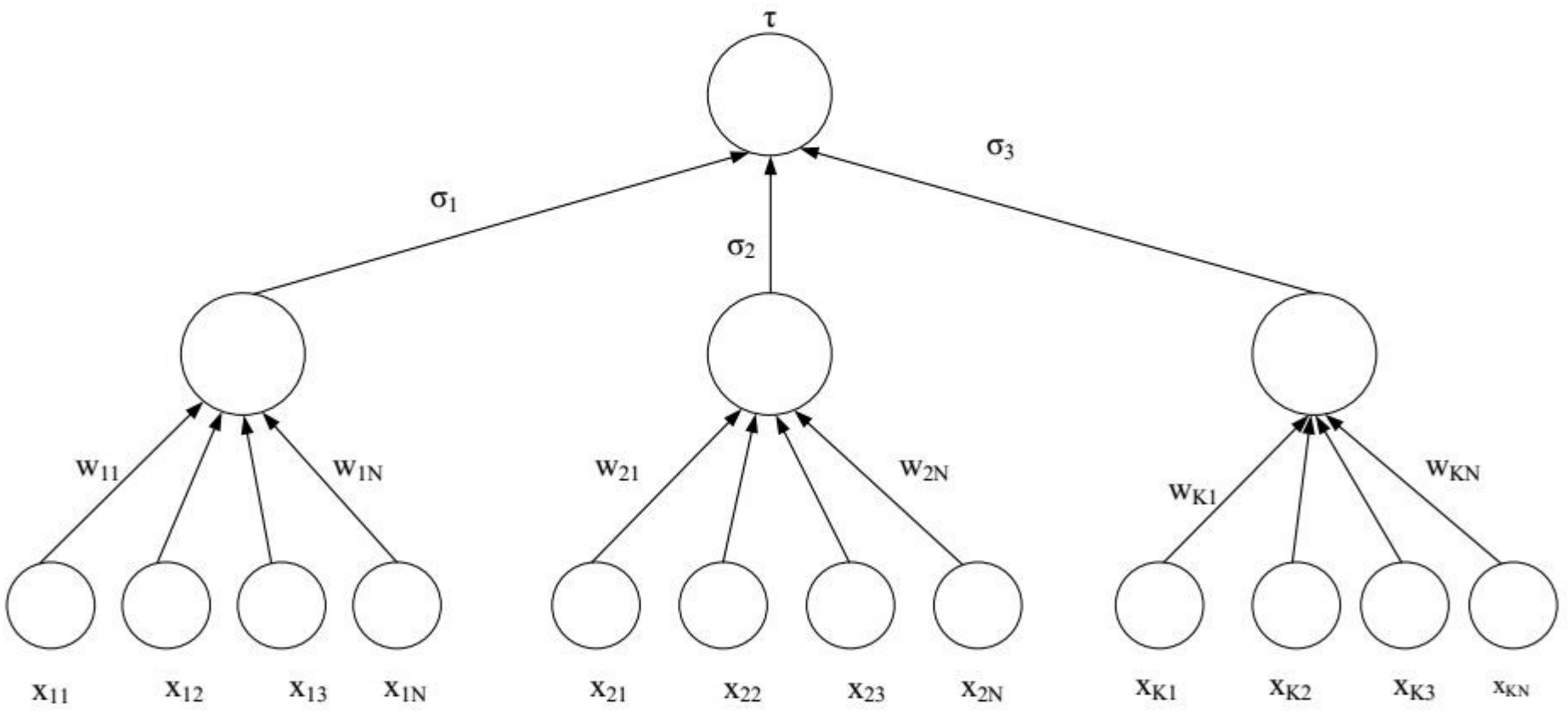


Рисунок 2.2 – Схема нейронної мережі

На рис. 2.2 X_{ij} - значення вхідних нейронів , одержувані з вектора вхідних значень X_{KN}, де i ∈ [1, K], а j ∈ [1, N], W_{ij} - значення вагових коефіцієнтів, що задаються випадковим чином при першій ініціалізації нейронної мережі.

Алгоритм обміну інформацією на основі ДМП. Розглядаємо дві мережі (A і B), кожна з яких побудована на основі архітектури ДМП (як на рис. 2.2). Мережі обмінюються ви вихідними величинами по відкритих каналах.

Синхронізація мереж відбувається відповідно до наступним алгоритмом:

1) Задаємо випадкові значення вагових коефіцієнтів w_{11}, w_{12}, ..., w_{KN}.
2) Виконуємо наступні кроки, поки не настане синхронізація.
3) Генеруємо випадковий вхідний вектор X (x_{11}, ..., x_{1N}, ..., x_{KN}).
4) Обчислюємо значення прихованих нейронів по формулі.
5) Розраховуємо значення вихідного нейрона за формулою .
6) Порівнюємо виходи двох ДМП.
7) Виходи різні: перехід до п. 2.
8) Виходи однакові: застосовуємо вбрання правило до вагових коефіцієнтів.

Після повної синхронізації (ваги W_{ij} обох ДМП однакові) абоненти (мережі) A і B можуть використовувати ваги в якості ключа. На рисунку 2.3 зображено загальну схему нейронних мереж.

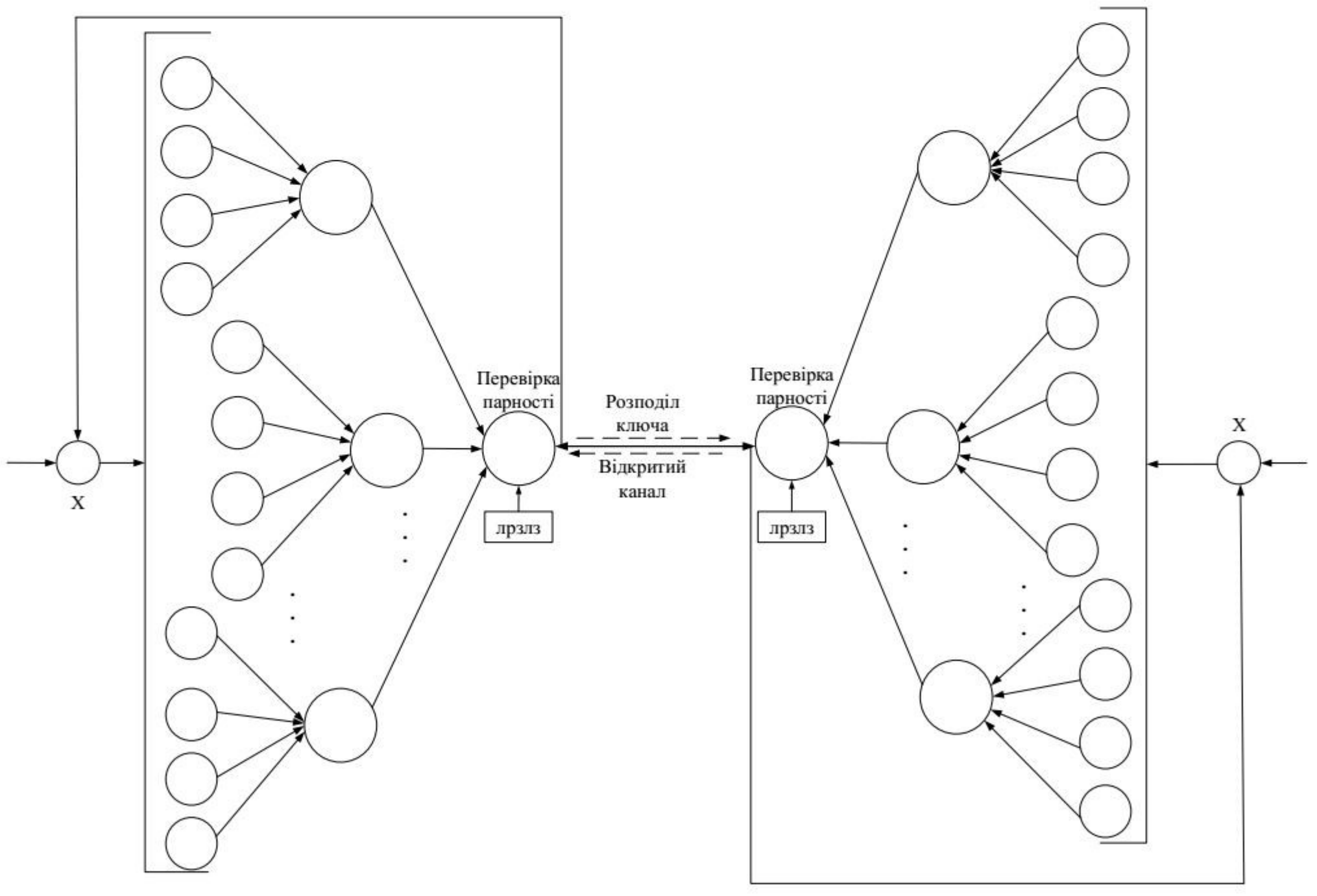


Рисунок 2.3 – Загальна схема двох нейромереж

Для поновлення вагових коефіцієнтів в ДМП можуть використовуватися такі правила:

правило Хебба:

$$w_{ij} = g(w_{ij} + x_{ij} \cdot \tau_{current} \cdot F(\sigma_i, \tau_{current}) \cdot F(\sigma_i, \tau_{other})), \quad (2.4)$$

де $\tau_{current}$ - вихідне значення ДМЧ, у якій змінюються ваги;

τ_{other} -Вихідний значення ДМЧ, з якої відбувається синхронізація;

x_{ij} - це значення -го вхідного сигналу -го прихованого нейрона;

$$F(x, y) = \begin{cases} +1, & при \quad x = y \\ -1, & при \quad x \neq y \end{cases}; \quad (2.5)$$

$$g(x) = \begin{cases} \text{sgn}(x) \cdot L, & при \quad |x| > L \\ x, & при \quad |x| \leq L \end{cases}; \quad (2.6)$$

- анти правило Хебба

$$w_{ij} = g(w_{ij} - x_{ij} \cdot \tau_{current} \cdot F(\sigma_i, \tau_{current}) \cdot F(\sigma_i, \tau_{other})), \quad (2.7)$$

- випадкового блукання

$$w_{ij} = g(w_{ij} + x_{ij} \cdot F(\sigma_i, \tau_{current}) \cdot F(\sigma_i, \tau_{other})) \quad (2.8)$$

При розгляді процесу навчання двох ДМП виникає питання про те, коли слід припиняти його виконання. Іншими словами, потрібно знайти критерій синхронізації, задовольнивши який, можна буде зупинити алгоритм навчання. Можливі такі підходи:

Повний перебір - обом ДМП подаються на входи всіх можливих вхідні вектори, а їх обчислені вихідні значення порівнюються. При досягненні синхронізації при всіх вхідних векторах всі відповідні виходи повинні збігатися. Такий підхід є найбільш неефективним з огляду на занадто великої кількості епох двонаправленого навчання. У разі збільшення числа порівнянь результатів роботи ДМП двох абонентів криптоаналітик з високою ймовірністю встигне навчити власну ДМП потрібної структури, і тим самим отримати генерується секретний ключ шифрування.

Ітеративний підхід - полягає в емпіричному оцінюванні необхідного числа ітерацій. Також є не найкращим варіантом. Залежно від початкових станів ДМП абонентів необхідну кількість ітерацій може сильно варіюватися. Результатом може стати як недостатня кількість епох навчання ДМП, так і перевищення необхідної їх кількості, що підвищить шанси криптоаналитика на отримання секретного ключа.

Явна порівняння (застосування дайджестів) - можливо при використанні хеш-функції для хешування поточних станів ДМП двох абонентів і їх порівняння з певною періодичністю (як правило, після кожного оновлення вагових коефіцієнтів). Це найпростіший і ефективний спосіб перевірки синхронізації станів двох ДМП. Єдиним нюансом є вибір надійної і швидкої хеш-функції з мінімальним шансом колізій;

Підхід з використанням дайджестів можна ускладнити для того, щоб мінімізувати число повідомлень, що пересилаються. Для цього кожним з абонентів і будуть використані по дві допоміжні ДМП. Так, наприклад, абонент буде мати не тільки машину парності, яка буде безпосередньо брати участь у взаємодії з абонентом при генерації загального ключа, а й її копію, початкові вагові коефіцієнти якої збігаються з початковими вагами оригіналу, а також ДМП, початкові значення вагових коефіцієнтів якої ініціалізовані випадково. Абонент буде спостерігати за своїми допоміжними машинами парності і визначить, коли настане їхня синхронізація. Далі, в процесі генерації загального секретного ключа, абонент буде пересилати дайджест набору ваг абоненту після числа ітерацій, яке знадобилося для досягнення синхронізації допоміжних ДМП абонента. Аналогічно, абонент може спостерігати за процесом синхронізації своїх допоміжних ДМП і використовувати отримане при цьому число ітерацій при взаємодії з абонентом. Учасники протоколу також можуть узгодити перед початком взаємодії число ітерацій, після яких почнеться обмін дайджестами [17].

Після синхронізації деревовидних машин парності двома абонентами мережі, у кожного з них буде послідовність з чисел в діапазоні від до. Ці числа можна буде,

за домовленістю, перевести в двійковий вигляд, представити їх як коди символів Юнікоду або інший символьної кодування, і взагалі привести їх до будь-якого зручного виду.

Модифікацією алгоритму передачі ключа полягає у використанні регістру зсуву з лінійним зворотнім зв'язком, вхідні параметри якого, вже розподілені між сторонами. Якщо регістр зсуву з лінійним зворотнім зв'язком видає значення 0, це означає, що вихідне значення τ самої нейронної мережі на сторона A можна передати через відкритий канал іншій стороні B. Якщо значення регістру дорівнює 1, то сторона A і B відправляють свої зворотні значення τ, які не обробляються.

2.3 Можливі атаки на перехоплення ключа

У кожному нападі вважається, що зловмисник E може повідомлення про підслуховування між сторонами A і B, але це робить не мати можливості їх змінити:

а) Метод грубої сили.

Для забезпечення жорстокої атаки зловмисник повинен протестувати всіх можливі клавіші (усі можливі значення ваг w_{kj}). За K прихованих нейронів, K * N вхідних нейронів і межі ваги L, це дає (2L + 1) можливості KH. Наприклад, конфігурація K = 3, L = 3 і N = 100 дає нам 3 * 10253 ключові можливості, що робить атаку неможливою сьогоднішньою живлення комп'ютера.

б) Навчання за допомогою власної деревовидної машини парності. Одну з основних атак може надати зловмисник, кому належить та сама машина парності дерев, що і сторони A та B. Він хоче синхронізувати свою машину з іншими двома. На кожному кроці є три ситуації можливо:

• Вихід (A) ≠ Вихід (B): жодна із сторін не оновлюється його ваги [18].

• Вихід (A) = Вихід (B) = Вихід (E): Усі три сторони оновлюють ваги в своїх машинах. Вихід (A) = Вихід (B) ≠ Вихід (E): Сторони A і B оновлюють свої деревовидної машини парності, але зловмисник не може цього зробити. Через цю ситуацію його навчання відбувається повільніше, ніж синхронізація сторін A і B.

Доведено, що синхронізація двох сторін швидше, ніж навчання нападника. Це можна покращити за допомогою збільшення синаптичної глибини L нейронної мережі. Це дає цьому протоколу достатню безпеку і зловмисник може з'ясувати ключ лише з невеликою ймовірністю. Змінивши цей параметр збільшує вартість успішної атаки експоненціально, тоді як зусилля для користувачів зростають поліноміально. Тому, порушуючи безпеку нейронного ключа обмін належить до класу складності NP. Існують і інші більш складні атаки проти цього протокол (наприклад, геометрична, більшість, генетична атака). Найбільший успішною є атака більшості. Поки що ніхто з відомих атаки можуть порушити безпеку обміну нейронними ключами протокол із запитами.

2.4 Аналіз характеристик мережі

ДМП - це нейронна мережа, яка формується прихованим шаром і єдиним виходом. Загальна структура складається з тріади значень K, N і L, де кількість нейронів у прихованому шарі, N це кількість записів кожного нейрона в прихованому шарі, і встановлює межа діапазону можливих цілих значень, пов'язаних з синаптичними вагами.

Головним параметром конфігурації деревовидної машини парності є кількість зв'язків між нейронами її вхідного і прихованого шарів. Число таких зв'язків обчислюється як K*N, Тобто добуток кількості прихованих нейронів і кількості вхідних нейронів, пов'язаних з кожним прихованим нейроном.

Для генерації та встановлення ключа для 512 бітів було перевірено деякі варіанти структури ДМП, синаптичні ваги яких дозволяють перевірити цю довжину ключа. Для її визначення, значення L було взято за основу, що у своєму бінарному позначенні встановлює остаточну довжину ключа. Для встановлення парних значень по довжині ключа границі діапазону значень від [-L, L] до [-L,L - 1], максимальне значення яких у десятковій нотації 2L-1 змінено, а довжина його

двійкового значення кратна 512. Завдяки цьому можливі значення, що дозволяють генерувати ключі бітів (див. таблицю 1).

Таблиця 1. Можливі значення L

Десяткове значення L	Розмір L в бітах
2^0	1
2^1	2
2^3	4
2^7	8
2^{15}	16
2^{31}	32
2^{63}	64
2^{127}	128
2^{255}	256
2^{511}	512

Для кожного значення L є значення K та N такі, що $len(L_{BIN})*K*N = 512$, враховуючи обмеження K<=N, як показано в таблиці 2.

Таблиця 2. Можливі комбінації з L

L	K	N	L	K	N
2^0	1	512	2^{15}	1	32
	2	256		2	16
	4	129		4	8
	8	64	2^{31}	1	16
	16	32		2	8
2^1	1	256		4	4
	2	128	2^{63}	1	8
	4	64		2	4
	8	32	2^{127}	1	4
	16	16		2	2
2^3	1	128	2^{255}	1	2
	2	64	2^{511}	1	1

	4	32			
	8	16			
2^7	1	64			
	2	32			
	4	16			
	8	8			

Обмеження K <= N пов'язано з тим, що якщо K > N, то нейронна мережа зможе досягти значень, які роблять синхронізацію неможливою. Потім можливі 30 комбінацій значень K, N та L, які дозволяють генерувати ключ довжиною 512 біт.

Найкращою конфігурацією ДМП є та, в якій число прихованих нейронів пропорційно числу входів у кожного з них. Така структура дає оптимальне співвідношення надійності і швидкості синхронізації машин парності. При цьому незначні зрушення в ту чи іншу сторону допустимі і не спричинять за собою серйозних наслідків. Також збільшення входів у кожного прихованого нейрона деревовидної машини парності призводить до збільшення здатності цієї ДМП до навчання, тобто до більш швидкому оновленню вагових коефіцієнтів зв'язків з цими входами. Збільшення ж кількості самих прихованих нейронів ускладнює процес виявлення залежностей між вектором вхідних сигналів і значенням, отриманим на виході ДМП і, отже, підвищує її надійність.

Наприклад у нас є 2 нейронні мережі з такими характеристиками K = 3, N = 9, L = 5.

На вхід сторони A і B подається такий вектор X:

[-2 4 1 -5 1 -1 2 4 -3]

[-5 3 5 -5 1 2 3 -5 5]

[0 -3 0 5 -5 -4 5 4 -3]

і генерується такий вектор W_A і W_B:

w_{1Aj} = [0 -5 -2 -5 -3 -4 -3 0 -4]

w_{2Aj} = [5 -1 -1 -4 2 -4 -5 4 4]

$w_{3A\,j}$ = [2 5 -3 -1 4 1 -3 1 -2]

$w_{1B\,j}$ = [-2 4 -5 -3 1 0 -2 -4 4]

$w_{2B\,j}$ = [1 3 -3 -1 5 -3 -4 5 -1]

$w_{3B\,j}$ = [-5 1 4 -1 -4 2 -1 2 -5]

Потім розраховується на 2 сторонах σ за формулою 2,1 і 2,2, де ми отримуємо значення на кожній стороні:

σ_A = [1 -1 -1]

σ_B = [-1 -1 1]

За формулою 2,3 ми отримуємо на сторонах значення $\tau_A = \tau_B$ і відбувається оновлення ваг, де значення приймаються за 1.

Оновлення ваг відбувається за правилом Хебба і на тих шарах, де значення дорівнюють 1:

$$w_{ij} = g(w_{ij} + x_{ij} \cdot \tau_{current} \cdot F(\sigma_i, \tau_{current}) \cdot F(\sigma_i, \tau_{other})) = -2 \cdot 1 \cdot 1 \cdot 1 = -2$$

Таким чином змінюється елементи матриці першого прихованого шару на стороні А, а на стороні В третього. Після розрахунків ми отримаємо такі оновлені ваги на сторонах:

w_{1A} [-2 -1 -1 -5 -2 -5 -1 4 -5]

w_{2A} [5 -1 -1 -4 2 -4 -5 4 4]

w_{3A} [2 5 -3 -1 4 1 -3 1 -2]

w_{1B} [-2 4 -5 -3 1 0 -2 -4 4]

w_{2B} [1 3 -3 -1 5 -3 -4 5 -1]

w_{3B} [-5 -2 4 4 -5 -2 4 5 -5]

Оновлення ваг продовжується до ln(2L+1)*K*N разів. Після проходження 65 оновлень ми отримуємо однакові ваги на стороні А і В:

[-1 -5 5 -2 5 5 -5 2 5]

[-5 5 4 -4 -1 3 -1 1 -5]

[5 -5 -2 -1 -3 4 -5 -5 -5]

Отримані значення перетворюються на хеш-значення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ

3.1 Обґрунтування вибору засобів програмної реалізації методу

Від вибору мови залежить подальша розробка програмного способи. Було проведено вибір між 4-х мов програмування MATLAB, Java, Python і C ++.

MATLAB - це високорівнева мова і інтерактивне середовище для програмування, чисельних розрахунків і візуалізації розрахунків. Мова, інструментарій та інтегровані математичні функції дають можливість вивчати всілякі розклади і отримувати висновок швидше, ніж у електронних таблицях або ж класичних мов програмування, таких як C/C++ або ж Java. Для роботи з нейронними мережами в MATLAB створений додаток Neural Network Toolbox. Neural Network Toolbox - це додаток розширення MATLAB, що має широкі можливості у роботі з нейронними мережами [23].

C++ вважається швидкою та потужною високорівневою мовою програмування. Її можливість швидко працювати на апаратному рівні дозволяє користувачам зменшити час виконання програми. C ++ вважається дуже корисною для проектів з розробки штучного інтелекту, де важлива швидкодія. Для розробки нейронних мереж є кілька бібліотек і навчальних статей.

Java це багатопарадигмальна мова програмування, яка підтримує об'єктно-орієнтоване програмування і може бути виконана на значній кількості обладнання без необхідності змінювати код під іншу платформу. Це мова програмування штучного інтелекту, яка може працювати на будь-якій платформі, яка підтримує її без перекомпіляції.

Python - інтерпретована мова програмування. З однієї сторони, це дозволяє сильно спростити відлагодження програм, з іншої - призводить до відносно невисоку швидкість виконання. Python має сильну підтримку модульності, завдяки якій можливо застосувати різноманітні модулі, написані для інших програм у власних програмах. У проекті розробки штучних нейронних мереж Python

вважається зручним, тому що для цієї мови є велика кількість навчальних прикладів з розробки штучних нейронних мереж, а ще бібліотек і модулів.

Одним з головних недоліків є його відносно низька швидкість виконання. Python є мовою з повною динамічною типізацією, автоматичним управлінням пам'яттю. Якщо на перший погляд це може здаватися перевагою, то при розробці програм з підвищеним вимогою до ефективності, Python може значно програвати за швидкістю схожим мовам (C / C ++, Java, Go). Що стосується динамічних побратимів (PHP, Ruby, JavaScript), то тут справи йдуть набагато краще, Python в більшості випадків виконує код швидше за рахунок попередньої компіляції в байт-код і значної частини стандартної бібліотеки, написаної на Сі.

Програмний додаток створений на мові програмування Python. Незважаючи на ряд проблем історично притаманних Python, він продовжує залишатися провідним інструментом:

– розробка веб-додатків.

– аналіз даних і машинне навчання (пакети scipy, scikit-learn, pandas, numpy визнані світовим ученим співтовариством).

– швидке прототипування ідей в бізнесі за рахунок великої кількості готових бібліотек, низький поріг входження в мову і високої продуктивності програмістів, які пишуть на Python.

– написання скриптів (сценаріїв) для автоматизації завдань. Python за замовчуванням поставляється з усіма збірками unix-like систем і є відмінною заміною Bash у всіх сенсах.

3.2 Розробка програмного засобу

Програмний додаток складається з 2 основних модулів:

– модуль обміну ключами - призначений для отримання на обох кінцях публічного каналу закритого ключа таким чином, щоб через публічний канал його

неможливо було перехопити. Даний модуль призначений для отримання сесійних ключів.

– криптографічний модуль, який здійснює, після процедури обміну ключами, шифрування і розшифровування за допомогою ключа.

Процес обміну ключами побудований на синхронізації двох деревовидних машин парності. При реалізації алгоритму синхронізації використано класичний алгоритм з модифікацією подачі помилкових бітів.

Деревоподібна машина парності є основним компонентом системи обміну ключами. Являє собою програмну реалізацію багаторівневої нейронної мережі і реалізованої для неї алгоритму навчання. На рисунку 3.2 зображено алгоритм роботи модулю обміну ключами.

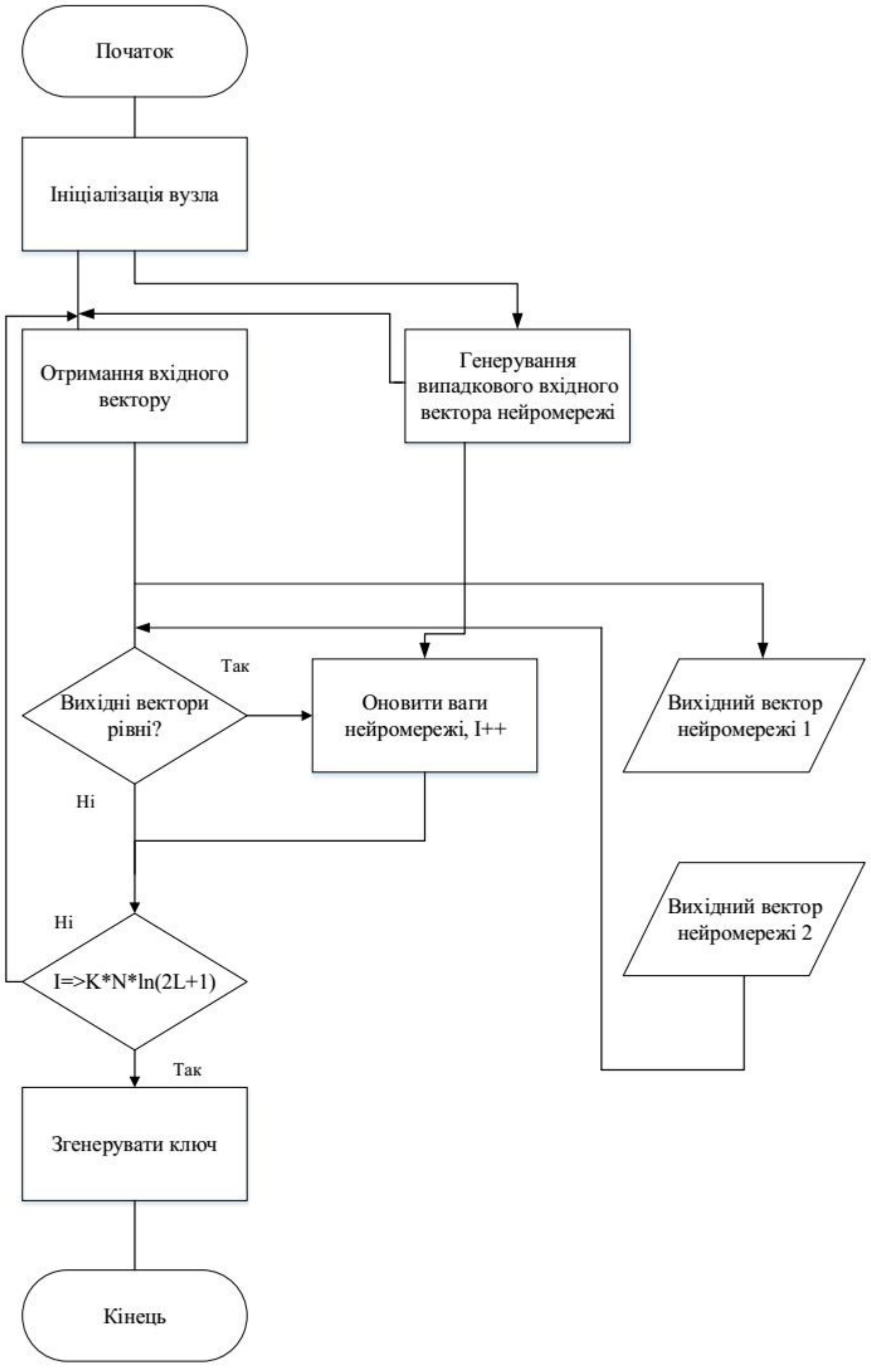


Рисунок 3.1 – Алгоритм роботи модулю обміну ключами

Процес навання реалізовано класом Update_rules. Даний клас містить у собі список нейронів, які не започатковано автоматично на підставі параметрів N, K, L, функція активації, які передаються користувачем в машину при ініціалізації. На рисунку 3.3 зображено частину коду з правилами: Хебіана, Анти-Хебіана та правило блукаючого кроку.

```
def hebbian(W, X, sigma, tau1, tau2, l):
    k, n = W.shape
    for (i, j), _ in np.ndenumerate(W):
        W[i, j] += X[i, j] * tau1 * theta(sigma[i], tau1) * theta(tau1, tau2)
        W[i, j] = np.clip(W[i, j] , -l, l)

def anti_hebbian(W, X, sigma, tau1, tau2, l):
    k, n = W.shape
    for (i, j), _ in np.ndenumerate(W):
        W[i, j] -= X[i, j] * tau1 * theta(sigma[i], tau1) * theta(tau1, tau2)
        W[i, j] = np.clip(W[i, j], -l, l)

def random_walk(W, X, sigma, tau1, tau2, l):
    k, n = W.shape
    for (i, j), _ in np.ndenumerate(W):
        W[i, j] += X[i, j] * theta(sigma[i], tau1) * theta(tau1, tau2)
        W[i, j] = np.clip(W[i, j] , -l, l)
```

Рисунок 3.3 – Правила навчання нейромереж

Другий основною частиною модуля безпеки є криптографічний модуль. Даний модуль здійснює шифрування вихідний і дешифрування вхідної інформації. Для цих операцій використовується ключ, який був згенерований шляхом виконання алгоритму обміну ключами в модулі обміну ключами. На рисунку 3.4 зображено алгоритм роботи криптографічного модулю.

Рисунок 3.4 – Алгоритм роботи криптографічного модулю

Шифрування здійснюється симетричним блоковим шифром. Шифр використовуваний в даному модулі - AES

3.3 Реалізація програмного засобу

На базі описаного вище протоколу було розроблено додаток для обміну мнгновенье повідомленнями. На рисунку 3.5 зображено інтерфейс програми.

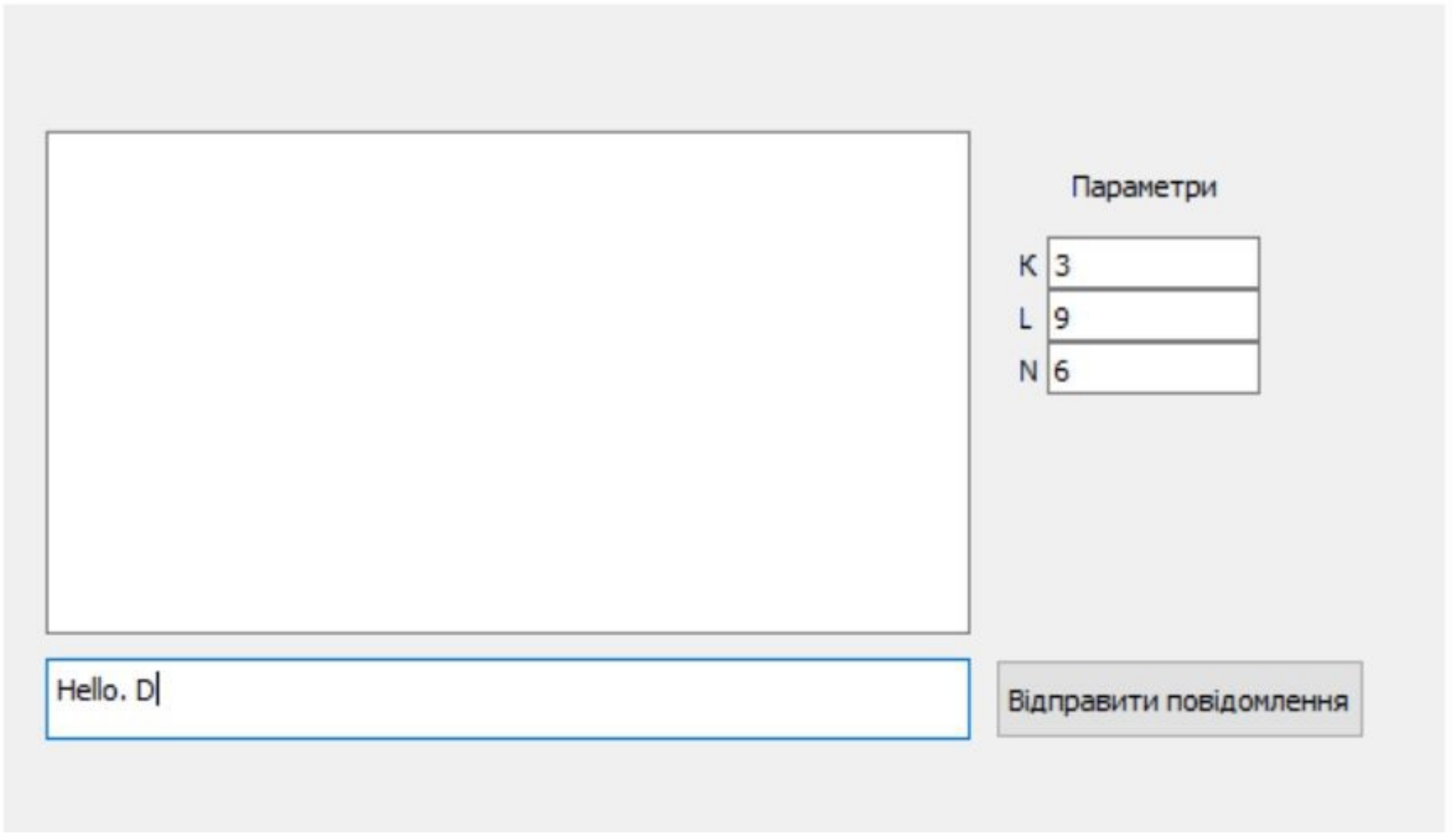


Рисунок 3.5 – Інтерфейс програмного модулю

Цей додаток має вікно для введення тексту, параметри підключення, параметри деревовидної машини парності, які користувач може задати при підключенні.

3.4 Тестування програмного засобу

Було проведено експерименти з вимірювання часу синхронізації мереж. Вимірювання часу синхронізації при зміні L. .В даному експерименті будуть проведені вимірювання швидкості синхронізації двох деревовидних машин парності при зміні параметра L (діапазон значень ваг). Для даного експерименту ми маємо дві деревовидних машини парності. Параметри N і K для даного експерименту будуть зафіксовані і набувати значень K = 3 і N = 1000. На рисунку 3.6 можна побачити залежність кількості ітерацій, необхідних для синхронізації від параметра L.

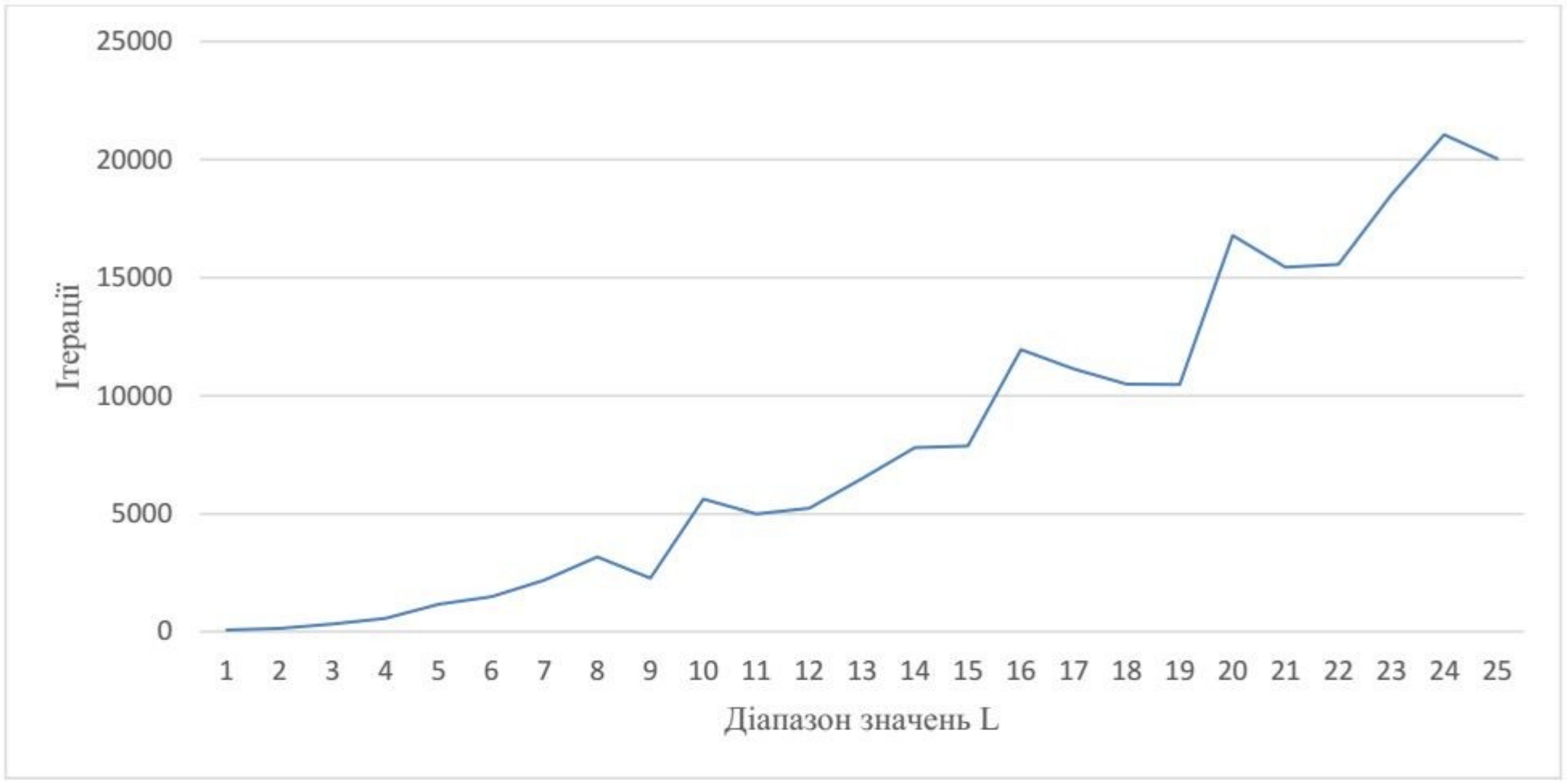


Рисунок 3.6 – Графік зміни кількості ітерацій синхронізації в залежності від L

З графіка можна побачити що, час синхронізації зростає нелінійно. При збільшенні L, швидкість зростання часу синхронізації постійно збільшується. Так як, дані параметри будуть використовуватися для протоколу обміну ключами, то нам необхідна швидкодія.

Таким чином можна зробити висновок, що при збільшенні параметра L в більшу сторону, час синхронізації істотно зростає. Так як для протоколу важливо швидкодію, то не бажано, щоб час процесу синхронізації перевищує 1000 мс. Тому при виконанні процесу синхронізації не варто брати значення L, що перевищують 10.

Вимірювання часу синхронізації при зміні K. В даному експерименті будуть проведені вимірювання швидкості синхронізації двох деревовидних машин парності при зміні параметра K (кількість нейронів). Для даного експерименту ми маємо дві деревовидних машини парності. Параметри N і L для даного експерименту будуть зафіксовані і набувати значень L = 3 і N = 1000. На рисунку 3.7 можна побачити залежність залежність кількості ітерацій, необхідних для синхронізації від параметра K.

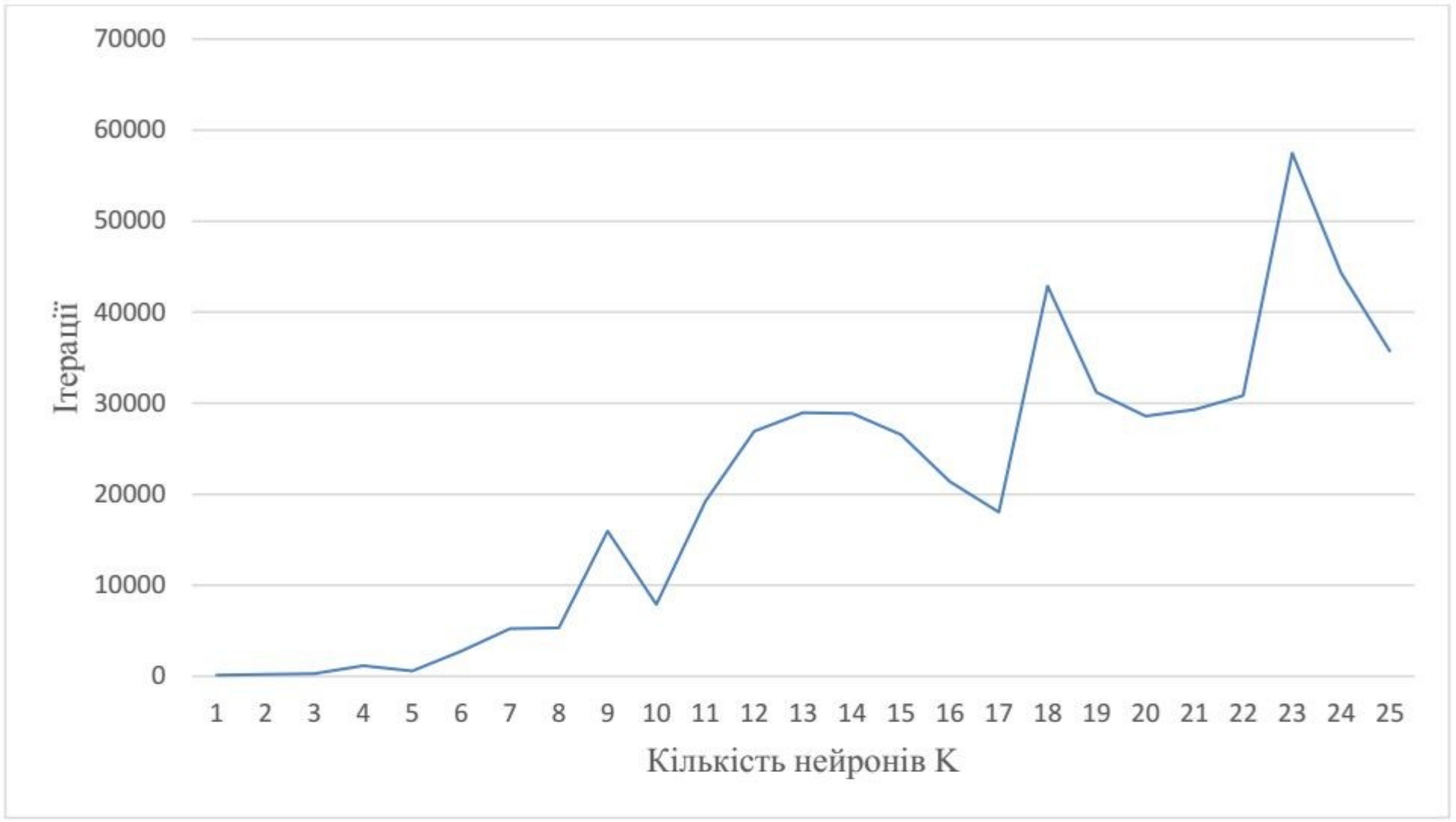


Рисунок 3.7 – Графік зміни кількості ітерацій синхронізації в залежності від K

На даному графіку можна помітити, що при зміні кількість нейронів в деревовидної машині парності, швидкість синхронізації збільшується швидше, ніж при зміні діапазону значення, таким чином можна сказати, що кількість нейронів в деревовидної машині парності впливає на загальний час синхронізації сильніше ніж діапазон значень.

Вимірювання часу синхронізації при зміні N. В даному експерименті будуть проведені вимірювання швидкості синхронізації двох деревовидних машин парності при зміні параметра N (кількість входів кожного нейрона). Для даного експерименту ми маємо дві деревовидних машини парності. Параметри K і L для даного експерименту будуть зафіксовані і набувати значень L = 3 і K = 3. На рисунку 3.8 можна побачити залежність кількості ітерацій, необхідних для синхронізації від параметра N.

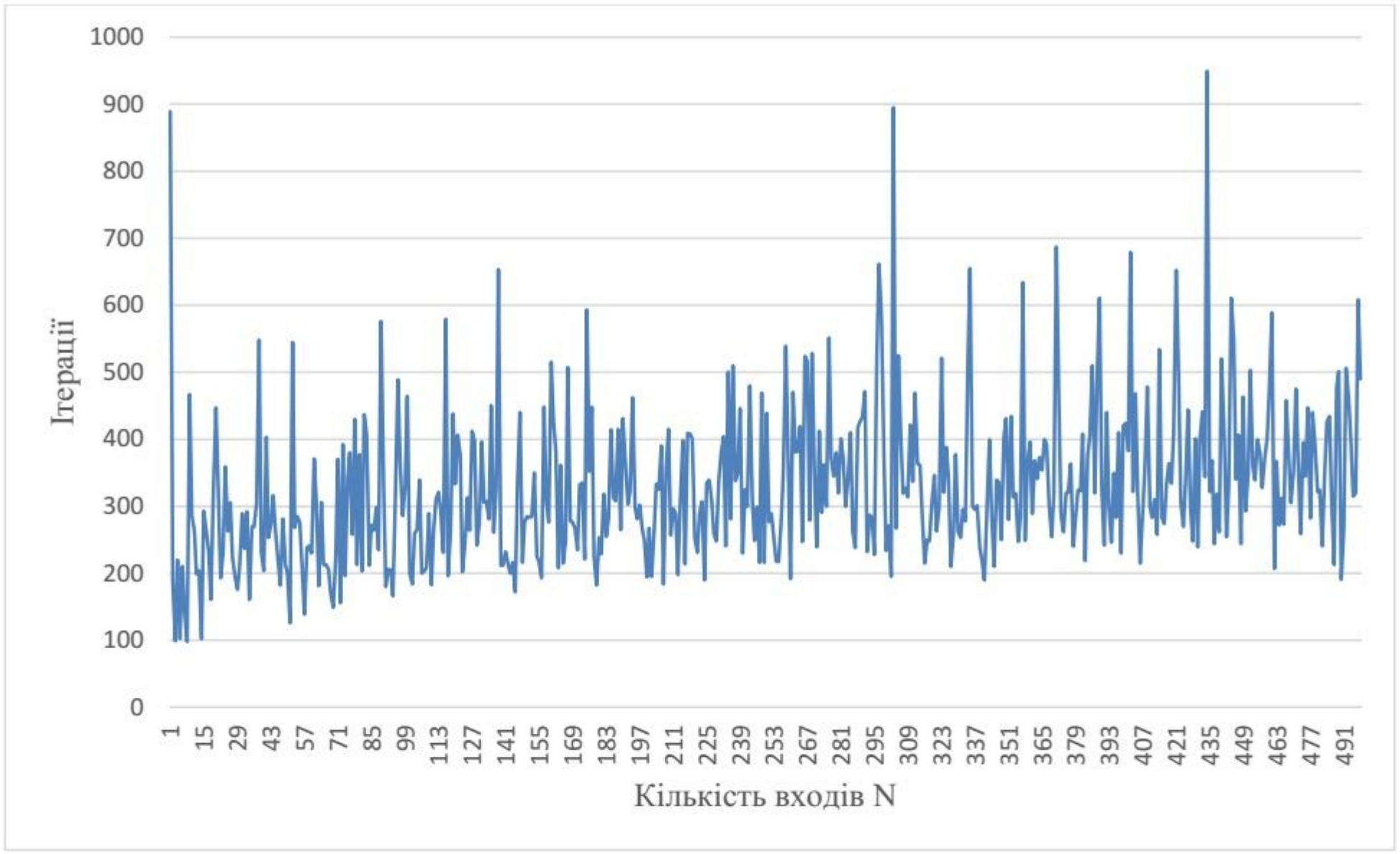


Рисунок 3.8 – Графік зміни кількості ітерацій синхронізації в залежності від К

З викладеного вище графіка, параметр N, чинить найменший вплив на кількість ітерацій, необхідних для синхронізації двох деревовидних машин парності, з усіх трьох параметрів. Крім того, можна помітити великий розкид значень при різній кількості входів.

На рисунку 3.9 показано інтерфейс виконання синхронізації. Як

```
Creating machines : k=3, n=9, l=9
Using hebbian update rule.
Synchronization = 100%   /  Updates = 71 / Eve's updates = 26
Machines have been synchronized.
Time taken = 0.43601942062377793 seconds.
Updates = 71.
Eve's machine is only 0 % synced with Alice's and Bob's and she did 26 updates.
Key: 1B0FD9EFA5279C4203B7C70233F86DBF
```

Рисунок 3.9 – Інтерфейс виконання синхронізації

На рисунку можна побачити, час синхронізації та синхронізацію зловмисника з іншими сторонами, хеш значення ключа при параметрах К = 3, N = 9, L = 5.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання економічного потенціалу розробки

Метою роботи є розробка та програмна реалізація методу криптографічного перетворення на основі нейронних мереж. Розробка нових засобів та проведення науково-конструкторських робіт потребує грошових витрат. Для оцінки результату проведених досліджень, їхньої практичної користі потрібно провести оцінку економічної ефективності виконаних робіт. Технологічний аудит проводиться з метою оцінювання комерційного потенціалу розробки, що створена в результаті наукової діяльності [24].

Для проведення технологічного аудиту залучені 3 незалежних експерти: керівник магістерської роботи доцент кафедри захисту інформації Вінницького національного технічного університету Куперштейн Л. М; доцент кафедри захисту інформації Вінницького національного технічного університету Войтович О. П; доцент кафедри захисту інформації Вінницького національного технічного університету Баришев Ю. В.. Оцінювання комерційного потенціалу розробки здійснюється за критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на малому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 4.1

3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
Ринкові переваги (недоліки):					
4	Технічні та споживчі властивості продукту значно гірші ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатацій-ні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатацій-ні витрати на рівні експлуатацій-них витрат аналогів	Експлуатацій-ні витрати трохи нижчі, ніж в аналогів	Експлуатацій-ні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці як з питань технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження таблиці 4.1

1	2	3	4	5	6
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використову-ють у військовому промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використо-вують у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10 років	Термін реалізації ідеї від 3-х до 5ти років. Термін окупності інвестицій більше 5 років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років.
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки виведені в таблиці 4.2.

Таблиця 4.2 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії	Прізвище, ініціали, посада експерта		
	Куперштейн Л. М. .	Войтович О. П.	Баришев Ю. В
	Бали, виставлені експертами		
1	3	3	4
2	3	2	2
3	4	3	3
4	4	4	4
5	3	3	3
6	3	0	2

Критерії	Прізвище, ініціали, посада експерта		
	Куперштейн Л. М. .	Войтович О. П.	Баришев Ю. В
	Бали, виставлені експертами		
7	4	3	4

Продовження таблиці 4.2

8	4	4	4
9	4	4	4
10	4	4	4
11	3	4	3
12	3	3	3
Сума балів (СБ)	$СБ_1 = 42$	$СБ_2 = 37$	$СБ_3 = 40$
Середньо-арифметична сума балів СБ	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = 39{,}6$		

З отриманих оцінок можна зробити висновки про економічний потенціал розробки. Для цього використовуються рекомендації, подані в таблиці 4.3.

Таблиця 4.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Виходячи з рекомендацій у таблиці можна побачити що розробка має потенціал вище середнього.

4.2 Розрахунок кошторису витрат на розробку

Розрахунок кошторису витрат на розробку може складатись з таких етапів:

1-й етап: розрахунок витрат, які безпосередньо стосуються розробників продукту.

2-й етап: розрахунок виробничої собівартості одиниці продукту.

3-й етап: розрахунок ціни реалізації та економічного ефекту для споживача.

Основна заробітна плата розробників, які працюють над проектом визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ (грн.)}, \quad (4.1)$$

де М – місячний посадовий оклад розробника,

$Т_р$ – число робочих днів в місяці $Т_р$= 22дні,

t – число днів роботи розробника.

Над створенням розробки працювали керівник проекту та інженер, отже, для них необхідно виконати такі розрахунки:

Таблиця 4.4 – Витрати на оплату праці

Найменування посади виконавця	Місячний посадовий оклад, грн. грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці. грн.
Програміст	13000	591	35	20682
Керівник проекту	10000	454,5	20	9091
Всього:				29773

Додаткова заробітна плата $З_{дод}$ розраховується як 10 % від основної заробітної плати робітників.

$$З_{дод} = Н_{дод} \cdot З_р \text{ [грн.]}, \quad (4.2)$$

де $Н_{дод}$ - норма нарахування додаткової заробітної плати.

$$З_{дод} = 29773 * 0,1 = 2977,3 \text{ (грн.)}$$

Нарахування на заробітну плату $З_н$ у 2019 році складають приблизно 22% від суми основної та додаткової заробітної плати розробників та робітників.

$$З_н = Н\% \cdot (З_о+Зд)\ (грн.), \qquad (4.3)$$

де, Н% - процент нарахування на заробітну плату.

Підставивши значення у формулу (4.3), отримується таке значення:

$$З_н = 22\% * (2977,3+29773)= 7205,06\ (грн.)$$

У спрощеному вигляді витрати на амортизацію обладнання можуть бути розраховані за формулою:

$$A = \frac{Ц \cdot H_a}{100} \cdot \frac{T}{12} \qquad (4.4)$$

де Ц – балансова вартість персонального комп'ютера, Ц = 15000 грн.;

H_a – річна норма амортизаційних відрахувань, H_a=20%;

Т – термін використання комп'ютера, Т = 1 місяць.

$$А= ((15000*20)/100)*(2/12) =500\ грн.$$

Таблиця 4.5 – Оренда обладнання та приміщень

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
Комп'ютер	15000	20	2	500
Оренда машинного часу в обчислювальному центрі	15000	10	2	250
WI-FI	750	15	2	18,75
Всього:				768,75

Витрати на матеріали та комплектуючі, що були використані на виготовлення розробки, розраховується за формулою:

$$K = \sum_{1}^{n} H_i \times Ц_i \times K_i \text{ [грн]}, \qquad (4.5)$$

де H_i – кількість комплектуючих і-го виду, шт;

$Ц_i$ – роздрібна ціна комплектуючих і-го виду, грн.;

K_i - коефіцієнт транспортних витрат, K_i =1,1;

п - кількість видів комплектуючих.

Проведені розрахунки наведено в таблиці 4.6.

Таблиця 4.6 – Витрати на матеріали та комплектуючі при розробці ПЗ

Найменування матеріалів та комплектуючих	Кількість, шт.	Ціна за штуку, грн.	Сума, грн.
Флеш накопичувач	1	550	550
Папір А4 формату	50	0,10	5
Ручки	4	9,20	36,80
Загальна вартість			591,8

Витрати на силову електроенергію розраховуються за формулою:

$$B_e = B \cdot П \cdot Ф \cdot K_п \text{ [грн.]}, \qquad (5.6)$$

де В - вартість 1 кВт-години електроенергії, 2,44 грн;

П - установлена потужність обладнання, кВт, 0,3кВт;

Ф- фактична кількість годин роботи обладнання, 250 годин;

$K_п$ - коефіцієнт використання потужності 0,6.

$$B_e = 2,44 \cdot 0,3 \cdot 250 \cdot 0,6 = 109,8 \text{ (грн.)}$$

Інші витрати охоплюють: загально виробничі витрати (витрати управління організацією, ремонт та експлуатація основних засобів, витрати на опалення, освітлення тощо), адміністративні витрати (проведення зборів, оплата юридичних

та аудиторських послуг, тощо), витрати на збут (витрати на рекламу, перепідготовка кадрів) на інші операційні витрати (штрафи, пені, матеріальні допомоги, втрати від знецінення запасів тощо).

Інші витрати доцільно прийняти як 200% від суми основної заробітної плати розробників, що виготовили дослідний зразок, тобто від 30.

$$З_о = 29773 \cdot 2 = 59546 \text{ (грн.)}$$

Сума всіх попередніх витрат дає загальні витрати на нову розробку:

$$В=29773+2977,3+7205,06+768,75+591,8+109,8+59546 = 100971,71 \text{ (грн.)}$$

На основі цих даних необхідно обчислити величину собівартості одиниці продукції.

$$В_{заг} = \frac{В}{\alpha} = 100971,71/1 = 100971,71 \text{ грн.}$$

Прогнозування загальних витрат на виконання та впровадження результатів магістерської кваліфікаційної роботи здійснено за формулою:

$$ЗВ = \frac{В_{заг}}{\beta},$$

де β – коефіцієнт, який характеризує етап виконання магістерської кваліфікаційної роботи, 0,7.

$$ЗВ = 100971,71/0,7= 144245,3 \text{ грн.}$$

Загальні витрати на виконання та впровадження результатів роботи становлять 144245,3 грн

4.3 Прогнозування прибутків від застосування результатів розробки

Спрогнозуємо кількісно, яку вигоду можна отримати у майбутньому від впровадження результатів виконаної наукової роботи.

Виконання наукової роботи та впровадження її результатів буде здійснюватися протягом одного року. Основні позитивні результати від впровадження розробки очікуються протягом трьох років після її впровадження. Одним із основних позитивних результатів є зростання величини прибутку.

При реалізації результатів наукової розробки цілісність фалу можна буде визначити щоразу приступаючи до роботи. Покращується якість робочого процесу, що дозволяє підвищити продуктивність роботи.

Збільшення чистого прибутку підприємства $\Delta \Pi_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}),$$

де $\Delta \Pi_o$ – покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

Δ N – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість встановлена на рівні 17%, а коефіцієнт $\lambda = 0,8547$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = 0,2...0,3$;

υ – ставка податку на прибуток 18%.

В результаті впровадження результатів наукової розробки покращується якість продукту, що дозволяє підвищити ціну його реалізації на 2000 грн.

Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 200 шт., протягом другого року – ще на 150 шт., протягом третього року – ще на 100 шт.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 1000 шт., а її ціна – 3000 грн.

Потрібно спрогнозувати збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta\,П_i$ протягом першого року складе:

$$\Delta П_1 = [150 * 1000 + (3000 + 2000) * 200] * 0{,}854 * 0{,}25\left(1 - \frac{18}{100}\right) = 201331 \text{ грн.}$$

Збільшення чистого прибутку підприємства $\Delta\,П_i$ протягом другого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_2 = [150 * 1000 + (3000 + 2000) * (200 + 150)] * 0{,}854 * 0{,}25\left(1 - \frac{18}{100}\right) = 332633 \text{ грн.}$$

Збільшення чистого прибутку підприємства $\Delta\,П_i$ протягом третього року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_3 = [150 * 1000 + (3000 + 2000) * (200 + 150 + 100)] * 0{,}854 * 0{,}25\left(1 - \frac{18}{100}\right) = 420168 \text{ грн.}$$

Результати вкладених у наукову розробку інвестицій почнуть виявлятися через два роки.

4.4 Розрахунок ціни реалізації виробу та чистого прибутку

Загальні витрати на виконання та впровадження результатів роботи дорівнює 144245,3 грн. Результати вкладених у наукову розробку інвестицій почнуть виявлятися через два роки.

Ці результати виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на 201331 тис. грн. відносно базового року, у другому році – збільшення чистого прибутку на 332633 тис. грн. (відносно базового року), у третьому році – збільшення чистого прибутку на 420168 тис. грн. (відносно базового року).

Рисунок, що характеризує рух платежів наведено на рисунку 4.1.

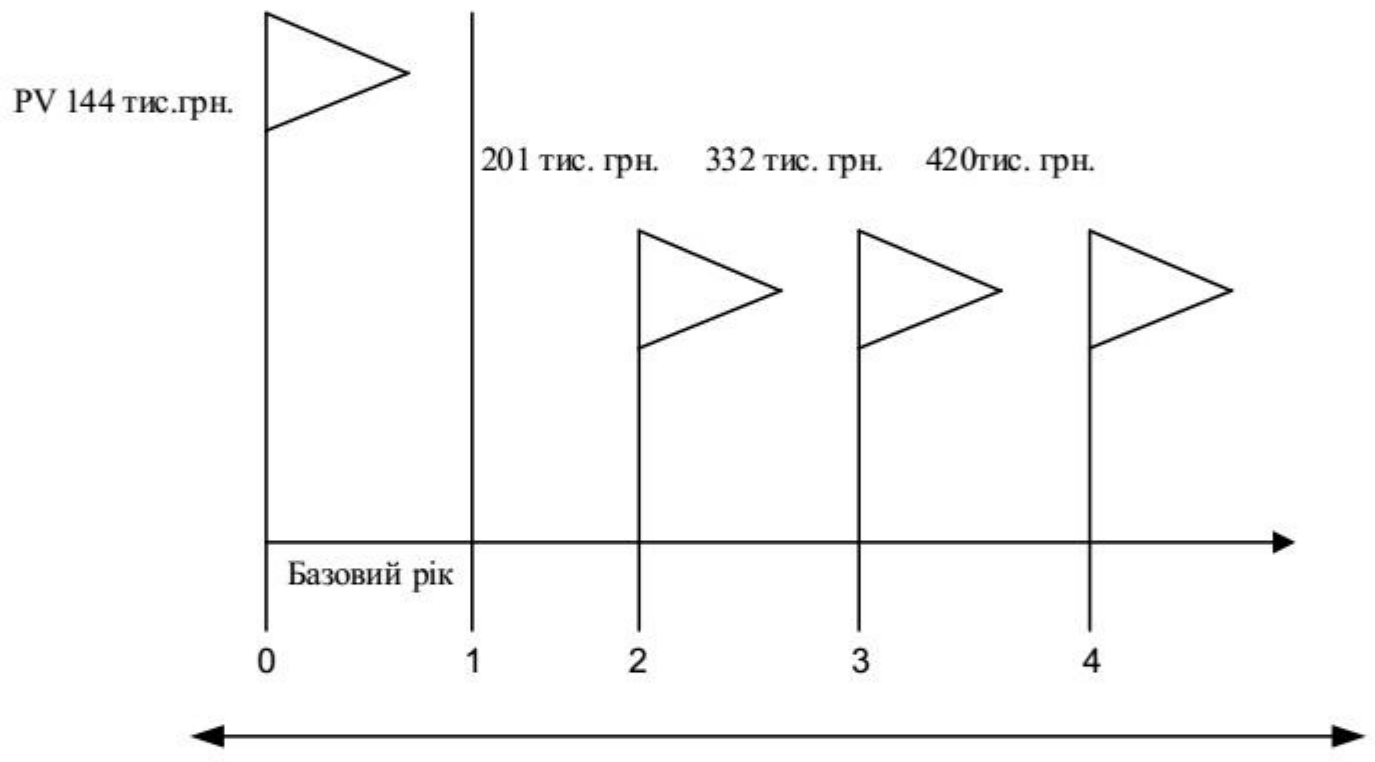


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів наукової робот

Для розрахунку абсолютної ефективності вкладених інвестицій користуються формулою:

$$E_{абс} = (ПП – PV),$$

де ПП – приведена вартість всіх чистих прибутків, які отримає підприємство (організація) від реалізації результатів наукової розробки;

PV – теперішня вартість інвестицій PV = ЗВ = 112173,78 грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{\tau} \frac{\Delta П_i}{(1+\tau)^t},$$

де $П_i$-збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ– ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки «0». <

Якщо $Е_{абс} = 0$, то результат від проведення наукових досліджень та їх впровадження буде збитковим, і вкладати кошти в проведення цих досліджень ніхто не буде.

Якщо $Е_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо приведену вартість усіх чистих прибутків:

183 028,18 274 903,3 315 678,43

$$ПП = \frac{144245,3}{(1+0,1)^0} + \frac{201331}{(1+0,1)^1} + \frac{332633}{(1+0,1)^2} + \frac{420168}{(1+0,1)^3} = 917855,21$$

Тепер, розрахуємо абсолютну ефективність вкладених інвестицій

$$Е_{абс} = 917855,21 – 144245,3 = 773609,91 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР є доцільним.

Розрахунок відносної ефективності вкладених у наукову розробку інвестицій. Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн; PV –теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

$E_в$= 0,48 (48%)

Розраховану величину $E_в$ порівняємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = (0,14...0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

Якщо величина $E_в > \tau_{мін}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки. В іншому випадку фінансування наукової розробки здійснюватися не буде.

Розрахуємо мінімальну ставку дисконтування:

$$\tau = 0,15 + 0,01 = 0,25$$

Оскільки $E_в$ = 48 % $\tau_{мін}$= 0,25 = 25 %, то у інвестора буде зацікавленість вкладати гроші в наукову розробку, оскільки значно більші прибутки він отримає від неї, ніж якби просто поклав свої гроші на депозит у комерційний банк.

Розрахунок терміну окупності інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

Якщо $T_{ок} < 3…5$-ти років, то фінансування наукової розробки є доцільним.

$$T_{ok} = \frac{1}{0,48} = 2,25$$

Фінансування наукової розробки є доцільним, оскільки термін окупності $Т_{ок}$ < 5 років.

Результати проведених розрахунків дають можливість зробити висновок про доцільність розробки та впровадження нашої наукової роботи. Це підтверджують такі показники як:

- абсолютна ефективність вкладених інвестицій, яка дорівнює 773609,912 грн., що є більшим 0 і вказує на те, що інвестор може бути зацікавленим у розробці;
- відносна ефективність наукової розробки становить 48%, що є вищим за мінімальну ставку дисконтування (25%), тому вкласти кошти у нашу розробку є вигідніше, ніж покласти кошти на депозит;
- термін окупності вкладених у реалізацію наукового проекту інвестицій складе 2,25 років, що є менше 5-ти і вказує на швидку окупність вкладених інвестицій.

Крім того, розраховано, що наукова розробка приноситиме підприємству додатковий прибуток протягом 3-х років за рахунок покращення її якості порівняно з існуючими аналогами.

ВИСНОВКИ

Результатом виконання магістерської кваліфікаційної роботи є метод передачі та шифрування інформації на основі нейронних мереж та програмний засіб для передачі симетричного ключа та шифрування на ньому інформації. Було досліджено базові принципи роботи деревовидних машин парності і їх синхронізації. Було проведено аналіз літературних джерел у напрямку криптографії, передачі секретного ключа, нейрокриптографії. Була проведена серія експериментів, для визначення впливу параметрів деревовидної машини парності на час синхронізації.

Був реалізований програмний засіб, що дозволяє здійснювати безпечну передачу повідомлень між абонентами. Даний алгоритм був реалізований на основі явища нейросинхронізації і модифікований за допомогою регістру зсуву з лінійним зворотнім зв’язком.

Таким чином можна зробити висновок, що на явищі нейросинхронізації можливо реалізувати аналог існуючих методів обміну ключами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нейронні мережі, сутність мереж [Електронний ресурс]. – Режим доступу: http://opticstoday.com/katalog-statej/stati-nakrainskom/nejromerezhi/nejronni-merezhi-sutnist-merezh.html – Назва з екрану.
2. Шнайер Б. Прикладная криптография. Протоколы,алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. – М.: Триумф, 2002. – 816 с. Дошина А. Д., Михайлова А. Е., Карлова В. В.
3. Криптография. Основные методы и проблемы. Современные тенденции криптографии [Текст] // Современные тенденции технических наук: материалы IV междунар. науч. конф. (г. Казань, октябрь 2015 г.). – Казань: Бук, 2015.
4. Gurney, Kevin (1997). An introduction to neural networks. UCL Press. ISBN 1857286731. OCLC 37875698
5. Венбо Мао Современная криптография. Теория и практика – Modern Cryptography: Theory and Practice. – М.: Вильямс, 2005. – 768 с. – 2 000 экз. – ISBN 5-8459-0847-7, ISBN 0-13-066943-1.
6. Dourlens, S., The first definition of the Neuro-Cryptography (AI Neural-Cryptography) applied to DES cryptanalysis by Sebastien Dourlens – 1995, France.
7. Latha P., Dr. L. Ganesan and Dr. . S. Annadurai, “Face Recognition using Neural Networks”, Signal Processing: An International Journal (SPIJ) Volume (3): Issue (5) pp153-160 Электронный ресурс.
8. Lian, J.S. Sun, Z.Q. Wang. Security Analysis of A Chaos-based Image Encryption Algorithm. Physica A: Statistical and Theoretical Physics, Vol. 351, No. 2-4, 15 June 2005, Pages 645-661
9. Shiguo Lian, Jinsheng Sun, Zhiquan Wang. One-way Hash Function Based on Neural Network.
10. Marshalko, “On the security of a neural network-based biometric authentication scheme”, Матем.
11. T. Wollinger, J. Guajardo, and C. Paar, “Cryptography in embedded systems: An overview (invited paper),” in Proc. of the Embedded World 2003 Exhibition and Conference. Nurnber ¨ g, Germany: Design & Elektronik, Nurnber ¨ g, Feb. 18-20 2003, pp. 735–744.
12. A. Klimov, A. Mityagin, and A. Shamir, “Analysis of neural cryptography,” in Proc. of AsiaCrypt 2002, ser. LNCS, vol. 2501. Springer Verlag, 2002, pp. 288–298
13. M. Rosen-Zvi, E. Klein, I. Kanter, and W. Kinzel, “Mutual learning in a tree parity machine and its application to cryptography,” Phys. Rev. E., vol. 66, no. 066135, 2002
14. R. Anderson, “Protecting embedded systems – the next ten years (invited talk),” in Proc. of the 3rd International Workshop on Cryptographic Hardware and Embedded

Systems, CHES 2001, ser. LNCS, vol. 2162. Paris, France: Springer Verlag, May 14-16, 2001, pp. 1–2.
15. M. Rosen-Zvi, I. Kanter, and W. Kinzel, “Cryptography based on neural networks – analytical results,” J. Phys. A: Math. Gen., vol. 35, no. 47, pp. L707–L713, 2002
16. CG Günther, An identity-based key-exchange protocol. (J-J Quisquater, J Vandewalle, eds.) (Springer, Berlin, Heidelberg, 1990).
17. S. Santhanalakshmi, K. Sangeeta, G. K. Patra, "Design of Stream Cipher for Text Encryption using Soft Computing based Techniques", IJCSNS International Journal of Computer Science and Network Security, pp. 149-152, 2012.
18. A. Ruttor and W. Kinzel, “Repulsive feedback mechanisms in neural cryptography,” 2003
19. R. Mislovaty, E. Klein, I. Kanter, and W. Kinzel, “Public channel cryptography by synchronization of neural networks and chaotic maps,” Phys. Rev. Lett., vol. 91, no. 118701, 2003.
20. W. Kinzel and I. Kanter, “Interacting neural networks and cryptography,” in Advances in Solid State Physics, B. Kramer, Ed. Springer Verlag, 2002, vol. 42
21. About Python [назва з екрану]. – Режим доступу до джерела: https://www.python.org/about/apps/
22. PyCharm features [назва з екрану]. – Режим доступу до джерела: https://www.jetbrains.com/pycharm/features/
23. PyQt5-tools Desighner [назва з екрану]. – Режим доступу до джерела: https://build-system.fman.io/pyqt5-tutorial
24. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.
25. Козловський В. О. Економіка, організація виробництва та менеджмент в дипломних роботах. Навчальний посібник / В. О. Козловський – Вінниця: ВНТУ, 2004. – 94 с.
26. Козловський В. О. Техніко-економічні обґрунтування та економічні розрахунки в дипломних проектах та роботах. Навчальний посібник / В. О. Козловський – Вінниця: ВНТУ, 2003. – 75 с

ДОДАТКИ

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д. т. н., проф
_____ В. А. Лужецький
«___» _____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу на тему:
«Інтелектуальна інформаційна технологія виявлення DDoS-атак»

08-20.МКР.014.00.000 ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.н., доц. кафедри ЗІ
_____ Л. М. Куперштейн
«___» _____ 2019 р.

1 Підстави для проведення робіт

Робота проводиться на підставі наказу № 254 ректора ВНТУ від 02.10.2019 р.
Дата початку роботи 01.09.19 р.
Дата закінчення роботи 12.12.19 р.

2 Мета та призначення НДР

Метою – є створення методу формування та передачі секретного ключа за допомогою нейронних мереж.
Об'єктом є процес передачі секретного ключа між сторонами..
Предметом є методи криптографічних перетворень.
Актуальність. Криптографія з відкритим ключем заснована на проблемі великих обчислень, або в якій невідомий ефективний алгоритм вирішення. Коли ефективний алгоритм вирішення не відомий, розвиток обчислювальної потужності та розвиток технологій зростає, наприклад, хмарні обчислення або використання великих ботнетів, дозволяють проводити жорстокі атаки. Як відповідь на це ми спостерігаємо постійне збільшення довжини ключів, що використовуються в криптографічному алгоритмі, до безпечного розміру, але це також збільшує час, необхідний для шифрування та дешифрування повідомлень. Альтернативний підхід - це пошук нових методів криптографії, який не покладається на обчислювально важкі проблеми. Цікавими є спроби застосування методів штучного інтелекту в криптографії. Використання штучних нейронних мереж для обміну криптографічними ключами для подальшого спілкування партнера за допомогою відкритого каналу зв'язку є перспективним напрямком.

3 Мета та призначення розробки

Вхідними даними НДР є:

1. Нейронні мережі, сутність мереж [Електронний ресурс]. – Режим доступу: http://opticstoday.com/katalog-statej/stati-nakrainskom/nejromerezhi/nejronni-merezhi-sutnist-merezh.html – Назва з екрану.
2. Шнайер Б. Прикладная криптография. Протоколы,алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. –М.: Триумф, 2002. – 816 с. Дошина А. Д., Михайлова А. Е., Карлова В. В.
3. Криптография. Основные методы и проблемы. Современные тенденции криптографии [Текст] // Современные тенденции технических наук: материалы IV междунар. науч. конф. (г. Казань, октябрь 2015 г.). – Казань: Бук, 2015.

4. Gurney, Kevin (1997). An introduction to neural networks. UCL Press. ISBN 1857286731. OCLC 37875698

4 Виконавці НДР

Студент групи 1БС-18м Татарчук Артем Євгенович

5 Вимоги до виконання НДР

Для розширення можливостей розробленого засобу необхідно розв'язати такі задачі:

– проаналізувати використання нейронних мереж в криптографії;

– розробити метод формування та передачі ключа шифрування за допомогою нейронних мереж;

– розробити програмний засіб;

– виконати тестування розробленого програмного засобу;

– виконати економічну доцільність обґрунтування методу.

6 Вимоги до програмної документації

Графічна і текстова документація повинна відповідати діючим стандартам України.

7 Етапи НДР

Робота з теми виконується у 8 етапів.

Зміст етапу	Початок	Закінчення	Очікувані результати	Звітна документація
Аналіз завдання. Вступ	01.09.2019	04.09.2019	Вступ	Чернетка вступу
Розробка технічного завдання	05.09.2019	15.09.2019	Технічне завдання	Проект технічного завдання
Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	15.09.2019	10.10.2019	Аналіз існуючих аналогів. Аналіз відомих методів. Постановка завдання	Чернетка першого розділу
Розробка методу криптографічного перетворення	11.10.2019	25.10.2019	Розроблено метод криптографічного перетворення	Чернетка другого розділу

Розробка засобу, що реалізує метод криптографічного перетворення	26.10.2019	01.11.2019	Розроблений засіб, що реалізує метод криптографічного перетворення	Чернетка третього розділу
Експериментальні дослідження	02.11.2019	05.11.2019	Експериментально перевірено розроблений засіб	Чернетка четвертого розділу
Розробка економічного розділу	06.11.2019	09.11.2019	Економічні показники дослідження	Чернетка економічного розділу
Оформлення пояснювальної записки	10.11.2019	20.12.2019	Пояснювальна записка	Пояснювальна записка

8 Очікувані результати та порядок реалізації НДР

Передбачається розробка нового (удосконалення існуючого) методу, який спрямований на підвищення конфіденційності інформації у відкритих мережах. Заплановане створення програмного засобу, який може бути використаний установами, органами державної влади, які є суб'єктами забезпечення кібербезпеки, суб'єктами сектору безпеки і охорони.

9 Матеріали які подаються після закінчення НДР

По завершенню роботи подається пояснювальна записка та ілюстративна частина.

10 Порядок приймання НДР та її етапів

Апробація на науково-технічних конференціях. Результати роботи будуть розглядатися на засіданні ДЕК із захисту магістерських кваліфікаційних робіт.
Попередній захист та доопрацювання МКР грудень 2019 р.
Представлення МКР до захисту 18 грудня 2019 р.

11 Вимоги до розроблення документації

Документація буде виконуватись за допомогою комп'ютерного набору у відповідності вимог ДСТУ 3008-95. «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення»

12 Вимоги щодо технічного захисту інформації з обмеженим доступом

У зв'язку з тим що дана робота не містить інформації, що потребує захисту у відповідності до законів України, заходи з її технічного захисту не передбачаються.

Розробив студент групи 1БС-18м _____ Татарчук Артем Євгенович

Додаток Б
Лістинг програми
Інтерфейс програмного засобу

```
import capture_module as cm
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QMessageBox
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(629, 363)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.tabs = QtWidgets.QTabWidget(self.centralwidget)
        self.tabs.setGeometry(QtCore.QRect(6, 9, 621, 221))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.tabs.setFont(font)
        self.tabs.setObjectName("tabs")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.tableWidget = QtWidgets.QTableWidget(self.tab)
        self.tableWidget.setGeometry(QtCore.QRect(0, 0, 611, 201))
        self.tableWidget.setAutoFillBackground(True)
        self.tableWidget.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        self.tableWidget.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        self.tableWidget.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustIgnored)
        self.tableWidget.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)
        self.tableWidget.setCornerButtonEnabled(False)
        self.tableWidget.setObjectName("tableWidget")
        self.tableWidget.setColumnCount(6)
        self.tableWidget.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(3, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
        self.tableWidget.setHorizontalHeaderItem(4, item)
        item = QtWidgets.QTableWidgetItem()
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        item.setFont(font)
```

```
self.tableWidget.setHorizontalHeaderItem(5, item)
self.tabs.addTab(self.tab, "")
self.tab_2 = QtWidgets.QWidget()
self.tab_2.setObjectName("tab_2")
self.textEdit = QtWidgets.QTextEdit(self.tab_2)
self.textEdit.setGeometry(QtCore.QRect(13, 10, 521, 171))
self.textEdit.setObjectName("textEdit")
self.groupBox_3 = QtWidgets.QGroupBox(self.tab_2)
self.groupBox_3.setGeometry(QtCore.QRect(540, 0, 71, 181))
self.groupBox_3.setObjectName("groupBox_3")
self.label_5 = QtWidgets.QLabel(self.groupBox_3)
self.label_5.setGeometry(QtCore.QRect(10, 20, 61, 31))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(False)
font.setItalic(True)
font.setWeight(50)
self.label_5.setFont(font)
self.label_5.setObjectName("label_5")
self.lineEdit_2 = QtWidgets.QLineEdit(self.groupBox_3)
self.lineEdit_2.setGeometry(QtCore.QRect(10, 60, 51, 21))
self.lineEdit_2.setObjectName("lineEdit_2")
self.label_6 = QtWidgets.QLabel(self.groupBox_3)
self.label_6.setGeometry(QtCore.QRect(0, 80, 71, 31))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(False)
font.setItalic(True)
font.setWeight(50)
self.label_6.setFont(font)
self.label_6.setObjectName("label_6")
self.checkBox = QtWidgets.QCheckBox(self.groupBox_3)
self.checkBox.setGeometry(QtCore.QRect(30, 110, 81, 20))
self.checkBox.setText("")
self.checkBox.setObjectName("checkBox")
self.monitorButton_3 = QtWidgets.QPushButton(self.groupBox_3)
self.monitorButton_3.setGeometry(QtCore.QRect(0, 140, 71, 31))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50
self.tabs.addTab(self.tab_2, "")
self.monitorButton = QtWidgets.QPushButton(self.centralwidget)
self.monitorButton.setGeometry(QtCore.QRect(510, 250, 101, 31))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
self.monitorButton.setFont(font)
self.monitorButton.setObjectName("monitorButton")
self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox.setGeometry(QtCore.QRect(20, 240, 451, 91))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.groupBox.setFont(font)
self.groupBox.setObjectName("groupBox")
self.label = QtWidgets.QLabel(self.groupBox)
self.label.setGeometry(QtCore.QRect(10, 20, 141, 31))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
self.label.setFont(font)
self.label.setObjectName("label")
self.label_2 = QtWidgets.QLabel(self.groupBox)
self.label_2.setGeometry(QtCore.QRect(10, 50, 141, 31))
font = QtGui.QFont()
```

```
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
self.label_2.setFont(font)
self.label_2.setObjectName("label_2")
self.radioButton = QtWidgets.QRadioButton(self.groupBox)
self.radioButton.setGeometry(QtCore.QRect(160, 26, 20, 21))
self.radioButton.setText("")
self.radioButton.setObjectName("radioButton")
self.radioButton_2 = QtWidgets.QRadioButton(self.groupBox)
self.radioButton_2.setGeometry(QtCore.QRect(160, 60, 16, 17))
self.radioButton_2.setText("")
self.radioButton_2.setChecked(True)
self.radioButton_2.setObjectName("radioButton_2")
self.groupBox_2 = QtWidgets.QGroupBox(self.groupBox)
self.groupBox_2.setEnabled(False)
self.groupBox_2.setGeometry(QtCore.QRect(220, 10, 201, 71))
font = QtGui.QFont()
font.setBold(False)
font.setUnderline(False)
font.setWeight(50)
font.setStrikeOut(False)
font.setKerning(True)
self.groupBox_2.setFont(font)
self.groupBox_2.setObjectName("groupBox_2")
self.label_3 = QtWidgets.QLabel(self.groupBox_2)
self.label_3.setGeometry(QtCore.QRect(10, 20, 31, 41))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setUnderline(False)
font.setWeight(50)
font.setStrikeOut(False)
font.setKerning(True)
self.label_3.setFont(font)
self.label_3.setObjectName("label_3")
self.radioButton_3 = QtWidgets.QRadioButton(self.groupBox_2)
self.radioButton_3.setEnabled(False)
self.radioButton_3.setGeometry(QtCore.QRect(40, 20, 20, 41))
self.radioButton_3.setText("")
self.radioButton_3.setChecked(True)
self.radioButton_3.setObjectName("radioButton_3")
self.label_4 = QtWidgets.QLabel(self.groupBox_2)
self.label_4.setGeometry(QtCore.QRect(60, 20, 51, 41))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setUnderline(False)
font.setWeight(50)
font.setStrikeOut(False)
font.setKerning(True)
self.label_4.setFont(font)
self.label_4.setObjectName("label_4")
self.radioButton_4 = QtWidgets.QRadioButton(self.groupBox_2)
self.radioButton_4.setGeometry(QtCore.QRect(110, 20, 20, 41))
self.radioButton_4.setText("")
self.radioButton_4.setObjectName("radioButton_4")
self.lineEdit = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit.setGeometry(QtCore.QRect(140, 30, 51, 21))
self.lineEdit.setObjectName("lineEdit")
self.monitorButton_2 = QtWidgets.QPushButton(self.centralwidget)
self.monitorButton_2.setEnabled(False)
self.monitorButton_2.setGeometry(QtCore.QRect(510, 300, 101, 31))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 629, 21))
```

```
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.retranslateUi(MainWindow)
        self.tabs.setCurrentIndex(1)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
        self.monitorButton.clicked.connect(self.monitor)
        self.radioButton.clicked.connect(self.radio_checked_neural)
        self.radioButton_2.clicked.connect(self.radio_checked_file)
        # self.radioButton_3.clicked.connect(self.pop_up_error)
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Neural Crypto"))
        self.tableWidget.setSortingEnabled(True)
```

Модуль передачі ключа

```
from machine import Machine
import numpy as np
import time
import sys

# Machine parameters
k = 3
n = 9
l = 5

# Update rule
update_rules = ['hebbian', 'anti_hebbian', 'random_walk']
update_rule = update_rules[0]

# Create 3 machines : Alice, Bob and Eve. Eve will try to intercept the communication
between
# Alice and Bob.
print("Creating machines : k=" + str(k) + ", n=" + str(n) + ", l=" + str(n))
print("Using " + update_rule + " update rule.")
Alice = Machine(k, n, l)
Bob = Machine(k, n, l)
Eve = Machine(k, n, l)


# Random number generator
def random():
    return np.random.randint(-l, l + 1, [k, n])


# Function to evaluate the synchronization score between two machines.
def sync_score(m1, m2):
    return 1.0 - np.average(1.0 * np.abs(m1.W - m2.W) / (2 * l))

# Synchronize weights

sync = False  # Flag to check if weights are sync
nb_updates = 0  # Update counter
nb_eve_updates = 0  # To count the number of times eve updated
start_time = time.time()  # Start time
sync_history = []  # to store the sync score after every update

while (not sync):

    X = random()  # Create random vector of dimensions [k, n]

    tauA = Alice(X)  # Get output from Alice
    tauB = Bob(X)  # Get output from Bob
```

```
tauE = Eve(X)  # Get output from Eve

Alice.update(tauB, update_rule)  # Update Alice with Bob's output
Bob.update(tauA, update_rule)  # Update Bob with Alice's output

# Eve would update only if tauA = tauB = tauE
if tauA == tauB == tauE:
    Eve.update(tauA, update_rule)
    nb_eve_updates += 1

nb_updates += 1

score = 100 * sync_score(Alice, Bob)  # Calculate the synchronization of the 2 machines
    sync_history.append(score)  # Add sync score to history, so that we can plot a graph
later.

    sys.stdout.write('\r' + "Synchronization = " + str(int(score)) + "%   /  Updates = " +
str(
        nb_updates) + " / Eve's updates = " + str(nb_eve_updates))
    if score == 100:  # If synchronization score is 100%, set sync flag = True
        sync = True

end_time = time.time()
time_taken = end_time - start_time  # Calculate time taken

# Print results
print('\nMachines have been synchronized.')
print('Time taken = ' + str(time_taken) + " seconds.")
print('Updates = ' + str(nb_updates) + ".")

# See if Eve got what she wanted:
eve_score = 100 * int(sync_score(Alice, Eve))
if eve_score > 100:
    print("Oops! Nosy Eve synced her machine with Alice's and Bob's !")
else:
    print("Eve's machine is only " + str(eve_score) + " % " + "synced with Alice's and
Bob's and she did " + str(
        nb_eve_updates) + " updates.")

# Plot graph
import matplotlib.pyplot as mpl
import hashlib

sum_w = 0
for (i, j), _ in np.ndenumerate(Alice.W):
    sum_w += Alice.W[i, j]
result_hash = hashlib.md5(str(sum_w).encode("utf-8")).hexdigest()

print('Key:', result_hash.upper())

mpl.plot(sync_history)
mpl.show()
```

```
from update_rules import hebbian, anti_hebbian, random_walk
import numpy as np


class Machine:
      '''Machine
      A tree parity machine. Generates a binary digit(tau) for a given random vector(X).
      The machine can be described by the following parameters:
      k - The number of hidden neurons
      n - Then number of input neurons connected to each hidden neuron
      l - Defines the range of each weight ({-L, ..., -2, -1, 0, 1, 2, ..., +L })
      W - The weight matrix between input and hidden layers. Dimensions : [K, N]
      '''
      def __init__(self, k=3, n=4, l=6):
          '''
```

```
        Arguments:
        k - The number of hidden neurons
        n - Then number of input neurons connected to each hidden neuron
        l - Defines the range of each weight ({-L, ..., -2, -1, 0, 1, 2, ..., +L })
'''
        self.k = k
        self.n = n
        self.l = l
        self.W = np.random.randint(-l, l + 1, [k, n])

def get_output(self, X):
        '''
        Returns a binary digit tau for a given random vecor.
        Arguments:
        X - Input random vector
        '''

        k = self.k
        n = self.n
        W = self.W
        X = X.reshape([k, n])

        sigma = np.sign(np.sum(X * W, axis=1)) # Compute inner activation sigma
Dimension:[K]
        tau = np.prod(sigma) # The final output

        self.X = X
        self.sigma = sigma
        self.tau = tau

        return tau

    def __call__(self, X):
        return self.get_output(X)

def update(self, tau2, update_rule='hebbian'):
        '''
        Updates the weights according to the specified update rule.

        Arguments:
        tau2 - Output bit from the other machine;

        update_rule - The update rule.
        Should be one of ['hebbian', 'anti_hebbian', random_walk']
        '''

        X = self.X
        tau1 = self.tau
        sigma = self.sigma
        W = self.W
        l = self.l

        if (tau1 == tau2):
                if update_rule == 'hebbian':
                        hebbian(W, X, sigma, tau1, tau2, l)
                elif update_rule == 'anti_hebbian':
                        anti_hebbian(W, X, sigma, tau1, tau2, l)
                elif update_rule == 'random_walk':
                        random_walk(W, X, sigma, tau1, tau2, l)
                else:
raise Exception("Invalid update rule. Valid update rules are: " +
        "\'hebbian\', \'anti_hebbian\' and \'random_walk\'.")
# Update rules for tree parity machine
import numpy as np
def theta(t1, t2):
    return 1 if t1 == t2 else 0
def hebbian(W, X, sigma, tau1, tau2, l):
    k, n = W.shape
    for (i, j), _ in np.ndenumerate(W):
```

```
        W[i, j] += X[i, j] * tau1 * theta(sigma[i], tau1) * theta(tau1, tau2)
        #print("First. Weight:", W[i, j], "Input X:", X[i, j], "Taul1:", tau1, "Taul2",
tau2)
        W[i, j] = np.clip(W[i, j], -l, l)
def anti_hebbian(W, X, sigma, tau1, tau2, l):
    k, n = W.shape
    for (i, j), _ in np.ndenumerate(W):
        W[i, j] -= X[i, j] * tau1 * theta(sigma[i], tau1) * theta(tau1, tau2)
        W[i, j] = np.clip(W[i, j], -l, l)
def random_walk(W, X, sigma, tau1, tau2, l):
    k, n = W.shape
    for (i, j), _ in np.ndenumerate(W):
        W[i, j] += X[i, j] * theta(sigma[i], tau1) * theta(tau1, tau2)
         W[i, j] = np.clip(W[i, j], -l, l)

        from machine import Machine
        import numpy as np
        import time
        import sys

        #Machine parameters
        k = 100
        n = 10
        l = 10

        #Update rule
        update_rules = ['hebbian', 'anti_hebbian', 'random_walk']
        update_rule = update_rules[0]

        #Create 3 machines : Alice, Bob and Eve. Eve will try to intercept the communication
between
        #Alice and Bob.
        print("Creating machines : k=" + str(k) + ", n=" + str(n) + ", l=" + str(n))
        print("Using " + update_rule + " update rule.")
        Alice = Machine(k, n, l)
        Bob = Machine(k, n, l)
        Eve = Machine(k, n, l)

        #Random number generator
        def random():
              return np.random.randint(-l, l + 1, [k, n])

        #Function to evaluate the synchronization score between two machines.
        def sync_score(m1, m2):
              return 1.0 - np.average(1.0 * np.abs(m1.W - m2.W)/(2 * l - 1))

        #Synchronize weights

        sync = False # Flag to check if weights are sync
        nb_updates = 0 # Update counter
        nb_eve_updates = 0 # To count the number of times eve updated
        start_time = time.time() # Start time
        sync_history = [] # to store the sync score after every update

        while(not sync):

              X = random() # Create random vector of dimensions [k, n]

              tauA = Alice(X) # Get output from Alice
              tauB = Bob(X) # Get output from Bob
              tauE = Eve(X) # Get output from Eve

              Alice.update(tauB, update_rule) # Update Alice with Bob's output
              Bob.update(tauA, update_rule) # Update Bob with Alice's output

              #Eve would update only if tauA = tauB = tauE
              if tauA == tauB == tauE:
                    Eve.update(tauA, update_rule)
                    nb_eve_updates += 1
```

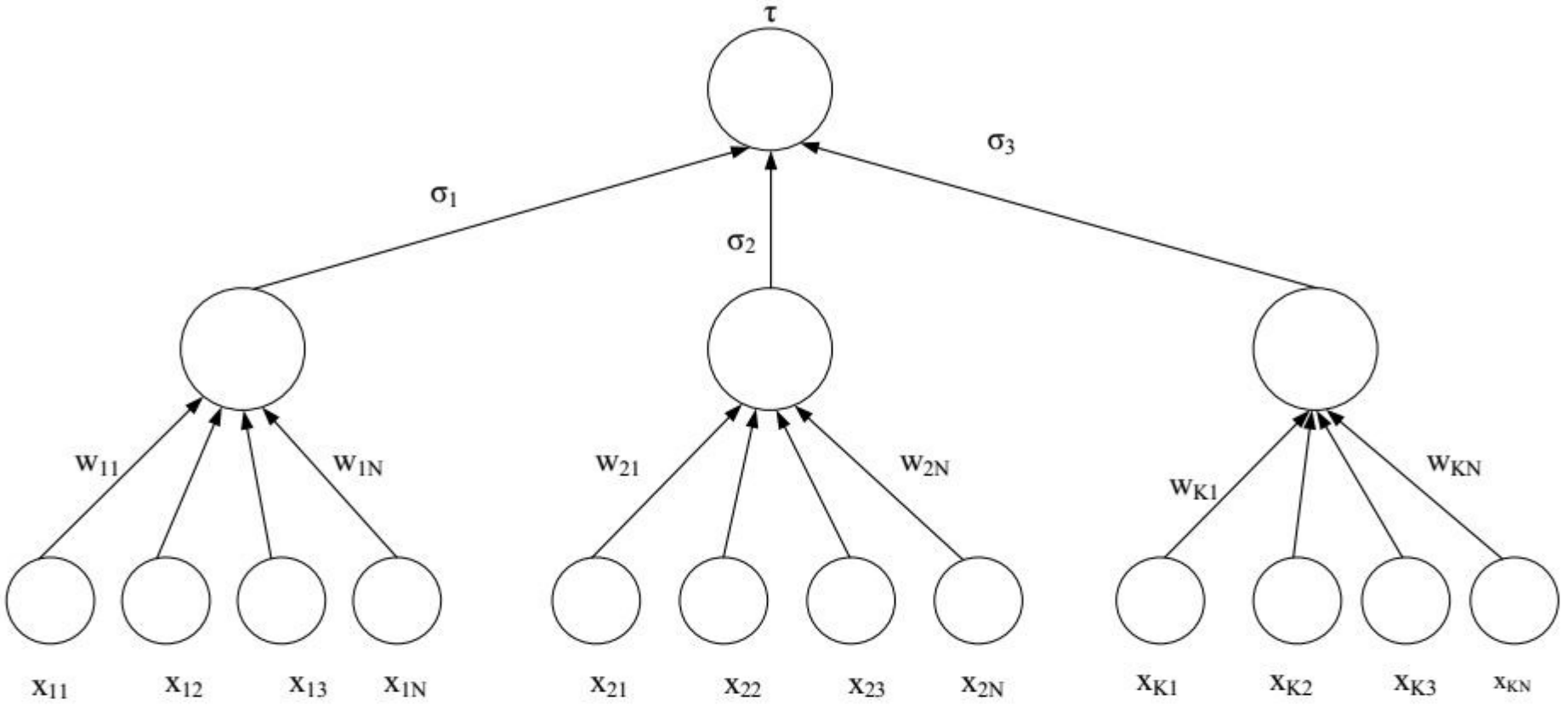
```
            nb_updates += 1

            score = 100 * sync_score(Alice, Bob) # Calculate the synchronization of the 2
machines

            sync_history.append(score) # Add sync score to history, so that we can plot a
graph later.

            sys.stdout.write('\r' + "Synchronization = " + str(int(score)) + "%   /
Updates = " + str(nb_updates) + " / Eve's updates = " + str(nb_eve_updates))
            if score == 100: # If synchronization score is 100%, set sync flag = True
                  sync = True

      end_time = time.time()
      time_taken = end_time - start_time # Calculate time taken

      #Print results
      print ('\nMachines have been synchronized.')
      print ('Time taken = ' + str(time_taken)+ " seconds.")
      print ('Updates = ' + str(nb_updates) + ".")

      #See if Eve got what she wanted:
      eve_score = 100 * int(sync_score(Alice, Eve))
      if eve_score > 100:
            print("Oops! Nosy Eve synced her machine with Alice's and Bob's !")
      else:
            print("Eve's machine is only " + str(eve_score) + " % " + "synced with Alice's
and Bob's and she did " + str(nb_eve_updates) + " updates.")

      #Plot graph
      import matplotlib.pyplot as mpl
      mpl.plot(sync_history)
      mpl.show()
```

ІЛЮСТРАТИВНА ЧАСТИНА

СХЕМА КРИПТОСИСТЕМИ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

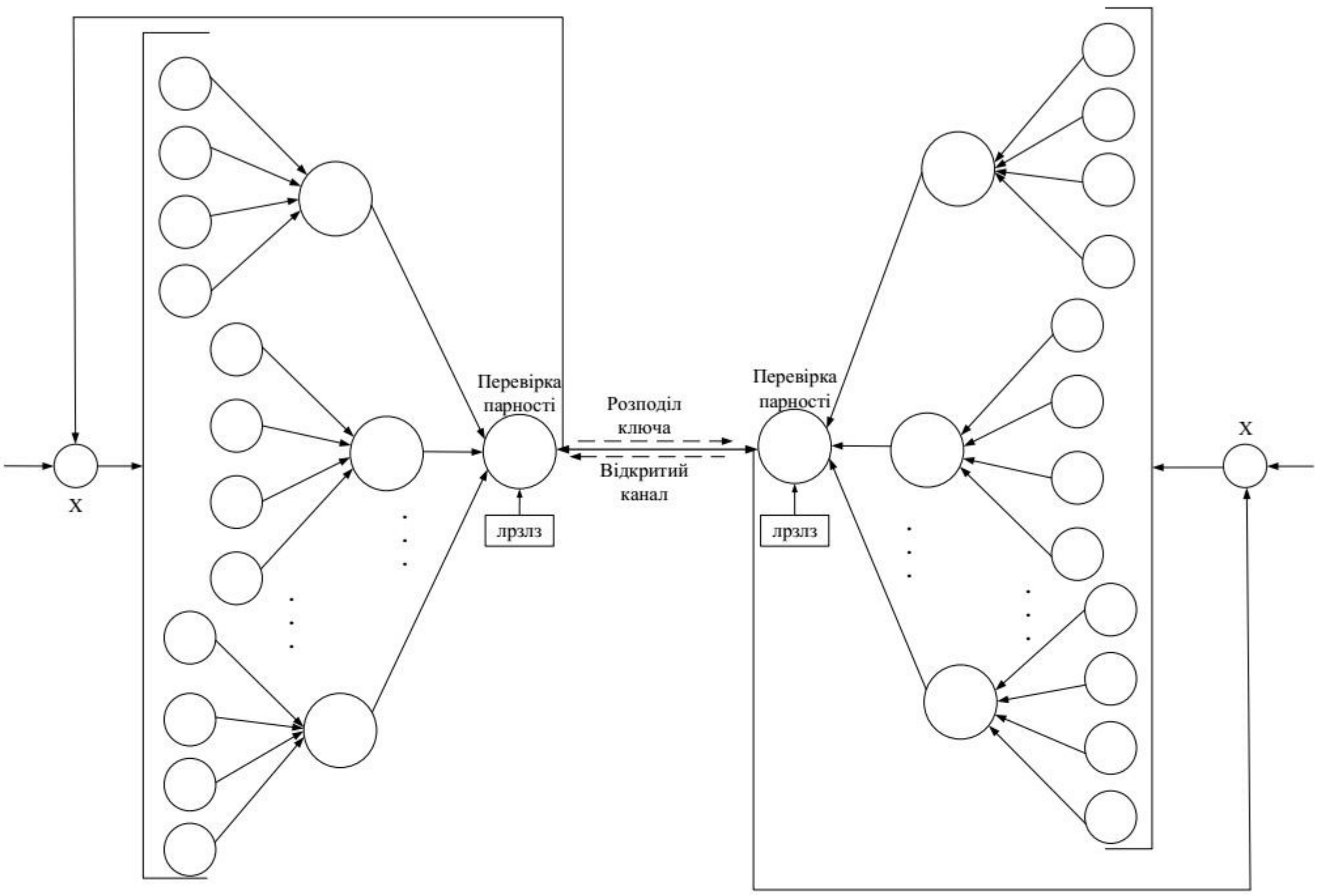
					08.20.МКР.014.00 ІЧ1			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Татарчук А.Є.			Метод криптографічного перетворення на основі нейронних мереж: Схема криптосистеми	Літ.	Арк.	Аркушів
Перевір.		Куперштейн					1	1
Реценз.		Крупельницький				ВНТУ, 1БС-18м		
Н. Контр.		Куперштейн						
Затверд.		Лужецький В.А.						

СХЕМА НЕЙРОННОЇ МЕРЕЖІ



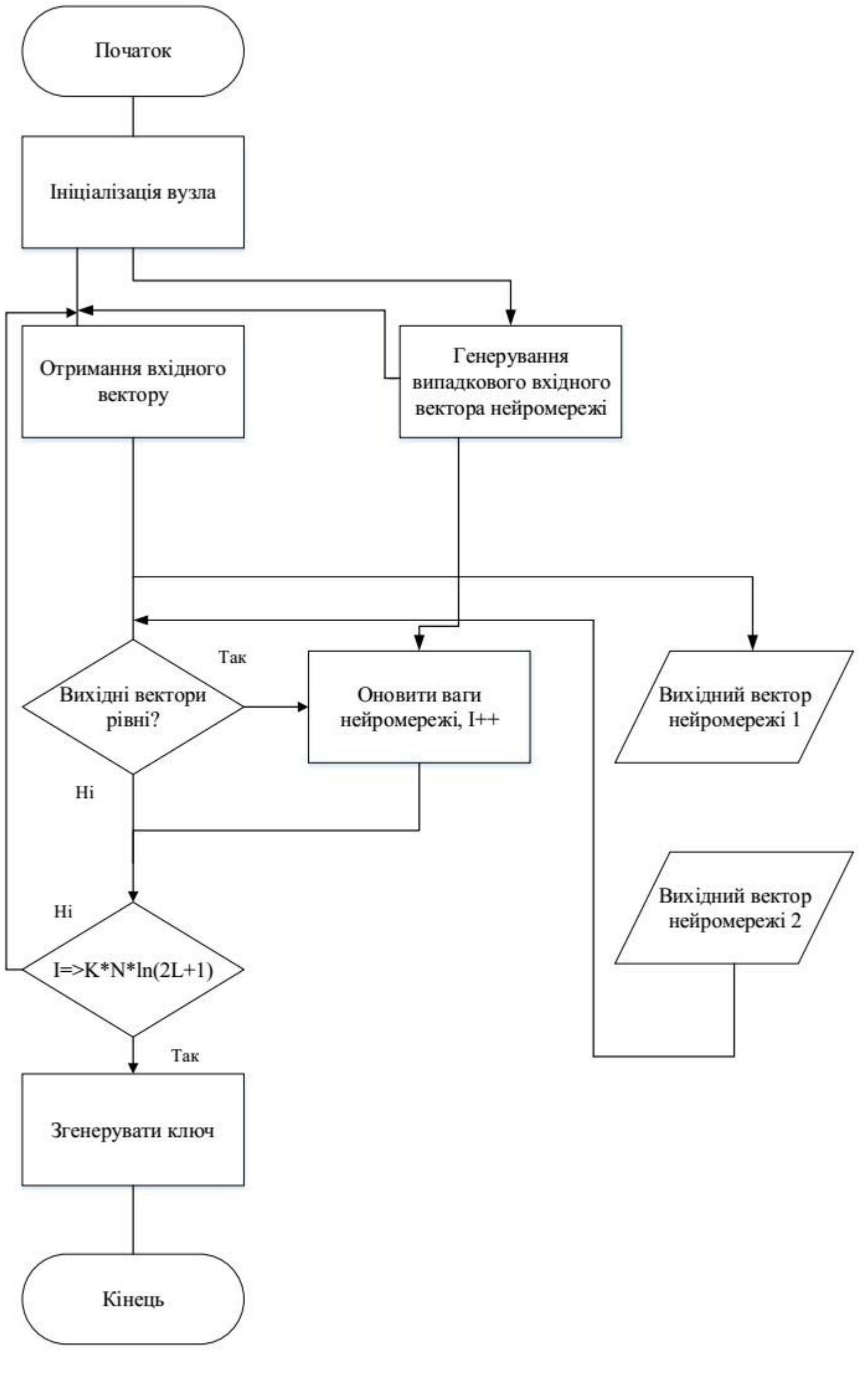
					08.20.МКР.014.00.000 ІЧ2			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Татарчук А.Є.			Метод криптографічного перетворення на основі нейронних мереж. Схема нейронної мережі	Літ.	Арк.	Аркушів
Перевір.		Куперштейн					1	1
Реценз.		Крупельницкьйи Л.				ВНТУ, 1БС-18м		
Н. Контр.		Куперштейн						
Затверд.		Лужецький В.А.						

ЗАГАЛЬНА СХЕМА ДВОХ НЕЙРОМЕРЕЖ



					08.20.МКР.014.00.000 І4			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Татарчук А.Є.			Метод криптографічного перетворення на основі нейронних мереж: Загальна схема двох мереж	Літ.	Арк.	Аркушів
Перевір.		Куперштейн					1	1
Реценз.		Крупельницкьйи Л.				ВНТУ, 1БС-18м		
Н. Контр.		Куперштейн						
Затверд.		Лужецький В.А.						

СХЕМА АЛГОРИТМУ ОБМІНУ КЛЮЧАМИ

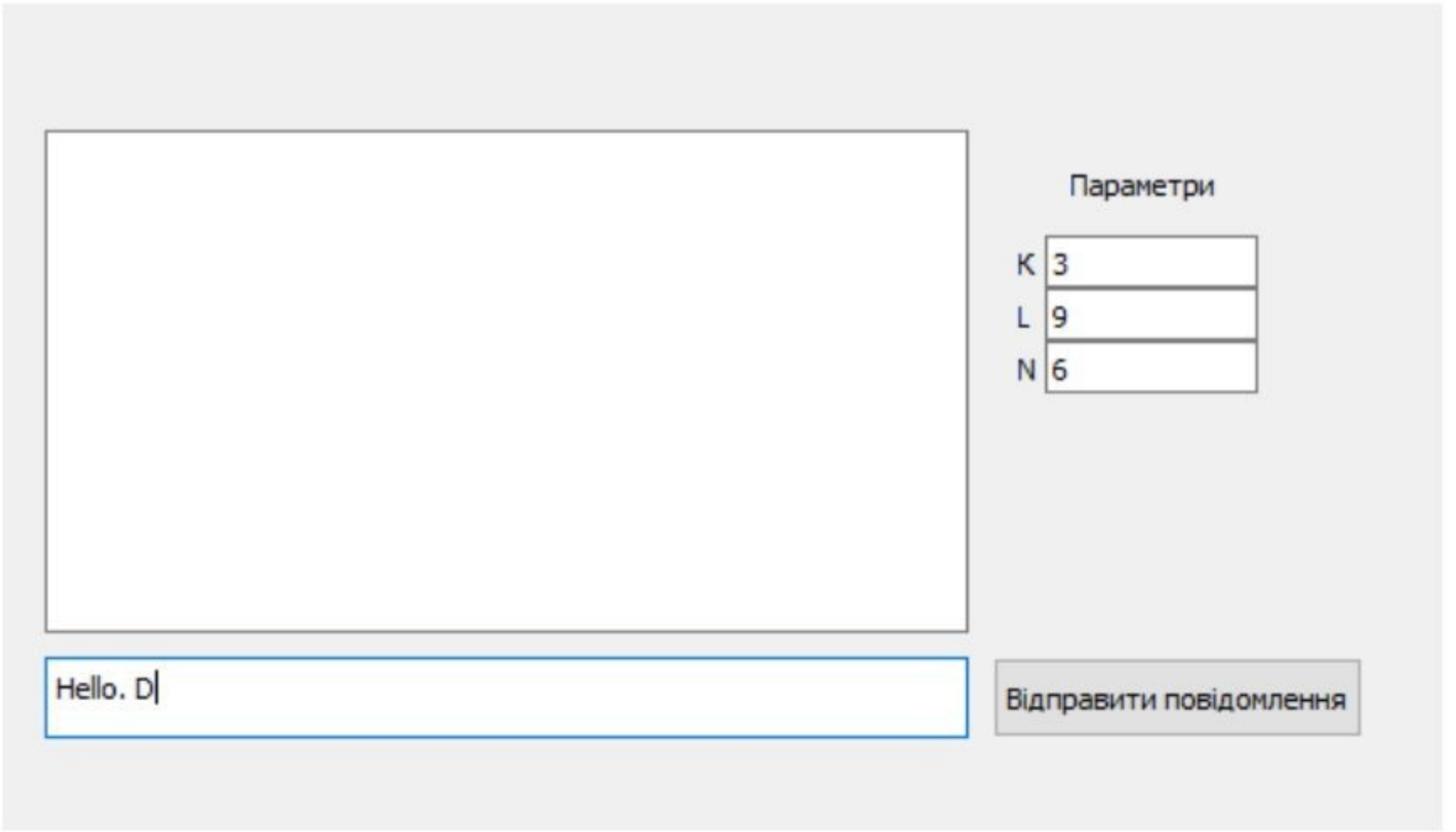


					08.20.МКР.014.00.000 І4Б			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Татарчук А.Є.			Метод криптографічного перетворення на основі нейронних мереж: Схема алгоритму обміну ключами	Літ.	Арк.	Аркушів
Перевір.		Куперштейн					1	1
Реценз.		Крупельницкьйи Л.				ВНТУ, 1БС-18м		
Н. Контр.		Куперштейн						
Затверд.		Лужецький В.А.						

АЛГОРИТМ РОБОТИ КРИПТОГРАФІЧНОГО МОДУЛЮ

					08.20.МКР.014.00.000 І Ч7		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Татарчук А.Є.			Метод криптографічного перетворення на основі нейронних мереж: Схема алгоритму роботи криптографічного модулю	Літ.	Арк. 1 / Аркушів 1
Перевір.		Куперштейн					
Реценз.		Крупельницкьйи Л.				ВНТУ, 1БС-18м	
Н. Контр.		Куперштейн					
Затверд.		Лужецький В.А.					

ФРАГМЕНТ ІНТЕРФЕЙСУ ДОДАТКУ




```
Creating machines : k=3, n=9, l=9
Using hebbian update rule.
Synchronization = 100%   /  Updates = 71 / Eve's updates = 26
Machines have been synchronized.
Time taken = 0.4360194206237793 seconds.
Updates = 71.
Eve's machine is only 0 % synced with Alice's and Bob's and she did 26 updates.
Key: 1B0FD9EFA5279C4203B7C70233F86DBF
```

					08.20.МКР.014.00.000 І4			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Татарчук А.Є.			Метод криптографічного перетворення на основі нейронних мереж: Фрагмент інтерфейсу додатку	Літ.	Арк.	Аркушів
Перевір.		Куперштейн					1	1
Реценз.		Крупельницкьйи Л.				ВНТУ, 1БС-18м		
Н. Контр.		Куперштейн						
Затверд.		Лужецький В.А.						