Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Метод захисту передачі даних у кіберфізичних системах
Назва теми

Галузь знань 12 – Інформаційні технології

Спеціальність 123 – Комп'ютерна інженерія

КРМКІ. 2001101.20.01.04 ПЗ

Виконав: студентка 2 курсу, група КІ1м-20-1 _____ Корецька Л.О.
Підпис

Керівник доц., к. т. н, зав. каф. Кб _____ Кльоц Ю.П.
Підпис

Нормоконтролер ст. викладач кафедри Кб _____ Мостовий С.В.
Підпис

До захисту допускаю:
Зав. кафедри Кб, к.т.н., доцент _____ Кльоц Ю.П.
Підпис

6.12.21 2021 р.

Хмельницький, 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КАФЕДРА КІБЕРБЕЗПЕКИ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ПРОГРАМУВАННЯ ТА ЗАХИСТ КОМП`ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

ЗАТВЕРДЖУЮ

Зав. кафедри Ю.П.Кльоц

" 01" 09 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Корецькій Людмилі Олександрівні
Прізвище, ім'я, по батькові студента

1. Тема роботи Метод захисту передачі даних у кіберфізичних системах

Керівник роботи Кльоц Ю.П., кандидат технічних наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом № 102 ректора університету додаток №23 від 25.08.2021

2. Строк подання студентом проекту (роботи) на кафедру 20.11.2021

3. Вихідні дані до проекту (роботи) атаки на дані, які передаються бездротовим зв'язком у кіберфізичній системі

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Проведення аналізу існуючих методів кодування та шифрування даних, виділити їх недоліки та особливості, провести аналіз можливих атак при передачі даних бездротовим зв'язком у кіберфізичній системі. Навести математичну модель методу та провести її апробацію. Розробити вимоги до системи захисту, алгоритм методу захисту передачі даних та алгоритм оцінки ефективності методу захисту передачі даних, провести моделювання запропонованого методу захисту передачі даних на даних різного об'єму. Висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Загальна характеристика магістерської роботи. Дослідження методів кодування та шифрування даних. Математична модель запропонованого методу. Основні положення методу. Алгоритмічна реалізація методу. Апробація методу – вхідні дані. Апробація методу - результати моделювання. Висновки.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Мостовий С..В. ст. викл. кафедри КБ		

7. Дата видачі завдання «_2_» _____вересня_____ 2021р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Вибір напряму дослідження та узгодження тематики КРМ з керівником	2.02.2021	
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	2.03.2021	
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	1.04.2021	
4	Робота над розділом 2 – розробка моделей і методів для вирішення поставленої задачі	1.05.2021	
5	Робота над науковою публікацією	1.06.2021	
6	Уточнення і затвердження теми	1.09.2021	
7	Робота над розділом 3 – розробка методу та алгоритмів, їх аналіз	2.09.2021	
8	Робота над розділом 4 – апробація запропонованих рішень	1.10.2021	
9	Узгодження отриманих результатів; оформлення пояснювальної записки згідно вимог	1.11.2021	
10	Оформлення графічної частини	8.11.2021	
11	Попередній захист роботи	10.11.2021	
12	Захист роботи на засіданні ЕК	8.12.2021	

Студент _____ Корецька Л.О
Підпис Ініціали, прізвище

Керівник роботи _____ Кльоц Ю.П.
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Метод захисту передачі даних у кіберфізичних системах.

Автор роботи: Корецька Людмила Олександрівна

Керівник роботи: к.т.н., доц., зав. каф. Кльоц Юрій Павлович

Загальний обсяг роботи: 89 сторінок, 28 рисунків, 12 таблиць, 5 додатків, 28 посилань

КРИПТОГРАФІЯ, МЕТОД СТИСКУ, ПОТОКОВИЙ ШИФР, КІБЕРФІЗИЧНІ СИСТЕМИ

Метою магістерського дослідження є підвищення захисту передачі даних у кіберфізчній системі бездротовим зв'язком.

Дана кваліфікаційна робота присвячена розробці методу захисту передачі даних у кіберфізичній системі із використанням словникового методу кодування вхідної послідовності та подальшим її шифрування потоковим алгоритмом. Захист даних із використанням запропонованого методу дає можливість досягти бажаного результату за рахунок відмінності статистичних властивостей вхідної та закодованої послідовностей символів.

06.12.2021р.

ANNOTATION

a qualification work of Liudmyla Koretska

entitled Method of data transmission protection in cyberphysical systems.

Mentor: Ph.D Klots Yuri

Total volume of work: 89 pages, 28 figures, 12 tables, 5 appendices, 28 references

CRYPTOGRAPHY, COMPRESSION METHOD, FLOW CODE, CYBERPHYSICAL SYSTEMS

The aim of the master's research is to increase the protection of data transmission in the cyberphysical system by wireless communication.

This qualification work is devoted to the development of a method for protecting data transmission in a cyberphysical system using a dictionary method of encoding the input sequence and its subsequent encryption by the streaming algorithm. Data protection using the proposed method makes it possible to achieve the desired result due to the difference between the statistical properties of the input and encoded character sequences.

09.12.2021 р.

ЗМІСТ

ВСТУП

Актуальність дослідження. Кіберфізичні системи заповнюють практично всі сфери нашого життя. Це і розумні будинки, розумні виробництва та мережі, Інтернет речей, безпілотний транспорт та транспортні мережі і навіть розумні міста та ін. З кожним днем в рамках четвертої промислової революції впровадження кіберфізичних систем відбувається у промисловість – індустрія 4.0. Разом із збільшенням охоплення сфер життя кіберфізичними системами відповідно і збільшується об’єми та цінність інформації, якими обмінюються елементи кіберфізичної системи.

Для передачі даних у кіберфізичних системах використовуються бездротові мережі. Це створює для зловмисників можливість перехоплення інформації з подальшим її дешифруванням та використанням у зловмисних цілях. Таке перехоплення може нанести непоправну шкоду тим, хто користується кіберфізичними системами й не тільки, оскільки вийшовши з ладу кіберфізична система може завдати не лише фінансових збитків власнику кіберфізичної системи, а нанести екологічних збитків та шкоду життю людини і т.д. А отже збільшується необхідність підвищувати захист даних, які передаються. Це стає можливим із використанням нових підходів щодо забезпечення криптостійкості системи захисту передачі даних бездротовим зв’язком.

У даній роботі було розглянуто удосконалення методу захисту передачі даних у кіберфізичній системі із використанням методу словникового стиснення та потокового шифрування. Використання методу кодування для даних кіберфізичної системи перед початком шифрування надає змогу змінити статистичні характеристики вхідної послідовності елементів (довжину повідомлення, об’єм даних, ентропію, ймовірності появи символів). Це саме ті статистичні характеристики, які потрібні зловмиснику для дешифрування даних. Надіславши тестову послідовність, зловмисник сподівається у результаті

дешифрування цієї послідовності (виконується зворотна операція сума за модулем 2 до зашифрованої послідовності), отримати ключову послідовність, яка використовувалась під час шифрування. Маючу кодову послідовність та перехопивши наступні блоки інформації дешифрування їх не складе ніяких складнощів. Використання кодування перед шифруванням не дає змогу виділити вхідну тестову послідовність та отримати ключ.

Отже тема захисту передачу даних у кіберфізичних системах є актуальною.

Об’єкт досліджень: процес кодування та шифрування даних у кіберфізичній системі із використанням словникового алгоритму кодування та потокового шифру.

Предмет досліджень: методи та алгоритми підвищення захисту передачі даних у кіберфізичній системі на основі потокових шифрів.

Метою магістерського дослідження є підвищення захисту передачі даних у кіберфізчній системі бездротовим зв’язком за рахунок використання кодування вхідної послідовності символів, яка дозволяє змінювати статистичні властивості цієї послідовності, з подальшим її шифруванням потоковим шифром.

Для досягнення мети поставлені наступні завдання:

- провести аналіз існуючих методів кодування та шифрування даних, виділено їх недоліки та особливості, провести аналіз можливих атак при передачі даних бездротовим зв’язком у кіберфізичній системі;
- навести математичну модель методу та провести її апробацію;
- розробити вимоги до системи захисту, алгоритм методу захисту передачі даних та алгоритм оцінки ефективності методу захисту передачі даних;
- провести моделювання запропонованого методу захисту передачі даних на даних різного об’єму.

Наукова новизна отриманих результатів: удосконалено метод потокового шифрування даних, який полягає у використанні перед шифруванням алгоритмів кодування, що відрізняється від відомих зміною статистичного розподілу елементів вхідного повідомлення для підвищення захисту передачі даних у кіберфізичній системі.

Практична цінність: проведене моделювання запропонованого методу захисту передачі даних у кіберфізичній системі показало їх ефективність і переваги у використанні порівняно із методом потокового шифрування.

Публікації. Результати досліджень опубліковано у одній статті у фаховому науковому виданні, що включений до переліку ВАК України.

1 АНАЛІЗ МЕТОДІВ ЗАХИСТУ ПЕРЕДАЧІ ДАНИХ У КІБЕРФІЗИЧНИХ СИСТЕМАХ

Кіберфізична система (КФС) – інформаційно-технологічна концепція, що передбачає інтеграцію обчислювальних ресурсів фізичних сутностей будь-якого виду, включаючи біологічні та рукотворні об'єкти. У кіберфізичних системах обчислювальна система є розподіленою по всій фізичній системі, яка є її носієм, та синергетично пов'язана з її елементами, що становлять елементи [1].

Термін кіберфізичні системи почав використовуватись у рамках четвертої промислової революції. Вперше про «Промисловість 4.0» було згадано у 2011 році у Німеччині. У основу цього поняття покладено те, що створюються ланцюги з машин, деталей, систем, а також інтелектуальних мереж. Такі елементи можуть керувати один одним автономно. Для керування процесами виробництва мережеві машини та виробничі системи повинні обмінюватись інформацією.

В основі кіберфізичних систем не лежить якась нова технологія, використовуються вже існуючі тенденції. Кіберфізичні системи складаються з давачів, пристроїв і механізмів. Завдяки вбудованому програмному забезпеченню та їх підключенню до мережі, елементи кіберфізичної системи стають інтерактивними. Інформацію, що отримана від давачів, кіберфізична система зберігає, обробляє та використовує для виконання інших процесів.

Компоненти кіберфізичної системи (враховуючи дротовий та бездротовий зв’язок), для ефективного моніторингу та керування фізичними системами, повинні бути ефективно інтегровані та захищені [2].

Вплив на кіберфізичні системи може носити наступний характер: вплив на систему керування; на людино-машинний інтерфейс; на пристрої, які входять до складу кіберфізичної системи; на мережеві протоколи та мережеве обладнання [3].

Виділяють низку ризиків під час побудови та функціонування кіберфізичних систем:

- різнорідність даних – повинна бути забезпечення можливість підтримки різних додатків та пристроїв;

- надійність – здатність продовжувати роботу в умовах непередбачуваних та адаптуватись до сторонніх впливів;

- управління даними – опрацювання даних повинно ґрунтуватись на постійних та адаптивних запитах;

- безпека – повинна забезпечуватись безпека комунікацій, оскільки зв'язок між елементами системи та керування нею відбувається у реальному часі. Також потрібно пам'ятати не лише про дані поточні, а й про ті, що зберігаються та обробляються, для використання пізніше;

- реальний час – великі обсяги даних, що надходять від сенсорів системи, повинні відповідним чином інтерпретуватись, бути ефективними та своєчасними, оскільки фізичні процеси не залежать від результатів обчислень та не зупиняються.

Отже, метою захисту кіберфізичних систем є забезпечення неперервності процесу керування, що забезпечуються конфіденційністю, цілісністю, доступністю даних.

1.1 Стандарти для кіберфізичних систем

Сфера виробництва із використанням кіберфізичних систем є сучасною. Тому дуже важливо виявити та усунути фактори, які заважають їх роботі. Наразі існує ряд стандартів, які регулюють певні аспекти, такі як термінологія, яка буде використовуватися, процеси, вимоги до інфраструктури тощо.

Міжнародними організаціями International Electrotechnical Commission (IEC), International Organization for Standardization (ISO), Institute of Electrical and Electronics Engineers (IEEE) було введено ряд стандартів для кіберфізичних систем. За рівнями їх функціонування ці стандарти можна систематизувати

наступним чином: стандарти рівня розумного з'єднання, який відповідає за отримання даних від фізичних об'єктів; стандарти рівня перетворення даних на інформацію; стандарти рівня кіберобчислень; стандарти рівня пізнання, на якому відбувається моніторинг та прийняття рішень; стандарти рівня конфігурації (стандарти загального контролю за кіберфізичними системами) [4, 5].

На рис. 1.1 подано структуру стандартів для кіберфізичних ситсем.

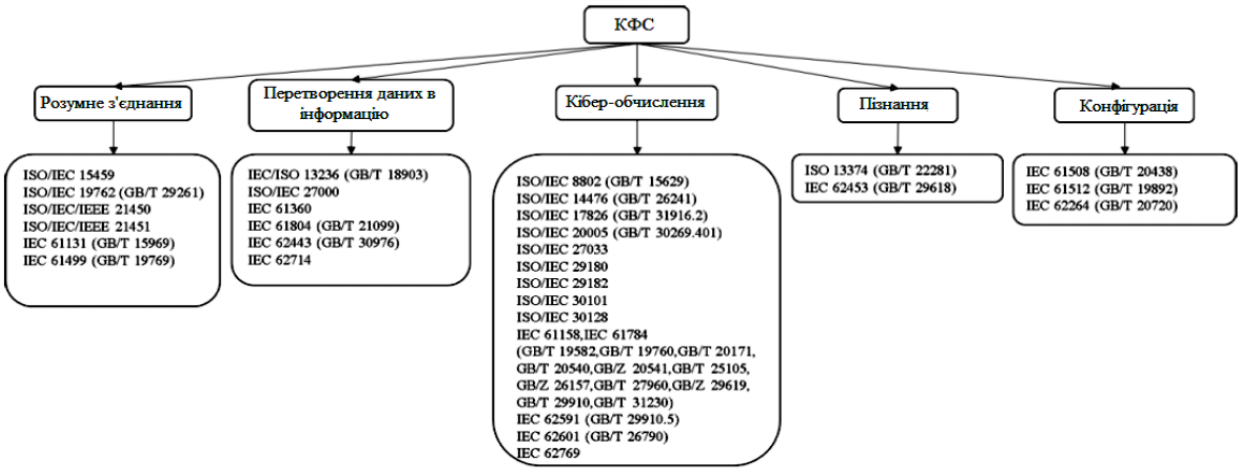


Рисунок 1.1 – Структура стандартів для кіберфізичних систем [4]

Щоб досягти однорідності та сумісності між різними елементами, дослідження в області кіберфізичних систем повинні відповідати цим стандартам.

В кіберфізичних системах використовуються різні технології для передачі даних: дротові та бездротові. На сьогодні існує велика потреба у використанні саме бездротових мереж для багатьох додатків, з метою забезпечення кращого сервісу, моніторингу за виробництвом та інше. Але програми, що використовують бездротові мережі, обмежені за максимальною пропускною здатністю.

Дротові мережі володіють такою головною характеристикою як висока швидкість передачі даних. Для отримання даних потрібно бути до неї фізично

підключеним. Недоліками такого з’єднання може бути вплив до зовнішніх сигналів, які випромінюють інші електронні пристрої.

Бездротові мережі володіють нижчою пропускною здатністю порівняно з дротовими, але інфраструктура, якщо вона встановлена, є простішою. Проте використання бездротової мережі представляє велику вразливість під час передачі даних через середовище передачі.

Бездротові технології поділяються на локальні мережі (LAN), персональні обчислювальні мережі (PAN), бездротові мережі масштабу міста (WMAN) і глобальні мережі (WAN). У табл. 1.1 наведено деякі характеристики цих типів мереж.

Таблиця 1.1 - Поділ бездротових технологій

Мережа	**Дистанція**	**Швидкість**	**Приклади технологій**
WPAN	<10 м	0,1-4 Мбіт/с	Bluetooth, ZigBee
WLAN	100 м	1-540 Мбіт/с	Wi-Fi
WMAN	> 100 м	8 Кбіт/с-2 Мбіт/с	WiMAX
WWAN	> 100 м	8 Кбіт/с-2 Мбіт/с	CSD, GPRS, EDGE, EV-DO, HSPA

Використання бездротових мереж для передачі даних у кіберфізичній системі накладає вимоги щодо пропускної здатності каналу зв’язку. Отже необхідно використовувати алгоритми стиснення даних. Використання бездротових мереж також накладає вимоги і на конфіденційність інформації, а отже необхідно використовувати систему безпеки передачі даних.

1.2 Огляд методів стиснення даних

Стиснення даних передбачає зменшення об’єму даних, які передаються, тобто потрібно усувати надмірність. Оскільки наразі суспільство, виробництво,

технології і т.д. все більше використовує поняття інформації, то і вимоги до обробки її, зберігання та використання також є досить важливими. Отже і стиснення інформації є важливим аспектом у галузі обробки даних. Згідно [6], Соломон визначає, що стисненням інформації є процес перетворення вхідного потоку даних у вихідний потік даних меншого розміру.

Отже стисненням даних є кодування, яке залежно від виду інформації, створює усунення надмірності із використанням алгоритму процесу стиснення інформації. Використання алгоритмів кодування інформації дозволяє зберігати великі об’єми інформації, при передачі даних із використанням бездротового зв’язку можна використовувати менші значення пропускної здатності каналу зв’язку для передачі такого ж об’єму інформації. Процес стиснення інформації дозволяє більш ефективно управляти ресурсами.

Час, який використовується на надсилання інформації без стиснення та інформації після стиснення, є відносно однаковий.

Значення коефіцієнту стиснення можна записати наступним чином (1.1):

$$k_{compr} = \frac{l_{compr}}{l_{origin}}, \tag{1.1}$$

де k_{compr} - коефіцієнт стиснення;

l_{compr} - довжина даних після проведення стиснення, біт;

l_{origin} - довжина даних до проведення стиснення, біт;

Значення k_{compr} буде змінним. На нього впливають:

- тип інформації (текст, відео, звук або фото) – для різних типів файлів використовуються різні алгоритми стиснення;
- алгоритм стиснення інформації, який використовується для стиснення.

Усі алгоритми стиснення інформації поділяються на алгоритми стиснення з втратами та без втрат. Перші алгоритми можуть надавати більший коефіцієнт стиснення, але відновити інформацію до першопочаткового стану вже не вийде. Алгоритми без втрат таку можливість надають, але і стиснення буде меншим. Методи другої групи використовують тоді, коли потрібно отримати інформацію точно, а першої – коли точність інформації не настільки важлива.

1.2.1 Стиснення даних із втратами

Використання алгоритмів цього типу [6-8] дозволяє отримати досить високий коефіцієнт стиснення, навіть вище 50%. Це відбувається за рахунок відкидання певної частини інформації.

Алгоритми стиснення даних із втратами в основному використовуються для стиснення відео, звукових файлів, зображень.

При використанні процесу відновлення даних із стиснутого формату, завжди буде різниця із початковими даними, оскільки при стисненні використовувалось видалення певної частини даних. Але гарний коефіцієнт стиснення (наприклад, файл .bmp можна зменшити на 95% із використанням алгоритму jpeg, а .wav – на 90% із використанням компресора mp3) зробив їх популярними у використанні.

1.2.2 Стиснення даних без втрат

Але якщо говорити про дані, які передаються у кіберфізичній системі, то застосування алгоритмів стиснення із втратами інформації, не є можливим. У [9] відзначається, що алгоритми стиснення даних без втрат, використовуються тоді, коли потрібно повністю відновити інформацію у першопочатковому вигляді. До таких типів інформації може відноситись стиснення тексту, код програми, дані системи та ін. Стиснення даних може сягати від 2 до 4 разів [6-8].

Алгоритми стиснення без втрат можна розділити на статистичні та словникові. Статистичні алгоритми стиснення використовують змінний розмір

коду, тобто ті елементи, які мають найвищу частоту появу у тексті, будуть закодовані меншою кількістю біт, і навпаки.

Нижче наведено приклади таких алгоритмів:

- статичний – ймовірності кожного символу залишаються незмінними. При такому варіанті алгоритму буде виконуватись лише одне зчитування даних. Ступінь стиснення таких даних є поганим. Перевага – швидкість;
- напівадаптивний – стиснення даних відбувається у два проходи: у першому відбувається визначення ймовірностей появи символів, а в другому – кодування;
- адаптивний – ймовірності символів змінюються під час виконання процесу стиснення. Стиснення даних відбувається за більш тривалий час, але й стиснення є кращим.

В основі використання словникових методів стиснення знаходиться використання словника. Словник може бути статичним або динамічним:

- статичний - використовується, коли повторювана структура даних є відомою ще до отримання вхідних даних. Словник є постійним і незмінним;
- динамічний – розмір словника змінюється під час надходження інформації у процесі стиснення даних. Коефіцієнт стиснення є кращим.

Перевагою статистичних методів над словниковими є їх більший коефіцієнт стиснення. Проте вони більш складні при проведенні обчислення. Перевагою словникових методів є їх можливість використання на великих об’ємах даних при досить непоганому коефіцієнті стиснення та більше просте і швидке декодування.

Метод RLE. Метод RLE (Run Lenght Encoding) [6] є одним із одним з найбільш зрозумілим, найпростішим та найшвидшим методом стиснення.

В основі методу лежить ідея проводити заміни символів, які повторюються, парою символів, в якій вказується символ та кількість

повторень цього символу. Недоліком методу є те, що якщо у вхідній послідовності недостатньо повторень, то стик даних не відбудеться, а навіть навпаки – об’єм даних збільшиться.

Модель методу RLE можна записати наступним чином:

Нехай x_1, x_2, x_3,…, x_n – символи, які потрібно закодувати, а i_1, i_2, i_3,… i_n – повтори, які мають елементи *X* відповідно, тоді (x_1, i_1), (x_2, i_2), (x_3, i_3),…, (x_n, i_n) - стиснення даних.

Процес декомпресії потоку RLE обмежується читанням стисненого рядка та передачею інформації впорядкованих пар.

Алгоритм статичний Хаффман. Алгоритм запропонований у 1952 році Девідом А. Хаффманом. В його основі лежить використання сталої статистичної моделі Хаффмана разом із кодером Хаффмана [10]. В цій моделі використовуються коди різної довжини: ті символи, які зустрічаються найчастіше, мають коротший код, і навпаки.

Алгоритм оснований на побудові структури даних зі списку всіх символів у вигляді дерева. Дані оброблятимуться згідно їх ймовірності появи та проходитиме впорядкування за спаданням. Алгоритм виглядає наступним чином:

- відбувається формування вузлів дерева за всіма символами вхідної послідовності. Кількість вузлів рівна кількості символів;
- для утворення вузла обирається два символи з мінімальними значеннями ймовірностей. Вони об’єднуються у вузол, який міститиме суму ймовірностей цих елементів;
- доки не отримається кореневий вузол потрібно повторювати попередній крок. Він міститиме суму всіх ймовірностей появи символів, тобто 1.

Для алфавіту A з частотним розподілом $\{f(a): a \in A\}$ на основі списку ймовірностей, які розташовані у порядку спадання, будується Хаффмана. Алгоритм зображено на рис. 1.2.

```
Huffman (A) {
        s = | A |
        X= A
        for i = 1 to s-1 {
                y = new node
                left [y] = min (X)
                right [y] = min (X)
                f[y] = f [left [y]] + f [right [y]]
         insert (X, y)
            }
 return (X)
        }
```

Рисунок 1.2 – Алгоритм побудови дерева Хаффмана

Для визначення коду елемента повідомлення потрібно перейти від вузла, який описує цей символ, до кореневого вузла. Кожна гілка, що знаходиться між відповідними вузлами, кодується 0 або 1 і поступового від вузла до вузла записується для символа. Після цього отримана послідовність біт інвертується. Середню довжину отриманих кодів можна за формулою (1.2) [6].

$$l_{сер} = \sum_{i=1}^{m} l_i p_i \,, \tag{1.2}$$

де $l_{сер}$ – середня довжина отриманого повідомлення, біт;

l_i – довжина символу, біт;

p_i – ймовірність появи символу i-ого символу.

Для визначення ефективності кодування, визначають ентропію вхідного потоку символів (1.3).

$$H = -\sum_{i=0}^{m} p_i \log\ p_i\ ,\ \sum p_i = 1. \tag{1.3}$$

а потім підставити значення в (1.4)

$$n = \frac{H(x)}{l_{сер}} \tag{1.4}$$

Обчислення надмірності кодів визначається за допомогою (1.5).

$$к = 1 - n = 1 - \frac{H(x)}{l_{сер}}. \tag{1.5}$$

Процес декомпресії відносно простий, достатньо знати структуру дерева, щоб здійснити подорож від кореневого вузла до вузла, зазначеного кодом Хаффмана, щоб визначити вхідний символ. Крім того, кодування даних дотримується тієї особливості, що елементи, які використовуються для заміни оригіналів, ні в якому разі не є префіксом іншого, тому декодування даних може здійснюватися негайно і однозначно.

Алгоритм динамічний Хаффман. Динамічний Хаффман був розроблений Фаллером і Галлагером [11, 12]. Його відмінність від статичного Хаффмана полягає у визначенні частот та ймовірності появи символів в процесі стиснення. Коефіцієнт стиснення для таких методів є кращим, а за часом не дуже відрізняється. Дерево будується поступово, з самого початку воно є порожнім, тобто без символів та ймовірностей.

Процес стиснення даних із використанням динамічного алгоритму є повільнішим, але надає кращий ступінь стиснення.

Арифметичний кодер. В основі алгоритму знаходиться запропонована Пітером Еліасом у 1960-х роках [13] статистичній моделі та арифметичний кодер. В цьому методі визначаються ймовірності кожного символу, для того, що виконати процес стиснення. Результатом є дійсне число R у діапазоні 0..1, причому на його точність впливає довжина стурни. Один і той же ланцюжок можна представити різними номерами, якщо числа знаходяться в інтервалі, в якому R є обмеженим.

Процес арифметичного кодера складається зі створення послідовності вкладених інтервалів у вигляді $\Phi_k(S) = [\alpha_k, \beta_k)$, $k = 0,1,...,N$, де S — вихідна послідовність , α_k, β_k – дійсні числа такі, що $0 \le \alpha_k \le \alpha_k + 10 \le$ і $\beta_k + 1 \le \beta_k \le 1$.

Для стиснення потрібно виконати наступні кроки:

1. Визначити частоти появи кожного символу вхідної послідовності.
2. Встановіть початковий діапазон [нижня межа, верхня межа) на [0,1).
3. Розбити отриманий інтервал відповідно до ймовірностей кожного символу та функції пропорційності.
4. Розбити отриманий початковий інтервал на кількість підінтервалів, яка рівна кількості символів в алфавіті для стиснення.

За допомогою рівнянь (1.6) можна визначити підінтервали:

$$\begin{gathered} difference = \max - \min; \\ \max = \min + difference \cdot \frac{cf_{S_{i-1}}}{cf_{S_0}}; \\ \min = \max + difference \cdot \frac{cf_{S_i}}{cf_{S_0}}; \end{gathered} \tag{1.6}$$

де cf – діапазон частот, а S_i – ймовірність символу i. Оскільки ймовірність кожного елемента еквівалентна інтервалу, у якому від розташовується, то кожен із символів буде в межах інтервалу [0,1) (1.7):

$$0 \le \sum_{i=1}^{n} P_i \le 1. \tag{1.7}$$

Декодування закодованих даних визначається значенням коду $\hat{v}$ послідовності стиснення [7], її послідовність (1.8):

$$\hat{S}(\hat{v}) = \{\hat{s}_1(\hat{v}), \hat{s}_2(\hat{v}), \ldots, \hat{s}_N(\hat{v})\} \tag{1.8}$$

Декодування відбуватиметься у тій же послідовності, що й кодування, - використовується процес визначення послідовності

Процес декомпресії відновлює інформацію в тій же послідовності, в якій вони були закодовані і задані рівнянням (1.8), визначається послідовність значень стандартизованих кодів $\hat{S}(\hat{v}) = \{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n\}$. Починаючи з $\hat{v}_1 = \hat{v}$, можна визначити s_k, що з $\hat{v}_k$, а потім реалізувати $\hat{v}_{k+1}$.

Найважливішою характеристикою арифметичного кодера є його гнучкість [14]. Його можна використовувати в поєднанні з будь-якою моделлю, з умовою, що вона зможе оперувати з послідовністю ймовірностей. Недоліком кодера є його низька швидкість обробки даних, оскільки використовується іноді більше ніж одне множення на подію, а іноді і більше двох множень та поділок на полію.

Метод LZSS. Метод створено Джеймсом Сторером і Томасом Шиманським у 1982 році [7]. За основу було взято алгоритм стиснення LZ77,

який розроблений Лемпелем і Зівом у 1977 році [15]. Обидва методи покладаються на стиснення словника.

Вхідна послідовність символів завантажується у ковзне вікно, яке складається з буферу пошуку (призначений для зберігання оброблених кодів - словник) та буферу обробки (призначений для зберігання символів, які підлягають кодуванню). Схематично компресор зображено на рис. 1.2

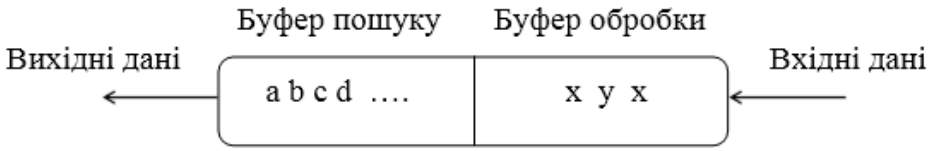


Рисунок 1.2 – Схема компресора, що побудований за алгоритмом LZ77

Процес кодування починається із введенням символів вхідної послідовності в буфер обробки. Кожен символ зчитується та порівнюється із буфером пошуку (словником). Якщо у словнику виявлено збіг, то він замінюється на впорядковану трійку кодів: [зміщення, довжина, символ]. Де зміщення – початкова точка збігу між словником та буфером обробки, довжина – вказує скільки позицій охоплює збіг, символ – вказує наступний елемент для обробки.

На початку кодування обидва буфера порожні. Під час процесу кодування словник і буфер обробки збільшуються до встановленої межі, яка визначається заздалегідь. Символи починають заповнювати буфер обробки та порівнюватись із вмістом словника. Потрібно віднайти найдовшу відповідність з буферу обробки із словником.

На ефективність стиснення спливає спосіб кодування цих впорядкованих трійок. Тому вдосконаленням методу LZ77 і є метод LZSS.

Алгоритм LZW. Алгоритм словникового стиснення LZW є одним з варіантів алгоритму LZ78. Був розроблений у 1984 році Террі Велчем [13].

LZW перед початком роботи містить всі можливі символи, тобто перший символ, який підлягатиме обробці, завжди буде знайдений у словнику.

Ідея алгоритму полягає у створенні словникових записів, що доповнюють вихідний словник, тобто буде доповнюватись словник по мірі використання процесу стиснення.

Алгоритм LZW:

1. Записати у словник всі можливі символи.
2. Зчитати початковий символ вхідної послідовності та повернути його код.
3. Зчитати наступний символ та об’єднати з попереднім. Переглянути чи є в словнику такий вміст. Якщо немає, то ввести у словник новий символ , якщо знайдено, то повернути код співпадіння.
4. Якщо дані ще існують, то продовжити починаючи з пункту 3, якщо ні, то вийти.

При декодуванні запускається процес відновлення словника. Це є перевагою, оскільки не має необхідності передавати весь словник, достатньо передати лише розмір словника, який створюється при кодуванні.

1.3 Огляд методів шифрування даних

Поняття шифрування даних походить від грецького cryptós і означає приховувати, a graphein – означає писати.

Шифрування даних – це процес, який призначено для створення захисту повідомлень від несанкціонованого доступу та зчитування, тобто відбувається кодування інформації таким чином, що без знання одного або кількох ключів не є можливим отримати вихідну інформацію.

Криптографія за визначенням Менезеса та ін. [16] передбачає вивчення математичних методів, пов'язаних з аспектами інформаційної безпеки, такими як цілісність даних, конфіденційність, автентифікація.

Процес, який зворотнім до шифрування, тобто виділення корисного тексту із зашифрованого, називається дешифруванням.

Із використанням криптографії стає можливим встановити безпечний зв’язок із використання таких аспектів:

- конфіденційність – оригінальну інформацію зможуть побачити лише ті суб'єкти, які є авторизовані;
- цілісність – зберігається зміст даних, тобто перевіряється чи не було вставок, заміна або видалення частини інформації;
- автентифікація – гарантування, що до процесу передачі даних залучені лише ті суб’єкти, які мають на це дозвіл;
- підтвердження – суб'єкт, що маю доступ до даних, не заперечує факту, що був частиною обміну або передачі зашифрованою інформацією.

Залежно від алгоритмів роботи, шифри поділяються на [17]: симетричні (для шифрування та дешифрування потрібен один ключ) та асиметричні (ключі поділяються на секретний, яким шифрують повідомлення, та відкритий – за допомогою нього відбувається процес дешифрування повідомлення).

Залежно від способу шифрування симетричні шифри поділяються на блочні шифри та потокові шифри.

1.3.1 Симетричні шифри

Швидкість шифрування симетричними шифрами є зазвичай висока, за рахунок використання одного ключа для відтворення процесу шифрування та дешифрування (рис. 1.3).

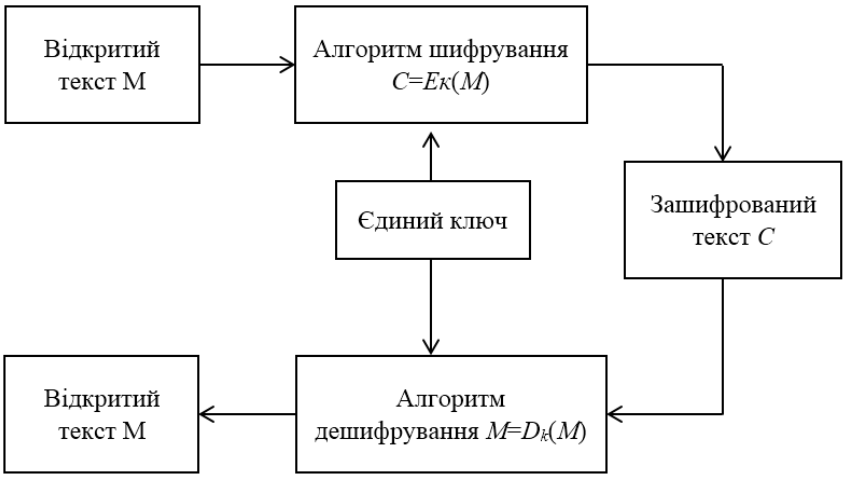


Рисунок 1.3-Схема симетричних шифрів

Потокові шифри. Нехай A – алфавіт складається з q символів, а E_e – простий шифр підстановки з блоком довжини l, де $e \in K$. Нехай $m_1 m_2 m_3$ – рядок звичайного тексту, а $e_1 e_2 e_3$ – потік ключів K. У алгоритмі потокового шифрування відбувається створення зашифрованого тексту $c_1 c_2 c_3$, де $c_1 = E_{ei}(m_i)$. Якщо d_i позначає зворотний процес e_i, то $D_{di}(c_i) = m_i$ містить розшифрування зашифрованого тексту.

Шифрування із використанням потокових шифрів може відбуватись побітно або побайтно, що робить їх досить швидкими. Також ці шифри є досить стійкими до помилок, оскільки навіть наявна помилка не поширюється на наступні шифри. Але складність виявлення будь-яких змін у ньому призводить до певних недоліків у використанні.

Шифрування даних із використанням потокових шифрів рекомендується, тоді, коли ресурси пам’яті є обмеженими, а дані повинні оброблятись посимвольно.

Метод RC4. Цей метод шифрування був створений Роном Рівестом у 1987 році і передбачає шифрування потоку ключів змінного розміру. Такі шифри призначені для роботи з байтами інформації [18].

Використання цього методу потокового шифрування передбачає наявність двох записів: перший – ключова послідовність, другий – інформація, яку потрібно зашифрувати. Відбувається генерування псевдовипадкової послідовності біт (потік ключів), який потім об’єднується із використанням операції сума за модулем 2 із вхідною послідовністю символів для їх шифрування.

У даному алгоритмі ключовий потік залежить не лише від простого тексту, а й від 256-байтового вектору стану S -box, який містить елементи $S[0]$, $S[1]$, …, $S[255]$, які переставляються по відношенню до вхідного ключа, а також по відношенню самих до себе.

Для шифрування та дешифрування використовується один і той же процес – операція сума за модулем 2.

Важливо зазначити, що як для шифрування даних, так і для їх розшифрування виконується один і той же процес. Потім, щоб ініціювати шифрування або дешифрування, спочатку генерується байт k символу S шляхом вибору одного з 255 записів. Кожен із записів S ініціалізується в порядку зростання від 0 до 255.

Щоб виконати процес шифрування, здійснюється операція XOR зі значенням k з наступним байтом інформації, яку потрібно зашифрувати.

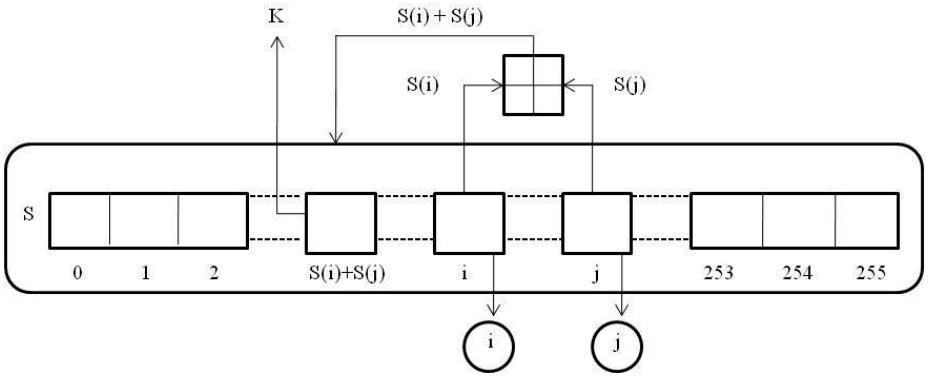


Рисунок 1.4 - Схема алгоритму шифрування RC4

Блокові шифри. В основі використання блокових шифрів для шифрування вхідної послідовності символів знаходиться розділення вхідної інформації на блоки символів. Шифрування відбувається поблоково, що призводить до значно меншої швидкості обробки інформації порівняно із потоковими шифрами. При виникненні помилки у одному символі, вона буде розповсюджена і на весь блок, що робить використання блокових алгоритмів не надто надійними. Проте, перевагою їх є те, що введення нового стороннього блоку, є неможливим без його виявлення.

Алгоритм DES. Алгоритм DES є комбінацією таких двох методів шифрування: перестановки та дифузії (заміна з наступною перестановкою). У цьому алгоритмі створюється 56-бітний ключ і відбувається шифрування даних розміром 64 біт. Шифратор DES містить два записи: вхідну послідовність символів та ключ, за допомогою якого буде виконуватись шифрування.

На першому етапі відбувається процес перестановки у 64-бітних блоках. Кожен з цих блоків розділяється на 2 рівні блоки по 32 біт і на наступних етапах відбуваються подальші перестановки у цих блоках, а також заміна елементів. На 16-ому етапі відбувається повернення 64-розрядного результату, в якому зашифровано вхідну послідовність символів із використанням ключа.

Поряд із використанням ключа, використовується зсув елементів. З перших початкових 56 елементів, обирається 48 елементів. Для кожного з етапів генерується підключ k_i на основі комбінації перестановок та зсуву елементів. При послідовному зміщенні елементів генеровані підключі будуть відрізнятись один від одного на всіх 16 етапах, щоб функція перестановки була різною на всіх етапах.

Отже, функція F складається в основному з чотирьох кроків: розширення, змішування, підстановка, перестановка.

Результат виконання функції F із використанням операції сума за модулем 2 поєднується із лівим блоком. У результаті відбувається генерування нового

блоку у правій частині. А з попередньої правої частини відбувається створення нового лівого блоку. Таких повторень відбувається 16 – 16 раундів DES.

Алгоритм шифру наведено на рис. 1.5. Використовуються наступні позначення: C_i - результат ітерації i, I_i та D_i є лівою та правою половинами C_i, K_i – 48-бітовий ключ для раунду i, F – функція, яка виконує необхідні перестановки та відповідні заміни, оперує ключем XOR.

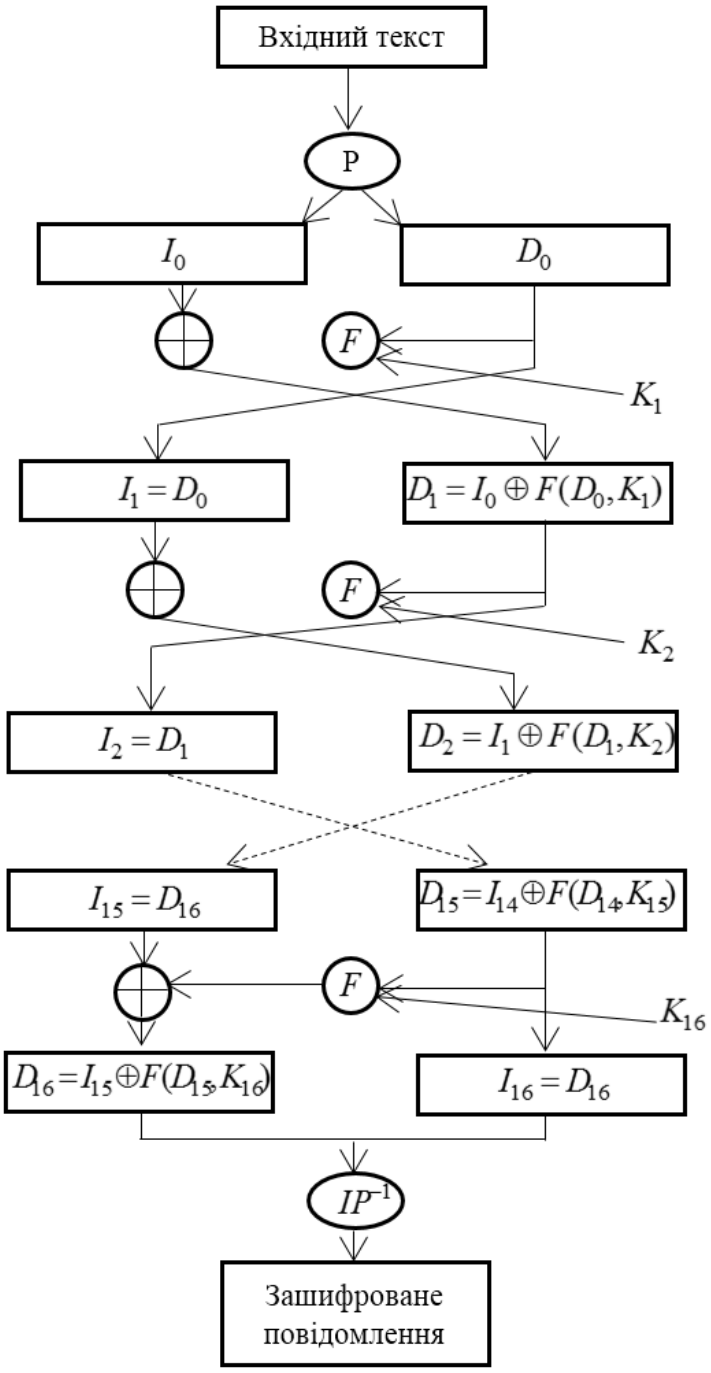


Рисунок 1.5 – Схема алгоритму шифрування DES

Із використанням встановлених таблиць визначаються початкова та обернені перестановки. Перші записи у таблицю мають номера від 1 до 64. Таблиця перестановок містить такі самі позиції пронумерованого біта у виводі і містить аналогічно 64 біти.

При дешифрування виконуються такі самі кроки, як при шифруванні, але підключі використовуються у зворотному порядку.

Алгоритм AES. Алгоритм AES був розроблений В. Рейменом і Дж. Демен, опублікований NIST у листопаді 2001 р. [19] і прийнятий як заміна DES.

В основі цього алгоритму знаходиться використання блоків вихідної послідовності символів по 128 біт, а ключ розміром 128, 192 і 256 біт. В кожній фазі шифрування/дешифрування у векторний масив перезаписуються 128 біт вхідної послідовності символів. Окремо в масиві зберігається ключ, який призначений для подальшої обробки вхідної послідовності символів. Всі елементи мартиці розташовуються послідовно у матриці 4x4 байтів, а ключ є вектором розміром 128 біт (16 байт), в який копіюється та розташовується вектор ключ, який містить слова обміну. Для вектору 128 біт таких слів є 44 і кожне складається з 4 біт. Процес шифрування складається з 10 етапів: перші 9 етапів мають 4 підетапи, а 10-ий етап – 3 підетапи. Розглянемо підетапи:

Важливо зазначити, що AES складається з десяти етапів, де кожен з перших дев’яти має чотири підетапи, а останній, десятий, містить три підетапи.

Такими підетапами, на яких здійснюється перестановка та заміна, є:

- SubBytes – відбувається заміна кожного з байтів на інший елемент використовуючи у якості посилання S-Box. В кожному байті вектора відображається інформації про номер рядка (старші 4 біти байта) та про номер стовпця (молодші 4 біти байта);
- ShiftRows – відбувається зсув вліво: перший рядок залишається без змін, в другому відбувається зсув на один байт, в третьому – на дві, в четвертому – на три.

- MixColums – відбувається дифузія в шифруванні: комбінація стовпців підлягає обробці необоротною функцією, при якій кожен стовпець розглядається у вигляді поліному GF(28). Модуль $(4x+1)$ об'єднується з фіксованим поліномом $c(x)$ за формулою:

$$c(x) = \{03\}x_3 + \{01\}x_2 + \{01\}x_1 + \{02\}x.$$

- AddRoundKey – після проведення відповідних перестановок та дифузій використовується об'єднання кожного елемента отриманої матриці із підключем, який отримується з оригінального ключа шифрування із використанням функції ітерації. Для кожного раунду буде створений новий підключ. Нарешті, на рис. 1.6 показана схема шифрування AES.

-

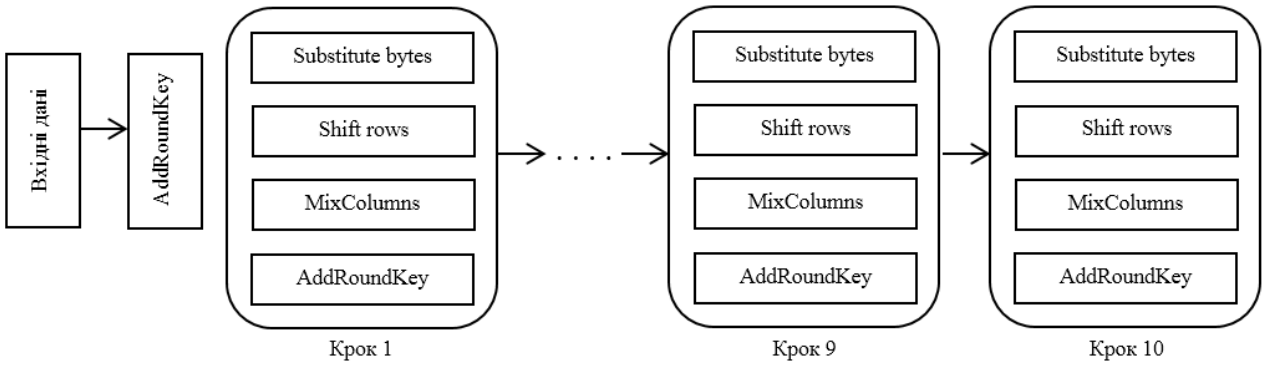


Рисунок 1.6 – Схема шифрування AES

1.3.2 Асиметричні шифри

У асиметричних ключах використовуються два ключі: A_k – секретний ключ (використовується для шифрування), B_k – відкритий ключ (використовується для дешифрування).

Ця техніка використовується у алгоритмі RSA, який був створений у 1977 р. Р. Рівестом, А. Шаміром та Л. Адлеманом [17].

Безпека даних, які зашифровані такими шифрами, залежить від складності математичних обчислень для отримання відкритого ключа з секретного.

В обчислювальному відношенні такі системи є більш дорогими в обчислювальному відношенні, мають вище, ніж симетричні шифри, енергоспоживання [20]. Такі шифри можуть використовуватись для шифрування елементів, які не потрібно передавати швидко, або підлягають лише зберіганню. Прикладом використання таких шифрів може бути цифровий підпис.

1.4 Обґрунтування вибору методів кодування та шифрування для досягнення поставленої мети

Аналіз отриманих результатів показав, що використання передачі даних із використанням бездротового зв’язку, призводить застосування додаткових мір забезпечення захисту передачі даних в кіберфізичній системі. У наявних протоколах передачі даних наявні властиві кожному з них вразливості. Окрім того, об’єми інформації, з якими працюють кіберфізичні системи невпинно збільшуються.

Встановлено, що всі методи кодування (стиснення) інформації володіють певними недоліками. Критеріями оцінки методів є час, за який відбувається кодування, продуктивність, ресурси, які потрібні для виконання операції кодування. У результаті аналізу встановлено, що за продуктивністю найкращим є LZ77, потім LZW, за часом – RLE, потім LZW та статичний Хаффман. У декомпересії найшвидший LZSS, потім RLE і LZW. Найкращий компроміс між стисненням, коефіцієнтом стиснення та часом декомпресії, є метод LZW. Також алгоритм LZW найкраще адаптується до схеми, має велику гнучкість для обробки рязних даних та може працювати у середовищах з обмеженими ресурсами.

Методи шифрування мають наступні особливості: найбезпечнішим методом шифрування є AES, але він є низьким за продуктивністю, потокові шифри мають здатність обробляти великі об’єми інформації за одиницю часу на відміну від блокових. Блочні шифри зазвичай використовуються для зберігання інформації, або для передачі даних в дротовому зв’язку, а потокові– використовуються в процесах передачі даних при бездротовому з’єднанні. Також недоліками блочних шифрів є їх можливість накопичувати та поширювати помилки, на відміну від потокового – помилка впливає лише на один елемент.

Проведений аналіз надав наступні результати. В умовах передачі даних кіберфізичних систем бездротовим зв’язком важливим є наступні параметри: надійність та швидкість передачі даних, безпека передачі даних. Використання сучасних методів обробки та захисту передачі даних призводить до тих чи інших недоліків. Недоліком методів шифрування даних є їх недостатня стійкість до різноманітних атак або низька швидкість та продуктивність. Зокрема, метод потокового шифрування, який оснований на реалізації процесу додавання за модулем 2 ключового потоку із послідовністю символів, має недолік, що пов’язаний з повторним виконанням операції за модулем 2 над прийнятою тестовою зашифрованою комбінацією та легким визначенням елементів ключового потоку. Процес дешифрування може стати неможливим, якщо додатково ввести процес кодування методами, які змінюють частоту появи символів. Таким методом обрано методу словникового кодування LZW, із використанням якого не лише змінюються частоти появи символів, а й їх кількість. Слід зазначити, що використовувати кодування даних потрібно перед шифруванням повідомлень, що надасть можливість не лише змінити статистичний розподіл символів, а й стиснути дані, а й відповідно підвищиться продуктивність використання методу захисту передачі даних у кіберфізичній системі.

1.5 Постановка задачі

Головною метою досліджень є захист передачі даних у кіберфізичній системі. Для досягнення цієї мети потрібно виконати наступні завдання:

1) розробити метод захисту передачі даних бездротовим зв’язком у кіберфізичній системі із використанням методу кодування (стиснення) даних, з метою зміни їх частотного розподілу, та подальшого шифрування отриманої кодової послідовності;

2) провести аналіз ефективності запропонованого методу захисту передачі даних у кіберфізичній системі;

3) провести моделювання запропонованого методу захисту передачі даних у кіберфізичній системі та зробити висновки щодо ефективності захисту передачі даних.

2 МОДЕЛЮВАННЯ ПРОЦЕСУ ЗАХИСТУ ПЕРЕДАЧІ ДАНИХ У КІБЕРФІЗИЧНИХ СИСТЕМАХ

2.1 Математична модель методу

Метод захисту передачі даних кіберфізичної системи основано на використанні двох методів кодування даних:

1. словниковий кодування (використання методу із сімейства LZ-методів словникового стиснення – LZW);
2. криптографічне шифрування даних (використання методу потокового шифрування – RC4).

Математична модель методу захисту передачі даних кіберфізної системи повинна відображати принципи реалізації створеного методу кодування та шифрування даних для забезпечення надійної передачі даних бездротовим зв’язком в умовах дії завад та можливих атак щодо перехвату даних сторонніми особами задля їх можливої зміни, підміни або видалення.

Для створення математичної моделі методу захисту передачі даних проведемо аналіз виконуваних операцій та їх послідовності виконання із використанням методу кодування даних LZW та криптографічного шифрування RC4.

Двома загальними параметрами методів кодування сімейства LZ є наявність словника та статистичного кодування. Основною концепцією кодування методом LZW є заміна підрядків тексту на місце їх попереднього положення, тобто де вони зустрічались раніше в тексті.

Дані кіберфізичної системи, які будуть надсилатись формуються у повідомлення певної довжини. Назвемо це повідомлення текстом.

Нехай $G = g_1 g_2 g_3 ... g_m$ - текст (або рядок), що створений з деякого алфавіту $\Sigma = \{a_1, a_2, a_3, ..., a_q\}$. Використання методу LZW призводить до стиснення тексту G у текстовий рядок:

$$c(G) = p_1 p_2 p_3 ... p_v, \quad (2.1)$$

де $p_i \le v + q - 1$, при $i = 1,...,v$, причому $m \ne v$.

Це перетворення відбувається за допомогою використання словника і може бути зображено у вигляді дерева. Словникове дерево формується під час кодування у режимі он-лайн. Позначимо словникове дерево літерою *T*. Тоді кожен вузол словникового дерева – це підрядок, який утворюється шляхом конкатенації символів в кожному вузлі на шляху від кореневого вузла словникового дерева. Далі відбувається заміна підрядка на відповідний код . Цей код віднаходиться наступним чином. Кожен вузол отримує номер – ціле число. Це число є вказівником на те, яким числом (кодом) повинна пройти заміна підрядка на текст для формування вихідного коду.

T створюється в якості кореневого дерева (*q*+1). Корінь цього дерева ініціалізується з міткою (0). За допомогою цього кореня представляється нульовий рядок (*A*). Корінь має дочірні вузли, що відмічені (1, 1), (2, a_2), …, (*q*, a_q) відповідно для представлення *q* рядків з одного символу.

Текст, який вводиться для кодування методом LZW, проходить перевірку символом і найдовший підрядок, який вже існує в дереві, буде замінюватись цілим числом, яке пов’язане з відповідним вузлом, який описує підрядок в дереві. Такий підрядок має назву – префікс (або префіксний рядок). Цей префіксний рядок (*w*) розширюється на наступний символ (*K*) і буде сформований новий рядок префікса (*wK*).

Дочірній вузол створюється на вузлі, який являє собою співпадаючий рядок в дереві для запису нового рядка префіксу.

На кожному кроці при $i > 0$, виводиться вказівник p_i, який знаходиться у поточному вузлі в гілці T_i.

Шляхом додавання вузла у майбутній новий рядок префікса, гілка T_i модифікується у гілку T_{i+1}, яка спочатку є неосновною.

Слід зазначити, що кількість гілок T_i завжди дорівнює (i+q). На кроках, що відповідають гілкам T_i , виводиться вказівник p_i для кожного кроку i. Результатом виконання описаних операцій є рядок $c(G) = p_1p_2p_3\ldots p_v$.

Визначимо довжину l_{p_i}, отриманого повідомлення $c(S)$. Довжина кожного закодованого символу залежить від об'єму словника і визначається (2.2):

$$l_{p_i} = \log_2 V_{dict}, \tag{2.2}$$

де l_{p_i} - довжина i-ого символу у отриманому закодованому повідомленні, біт; V_{dict}, - об'єм словника, який динамічно створюється під час кодування повідомлення. Основа логарифму обирається 2, оскільки всі символи переводяться у двійкову систему числення.

Довжину $l_{c(S)}$ отриманої кодової послідовності визначаємо наступним чином (2.3):

$$l_{c(S)} = l_{p_i} \cdot v, \tag{2.3}$$

де $l_{c(S)}$ – довжина отриманої кодової послідовності, v - кількість символів, що склали отримане повідомлення (кількість вказівників).

Наступним кроком після кодування слідує шифрування.

Для реалізації шифрування щодо закодованої кодової послідовності $c(S)$, згідно алгоритму шифрування RC4, необхідно забезпечити генерування блоку підстановки S-блок. Розмір S-блоку залежить від параметра (модуля) n. Отже довжина кожного символу S-блоку становить (2.4):

$$l_{s[k]} = \log_2 n, \tag{2.4}$$

де $l_{s[k]}$ - довжина *i*-ого елемента блоку підстановки *S*-блоку, *s*[*k*] - елемент, який знаходиться на *k*-ому місці *S*-блоку.

Отже, потоковий шифр RC4 працюватиме з блоками даних, довжина кожного з яких також дорівнюватиме $l_{s[k]}$ біт.

Отже, перед шифруванням отриману кодову послідовність потрібно поділити на рівні блоки по $l_{s[k]}$ біт. Кількість таких блоків k_{blocks}, визначається наступним чином (2.5):

$$k_{blocks} = \frac{l_{c(S)}}{l_{s[k]}}, \tag{2.5}$$

де k_{blocks} - кількість блоків розміром 1 байт (8 біт) у отриманій кодовій послідовності.

У випадку якщо остача від ділення $l_{c(S)}$ на $l_{s[k]}$ не становить 0, тобто $l_{c(S)}$ не є кратним $l_{s[k]}$, а отже і k_{blocks} не буде цілим числом, то потрібно на початку отриманої кодової послідовності (ще до початку її ділення на рівня блоки по $l_{s[k]}$ біт), встановити стільки нулів, щоб k_{blocks} стало цілим числом, тобто $l_{c(S)}$ було кратним $l_{s[k]}$. У результаті отримується послідовність $Y=y_0y_1\ldots y_{kblocks}$.

Станом схеми системи захисту є підстановка – S-блок (2.6). S-блок – матриця з *n* елементів, в якій однакова кількість рядків та стовпців.

$$S = \begin{bmatrix} 0 & \ldots\ldots & \sqrt{n}-1 \\ \cdot & & \cdot \\ \cdot & \ldots\ldots & \cdot \\ \cdot & & \cdot \\ n-\sqrt{n} & \ldots\ldots & n-1 \end{bmatrix}. \tag{2.6}$$

Для нумерації рядка та стовпця, в якому розташований елемент S-блоку $s[k]$ введемо два лічильники: i, j.

Кожен стан схеми системи захисту в момент часу t будемо позначати з індексом t: (S_t, i_t, j_t). Початкові значення $i_0=0$, $j_0=0$.

Ще одним параметром схеми системи захисту є ключ k_l, де $l=0,1,\ldots,L$-порядковий номер елемента у ключі, а L – загальна довжина ключа. Із ключового рядка формується масив, який заповнюватиметься ключем. Припустимо, що ключ має довжину $L=10$. При розмірності матриці n кількість рядків та стовпців матриці становитиме $\sqrt{n}$, а ключ заповнюватиме матрицю наступним чином (2.7):

$$K = \begin{bmatrix} k_0 & k_1 & \ldots & k_9 & k_0 & \ldots & k_5 \\ k_6 & & & \ldots & & & k_1 \\ . & & & . & & & . \\ . & & & . & & & . \\ . & & & . & & & . \\ . & & & . & & & . \\ k_0 & k_1 & \ldots & k_9 & k_0 & \ldots & k_5 \end{bmatrix}. \qquad (2.7)$$

Формування кожного стану схеми системи захисту відбувається за функцією переходів станів F (2.8):

$$F = \begin{cases} j_t = (j_t + s[\sqrt{n}(j_t+1) + i_t] + k_l(i_t \bmod L)) \bmod n, \\ i_t = (i_t + 1) \bmod n, \\ \text{перестановка : } s[i_t] \text{ т } s[j_t]. \end{cases} \qquad (2.8)$$

А функція виходу f дорівнює (2.9):

$$z_t = s_t[(s_t[i_t] + s_t[j_t]) \bmod n]. \qquad (2.9)$$

У результаті сформується послідовність $Z=z_0z_1\ldots z_{n-1}$.

Зашифроване повідомлення H отримуємо шляхом накладання сумою за модулем 2 (операція XOR) послідовності Z та Y.

$$h_i = z_i \oplus y_i.$$

Результатом є послідовність $H = h_1h_2..h_{k_{blocks}}$.

2.2 Аналіз адекватності побудови математичної моделі методу

Рекомендовано обирати модуль S-блоку рівним 256 елементів. Отже довжина кожного символу S-блоку становить (2.5):

$$l_{s[k]} = \log_2 n = \log_2 256 = 8 \text{ біт.}$$

Отже, потоковий шифр RC4 працюватиме з блоками даних, довжина кожного з яких також становить 8 біт (1 байт).

Для проведення апробації математичної моделі оберемо невелику послідовність символів, яка складається з трьох символів алфавіту: x, y, z. Загалом у символьній послідовності 16 символів (2.10).

$$G=xyzxyyzxyyxxxxxx, \qquad (2.10)$$

Оберемо для проведення кодування (стиснення) два словника: перший буде заповнюватись значеннями від 4 до кількості можливих входжень, його об’єм V_{dict_1}, а другий – наближений до реального розміру словника – 512 символів (першу частину заповнюватимуть символи з таблиці ASCII, другу – символи створені входженнями повторюваними символами у словник), його

об'єм V_{dict_2}. На рис. 2.1 зображено схему кодування (стиснення) послідовності (2.10) методом LZW:

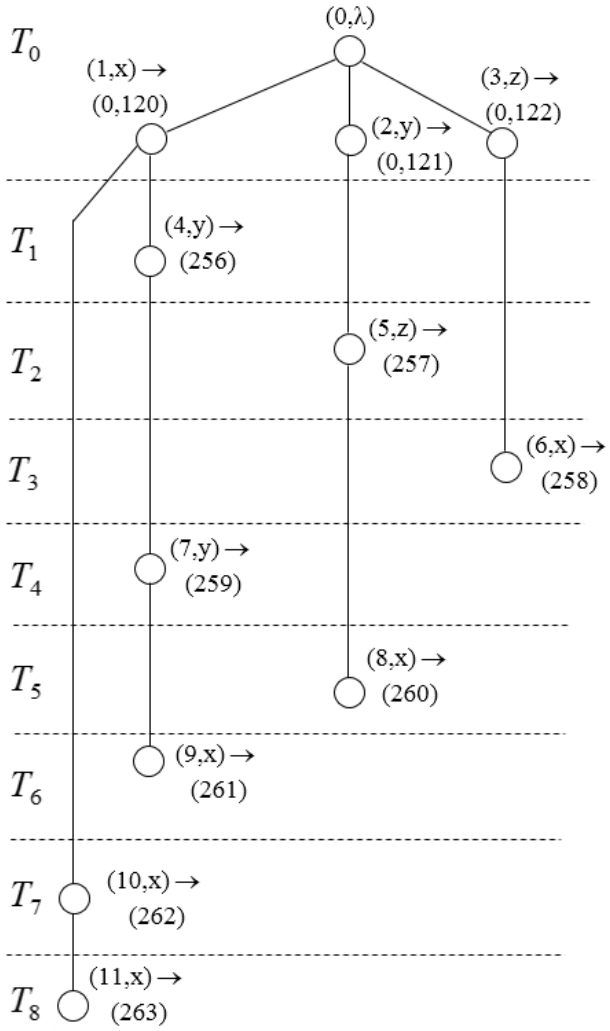


Рисунок 2.1 – Схема кодування послідовності

У результаті кодування (стиснення) отримано дві закодовані послідовності:

$$c(G)_1 = 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 7\ \ 1\ \ 10\ \ 11, \qquad (2.11)$$

$$c(G)_2 = 0{,}120\ \ 0{,}121\ \ 0{,}122\ \ 256\ \ 257\ \ 259\ \ 0{,}120\ \ 262\ \ 263\ . \qquad (2.12)$$

Послідовність символів (2.11) утворена із використанням словника $V_{dict_1} = 11$ симв., а послідовність (2.12) - із використанням словника $V_{dict_2} = 512$ симв.

Переведемо першу послідовність (2.11) у послідовність двійкових символів (2.13). Довжина кожного символу становить за (2.2) $\left(l_{p_i}\right)_1 = \log_2 V_{dict_1} = \log_2 11 = 4$ біт:

$$\begin{array}{c} 0001\ 0010\ 0011 \\ 0100\ 0101\ 0111 \\ 0001\ 1010\ 1011 \end{array} \qquad (2.13)$$

Кількості блоків у кожній з послідовностей (2.11) та (2.12) рівні і становлять $v_1 = v_2 = 9.$

Для другої послідовності (2.12) довжина кожного двійкового блоку за (2.2) становитиме $\left(l_{p_i}\right)_2 = \log_2 V_{dict_2} = \log_2 512 = 9$ біт, а послідовність буде виглядати наступним чином (2.14):

$$\begin{array}{c} 001111000\ 001111001\ 001111010 \\ 100000000\ 100000001\ 100000011 \\ 001111000\ 100000110\ 100000111 \end{array} \qquad (2.14)$$

За (2.3) довжина кодової послідовності (2.13) та (2.14) становитиме:

$$l_{c(G)_1} = \left(l_{p_i}\right)_1 \cdot v = 4 \cdot 9 = 36 \text{ біт},$$

$$l_{c(G)_2} = \left(l_{p_i}\right)_2 \cdot v = 9 \cdot 9 = 81 \text{ біт},$$

Для першої послідовності (2.13) перед початком шифрування визначимо довжину блоків $l_{s[k]}$. Довжина блоків залежить від кількості елементів S-блоку. Для можливості зручного відображення та аналізу обраної математичної моделі оберемо розмір S-блоку рівним $n_1 = 8$. Отже $l_{s[k]}$ за (2.4) становитиме:

$$l_{s[k]_1} = \log_2 n_1 = \log_2 3 = 3 \text{ біт}.$$

Для послідовності (2.14) обираємо розмір S-блоку $n_2 = 256$ елементів. Тоді довжина блоку за формулою (2.4) становитиме:

$$l_{s[k]_2} = \log_2 n = \log_2 256 = 8 \text{ біт}.$$

Визначимо на скільки блоків k_{blocks_1} по 3 біт буде розділено вихідну (вже закодовану) послідовність (2.13) за (2.5):

$$k_{blocks_1} = \frac{l_{c(G)_1}}{l_{s[k]_1}} = \frac{36}{3} = 12 \text{ блоків}.$$

Для послідовності (2.14) за (2.5) кількість блоків k_{blocks_2}:

$$k_{blocks_2} = \frac{l_{c(G)_2}}{l_{s[k]_2}} = \frac{81}{8} = 10{,}125 \text{ блоків}.$$

та округлюємо отримане значення до найближчого цілого числа в більшу сторону $k_{blocks_2} = 11$.

Аналіз отриманих значень кількості блоків, на які повинне бути розподілено закодоване повідомлення, показав, що вони мають різні значення,

що також може бути використано при побудові методу захисту передачі даних у кіберфізичній системі.

Оскільки кількість блоків k_{blocks_2}, не є цілим числом, то додамо на початку отриманої кодової послідовності (ще до початку її ділення на рівня блоки по 8 біт), 7 нулів. Таке додавання зміст самої кодової комбінації не змінює.

Після поділу кодової послідовності (2.13) на 12 блоків по 3 символи отримаємо послідовність біт (2.15):

$$\begin{array}{c} 000\ 100\ 100\ 011 \\ 010\ 001\ 010\ 111 \\ 000\ 110\ 101\ 011, \end{array} \qquad (2.15)$$

а поділ кодової послідовності (2.14) на 11 блоків по 4 символи призведе до появи послідовності біт (2.16):

$$\begin{array}{c} 00000000\ 01111000\ 00111100 \\ 10011110\ 10100000\ 00010000 \\ 00011000\ 00011001\ 11100010 \\ 00001101\ 00000111. \end{array} \qquad (2.16)$$

Для зручності переведемо отримані кодові послідовності у послідовності десяткових чисел (2.17) та (2.18) відповідно:

$$Y_1 = 0\ \ 4\ \ 4\ \ 3\ \ 2\ \ 1\ \ 2\ \ 7\ \ 0\ \ 6\ \ 5\ \ 3, \qquad (2.17)$$

$$Y_2 = 0\ \ 120\ \ 60\ \ 158\ \ 160\ \ 16\ \ 24\ \ 25\ \ 226\ \ 13\ \ 7. \qquad (2.18)$$

Проведемо аналіз моделі методу на основі використання для значення параметру n_1 рівним 8. Довжина ключа становить $L = 8$, його вміст приймемо $k_l = 1\,2\,3$, тоді значення в S-блоці згідно математичної моделі методу сформуються наступним чином (табл. 2.1):

Таблиця 2.1 – Вміст S-блоку

Позиція в S-блоці	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Значення	0	1	2	3	4	5	6	7

Формування K-блоку (табл. 2.2):

Таблиця 2.2 – Вміст K-блоку

Позиція в K-блоці	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
Значення	k_0	k_1	k_2	k_0	k_1	k_2	k_0	k_1
	1	2	3	1	2	3	1	2

Проведемо перемішування вмісту S-блоку за (2.8) (табл.2.3):

Таблиця 2.3 – Переміщування елементів в S-блоці

Номер дії	Дія, яка виконується (всі дії виконуються по модулю 8)	Нове значення i	Нове значення j
1	$j=0,\ i=0$	0	
2	$j=j+S_i+K_i=0+0+1=1$		1
3	Змінити місцями S_i та S_j, тобто S_0 та S_1		
4	$i=i+1$	1	
5	$i<8$, тому перейти до пункту 2		
2	$j=j+S_i+K_i=1+0+2=3$		3
3	Змінити місцями S_i та S_j, тобто S_1 та S_3		
4	$i=i+1$	2	
5	$i<8$, тому перейти до пункту 2		
2	$j=j+S_i+K_i=(3+2+3)\bmod 8=0$		0
3	Змінити місцями S_i та S_j, тобто S_2 та S_0		
4	$i=i+1$	3	
5	$i<8$, тому перейти до пункту 2		
2	$j=j+S_i+K_i=0+3+1=4$		4
3	Змінити місцями S_i та S_j, тобто S_3 та S_4		
4	$i=i+1$	4	
5	$i<8$, тому перейти до пункту 2		

Завершення таблиці 2.3

2	$j = j + S_i + K_i = (4+4+2) \bmod 8 = 2$		2
3	Змінити місцями S_i та S_j, тобто S_4 та S_2		
4	$i = i + 1$	5	
5	$i < 8$, тому перейти до пункту 2		
2	$j = j + S_i + K_i = (2+5+3) \bmod 8 = 2$		
3	Змінити місцями S_i та S_j, тобто S_5 та S_2		2
4	$i = i + 1$	6	
5	$i < 8$, тому перейти до пункту 2		
2	$j = j + S_i + K_i = (2+6+1) \bmod 8 = 1$		1
3	Змінити місцями S_i та S_j, тобто S_6 та S_1		
4	$i = i + 1$	7	
5	$i < 8$, тому перейти до пункту 2		
2	$j = j + S_i + K_i = (1+7+2) \bmod 8 = 2$		2
3	Змінити місцями S_i та S_j, тобто S_7 та S_2		
4	$i = i + 1$	8	
5	$i = 8$, тому завершити		

В результаті проведення дій у табл. 2.3 отримаємо оновлену таблицю значень елементів S-блоку (табл. 2.4):

Таблиця 2.4 – Оновлений вміст S-блоку

Позиція в S-блоці	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Значення	2	6	7	4	1	0	3	5

Далі відбувається обчислення елементів псевдовипадкової послідовності Z (табл. 2.5).

Таблиця 2.5 – Визначення значень елементів псевдовипадкової послідовності Z

Обчислення значень елементів	Дія, яка виконується (всі дії виконуються по модулю 8)	Нове значення i	Нове значення j	Нове значення b
z_0	$i = i + 1 = 0 + 1 = 1$	1		
	$j = j + S_i = 0 + 6 = 6$		6	
	Змінити місцями S_i та S_j, тобто S_1 та S_6			
	$b = S_i + S_j = (3 + 6) \bmod 8 = 1$			1
	$z_0 = S_b = S_1 = 1$			
z_1	$i = i + 1 = 1 + 1 = 2$	2		
	$j = j + S_i = (6 + 7) \bmod 8 = 5$		5	
	Змінити місцями S_i та S_j, тобто S_2 та S_5			
	$b = S_i + S_j = (5 + 7) \bmod 8 = 4$			4
	$z_1 = S_b = S_4 = 1$			
z_2	$i = i + 1 = 2 + 1 = 3$	3		
	$j = j + S_i = (4 + 4) \bmod 8 = 0$		0	
	Змінити місцями S_i та S_j, тобто S_3 та S_0			
	$b = S_i + S_j = (2 + 4) \bmod 8 = 6$			6
	$z_2 = S_b = S_6 = 6$			
z_3	$i = i + 1 = 3 + 1 = 4$	4		
	$j = j + S_i = (6 + 1) \bmod 8 = 7$		7	
	Змінити місцями S_i та S_j, тобто S_4 та S_7			
	$b = S_i + S_j = (5 + 1) \bmod 8 = 6$			6
	$z_2 = S_b = S_6 = 6$			
z_4	$i = i + 1 = 4 + 1 = 5$	5		
	$j = j + S_i = (7 + 7) \bmod 8 = 6$		6	
	Змінити місцями S_i та S_j, тобто S_5 та S_6			
	$b = S_i + S_j = (6 + 7) \bmod 8 = 5$			5
	$z_2 = S_b = S_5 = 6$			

Завершення таблиці 2.5

z_5	$i = i + 1 = 5 + 1 = 6$	6		
	$j = j + S_i = (6 + 7) \bmod 8 = 5$		5	
	Змінити місцями S_i та S_j, тобто S_6 та S_5			
	$b = S_i + S_j = (6 + 7) \bmod 8 = 5$			5
	$z_2 = S_b = S_5 = 7$			
z_6	$i = i + 1 = 6 + 1 = 7$	7		
	$j = j + S_i = (5 + 1) \bmod 8 = 6$		6	
	Змінити місцями S_i та S_j, тобто S_7 та S_6			
	$b = S_i + S_j = (6 + 1) \bmod 8 = 7$			7
	$z_2 = S_b = S_7 = 7$			
z_7	$i = i + 1 = (7 + 1) \bmod 8 = 0$	0		
	$j = j + S_i = (6 + 4) \bmod 8 = 2$		2	
	Змінити місцями S_i та S_j, тобто S_0 та S_2			
	$b = S_i + S_j = (6 + 1) \bmod 8 = 7$			7
	$z_2 = S_b = S_7 = 6$			

Запишемо отриману послідовність:

$$z_t = 1 \quad 1 \quad 6 \quad 6 \quad 6 \quad 7 \quad 7 \quad 6.$$

Отримана послідовність явно вказує на те, що розмір S-блоку, розмір ключа були обрані досить замалими. Тому можна спостерігати багато повторень в утвореній послідовності.

Із використанням отриманої послідовності z_t проведемо побітове шифрування кодової послідовності $Y=y_0y_1\ldots y_{kblocks}$ із використання операції суми за модулем 2 щодо послідовності Y (2.17) та Z (табл. 2.6):

Таблиця 2.6 – Шифрування даних

Y	0	4	4	3	2	1	2	7	0	6	5	3
	000	100	100	011	010	001	010	111	000	110	101	011
Z	1	1	6	6	6	7	7	6	1	1	6	6
	001	001	110	110	110	111	111	110	001	001	110	110
$H=Y \oplus Z$	001	101	010	101	100	110	101	001	001	111	011	101
	1	5	2	5	4	6	5	1	1	7	3	5

Отримана кодова послідовність має вигляд (2.19):

$$H= 1 \quad 5 \quad 2 \quad 5 \quad 4 \quad 6 \quad 5 \quad 1 \quad 1 \quad 7 \quad 3 \quad 5. \qquad (2.19)$$

Реалізація процесу шифрування із використанням методу словникового стиснення LZW для захисту передачі даних у кіберфізичній системі дозволяє отримати зашифроване повідомлення, статистичні властивості якого відмінні від статистичних властивостей вхідного тексту.

2.3 Висновки

У другому розділі магістерської роботи було розроблено математичну модель методу захисту передачі даних у кіберфізичних системах, що оснований на використанні стиснення інформації та потокового шифрування даних. Встановлено, що використання методу кодування (стиснення) перед початком шифрування є можливим. Математична модель методу захисту передачі даних у кіберфізичній системі описує наступні процеси:

- процесу отримання вхідної послідовності та представлення її у вигляді десяткових чисел, кількість яких визначається алфавітом кодування;
- процесу кодування отриманої послідовності та представлення її у двійковому вигляді;
- процес суцільного суміщення блоків отриманих двійкових повідомлень та подальшим їх розподілом на рівні блоки довжина яких рівна довжині елементів у S-блоці;

- процес отримання ключової послідовності для шифрування вхідного потоку біт;
- процес шифрування потоковим методом.

Використання запропонованої математичної моделі при побудові методу захисту передачі даних у кіберфізичній системі дає можливість отримати чіткий набір опису необхідних процесів для створення зашифрованого повідомлення.

Встановлено, що запропонований метод у порівнянні із звичайним потоковим шифруванням, дає можливість змінити статистичний розподіл появи символів у вхідній послідовності. Також отримано результат, який показав, що використання різної кількості символів у елементах S-блоку, також впливає на статистичний розподіл символів та їх кількість у послідовності біт, яка підготовлена для шифрування. Цей результат також може бути використаний для підвищення захисту передачі даних у кіберфізичних системах.

3 МЕТОД ФОРМУВАННЯ ПОТОКОВОГО ШИФРУ RC4 ІЗ ВИКОРИСТАННЯМ СЛОВНИКОВОГО МЕТОДУ СТИСНЕННЯ ІНФОРМАЦІЇ ДЛЯ ПІДВИЩЕННЯ ЗАХИСТУ ПЕРЕДАЧІ ДАНИХ У КІБЕРФІЗИЧНИХ СИСТЕМАХ

3.1 Аналіз даних та висування вимог до системи захисту передачі даних

Кіберфізична система – це комп’ютерна система, в якій фізичні та програмні компоненти тісно пов’язані між собою. Повноцінна кіберфізична система проєктується у вигляді мережі, в якій елементи знаходяться елементи у певній взаємодії, а не як окремі елементи.

Окремим видом КФС є мобільні КФС. Такі КФС включають мобільну робототехніку та електроніку, які можуть переноситись людьми або пересуватись самостійно залежно від спроєктованої системи. А отже питання щодо проєктування провідної системи зв’язку між елементами КФС не є можливим. Потрібно звернути увагу на бездротовий зв’язок, наприклад, із використанням Wi-Fi.

Перевагами бездротового зв’язку є можливість організувати зв’язок між елементами КФС швидко та просто. Вони швидко конструюються та реструктуруються. Оскільки передача даних відбувається радіохвилями, на відміну від провідного зв’язку, в якій передача відбувається електричними сигналами, то радіохвилі достатньо легко можна перехопити при наявності певного обладнання.

Радіоперехоплення сигналу між елементами КФС дасть змогу зловмисникам керувати цією системою А це в свою чергу може призвести до поломок, виходу з ладу окремих елементів КФС або повністю до виходу з ладу системи, керувати виконанням дій, які можуть нанести шкоди оточуючому середовищу або людині, що може призвести до великих втрат власнику цієї системи.

Отже, найбільш гостро при проєктуванні бездротових мереж є забезпечення їх захисту від доступу, використання, виявлення, перекручування, знищення, модифікації даних. Інформаційна безпека повинна забезпечувати доступність, цілісність, конфіденційність даних, які передаються. Сучасні методи автентифікації та шифрування у сучасних бездротових мережах є досить недосконалими.

На відміну від провідних мереж, в бездротових мережах можуть використовуватись зовсім інші види хакерських нападів. Тому і механізм захисту повинні відрізнятись.

Виділяють чотири основні категорії атак [22]:

1) атаки доступу;

2) атаки модифікації;

3) атаки на відмову в обслуговуванні;

4) атаки на відмову від зобов’язань.

Атака доступу – направлена зловмисником на отримання доступу до даних, до перегляду яких у нього немає прав, з метою порушення конфіденційності інформації.

Атака модифікації – має на меті зміну даних, тобто на порушення її цілісності. Виділяють три види модифікації:

1. заміна даних – направлена на заміну даних як у секретній так і загальнодоступній частині інформації;
2. додавання даних – додавання нових даних;
3. зміна даних – переміщення існуючих даних.

Атаки на відмову в обслуговуванні (Denial-of-service, DoS) – направлена на створення заборони користування системою легальному користувачу. У результаті створення DoS атаки не має доступу до комп’ютерної системи. DoS атаки направлені на застосунки, які оперують з даними КФС. Тобто намагаються створити неможливим розв’язання задач, на виконання яких націлений відповідний застосунок.

Атака, що направлена на відмову від зобов’язань, має за мету унеможливити ідентифікацію даних, або надати невірну інформацію про реальну подію або транзакцію. До даного виду атак відносять: маскарад, заперечення події, DoS проти мережі. Зменшити ризики можливо із використанням наступних простих кроків:

- змінити пароль, який встановлюється за замовчування, на пароль адміністратора у своєму бездротовому пристрої. Пароль повинне бути складним;

- відключити трансляцію ідентифікатора мережі або змінити його, не використовувати імена, які легко вгадати;

- використовувати шифрування трафіка: WPA протокол або WEP-шифр

Найбільш розповсюджена загроза у бездротових мережах – перехоплення сигналу та його дешифрування. При навіть можливому перехваті повідомлень, потрібно забезпечити підвищення надійності захисту, який стосується неможливості прочитати інформацію – тобто забезпечити шифрування даних.

Таким алгоритмом може бути RC4. Він широко використовується в комп’ютерних мережах в системах захисту даних. Може використовуватись, наприклад, в протоколах SSL та TLS, алгоритмах забезпечення безпеки бездротових мереж WEP, WPA.

Всі потокові шифри, зокрема і алгоритм RC4, будується на основі використанні випадкових бітів, які генеруються генератором випадкових бітів. На вхід генератора подається деякий ключ, а на йог виході формується випадкова послідовність. Але встановлено, що у такої послідовності існує певна залежність, тобто з певною періодичністю відбувається повторення символів. Тому послідовність є псевдовипадковою, а генератор – генератором псевдовипадкових біт.

Для створення реально випадкових чисел (бітів) вони мають ґрунтуватись на фізичному джерелі випадкової інформації, яку не має можливості передбачити: напівпровідникові прилади шуму, інтервали між перериваннями

пристроїв або натискання клавіш, молодші біти оцифрованого звуку. Але, якщо немає можливості використовувати такі джерела, то використовують джерела псевдовипадкових бітів.

Основними перевагами потокового шифру RC4 є висока швидкість роботи та змінний розмір ключа, а недоліками є використання пов'язаних або не випадкових ключів, і якщо використовувати один ключовий потік двічі.

Отже, основним недоліком методів шифрування, що основані на використанні потокового шифру, є формування випадкової послідовності. Наприклад, у роботі [23], пропонується, щоб генератор працював на основі лінійного рівняння з конкретним простим числом. Замість заповнення змінної стану (S) значеннями від 1 до 256, вона буде заповнена значеннями, які обчислюються за наступним рівнянням:

$$Y_{n+1} = (a \cdot Y_n + b) \bmod p,$$

де a - множник; b - приріст; Y_n - початкове значення; а p – велике просте число у якості модуля.

В роботі [24] пропонується використання зсувного ключа

$$k(t) = k(t) + \mu,$$

де μ - певний зсув.

3.2 Розробка алгоритму формування потокового шифру RC4 із використанням словникового методу стиснення інформації

В алгоритмі формування потокового шифру із використанням словникового методу стиснення інформації можна виділити наступні кроки:

- отримання вхідного текстового потоку $G = g_1 g_2 g_3 ... g_m$, що створений з деякого алфавіту $\Sigma = \{a_1, a_2, a_3, ..., a_q\}$. на основі отриманих даних від елементів кіберфізичної системи;

- формування вихідної текстової послідовності у вигляді двійкових символів використавши для перетворення таблицю ASCII;

- кодування (стиснення) отриманої вхідної комбінації із використанням словникового методу стиснення LZW – отримання послідовності 9-бітних послідовностей $c(G) = p_1 p_2 p_3 ... p_v$;

- об'єднання отриманої послідовності $c(G) = p_1 p_2 p_3 ... p_v$ у нерозривну послідовність біт довжиною $l_{c(S)}$;

- розділення бітової послідовності довжиною $l_{c(S)}$ на блоки по $l_{s[k]}$, кількість яких становить k_{blocks}. За необхідності, у випадку якщо $l_{c(S)}$ не є кратним $l_{s[k]}$, потрібно додати додаткові нульові біти на початку $c(G) = p_1 p_2 p_3 ... p_v$. Результатом є отримання послідовності $Y=y_0 y_1 \ldots y_{kblocks}$;

- формування підстановки - S-блоку;

- формування ключового потоку $Z=z_0 z_1 \ldots z_{n-1}$;

- шифрування послідовності $Y=y_0 y_1 \ldots y_{kblocks}$ ключовим потоком $Z=z_0 z_1 \ldots z_{n-1}$ із використанням операції сума за модулем 2.

Опишемо алгоритми основних етапів методу захисту даних кіберфізичної системи із використанням методу кодування (стиснення) словниковим алгоритмом LZW та шифрування потоковим шифруванням із використанням методу RC4. Згідно розробленої моделі захисту даних на першому етапі відбувається кодування даних. Розглянемо алгоритм перетворення потоку $G = g_1 g_2 g_3 ... g_m$ у потік $c(G) = p_1 p_2 p_3 ... p_v$.

Алгоритм формування кодової послідовності $c(G) = p_1 p_2 p_3 ... p_v$:

1) ініціалізація словника усіма можливими символами алфавіту $\Sigma = \{a_1, a_2, a_3, ..., a_q\}$ – {A '..' Z ',' a '..' z ',' 0 '..' 9 '};

2) надходження першого символу кодової послідовності p_1;

3) надходження наступного символу кодової послідовності p_2;

4) перевірка на наявність рядка $(p_1 + p_2)$:

a) якщо такий рядок виявлено, то відбувається об'єднання p_1 та p_2 у новий рядок;

b) якщо ні, то:

- вивести рядок коду для проведення заміни p_1;

- додати рядок $(p_1 + p_2)$ в словник та вказати номер/код для подальшого використання в словнику для цього рядка;

- присвоїти вхідній фразі p_2;

5) перевірка на наявність наступного символу в потоці символів:

a) якщо так, то повернутись до кроку 2;

b) якщо ні, то вихідний код, що замінює рядок p_i завершує процес.

Алгоритм методу кодування LZW наведено на рис. 3.1 [25].

Одержану послідовність потрібно «підготувати» до шифрування даних методом RC4. Алгоритм перетворення послідовності 9-бітних символів у 8-бітні складається з наступних кроків:

1) перетворити отриману послідовність у нерозривну послідовність біт;

2) визначити довжину утвореної послідовності $l_{c(S)}$;

3) перевірити, чи значення довжини $l_{c(S)}$ є кратним 8. Якщо ні – додати на початку рядка від 1 до 7 нулів таким чином, щоб значення довжини сформованого рядка було кратним 8;

4) сформований рядок бітів потрібно поділити на блоки бітів довжиною $l_{s[k]}$ біт;

5) отримати послідовність $Y=y_0y_1\ldots y_{kblocks}$, яка подаватиметься для проведення шифрування.

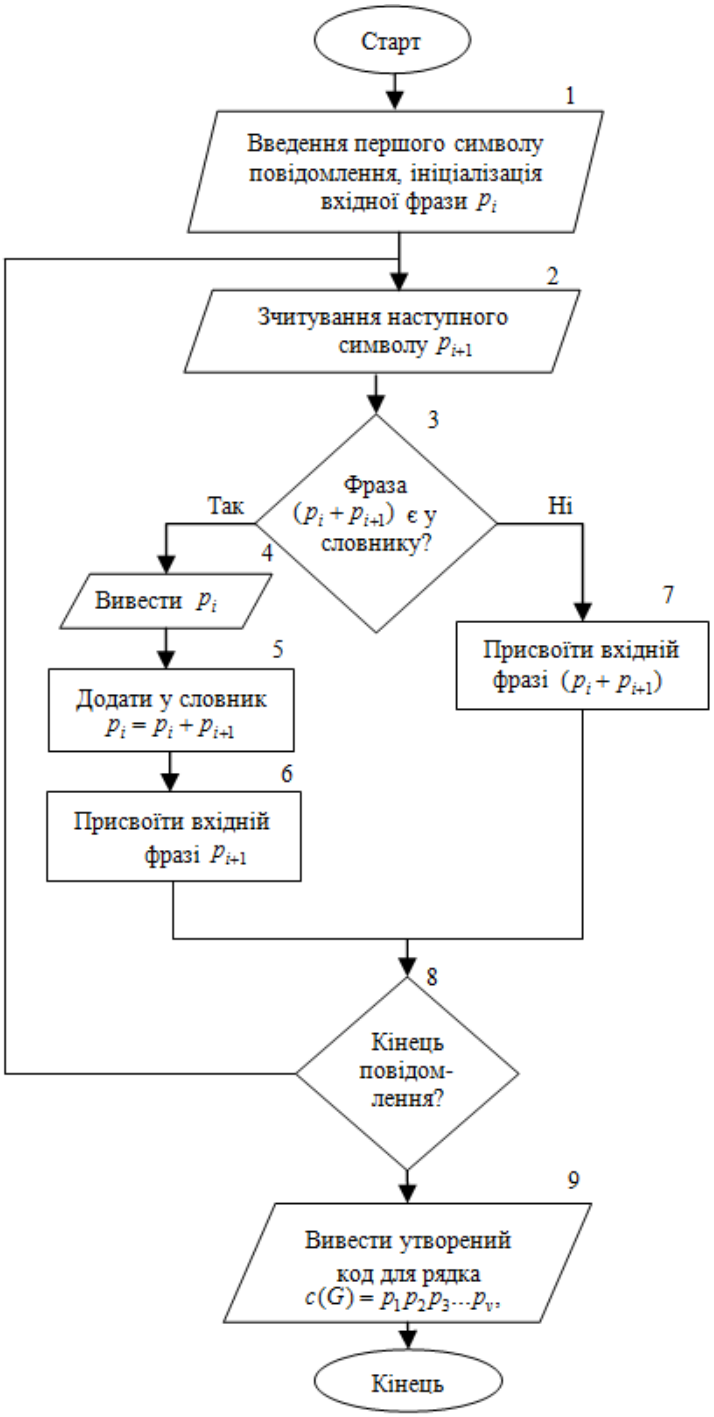


Рисунок 3.1 – Алгоритм кодування LZW [25]

Після реалізації кодування до отриманої кодової послідовності $Y=y_0y_1\ldots y_{kblocks}$ для можливості підвищення захисту передачі даних у кіберфізичній системі застосовується потоковий алгоритм шифрування даних RC4. Алгоритм реалізації методу шифрування даних RC4 [26]:

1) отримати послідовність $Y=y_0y_1\ldots y_{kblocks}$ та обраний ключ k_l;

2) створити два рядкових масиви S-блок та K-блок;

3) ініціалізувати масив S-блоку з числами від 0 до 255;

4) заповнити масив K обраним ключем. Якщо довжина ключа менша за 256 символів, то заповнення масиву ключем продовжують записуючи ключову послідовність одну за одною доки не заповниться масив 256-ма символами;

5) провести перестановки у масиві S-блоку залежно від масиву K-блоку;

6) знову провести перестановки у масиві S-блоку, щоб згенерувати остаточну ключову послідовність;

7) виконати операцію сума за модулем 2 відносно сформованого ключового потоку $Z=z_0z_1\ldots z_{n-1}$ та даними $Y=y_0y_1\ldots y_{kblocks}$, які потрібно зашифрувати. У результаті отримується послідовність $H = h_1h_2..h_{k_{blocks}}$.

Алгоритм методу захисту передачі даних зображено на рис. 3.2.

На рис. 3.2 із зображеної схему алгоритму методу захисту передачі даних у кіберфізичній системі із використанням словникового методу кодування LZW та шифрування даних RC4 можна виділити наступні основні етапи виконання:

- блоки 1-3 – введення вхідних даних, які необхідні для реалізації методу захисту передачі даних;
- блоки 4-6 – відповідають за кодування кодової послідовності та отримання закодованої послідовності $c(G) = p_1 p_2 p_3 ... p_v$;
- блоки 7-11 – шифрування даних
- блок 12 - отримання вихідної послідовності $H = h_1h_2..h_{k_{blocks}}$.
- блок 13 – передача зашифрованого тексту H.

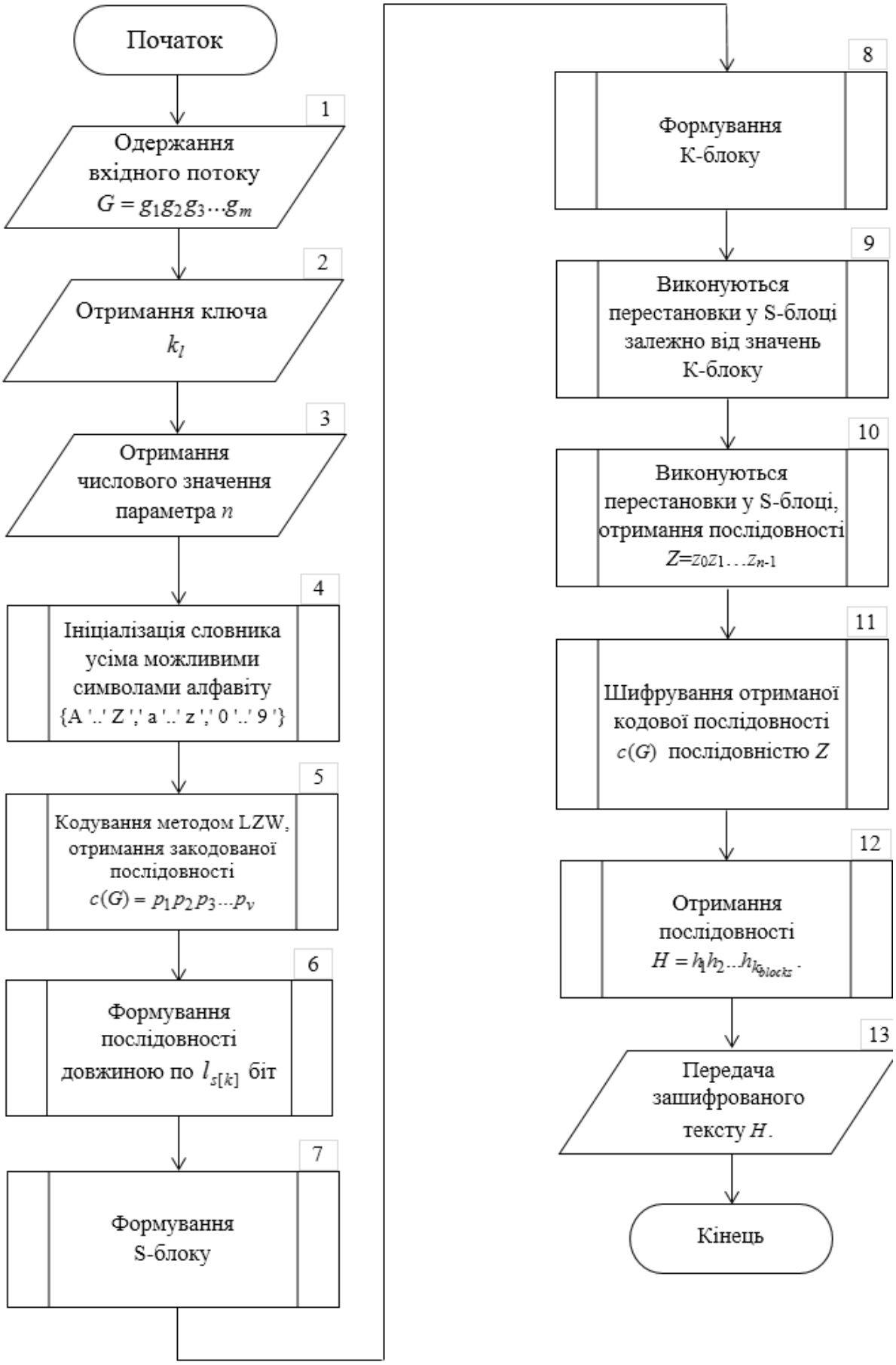


Рисунок 3.2 – Алгоритм методу захисту передачі даних

3.3 Аналіз ефективності захисту передачі даних у кіберфізичній системі із використання запропонованого методу

Передача даних у кіберфізичній системі відбувається із використанням бездротового зв'язку, а саме пропонується використовувати Wi-fi. Стандарт IEEE 802.11 передбачає використання алгоритмів захисту WEP (Wired Equivalent Privacy) [27] або WPA (Wi-Fi Protected Access) [28], в яких використовується шифрування RC4. На рис. 3.3 та 3.4 зображено алгоритми отримання ключів:

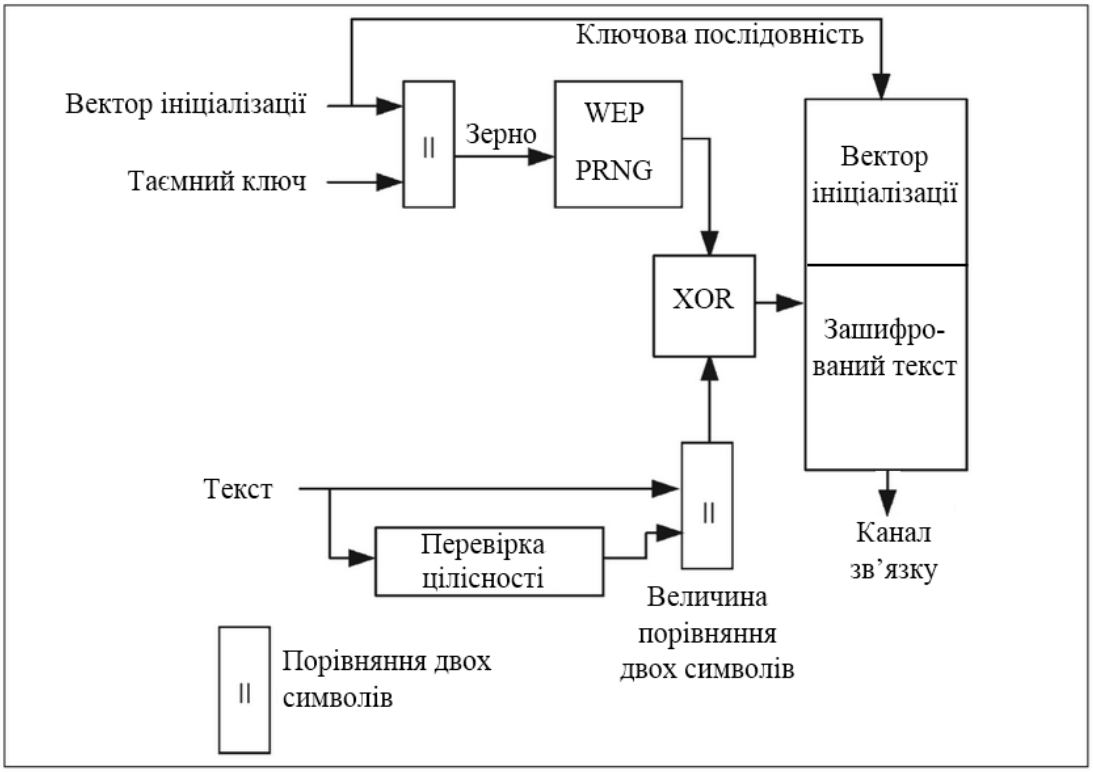


Рисунок 3.3 – Алгоритм шифрування в WEP [27]

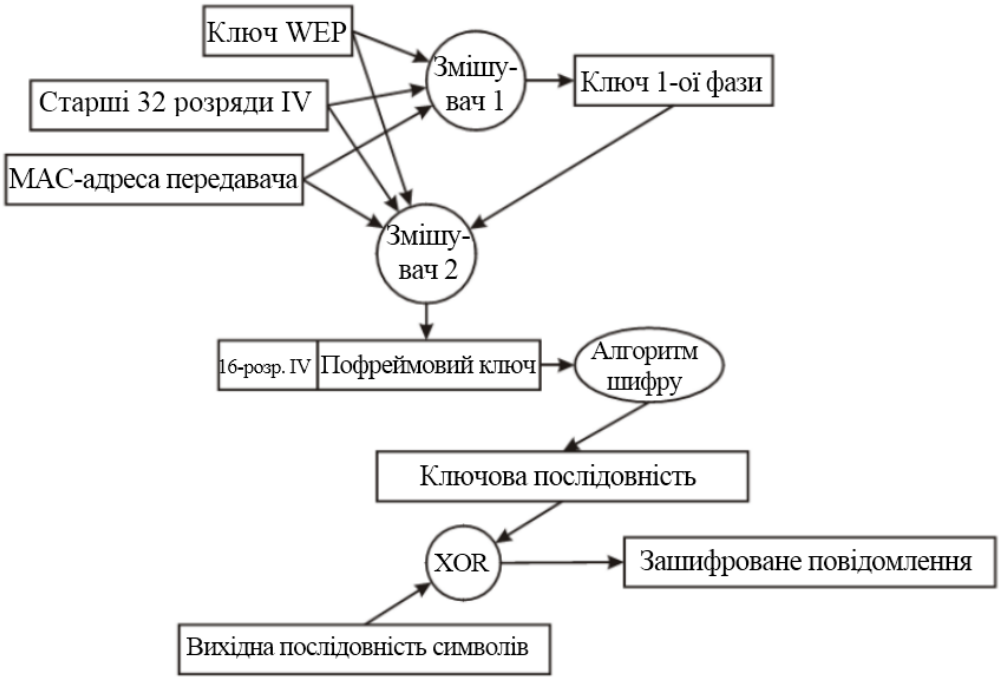


Рисунок 3.4 – Алгоритм шифрування в WPA [28]

Алгоритм WPA – це удосконалений алгоритм WEP, в який IEEE запропонувало ввести тимчасовий протокол цілісності ключа – TKIP.

Секретний ключ поєднується з вектором ініціалізації. Вихідна послідовність є вхідною інформацією для псевдовипадкового генератора ключів, який використовує шифрування RC4, для отримання октетів ключової послідовності. Кількість згенерованих октетів ключової послідовності рівна кількості октетів, на яку ділиться послідовність, що буде зашифрована.

Однією з відомих проблем потокового шифрування є те, що якщо всі октети вхідної послідовності, шифруються одним і тим самим вектором ініціалізації, то зашифрована послідовність, яка передається каналом зв’язку, може бути перехоплена, а вектор ініціалізації легко визначений, оскільки із вхідним повідомленням використовується операція сума за модулем 2. До того ж, якщо дані є досить довгими, а вектор ініціалізації є один і той же, то проблема визначення вектора ініціалізації значно спрощується.

Зловмиснику достатньо надіслати тестовий пакет даних, а на виході отримати зашифровану послідовність. При прийманні цієї послідовності

зловмисник знає і дані, які були зашифровані, і зашифровану послідовність. Використавши повторно до цієї послідовності операції сума за модулем 2, зловмисник легко отримує ключову послідовність для шифрування. Відповідно, наступні пакети даних, які будуть надсилатись каналом зв'язку, будуть легко дешифровані.

Запропонований метод у даній магістерській роботі передбачає введення додаткового рівня захисту. Використання запропонованого методу дозволяє:

- змінює частотний розподіл вхідних даних,

- змінює кількість вихідних (зашифрованих) октет, що унеможливлює атаку шляхом надсилання тестових даних, та застосування до отриманої послідовності операції сума за модулем 2, а відповідно і дешифрувати їх.

Проведемо аналіз ефективності захисту передачі даних для отриманої послідовності (2.10) ентропійним методом.

Для цього визначимо ентропію вхідного повідомлення (2.10):

$$H = \sum_{i=0}^{a} p_i \log\ p_i , \qquad (3.1)$$

де H_G - ентропія вхідної послідовності;

a – кількість різноманітних символів у вхідній послідовності;

p_i - ймовірність появи символів у вхідній послідовності.

Визначимо значення ймовірностей символів у вхідній послідовності (2.10) та запишемо їх у табл. 3.1. Аналіз вхідної послідовності символів показав, що в ній є 3 символи, які з'являються з різною ймовірністю:

Таблиця 3.1 – Статичний аналіз вхідної послідовності (2.10)

Символ	Кількість входжень у послідовності	Ймовірність появи
x	9	0,56
y	5	0,31
z	2	0,13

Ентропія за (3.1):

$$H = p_x \log p_x + p_y \log p_y + p_z \log p_z =$$
$$= 0{,}56 \log 0{,}56 + 0{,}31 \log 0{,}31 + 0{,}13 \log 0{,}13 = 1{,}37 (\text{біт/симв.})$$

Побудуємо гістограму, за допомогою якої зобразимо частотний розподіл символів у вхідній послідовності (рис. 3.5)

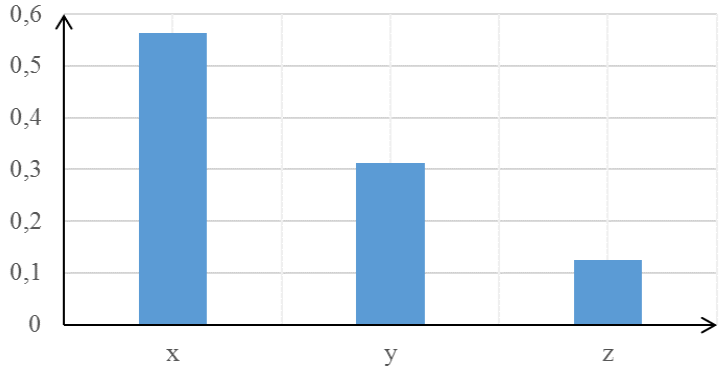


Рисунок 3.5 – Гістограма частотного розподілу символів у вхідній послідовності

Об’єм даних вхідної послідовності становить:

$$V = kn, \tag{3.2}$$

де V - об’єм вхідних даних, біт;

k - кількість символів у вхідній послідовності;

n - розрядність символу, біт.

Кількість символів у вхідній послідовності становить 16, всі символи кодуються за допомогою таблиці ASCII октетами, тобто $n = 8$. Об’єм становить:

$$V = 16 \cdot 8 = 128 \text{ біт}.$$

Визначимо ентропію та об’єм інформації для закодованого тексту $c(G)_2$ (2.12). Розподіл ймовірностей наведемо в табл. 3.2

Таблиця 3.2 – Статистичний розподіл символів у закодованому тексті

Символ	Кількість входжень у послідовності	Ймовірність появи
0,120	2	0,22(2)
0,121	1	0,11(1)
0,122	1	0,11(1)
256	1	0,11(1)
257	1	0,11(1)
259	1	0,11(1)
262	1	0,11(1)
263	1	0,11(1)

Ентропія закодованої послідовності становить:

$$H = p_{0,120} \log\, p_{0,120} + p_{0,121} \log\, p_{0,121} + p_{0,122} \log\, p_{0,122} + p_{256} \log\, p_{256} +$$
$$+ p_{257} \log\, p_{257} + p_{259} \log\, p_{259} + p_{0,120} \log\, p_{0,120} + p_{262} \log\, p_{262} + p_{263} \log\, p_{263} =$$
$$= 0,22 \log 0,22 + 7 \cdot 0,11 \log 0,11 = 2,93 \text{ (біт/симв.)}$$

Різне значення ентропій показує, що частотний розподіл символів у вхідній послідовності та у закодованій послідовності значно різняться.

При створенні суцільної послідовності біт з отриманих закодованих символів і їх поділі на октети, отримаємо наступний розподіл ймовірності символів (табл. 3.3)

Таблиця 3.3 – Статистичний аналіз послідовності символів після їх поділу на октети

Октет	Порядковий номер символу у таблиці ASCII	Кількість входжень	Ймовірність появи
00000000	0	1	0,091
00000111	7	1	0,091
00001101	13	1	0,091
00010000	16	1	0,091
00011000	17	1	0,091
00011001	18	1	0,091
00111100	60	1	0,091
01111000	120	1	0,091
10011110	158	1	0,091
10100000	160	1	0,091
11100010	226	1	0,091

$$H = p_0 \log p_0 + p_7 \log p_7 + p_{13} \log p_{13} + p_{16} \log p_{16} + p_{17} \log p_{17} + p_{18} \log p_{18} +$$
$$+ p_{60} \log p_{60} + p_{120} \log p_{120} + p_{158} \log p_{158} + p_{160} \log p_{160} + p_{226} \log p_{226} =$$
$$= 11 \cdot 0{,}091 \log 0{,}091 = 3{,}46 \text{ (біт/симв.)}$$

На рис. 3.6 наведено гістограму розподілу ймовірностей появи створених октетів для шифрування.

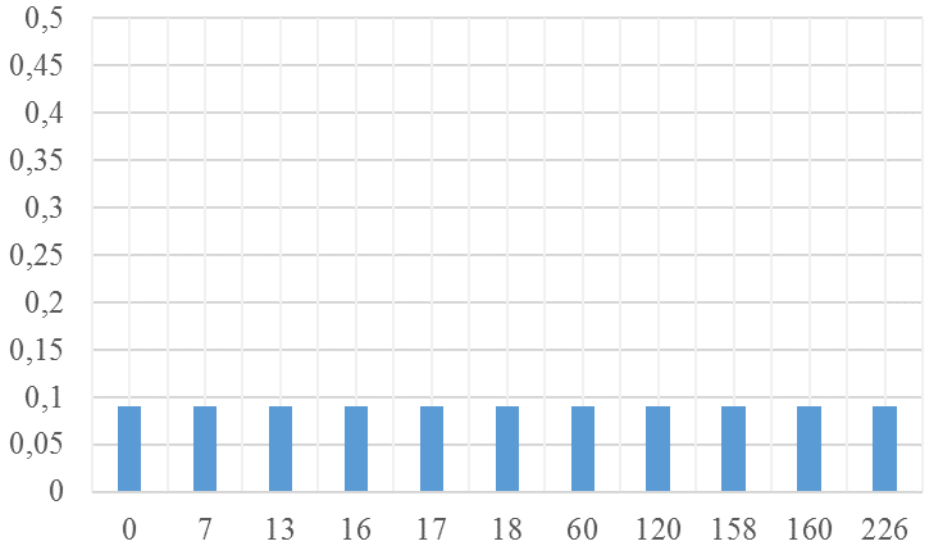


Рисунок 3.6 – Гістограма розподілу ймовірностей даних, що підлягають шифруванню

Кількість символів у отриманій кодовій послідовності становить 9, їх розрядність - 9 біт. Об’єм даних за (3.2):

$$V_1 = 9 \cdot 9 = 81 \text{ біт.}$$

Після їх розбиття на октети символів стало 11, а їх розрядність – 8 біт. Об’єм даних за (3.2) у закодованій послідовності становить:

$$V_2 = 11 \cdot 8 = 88 \text{ біт.}$$

Об’єм збільшився штучно за рахунок додавання 7 біт до початку кодової послідовності для забезпечення кратності довжини повідомлення восьми, тобто, щоб була змога створити октети.

Оскільки при шифруванні використовується операція сума за модулем 2, то довжина отриманої зашифрованої послідовності біт та об’єм не змінюються. Для порівняння отриманих результатів зведемо отримані результати у табл. 3.4:

Таблиця 3.4 – Зведений статистичний аналіз символів у вхідній та у закодованій послідовності

	Вхідна послідовність	Закодована послідовність	Кількість октет, на які розподілено закодовану послідовність
Кількість символів алфавіту, що зустрічаються	3	8	11
Ентропія, біт/симв.	1,37	2,93	3,46
Довжина, симв.	16	9	11
Об’єм, біт	128	81	88

З отриманих у таблиці 3.4 даних можна відмітити наступне. У всіх трьох послідовностях різні ентропії, і до того ж вона постійно збільшується починаючи від вхідної послідовності до закодованої і завершуючи коли вся послідовність розбивається на октети, а ймовірності появи символів навпаки зменшуються. Тобто передбачення кодової послідовності шляхом частотного аналізу шифрованої послідовності значно ускладнює процес дешифрування даних зловмисником. Об’єм вхідних даних та даних, що отримані після кодування та розбиття закодованої послідовності на октети, відрізняється від початкового значення, що унеможливлює дешифрування тестової послідовності, яку надсилає зловмисник.

Таким чином використання додаткового кодування методом LZW, дозволяє досягнути захисту передачі даних в кіберфізичній системі.

На рис. 3.7 та рис. 3.8 зображено запропоновану схему використання блоку кодування даних.

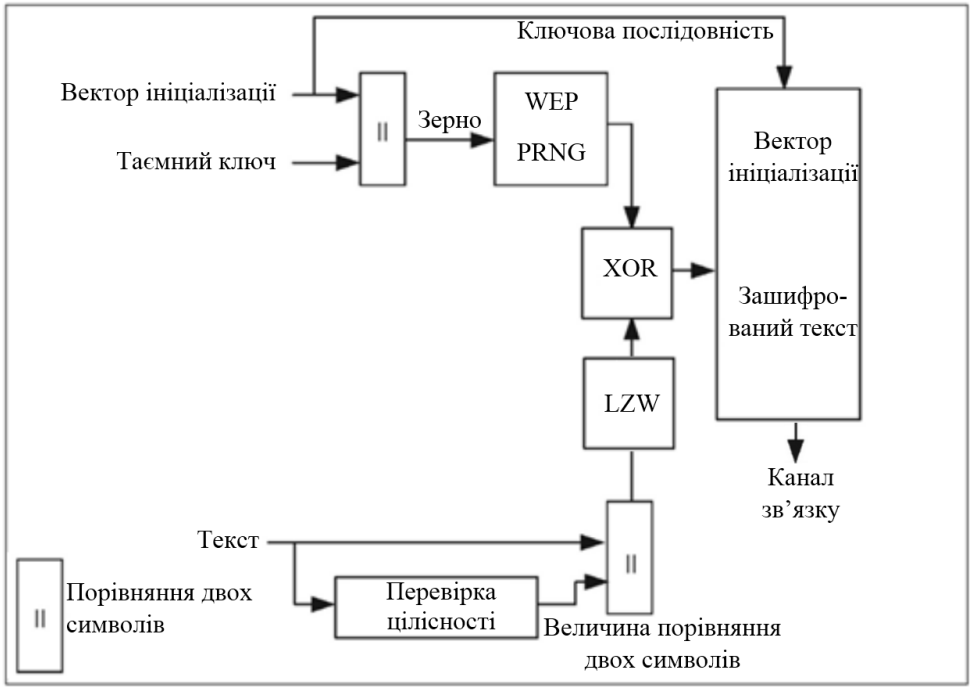


Рисунок 3.7 – Схема використання запропонованого методу захисту передачі даних у алгоритмі WEP

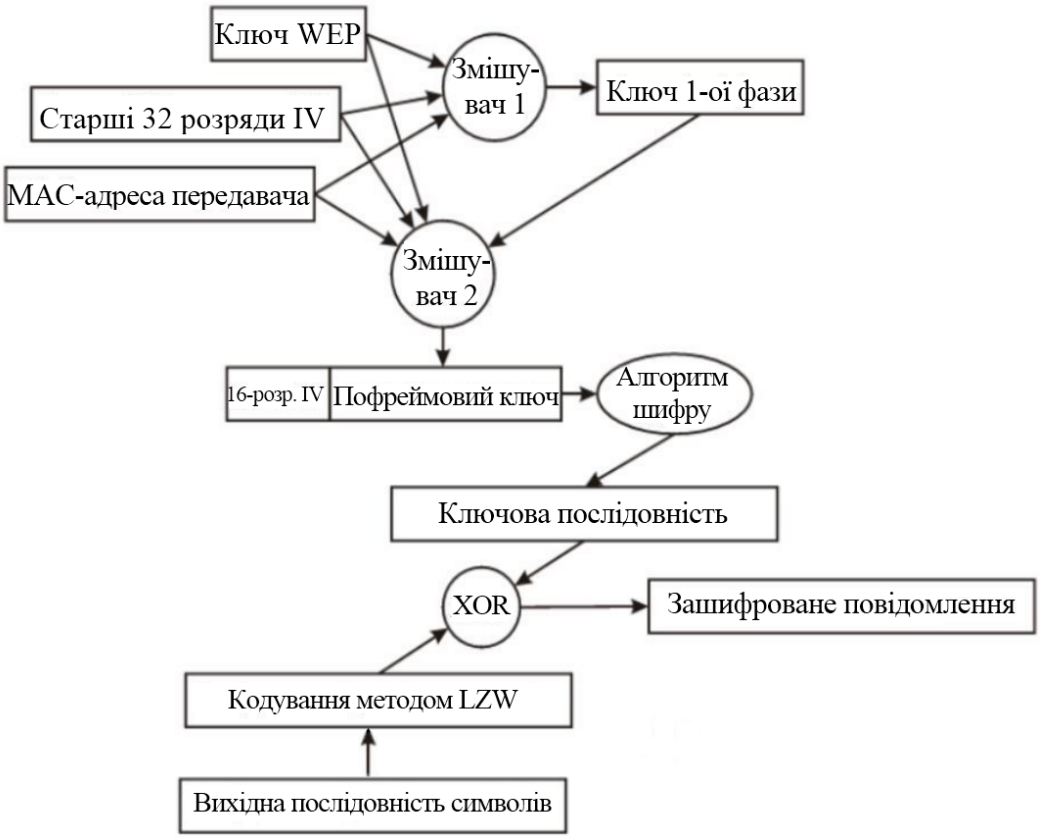


Рисунок 3.8 – Схема використання запропонованого методу захисту передачі даних у алгоритмі WPA

На рисунках показано, що запропоновано використовувати словникового методу стиснення при підготовці вихідної кодової послідовності символів до шифрування, що змінить статистичні характеристики вхідної послідовності.

3.4 Висновки

У третьому розділі було досягнуто наступних результатів:

- проведено аналіз даних та сформульовано основні вимоги до системи захисту передачі даних у кіберфізичній системі: дані, що підлягають передачі бездротовим каналом зв’язку, повинні володіти властивостями відмінними від вхідних даних, тобто повинно бути здійснено перетворення, яке змінить статистичні характеристики вхідних даних;

- розроблено основні етапи роботи методу захисту передачі даних у кіберфізичній системі;
- розроблено алгоритм методу захисту передачі даних бездротовим зв’язком у кіберфізичній системі;
- розроблено алгоритм оцінки ефективності впровадження запропонованого методу захисту передачі даних у кіберфізичній системі.

4 ДОСЛІДЖЕННЯ МЕТОДУ ЗАХИСТУ ПЕРЕДАЧІ ДАНИХ У КІБЕРФІЗИЧНИХ СИСТЕМАХ

4.1 Вибір програмних засобів дослідження побудованої моделі методу

Для апробації створеного методу потокового шифрування даних із попереднім їх стисненням, було обрано використати прототипно-орієнтовану, мультипарадигмену скриптову мову програмування JavaScript. Створення програмного продукту на мові програмування JavaScript дозволяє створити зручний інтерфейс для проведення експериментальної частини проєкту. Це дозволить без значних затрат часу провести кодування (стиснення) даних та їх потокове шифрування, визначити статичні (частотні) характеристики даних, що отриманні від елементів кіберфізичної системи, до кодування даних, під час кодування даних (стиснення) та у результаті їх потокового шифрування.

Програма містить три програмних блоки (рис. 4.1):

- блок введення даних для кодування (стиснення) та шифрування;
- блок кодування (стиснення) даних;
- блок шифрування даних методами RC4, RC4+, RC4A.

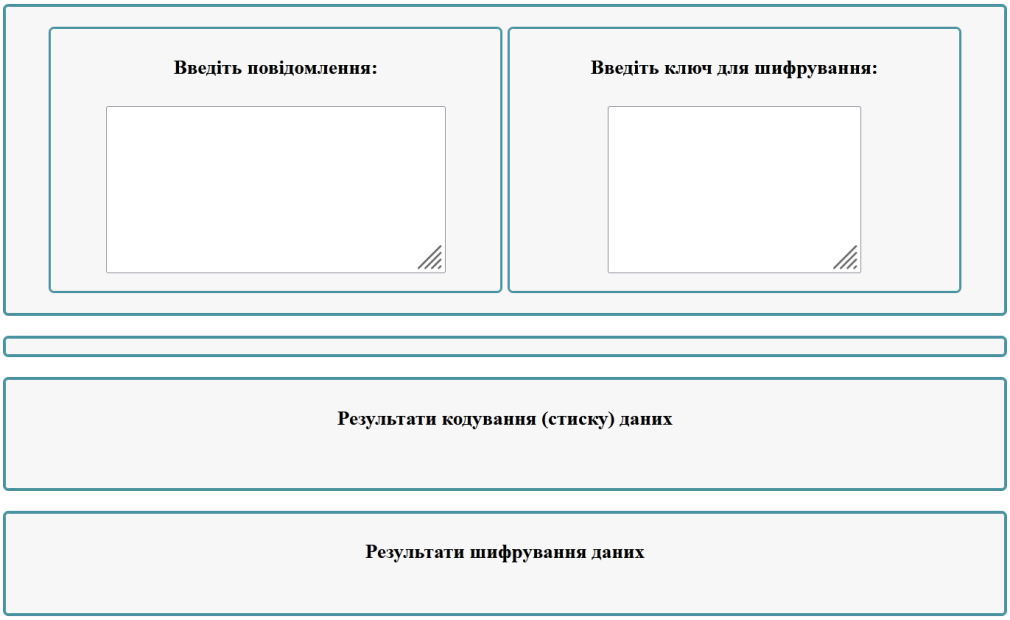


Рисунок 4.1 – Головна форма проєкту

Всі необхідні дослідження можуть проводитись на одній сторінці. На сторінці розташовані усі необхідні елементи для проведення експериментальних досліджень.

Сторінка містить текстове поле, в яке потрібно вводити дані, що отримані від елементів кіберфізичної системи. Кодування даних відбувається одразу ж під час введення даних. Це дозволяє впевнитись у тому, що відбувається кодування та стиснення даних. Для тестування введемо, наприклад, блок даних з 10 однакових символів: kkkkkkkkkk (рис. 4.2)

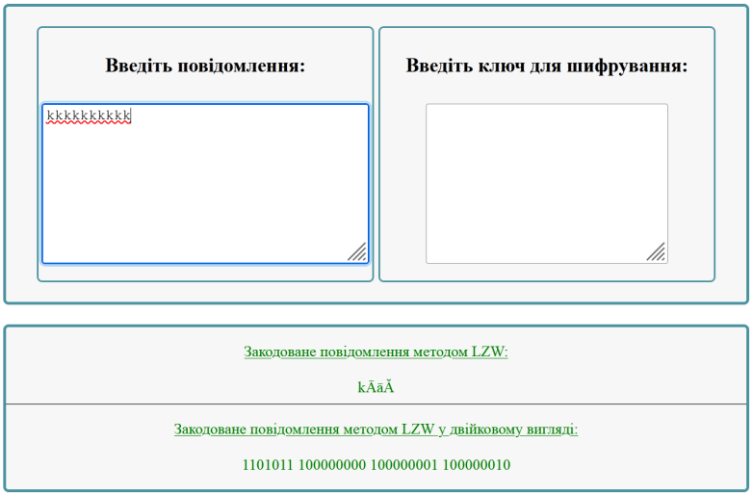


Рисунок 4.2 - Кодування (стсинення) даних

Як видно з рис. 4.2 закодовані дані вже мають меншу довжину, а нижче можна побачити закодовані (стиснені) дані у двійковому вигляді.

Для кодування методом LZW створено функцію lzw_encode.js, що підключена до головного файлу проєкту та обробляє дані з метою їх стиснення:

```
function lzwEncode(data) {
    let dictionary = {};
    let input_data = (data + "").split("");
    let output_data = [];
    let char;
    let word = input_data[0];
    let code = 256;
    for (let i=1; i<input_data.length; i++) {
        char=input_data[i];
```

```
        if (dictionary[word + char] != null) {
            word += char;
        }else {
            output_data.push(word.length > 1 ? dictionary[word] : word.charCodeAt(0));
            dictionary[word + char] = code;
            code++;
            word=char;
        }
    }
    output_data.push(word.length > 1 ? dictionary[word] : word.charCodeAt(0));
    for (let i=0; i<output_data.length; i++) {
        output_data[i] = String.fromCharCode(output_data[i]);
    }
    return output_data.join("");
}
```

Згідно методу для реалізації кодування (стиснення) потрібно використати словник. В цій функції оголошується та ініціюється об’єкт, в який і буде оновлюватись та доповнюватись словник.

Після натискання клавіші Enter в текстовому полі для введення ключа для шифрування у блоці статистичних даних запускається процес обробки вхідних даних до кодування та отриманих даних після кодування (рис. 4.3). Звідси можна отримати статистичний розподіл символів. У таблицях висвітлено статистичні дані символів вхідної послідовності та послідовності після кодування.

Для того, щоб перейти до шифрування отримані закодовані дані потрібно «підготувати». Метод шифрування RC4 передбачає роботу з байтами інформації (тобто кожен вхідний символ повинен містити 8 біт). А після кодування всі символи стали довжиною 9 біт. Тому перед шифруванням, використовуючи функції join, reduce, JSON.stringify, JSON.parse, елементи масиву із даними довжиною 9 біт об’єднуються у рядок. Після цього отриманий рядок ділиться на дані довжиною по 8 біт і записуються у об’єкт. Разом із цим в об’єкт записується кількість повторень тієї чи іншої 8-бітної комбінації.

Результати кодування (стиску) даних

Статистика символів до кодування:

Довжина повідомлення: 10

символ ASCII	символ ASCII у двійковому вигляді	позиція символу у таблиці ASCII	кількість	ймовірність появи
h	1101000	104	11	1.00
k	1101011	107	10	0.909
h	1101000	104	1	0.0909

Символи у словнику LZW:

Довжина повідомлення: 4

символ	позиція у словнику LZW	кількість	ймовірність появи
100000000	256	1	0.0909
100000001	257	1	0.0909
100000010	258	1	0.0909
001101000	104	2	0.182
100000000	256	1	0.0909
100000001	257	1	0.0909
100000010	258	1	0.0909
001101011	107	1	0.0909
000100000	32	1	0.0909

Рисунок 4.3 – Блок обробки даних вхідної послідовності до та після кодування (стиснення) даних

Підготовлені дані направляються на блок шифрування. Цей блок представлений функцією RC4.js:

```
function rc4(key, str) {
	let s = [], j = 0, x, res = '';
	for (let i = 0; i < 256; i++) {
		s[i] = i;
	}
	for (i = 0; i < 256; i++) {
		j = (j + s[i] + key.charCodeAt(i % key.length)) % 256;
		x = s[i];
		s[i] = s[j];
		s[j] = x;
	}
	i = 0;
	j = 0;
	for (let y = 0; y < str.length; y++) {
		i = (i + 1) % 256;
		j = (j + s[i]) % 256;
		x = s[i];
		s[i] = s[j];
		s[j] = x;
		res += String.fromCharCode(str.charCodeAt(y) ^ s[(s[i] + s[j]) % 256]);
```

```
}
return res;
}
```

Підготовлений масив даних обробляється створеною функцією із використанням ключа, який введено в текстовому першому блоці програми. Також створюється масив ключів довжиною 256 символів, за допомогою якого відбувається шифрування повідомлення (рис. 4.4).

символ	у двійковій системі числення	у десятковій системі числення
	10010011	147
	11000	24
Õ	11010100	212
¾	10111110	190
	10011101	157

Рисунок 4.4 – Зашифроване повідомлення методом RC4

Отримані статистичні дані щодо повідомлення до кодування, після кодування та після шифрування використовуються для подальшої обробки.

4.2.1 Апробація методу кодування даних кіберфізичної системи методом LZW

Для апробації методу захисту передачі даних в кіберфізичних системах було запропоновано використати декілька файлів із даними, які надходять від елементів кіберфізичної системи. Такими елементами можуть бути різноманітні кнопки, сенсори температури, сенсори рівня, сенсори тиску і т.д. Передбачається, що дані від сенсорів записуються у деякий файл. Опитування проводиться із деякою періодичністю залежно від вимог, що ставлять до певної кіберфізичної системи.

Відомо, що для того, щоб стиснення стало можливим, то в тексті повинні бути однакові елементи.

Проведемо дослідження для трьох різних файлів. Перший файл міститиме дані, що надходять від кіберфізичної системи, що містить в собі невелику кількість елементів. Наприклад, сенсор температури, дві кнопки, АЦП. Ці дані записані у файл KPS1.txt у наступному вигляді: bt0=1^bt1=0^t1=6A0E^adc1=30, де bt0, bt1 – умовні імена (змінні) першої та другої кнопок відповідно. Значення, що записуються під цими іменами можуть набувати значень 0 або 1 («ввімкнено», «вимкнено»). З виходу сенсору температури під ім’ям t1 буде записане відповідне значення, а із використанням АЦП – представлено у десятковому вигляді та записано під ім’ям adc1.

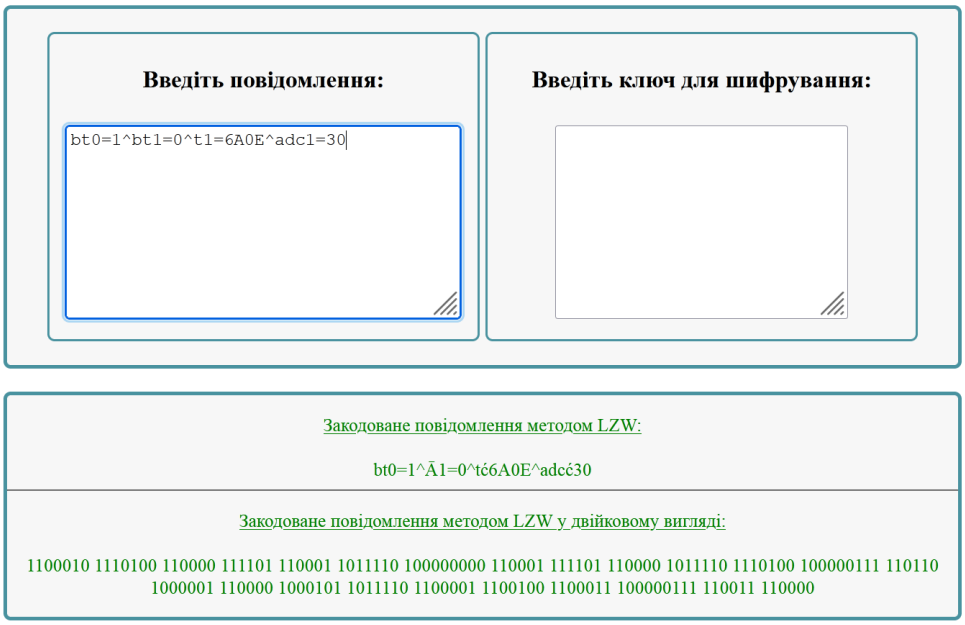
Проведемо стиснення даних використавши компресор на основі методу LZW. Результати наведено на рис. 4.5 а, б.

Аналіз отриманих даних показав, що коефіцієнт стиснення при використанні методу LZW становить:

$$k_{\text{ст}} = \frac{L_{code}}{L_{\text{LZW}}} = \frac{27}{24} = 1{,}125,$$

де L_{code} - довжина повідомлення до кодування, L_{LZW} - довжина повідомлення після кодування (стиснення).

Стиснення відбулось на 11,11%.



а)

Статистика символів до кодування:

Довжина повідомлення: 27

символ ASCII	символ ASCII у двійковому вигляді	позиція символу у таблиці ASCII	кількість	ймовірність появи
0	110000	48	4	0.148
1	110001	49	4	0.148
3	110011	51	1	0.0370
6	110110	54	1	0.0370
b	1100010	98	2	0.0741
t	1110100	116	3	0.111
=	111101	61	4	0.148
^	1011110	94	3	0.111
A	1000001	65	1	0.0370
E	1000101	69	1	0.0370
a	1100001	97	1	0.0370
d	1100100	100	1	0.0370
c	1100011	99	1	0.0370

б)

Символи у словнику LZW:

Довжина повідомлення: 24 символи.
Об'єм даних: 216 біт

символ ASCII у двійковому вигляді	символ ASCII	кількість	ймовірність появи
100000000	256	1	0.0370
100000111	263	2	0.0741
001100010	98	1	0.0370
001110100	116	2	0.0741
000110000	48	4	0.148
000111101	61	2	0.0741
000110001	49	2	0.0741
001011110	94	3	0.111
000110110	54	1	0.0370
001000001	65	1	0.0370
001000101	69	1	0.0370
001100001	97	1	0.0370
001100100	100	1	0.0370
001100011	99	1	0.0370
000110011	51	1	0.0370

в)

Рисунок 4.5 – Результати стиснення даних з файлу KPS1.txt: а – вихідне повідомлення та його стиснення, б, в – статистичні дані отриманих символів до та після кодування

Розглянемо кіберфізичну систему із більшою кількістю елементів. Наприклад, з шести кнопок, трьох сенсорів температури та АЦП, з виходу якого формуються дані про температуру та записуються у файл. Значення, що отримані з цих елементів, записані у файл KPS2.txt і виглядають наступним

чином:

bt0=1^bt1=0^bt2=1^bt3=0^bt4=1^bt5=0^t1=2AAD^adc1=25^t2=2AAD^adc2=25^t3=3336^adc3=30. Візуально вже можна оцінити, що коефіцієнт стиснення буде більший для цього файлу, оскільки багато однакових символів та груп символів. Проведемо стиснення даних. Його результати представлені на рис. 4.6, а статистичні дані наведено у додатку Г табл. Г.1 та табл. Г.2:

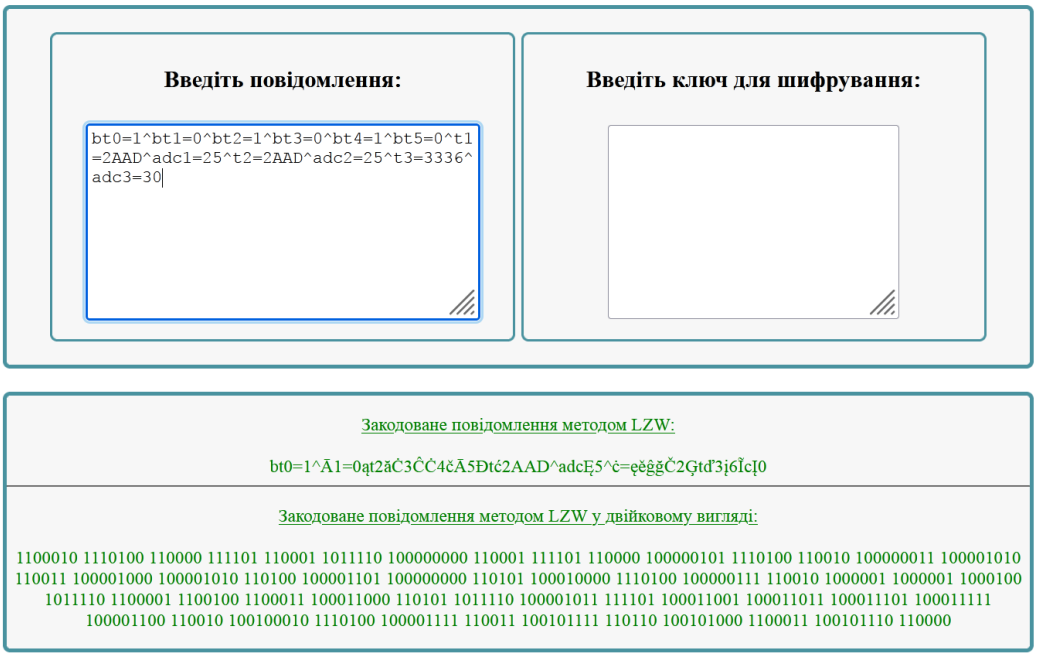


Рисунок 4.6 – Результати стиснення даних з файлу KPS2.txt

Отриманий результат показує, що коефіцієнт стиснення становить $k_{ст} = 1{,}54$, а стиснення відбулось на 34,94%.

Проаналізуємо кіберфізичну систему ще з більшою кількість кнопок, сенсорів та відповідних даних АЦП. Наприклад, додатково введемо ще сенсори руху, сенсори світла. Отримані дані записані у файл KPS3.txt у вигляді:

bt0=1^bt1=0^bt2=1^bt3=0^bt4=1^bt5=0^t1=2AAD^adc1=25^t2=2AAD^adc2=25^t3=3336^adc3=30^t4=3336^adc4=30^t5=2E17^adc5=27^t6=2224^adc6=20^tr1=0^tr2=1^tr3=0^tr4=0^tr5=0^tr6=1^lg1=1^lg2=0^lg3=0^lg4=1^lg5=1^lg5=1

Проведемо стиснення отриманого файлу. Результати стиснення на рис. 4.7, а статистичні дані отриманих результатів до та після стиснення наведено у додатку Г табл. Г.3 та табл. Г.4 відповідно

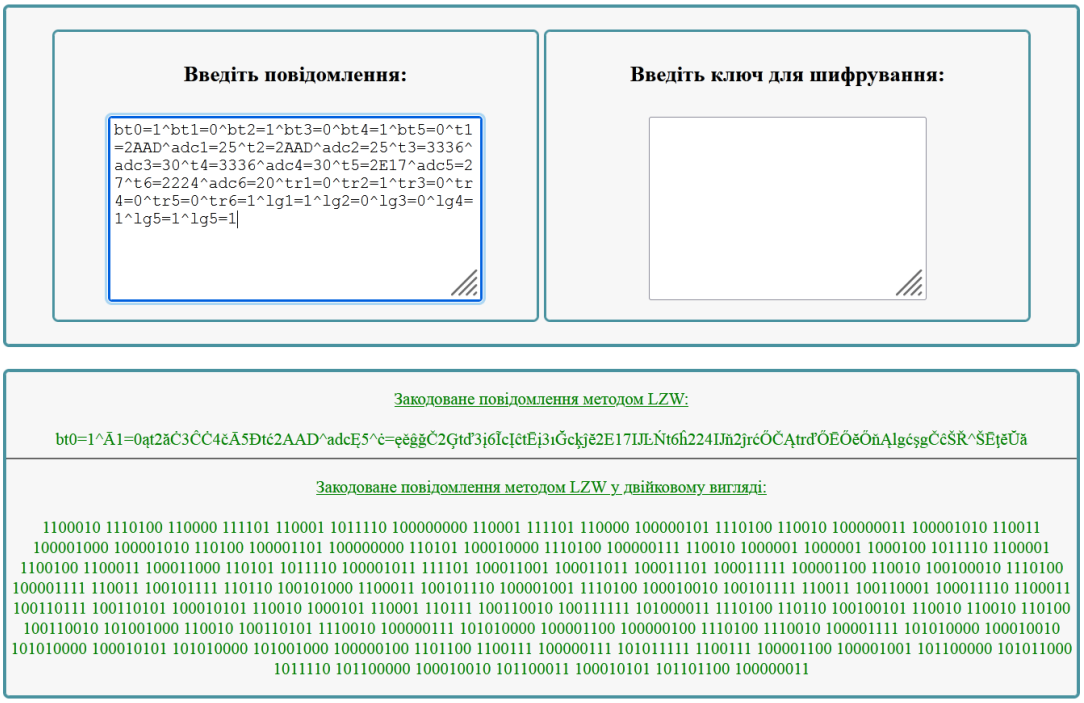
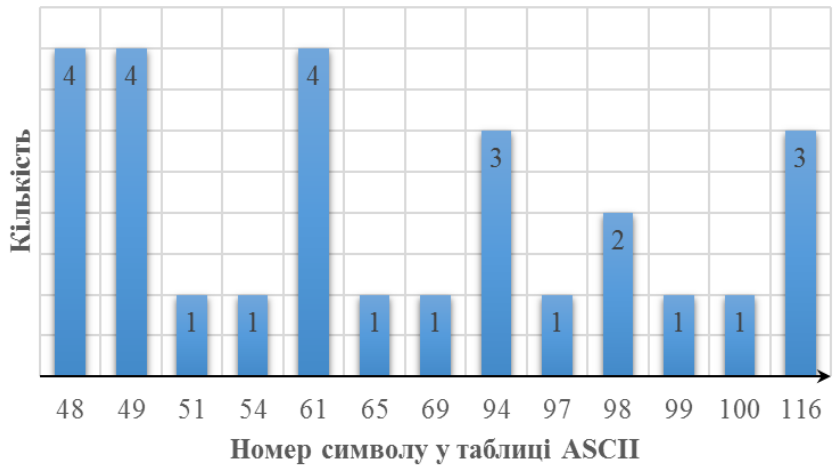


Рисунок 4.7 – Результати стиснення даних з файлу KPS3.txt

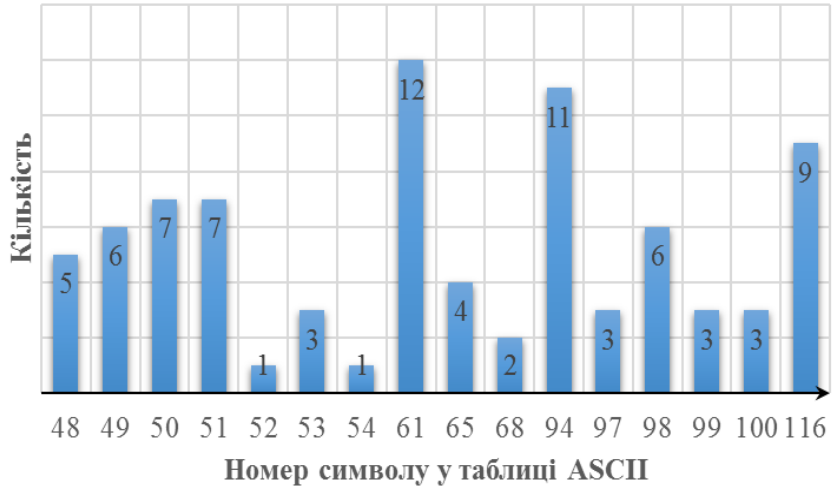
Коефіцієнт стиснення становить $k_{ст} = 1{,}81$, тобто стиснення відбулось на 44,83%

Слід зазначити, що коефіцієнт стиснення залежить не лише від об’єму даних, а й від їх повторюваності.

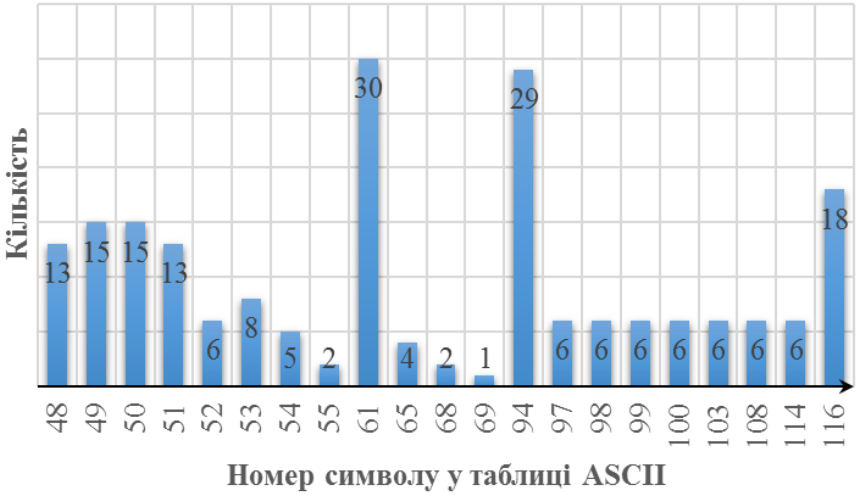
Для ілюстрації наведемо статистичний розподіл у початкових текстах, які записані у файли KPS1.txt (рис. 4.8, а), KPS2.txt (рис. 4.8, б), KPS3.txt (рис. 4.8, в)



а)



б)



в)

Рисунок 4.8 – Гістограма кількості символів у вхідній послідовності, що записана у файли: а) KPS1.txt, б) KPS2.txt, в)KPS3.txt

4.2 Апробація запропонованого методу захисту передачі даних у кіберфізичній системі

4.2.1 Апробація методу кодування даних кіберфізичної системи методом LZW

Для отримання даних щодо доцільності використання саме методу стиснення LZW, проведемо дослідження із використанням методу стиснення даних LZSS. Його особливістю є необхідність використання буферу для вхідних даних та створення на його основі словника. Проведемо стиснення методом LZSS для тих же даних, що і в попередньому підрозділі роботи. Об’єм буферу обрано 32 байти, а словник – об’ємом 4096 байт. Результати кодування наведено у додатку Д.

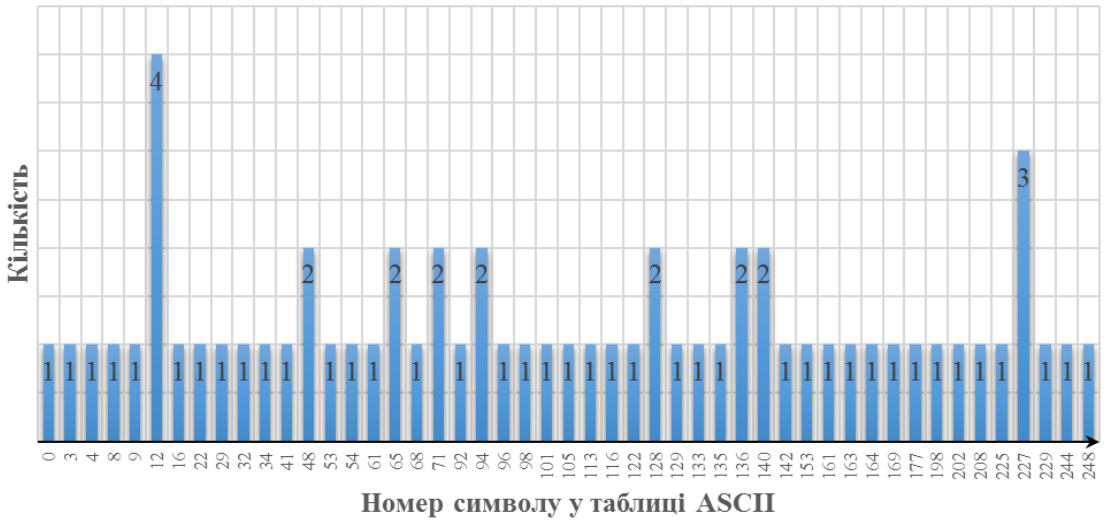
Отримані результати щодо коефіцієнту стиснення та часу, що витрачений на кодування методами LZW та LZSS, наведено у табл. 4.1.

Таблиця 4.1 – Зведені результати кодування даних методом LZW та LZSS

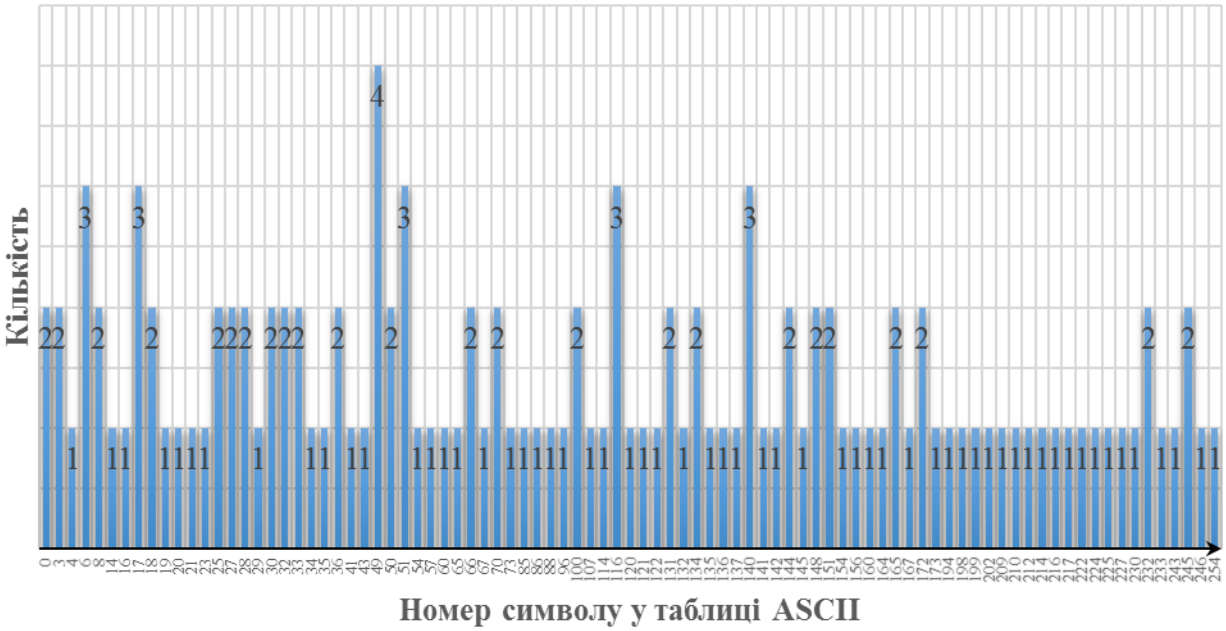
Назва файлу	Об’єм даних до стиснення, байт	Об’єм даних після стиснення, байт		Коефіцієнт стиснення		Стиснення, %		Час стиснення, мс	
		LZW	LZSS	LZW	LZSS	LZW	LZSS	LZW	LZSS
KPS1.txt	27	24	36	1,125	0,75	11,11	-33,33	<1	0,03
KPS2.txt	83	54	70	1,54	1,19	34,94	15,66	<1	<0,01
KPS3.txt	203	112	142	1,81	1,43	44,83	30,05	<1	0,03

На рис. 4.9 зображено гістограми кількості символів у закодованій послідовності, що були записані у файли KPS1.txt (рис. 4.9, а), KPS2.txt (рис. 4.9, б), KPS3.txt (рис. 4.9, в).

а)



б)



в)

Рисунок 4.9 – Гістограма кількості символів у закодованій послідовності, що були записані у файли: а) KPS1.txt, б)KPS2.txt, в)KPS3.txt

Аналіз отриманих гістограм рис. 4.8 та рис. 4.9 статистичних властивостей вхідного та закодованого тексту свідчить про їх різний склад, тобто статистичні властивості тесту закодованого відрізняються від початкових значень.

Аналіз отриманих даних показав, що для всіх об’ємів даних починаючи від найменших (27 символів – в даній роботі) до більших (203 символи), коефіцієнт стиснення має суттєву різницю, якщо порівнювати методи LZSS та LZW. Встановлено, що використання методу LZSS для невеликих об’ємів даних (або для даних, які не мають повторів), є недоцільним. Окрім цього алгоритм LZSS є доволі асиметричним, а процедура стиснення є доволі складною та здійснює доволі великий об’єм роботи при обробці кожного символу повідомлення, яке кодується. Це унеможливлює його використання при передачі даних бездротовим каналом зв’язку у кіберфізичних системах.

4.2.2 Апробація методу шифрування попередньо закодованих даних кіберфізичної системи методом RC4

Згідно розробленого методу після кодування методом LZW потрібно здійснити потокове шифрування методом RC4. Перед шифруванням виконується генерація ключа. Генерація потоку ключів відбувається за допомогою генератора псевдовипадкових символів. Далі між ключовим потоком та відкритим повідомленням відбувається операція сума за модулем 2. В алгоритмі RC4 використовується блок підстановки – S-блок. S-блок – це 256-байтовий масив розміром 16x16 (рис. 4.10):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Рисунок 4.10 – Масив блоку підстановки – S-блок

Отже, процес генерації ключової послідовності містить наступні етапи:

1) ініціалізація масиву блоку підстановки S-блоку довжиною 255 байтів S [0] = 0, S [1] = 1, S [2] = 2, ..., S [255] = 255 – як показано на рис.4.10;

2) заповнення масиву ключа К-блоку. Цей процес відбувається наступним чином. У якості ключа обирається деяка послідовність символів, наприклад, «encryption method», яку потрібно ввести у поле «Введіть ключ для шифрування» (рис. 4.11) та записується у вигляді номерів позицій у таблиці ASCII: «101, 110, 99, 114, 121, 112, 116, 105, 111, 110, 32, 109, 101, 116, 104, 111, 100». Секретний ключ буде повторювати ключ до того часу, поки він заповнить весь масив К-блоку (рис. 4.12):

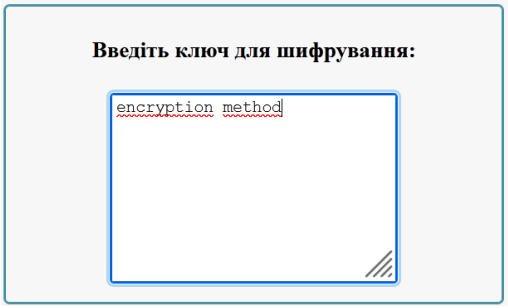


Рисунок 4.11 – Поле для введення ключа для шифрування

101	110	99	114	121	112	116	105	111	110	32	109	101	116	104	111
100	101	110	99	114	121	112	116	105	111	110	32	109	101	116	104
111	100	101	110	99	114	121	112	116	105	111	110	32	109	101	116
104	111	100	101	110	99	114	121	112	116	105	111	110	32	109	101
116	104	111	100	101	110	99	114	121	112	116	105	111	110	32	109
101	116	104	111	100	101	110	99	114	121	112	116	105	111	110	32
109	101	116	104	111	100	101	110	99	114	121	112	116	105	111	110
32	109	101	116	104	111	100	101	110	99	114	121	112	116	105	111
110	32	109	101	116	104	111	100	101	110	99	114	121	112	116	105
111	110	32	109	101	116	104	111	100	101	110	99	114	121	112	116
105	111	110	32	109	101	116	104	111	100	101	110	99	114	121	112
116	105	111	110	32	109	101	116	104	111	100	101	110	99	114	121
112	116	105	111	110	32	109	101	116	104	111	100	101	110	99	114
121	112	116	105	111	110	32	109	101	116	104	111	100	101	110	99
114	121	112	116	105	111	110	32	109	101	116	104	111	100	101	110
99	114	121	112	116	105	111	110	32	109	101	116	104	111	100	101

Рисунок 4.12 - Масив блоку ключа – К-блок

3) Перестановка для S-блоку. Операція перестановки призведе до змін у масиві S-блоку. Операція перестановки виконується 256 разів за наступною формулою: (j + S[i] + K[i])mod256. На цьому етапі зміні i та j використовуються для індексації масиву S[i] та K[i] відповідно. Спочатку ініціалізуються нульовими значеннями: i = 0, j = 0. Далі для кожного i від 0 до 255 визначається j:

$$j = (j + S[i] + K[i]) \bmod 256 = (0 + 0 + 101) \bmod 256 = 101.$$

Отже нове положення байту на 0-ій позиції у масиві S-блоку отримає значення 101. Відповідно байт зі 101-ї позиції переміститься на 0-у позицію (рис. 4.13).

101	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	0	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Рисунок 4.13 – Змішування елементів S-блоку з елементами K-блоку

Всього відбувається 256 таких ітерацій. Позиції інших байтів визначається аналогічним чином та перезаписується у масив S-блоку (рис. 4.14):

101	212	57	174	43	160	149	138	1	120	162	26	139	137	130	0
150	234	106	224	102	244	122	5	134	14	192	68	75	170	95	230
117	250	129	18	153	48	207	37	2	148	38	159	235	146	21	184
69	229	91	19	70	77	245	60	226	136	74	16	186	23	216	67
100	195	171	27	196	87	99	185	220	51	241	118	49	215	93	176
131	123	53	254	25	105	45	181	7	20	30	228	175	62	52	133
90	24	238	185	64	240	73	219	71	13	152	114	66	96	6	227
145	81	40	12	182	4	197	161	44	243	76	132	112	97	72	15
253	63	17	190	143	33	128	11	248	239	89	59	180	177	179	210
191	119	41	205	214	155	218	80	217	225	223	22	127	141	147	115
169	209	88	165	36	200	85	237	251	10	35	50	124	110	58	31
84	104	113	246	135	46	29	242	79	98	47	172	204	54	255	166
32	236	156	78	111	56	233	3	103	109	8	39	107	125	247	61
121	126	193	118	138	199	158	144	168	232	42	86	164	249	183	65
142	198	187	194	167	55	188	154	108	189	202	92	83	9	222	221
208	151	252	116	28	173	34	206	157	140	231	82	163	178	201	203

Рисунок 4.14 – Остаточний масив S-блоку

4) Далі відбувається процес генерування ключового потоку, який є псевдовипадовим і отримується шляхом операцій обчислення. Кількість таких

операцій дорівнює кількості байтів відкритого тексту. Нижче наведемо приклад процесу розрахунку елементів ключового потоку.

Генерування першого елемента псевдовипадкової ключової послідовності:

- ініціалізація i = 0; j = 0
- i = (i + 1) mod 256 = (0 + 1) mod 256 = 1
- j = (j + S [i]) mod 256 = (0 + S [1]) mod 256 = (0 + 212) mod 256= 212;
- S[i] = S[1];
- S[j] = S[212];
- S[i] = S[212] = 138;
- S[j] = S[1] = 212;
- t = (138 + 212) mod 256;
- k = S[t] = S[94] = 52.

Генерування ключового потоку триває до тих пір, доки буде зашифровано цілу кількість байтів відкритого тексту і згенерований псевдовипадковий ключ (рис. 4.15).

52	62	5	251	165	85	197	62	43	189	251	97	66	249	23	246
152	241	121	161	94	195	14	233	81	118	152	8	239	120	110	230
60	21	117	136	39	47	221	69	187	200	182	158	19	57	240	15
9	202	57	44	108	185	91	65	193	252	175	193	104	89	135	89
43	83	64	210	221	201	246	151	7	216	136	195	78	61	157	71
156	214	35	38	6	61	127	43	60	162	4	11	109	71	45	163
155	6	240	128	193	33	179	27	……							

Рисунок 4.15 – Псевдовипадковий ключ RC4

Кожен байт отриманого ключа обробляється із кожним байтом відкритого тексту із використанням операції сума за модулем 2.

Для першого повідомлення з файлу KPS1.txt після шифрування відкритого тексту, отримані наступні результати (табл. 4.2)

Таблиця 4.2 – Результати шифрування методом RC4 першого повідомлення з файлу KPS1.txt

Псевдовипадковий ключ			**Текст**			**Операція додавання за модулем 2 (XOR) - шифрування**		
Символ ASCII	**10-ий код символу**	**2-ий код символу**	**Символ ASCII**	**10-ий код символу**	**2-ий код символу**	**Символ ASCII**	**10-ий код символу**	**2-ий код символу**
4	52	00110100	1	49	00110001		5	101
>	62	00111110		29	00011101	#	35	100011
	5	00000101		6	00000110		3	11
û	251	11111011		3	00000011	ø	248	11111000
¥	165	10100101	Ñ	209	11010001	t	116	1110100
U	85	01010101	□	137	10001001	Ü	220	11011100
Å	197	11000101	z	122	01111010	¿	191	10111111
>	62	00111110		0	00000000	>	62	111110
+	43	00101011	1	49	00110001		26	11010
½	189	10111101	-	30	00011110	£	163	10100011
û	251	11111011	□	140	10001100	w	119	1110111
a	97	01100001		11	00001011	j	106	1101010
B	66	01000010	Ç	199	11000111	□	133	10000101
ù	249	11111001	H	72	01001000	±	177	10110001
	23	00010111	8	56	00111000	/	47	101111
ö	246	11110110	Ø	216	11011000	.	46	101110
□	152	10011000	□	130	10000010		26	11010
ñ	241	11110001	0	48	00110000	Á	193	11000001
y	121	01111001	"	34	00100010	[	91	1011011
¡	161	10100001	□	151	10010111	6	54	110110
^	94	01011110	□	140	10001100	Ò	210	11010010
	195	11000011	&	38	00100110	å	229	11100101
é	14	00001110	C	67	01000011	M	77	1001101
Q	233	11101001		28	00011100	õ	245	11110101
v	81	01010001		28	00011100	M	77	1001101
	118	01110110	f	102	01100110		16	10000
	152	10011000	0	48	00110000	¨	168	10101000

Результати шифрування RC4 другого та третього повідомлень з файлів KPS2.txt та KPS3.txt наведено у додатку Е табл. Е.1 та табл. Е.2 відповідно.

4.3 Дослідження ефективності побудованої моделі

Дослідження ефективності побудованої моделі методу захисту передачі даних у кіберфізичних системах проведемо на основі алгоритму, що викладений у розділі 3.3. У табл. 4.3 наведено зведені дані, щодо оцінки статистичних параметрів даних у файлах KPS1.txt, KPS2.txt, KPS3.txt

Таблиця 4.3– Зведені дані статистичних параметрів даних у кіберфізичній системі перед шифруванням

Файл	Параметр	Кількість символів алфавіту, що зустрічаються	Ентропія, біт/симв.	Довжина, симв.	Об’єм, біт
KPS1.txt	Вхідна послідовність	13	3,43794	27	216
	Закодована послідовність	15	3,45662	24	216
	Октети, на які розподілено закодовану послідовність	23	4,46	27	216
KPS2.txt	Вхідна послідовність	16	3,717	83	664
	Закодована послідовність	36	3,64722	54	486
	Октети, на які розподілено закодовану послідовність	49	5,49375	61	488
KPS3.txt	Вхідна послідовність	21	3,98084	203	1624
	Закодована послідовність	60	3,5934	112	1008
	Октети, на які розподілено закодовану послідовність	89	6,34787	126	1008

Аналіз отриманих даних свідчить про те, що статистичний розподіл символів вхідної послідовності, закодованої послідовності та послідовності

октет, на які поділено закодовану послідовність є різною. Це дає можливість стверджувати, що шифрування даних, які попередньо закодовані словниковим методом стиснення, унеможливлює дію атаки яка передбачалась введенням тестової послідовності з подальшим виконанням операції за модулем 2 над зашифрованою послідовністю з метою визначення ключа. Тобто перед шифруванням даних вхідна послідовність буде видозмінена, а також порушено її статичний розподіл.

4.4 Висновки

У четвертому розділі роботи було проведено моделювання запропонованого методу захисту передачі даних із використанням кодування (стиснення) даних та їх шифрування потоковим шифром у кіберфізичних системах бездротовим зв’язком.

Моделювання було проведено із використанням даних, що отримані від елементів кіберфізичної системи, різного об’єму.

Результати проведеного моделювання:

- було визначено статистичні характеристики вхідних даних та даних, що отримані у результаті кодування (стиснення) перед шифрування;
- встановлено, що використання запропонованого методу дозволяє отримати дані, які статистично незалежні між собою у вхідній та закодованій послідовності;
- показано, що метод захисту передачі даних у кіберфізичній системі є працюючим і дозволяє здійснити захист даних різного об’єму.

ВИСНОВКИ

У кваліфікаційній роботі магістра було розроблено метод захисту передачі даних у кіберфізичній системі бездротовим зв’язком із сумісним використанням словникового методу кодування даних та потокового шифрування даних, який використовується у технологіях WEP та WPA:

- проведено аналіз існуючих методів кодування та шифрування даних, виділено їх недоліки та особливості, проведено аналіз можливих атак при передачі даних бездротовим зв’язком у кіберфізичній системі;
- наведено математичну модель методу та проведено її апробацію;
- розроблено вимоги до системи захисту, алгоритм методу захисту передачі даних та алгоритм оцінки ефективності методу захисту передачі даних;
- проведене моделювання за допомогою розробленого програмного коду методу захисту передачі даних на даних різного об’єму довело ефективність при його використанні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. R. G. Sanfelice. Analysis and Design of Cyber-Physical Systems. A Hybrid Control Systems Approach // Cyber-Physical Systems: From Theory to Practice / D. Rawat, J. Rodrigues, I. Stojmenovic. — CRC Press, 2016. — ISBN 978-1-4822-6333-6.]

2. Pajic M. Robust architectures for embedded wireless network control and actuations / M. Pajic, A. Chernoguzov, R. Mangharam. // Transactions on Embedded Computing System. – 2012.

3. Зегжда П. Д. Подходы к оценке безопасности киберфизических систем [Електронний ресурс] / П. Д. Зегжда // Санкт - Петербургский политехнический университет Петра Великого, Рускрипто. – 2017. – Режим доступу до ресурсу: https://docplayer.com/53543447-Podhody-k-ocenke-bezopasnosti-kiberfizicheskih-sistem.html.

4. Бубела Т. З. Аналіз потенціалу кіберфізичних систем для застосування у агросекторі / Т. З. Бубела, Т. І. Федишин. // Національний університет “Львівська політехніка” Вимірювальна техніка та метрологія. – 2019. – №4.

5. A Review of Technology Standards and Patent Portfolios for Enabling Cyber-Physical Systems in Advanced Manufacturing / AMY J. C. TRAPPEY, HARLES V. TRAPPEY, USHARANI HAREESH GOVINDARAJAN, JOHN J. SUN. // IEEE Access - SPECIAL SECTION ON INDUSTRY 4.0. – 2016.

6. Salomon D. Data Compression: The Complete Reference, 3rd Edition / David Salomon. – USA: HAM, 2004. – 920 с.

7. Sayood K. Lossless Compression Handbook / Khalid Sayood. Elsevier Science, 2003. – 454 с.

8. Witten I. H. Managing Gigabytes: Compressing and Indexing Documents and Images / I. H. Witten, A. Moffat, T. C. Bell., 1999. – 560 с.

9. Smith S. Digital Signal Processing: A Practical Guide for Engineers and Scientists / Steven Smith. – San Diego, California: California Technical Publishing, 2002.

10. D. A. Huffman, “A method for the construction of minimum redundancy codes,” Proc. IRE, vol. 40, no. 2, pp. 1098–1101, Sept. 1952

11. Основи теорії інформації та кодування. Конспект лекцій: [Електронний ресурс]: навч. посіб. для студ. спеціальності 171 «Електроніка», спеціалізації «Електронні та інформаційні системи і технології телебачення, кінематографії та звукотехніки»/ М.І. Романюк; Ю. Г. Савченко; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 1,86 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. –70 с.

12. Коротєєва Т. Алгоритми та структури даних / Т. О. Коротєєва. – Львів: Львівська політехніка, 2014. – 280 с.

13. Майданюк В. П. Кодування та захист інформації. Навчальний посібник. - Вінниця: ВНТУ, 2009. - 164 с.

14. Kumar G. Analysis of Arithmetic and Huffman Compression Techniques by Using DWT-DCT / G. Kumar, R. Kumar. // Graphics and Signal Processing. – 2021. – №4. – С. 63–70.

15. Ziv J. A Universal Algorithm for Sequential Data Compression / J. Ziv, A. Lempel. // IEEE Transactions on Information Theory. – 1977. – №23.

16. Гапак О.М. Захист інформації в комп’ютерних системах. Підручник / О.М. Гапак, С.І. Балога .- Ужгород: видавництво ПП «АУТДОР-ШАРК», 2021. – 184 с.

17. Stallings W. Cryptography and Network Security Principles and Practices / William Stallings. Prentice Hall, 2005. – 592 с.

18. Юдін О. К. Аналіз захищеності бездротових мереж з використанням WEP-технології / О. К. Юдін, О. Весельська. // Наукоємні технології: Інформаційна безпека. – 2012. – №3. ISSN 2075-0781

19. Тарнавський Ю. А. Технології захисту інформації: підручник / Ю. А. Тарнавський. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 162 с.

20. Остапов С. Е. Технології захисту інформації : навчальний посібник / С. Е. Остапов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2013. – 476 с.

21. Кузнецов О. О. Захист інформації в інформаційних системах / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2011. – 512 с.

22. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

23. Kareem S. Modification on Key Stream Generator for RC4 Algorithm / S. Kareem, A. Rahma. // Engineering and Technology Journal. – 2020. – №38. – p. 54–60.

24. MRC4: A Modified RC4 Algorithm Using Symmetric Random Function Generator for Improved Cryptographic Features / Rahul Saha, G. Geetha, Gulshan Kumar, Tai-Hoon Kim. // IEEE Access - Special section on emerging approaches to cyber security. – 2019.

25. Ковтун Л.О. Основні ідеї та алгоритми адаптивного словникового кодування методом Лемпеля-Зіва [Текст] / Л. О. Ковтун, Д. М. Медзатий // Вісник Хмельницького національного університету. Технічні науки. – 2017. – № 4. – С. 129-136.

26. Mousa A. Evaluation of the RC4 Algorithm for Data Encryption / A. Mousa, A. Hamad. // International Journal of Computer Science and Applications. – 2006. – №2.

27. Гетьман, І. А. Технології захисту інформації: посібник для студентів вищих навчальних закладів спеціальності «Комп’ютерні науки» / І. А. Гетьман, О. В. Алтухов. – Краматорськ : ДДМА, 2018. – 120 с.

28. Беспроводная безопасность — шифрование [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/izuchite-besprovodnuiu-bezopasnost/besprovodnaia-bezopasnost-shifrovanie.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного коду блоку введення даних, їх кодування та шифрування

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title></title>
	<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
	<div id="in" >
	<div id="in1" >
		<h4>Введіть повідомлення:</h4>
		<textarea id="input" cols="35" rows="8"></textarea>
		<p></p>
	</div>
	<div id="in1">
		<h4>Введіть ключ для шифрування:</h4>
		<textarea id="input1" cols="25" rows="8" ></textarea>
		<p></p>
	</div>
	</div>
	<div id="in">
		<p id="p1"></p>
		<p id="p2"></p>
	</div>

	<div id="in" >
		<h4>Результати кодування (стиснення) даних</h4>
		<div class="inline">
			<h4 id="h41"></h4>
			<p id="p5"></p>
			<table id="table1" class=" visible "></table>
		</div>
		<div class="inline">
			<h4 id="h44"></h4>
			<p id="p6"></p>
			<table id="table4" class=" visible "></table>
		</div></div>
	<div id="in">
```

```
        <div class="inline">
            <h4 id="h42"></h4>
            <p id="p4"></p>
            <table id="table2" class=" visible "></table>
        </div>
        <div class="inline">
            <h4 id="h47"></h4>
            <p id="p9"></p>
            <table id="table7" class=" visible "></table>
        </div>
        </div>
        <div id="in" >
        <h4>Результати шифрування даних</h4>
        <h4 id="h48"></h4>
            <div class="inline"><p id="p3"></p>
                <table id="table3" class=" visible "></table>
            </div>
            <div class="inline"><p id="p7"></p>
                <table id="table5" class=" visible "></tr>
                </table>
            </div>
            <div class="inline"><p id="p8"></p>
                <table id="table6" class=" visible "></table>
            </div>
        </div>
<script src="lzw_encode.js"></script>
<script src="binaryAgent.js"></script>
<script src="rc4.js"></script>
<script src="rc4plus.js"></script>
<script src="rc4a.js"></script>
<script src="random.js"></script>
<script src="split.js"></script>

<script>
        console.time('FirstWay');
        let s1 = 'bt0=1^bt1=0^t1=2AAD^adc1=25';
        let result3;
        let input = document.querySelector('#input');
        let input1 = document.querySelector('#input1');
        let elem = document.querySelector('#elem');
        let p1 = document.querySelector('#p1');
        let p2 = document.querySelector('#p2');
        let p3 = document.querySelector('#p3');
```

```
let p4 = document.querySelector('#p4');
let p5 = document.querySelector('#p5');
let p6 = document.querySelector('#p6');
let p7 = document.querySelector('#p7');
let p8 = document.querySelector('#p8');
let p9 = document.querySelector('#p9');
let h41 = document.querySelector('#h41');
let h42 = document.querySelector('#h42');
let h44 = document.querySelector('#h44');
let h47 = document.querySelector('#h47');
let h48 = document.querySelector('#h48');
let button = document.querySelector('#button');
let button1 = document.querySelector('#button1');
let button2 = document.querySelector('#button2');
let table1 = document.querySelector('#table1');
let table2 = document.querySelector('#table2');
let table3 = document.querySelector('#table3');
let table4 = document.querySelector('#table4');
let table5 = document.querySelector('#table5');
let table6 = document.querySelector('#table6');
let table7 = document.querySelector('#table7');
let result1 = '';
let arr1 = [];
let str;
let str2;
let str3;
let str7;
let arr3 = [];
let arr4 = [];
let arr = [];

//Кодування методом LZW
input.addEventListener('input', function(){
let result2 = input.value;
result3 = lzwEncode(result2);
p1.innerHTML = '<u>Закодоване повідомлення методом LZW: </u><br><br>' + result3 + '<hr>';
for (var i = 0; i < result3.length; i++) {
    arr[i] = result3[i].charCodeAt(0).toString(2);
    p2.innerHTML = '<u>Закодоване повідомлення методом LZW у двійковому вигляді: </u><br><br>' + arr.join(' ');
    str1 = arr.join(' ');
    console.log(arr);
```

```
        }
    })
        input.addEventListener('keydown', function(event) {
            if (event.keyCode === 13) {
            elem1.focus();
        }})
        input1.addEventListener('keydown', function(event) {
            if (event.keyCode === 13) {

                for (var i = 0; i < input.value.length; i++) {
            arr4[i] = input.value[i];
        }
        //із масиву в об'єкт
        var result5 = arr4.reduce(function(acc, el) {
            acc[el] = (acc[el] || 0) + 1;
            return acc;
        }, {});
        let str8 = JSON.stringify(result5, null, 2);
        let t1 = JSON.parse(str8);

            h41.innerHTML = 'Статистика символів до кодування: ';

            let tr11 = document.createElement('tr');

            let th11 = document.createElement('th');
            let th12 = document.createElement('th');
            let th13 = document.createElement('th');
            let th14 = document.createElement('th');
            let th15 = document.createElement('th');

            th11.innerHTML = 'символ ASCII ';
            th12.innerHTML = 'символ ASCII <br> у двійковому вигляді ';
            th13.innerHTML = 'позиція символу <br> у таблиці ASCII';
            th14.innerHTML = 'кількість   ';
            th15.innerHTML = 'ймовірність <br>появи ';

            tr11.appendChild(th11);
            tr11.appendChild(th12);
            tr11.appendChild(th13);
            tr11.appendChild(th14);
            tr11.appendChild(th15);
```

```
                table1.appendChild(tr11);
                for (let obj in t1) {
                table1.classList.remove('visible');
                let tr = document.createElement('tr');

                for (let i = 0; i < 1; i++) {
                        let td1 = document.createElement('td');
                        let td2 = document.createElement('td');
                        let td3 = document.createElement('td');
                        let td4 = document.createElement('td');
                        let td5 = document.createElement('td');

                        let obj1 = obj.charCodeAt(0).toString(2);
                        if (obj1.length == 7) {
                                obj1 =  '0' + obj1;
                        }

                        if (obj1.length == 6) {
                                obj1 =  '0' +'0' + obj1;
                        }

                        td1.innerHTML = obj;
                        td2.innerHTML = obj1;
                        td3.innerHTML  =  parseInt(obj.charCodeAt(0).toString(2),
2);
                        td4.innerHTML = t1[obj];
                        td5.innerHTML = (t1[obj] / arr4.length).toPrecision(3);

                        tr.appendChild(td1);
                        tr.appendChild(td2);
                        tr.appendChild(td3);
                        tr.appendChild(td4);
                        tr.appendChild(td5);
                }

                table1.appendChild(tr);
        }
        p5.innerHTML = 'Довжина повідомлення: ' + input.value.length + "
символи. <br>Об'єм даних: " + (input.value.length * 8) + ' біт';
                }
        })
        //Вивести повідомлення у двійковому вигляді при натисканні
клавіші Enter (повідомлення довжиною 9 символів)
```

```
input1.addEventListener('keydown', function(event) {
	if (event.keyCode === 13) {
result2 = input.value;
result3 = lzwEncode(result2);

for (var i = 0; i < result3.length; i++) {
	arr[i] = result3[i].charCodeAt(0).toString(2);
		if (arr[i].length == 7) {
			arr[i] = '0' + '0' + arr[i];
		}
		if (arr[i].length == 6) {
			arr[i] = '0' + '0' + '0' + arr[i];
		}
		str1 = arr.join(' ');
	}
for (let i = 0; i < arr.length; i++) {
	if (arr[i].length == 6) {
		arr[i] = '0' + '0' + '0' + arr[i];
	}
}
//підрахунок однакових елементів
var result1 = arr.reduce(function(acc1, el1) {
	acc1[el1] = (acc1[el1] || 0) + 1;
	return acc1;
}, {});
let str9 = JSON.stringify(result1, null, 2);
table4.classList.add('white');
let t2 = JSON.parse(str9);
h44.innerHTML = 'Символи у словнику LZW: ';

	let tr41 = document.createElement('tr');
	let th41 = document.createElement('th');
	let th42 = document.createElement('th');
	let th43 = document.createElement('th');
	let th44 = document.createElement('th');

	th41.innerHTML = 'символ ASCII ';
	th42.innerHTML = 'символ ASCII <br> у двійковому вигляді ';
	th43.innerHTML = 'позиція символу <br> у таблиці ASCII';
	th44.innerHTML = 'кількість   ';

	tr41.appendChild(th41);
```

```
                tr41.appendChild(th42);
                tr41.appendChild(th43);
                tr41.appendChild(th44);

                table4.appendChild(tr41);

                for (let obj in t2) {
                table4.classList.remove('visible');
                let tr = document.createElement('tr');

                        let td1 = document.createElement('td');
                        let td2 = document.createElement('td');
                        let td3 = document.createElement('td');
                        let td4 = document.createElement('td');

                        if (obj.length == 6) {
                                td1.innerHTML = '0' + '0' + '0' + obj;
                        }

                        if (obj.length == 9) {
                                td1.innerHTML = obj;
                        }

                        td2.innerHTML = parseInt(obj, 2);
                        td3.innerHTML = t2[obj];
                        td4.innerHTML = (t2[obj] / arr4.length).toPrecision(3);

                        tr.appendChild(td1);
                        tr.appendChild(td2);
                        tr.appendChild(td3);
                        tr.appendChild(td4);

                table4.appendChild(tr);
        }

        p6.innerHTML = 'Довжина повідомлення: ' + result3.length + " символи. <br>Об'єм даних: " + (result3.length * 9) + ' біт';
        //об'єднання елементів масиву у рядок після кодування, розбиття його на символи по 8 двійкових символів та запис у масив myResult, перетворення у об'єкт
        str1 = str1.replace( /\s/g, '');
        let myResult = split(str1,8);
```

```
            //додавання нулів до першого елемента масиву, якщо їх не 8
                if (myResult[0].length == 7) {
                    myResult[0] = '0' + myResult[0];
                }

                 if (myResult[0].length == 6) {
                    myResult[0] = '0' + '0' + myResult[0];
                }

                if (myResult[0].length == 5) {
                    myResult[0] = '0' + '0' + '0' + myResult[0];
                }

                if (myResult[0].length == 4) {
                    myResult[0] = '0' + '0' + '0' + '0' + myResult[0];
                }

                if (myResult[0].length == 3) {
                    myResult[0] = '0' + '0' + '0' + '0' + '0' + myResult[0];
                }

                if (myResult[0].length == 2) {
                    myResult[0] = '0' + '0' + '0' + '0' + '0' + '0' + myResult[0];
                }

                if (myResult[0].length == 1) {
                    myResult[0] = '0' + '0' + '0' + '0' + '0' + '0' + '0' +
myResult[0];
                }

                //виведення символів по 8 біт по порядку їх появи у тексті
                h47.innerHTML = 'символи по 8 біт за порядком появи';

                let tr71 = document.createElement('tr');
                let th71 = document.createElement('th');
                let th72 = document.createElement('th');
                let th73 = document.createElement('th');

                th71.innerHTML = 'символ';
                th72.innerHTML = 'у двійковій <br>системі числення';
                th73.innerHTML = 'у десятковій <br>системі числення';

                tr71.appendChild(th71);
```

```
                tr71.appendChild(th72);
                tr71.appendChild(th73);

                table7.appendChild(tr71);

                for (let elem of myResult) {
                let tr = document.createElement('tr');
                let td0 = document.createElement('td');
                let td1 = document.createElement('td');
                let td2 = document.createElement('td');

                td0.innerHTML = parseInt(elem, 2);
                td1.innerHTML = elem;
                td2.innerHTML = binaryAgent(elem);

                tr.appendChild(td2);
                tr.appendChild(td1);
                tr.appendChild(td0);
                table7.appendChild(tr);
            }
            //перетворення масиву myResult в рядок (з пробілами), підрахунок однакових елементів, запис результатів у об'єкт
            str3 = myResult.join(' ');

            var result = myResult.reduce(function(acc, el) {
              acc[el] = (acc[el] || 0) + 1;
              return acc;
            }, {});

            let str7 = JSON.stringify(result, null, 2);
            let t = JSON.parse(str7);

            var result = myResult.reduce(function(acc, el) {
                    acc[el] = (((acc[el] || 0) + 1)/myResult.length).toPrecision(4);
                     return acc;
            }, {});

            //виведення підрахунку однакових елементів у таблицю до шифрування
                    h42.innerHTML = 'Підготовка до шифрування - <br>розбиття символів на групи по 8 біт ';

                    let tr21 = document.createElement('tr');
```

```
        let th21 = document.createElement('th');
        let th22 = document.createElement('th');
        let th23 = document.createElement('th');
        let th24 = document.createElement('th');
        let th25 = document.createElement('th');

        th21.innerHTML = 'символ';
        th22.innerHTML = 'у двійковій <br>системі числення';
        th23.innerHTML = 'у десятковій <br>системі числення';
        th24.innerHTML = 'кількість';
        th25.innerHTML = 'ймовірність <br>появи';

        tr21.appendChild(th21);
        tr21.appendChild(th22);
        tr21.appendChild(th23);
        tr21.appendChild(th24);
        tr21.appendChild(th25);

        table2.appendChild(tr21);
for (let obj in t) {

        let tr = document.createElement('tr');
        table2.classList.remove('visible');
        for (let i = 0; i < 1; i++) {
                let td0 = document.createElement('td');
                let td1 = document.createElement('td');
                let td2 = document.createElement('td');
                let td3 = document.createElement('td');
                let td4 = document.createElement('td');

                td0.innerHTML = binaryAgent(obj);
                td1.innerHTML =  obj;
                td2.innerHTML = parseInt(obj, 2);
                td3.innerHTML = t[obj];
                td4.innerHTML = (t[obj] / myResult.length).toPrecision(3);

                tr.appendChild(td0);
                tr.appendChild(td1);
                tr.appendChild(td2);
                tr.appendChild(td3);
                tr.appendChild(td4);
        }
```

```
                table2.appendChild(tr);
            }
            p4.innerHTML = 'Довжина повідомлення: ' + myResult.length + " символи. <br>Об'єм даних: " + (myResult.length * 8) + ' біт';

            //перетворення закодованих елементів у ASCII
            let result4 = binaryAgent(str3);

            //створення рандомного ключа
            /*let key1 = 'encryption method';*/
            let key1 = input1.value;
            h48.innerHTML = 'ключ: ' + input1.value;
            let result5 = rc4(key1,result4);
            p3.innerHTML = '<u><b>Зашифроване повідомлення <br>методом <span style="color:red">RC4</span>: </b></u><br>' + result5;
            for (var i = 0; i < result5.length; i++) {
                arr3[i] = result5[i].charCodeAt(0).toString(2);
                }

                let tr31 = document.createElement('tr');

                let th31 = document.createElement('th');
                let th32 = document.createElement('th');
                let th33 = document.createElement('th');

                th31.innerHTML = 'символ ';
                th32.innerHTML = 'у двійковій системі числення';
                th33.innerHTML = 'у десятковій системі числення';

                tr31.appendChild(th31);
                tr31.appendChild(th32);
                tr31.appendChild(th33);

                table3.appendChild(tr31);

                for (let elem of arr3) {

                let tr = document.createElement('tr');
                table3.classList.remove('visible');

                for (let i = 0; i < 1; i++) {
                    let td0 = document.createElement('td');
```

```
                let td1 = document.createElement('td');

                let td2 = document.createElement('td');

                let elem1 = elem;
                if (elem1.length == 7) {
                        elem1 = '0' + elem1;
                }

                if (elem1.length == 6) {
                        elem1 = '0' + '0' + elem1;
                }

                if (elem1.length == 5) {
                        elem1 = '0' + '0' + '0' + elem1;
                }

                if (elem1.length == 4) {
                        elem1 = '0' + '0' + '0' + '0' + elem1;
                }

                if (elem1.length == 3) {
                        elem1 = '0' + '0' + '0' + '0' + '0' + elem1;
                }

                if (elem1.length == 2) {
                        elem1 = '0' +'0' + '0' + '0' + '0' + '0' + elem1;
                }

                td0.innerHTML = binaryAgent(elem);
                td1.innerHTML = elem1;
                td2.innerHTML = parseInt(elem,2);

                tr.appendChild(td0);
                tr.appendChild(td1);
                tr.appendChild(td2);
        }
        table3.appendChild(tr);
}
let arr5 = [];
let result6 = rc4plus(result4, key1);
```

```
p7.innerHTML = '<u><b>Зашифроване повідомлення <br>методом <span style="color:red">RC4+</span>: </b></u><br>' + result6;

for (var i = 0; i < result6.length; i++) {
	arr5[i] = result6[i].charCodeAt(0).toString(2);
	}

let tr51 = document.createElement('tr');

let th51 = document.createElement('th');
let th52 = document.createElement('th');
let th53 = document.createElement('th');

th51.innerHTML = 'символ ';
th52.innerHTML = 'у двійковій системі числення';
th53.innerHTML = 'у десятковій системі числення';

tr51.appendChild(th51);
tr51.appendChild(th52);
tr51.appendChild(th53);

table5.appendChild(tr51);

	for (let elem of arr5) {

	let tr = document.createElement('tr');
	table5.classList.add('black');
	table5.classList.remove('visible');

	for (let i = 0; i < 1; i++) {
		let td0 = document.createElement('td');
		let td1 = document.createElement('td');

		let td2 = document.createElement('td');

		td0.innerHTML = binaryAgent(elem);
		td1.innerHTML = elem;
		td2.innerHTML = parseInt(elem, 2);

		tr.appendChild(td0);
		tr.appendChild(td1);
		tr.appendChild(td2);
```

```
                }
                table5.appendChild(tr);
            }

        let arr6 = [];
            let result7 = rc4a(result4, key1);
                p8.innerHTML = '<u><b>Зашифроване повідомлення <br>методом <span style="color:red">RC4A</span>: </b></u><br>' + result7;

                for (var i = 0; i < result7.length; i++) {
                    arr6[i] = result7[i].charCodeAt(0).toString(2);
                    }
                let tr61 = document.createElement('tr');

                let th61 = document.createElement('th');
                let th62 = document.createElement('th');
                let th63 = document.createElement('th');

                th61.innerHTML = 'символ ';
                th62.innerHTML = 'у двійковій системі числення';
                th63.innerHTML = 'у десятковій системі числення';

                tr61.appendChild(th61);
                tr61.appendChild(th62);
                tr61.appendChild(th63);

                table6.appendChild(tr61);

                    for (let elem of arr6) {

                    let tr = document.createElement('tr');
                    table6.classList.add('black');
                    table6.classList.remove('visible');

                    for (let i = 0; i < 1; i++) {
                        let td0 = document.createElement('td');
                        let td1 = document.createElement('td');

                        let td2 = document.createElement('td');


                        td0.innerHTML = binaryAgent(elem);
```

```
                td1.innerHTML = elem;
                td2.innerHTML = parseInt(elem, 2);

                tr.appendChild(td0);
                tr.appendChild(td1);
                tr.appendChild(td2);

            }
            table6.appendChild(tr);
        }
})
</script>
</body>
</html>
```

ДОДАТОК Б

(обов'язковий)

Публікації за темою досліджень:

Ю.П. КЛЬОЦ, Л.О. КОРЕЦЬКА
Хмельницький національний університет

МЕТОД ЗАХИСТУ ПЕРЕДАЧІ ДАНИХ У КІБЕРФІЗИЧНИХ СИСТЕМАХ

Стаття присвячена удосконаленню методу захисту передачі даних у кіберфізичних системах із використанням словникового методу кодування (стиснення) та потокового шифрування. Підвищення захисту відбувається за рахунок зміни статистичних характеристик вхідної послідовності символів. Встановлено, що використання кодування перед проведенням шифрування, дозволяє змінити ентропію появи символів, довжину та об'єм повідомлення, кількість символів алфавіту, які використовуються для створення послідовності.

Ключові слова: криптографія, метод стиску, потоковий шифр, кіберфізичні системи.

Y.P. KLOTS, L.O. KORETSKA
Khmelnytskyi National University

METHOD OF DATA TRANSMISSION PROTECTION IN CYBERPHYSICAL SYSTEMS

Abstract – Wireless networks are used to transmit data in cyberphysical systems. This creates an opportunity for attackers to intercept information and then decrypt it and use it for malicious purposes. Consequently, the need to increase the protection of transmitted data increases. This is made possible by new approaches to ensuring the cryptographic stability of the wireless security system.

The purpose of ensuring the protection of data transmission in a cyberphysical system is to create a method of encryption that even when intercepting data by decryption was not possible. After sending the test sequence, the attacker hopes to obtain the key sequence that was used during the encryption as a result of decrypting this sequence (reverse operation the sum of module 2 to the encrypted sequence). Having a code sequence and intercepting the following blocks of information, their decryption will not be difficult. Using the coding method for cyberphysical system data before encryption allows you to change the statistical characteristics of the input sequence of elements (message length, data volume, entropy, probability of occurrence of characters). These are the statistical characteristics that an attacker needs to decrypt data. Using encryption before encryption does not allow you to select the input test sequence and obtain the key.

The statistical characteristics of the input data and the data obtained as a result of encoding (compression) before encryption were determined; it is established that the use of the proposed method allows to obtain data that are statistically independent of each other in the input and coded sequence; it is shown that the method of protection of data transmission in the cyberphysical system is working and allows to protect data of different volumes.

Keywords: cryptography, compression method, flow code, cyberphysical systems.

Постановка проблеми

Кіберфізичні системи (КФС) заповнюють практично всі сфери нашого життя. Це і розумні будинки, розумні виробництва та мережі, Інтернет речей, безпілотний транспорт та транспортні мережі і навіть розумні міста та ін. З кожним днем в рамках четвертої промислової революції впровадження КФС відбувається у промисловість – індустрія 4.0. Разом із збільшенням охоплення сфер життя КФС відповідно і збільшується об'єми та цінність інформації, якими обмінюються елементи КФС.

Для передачі даних у КФС використовуються бездротові мережі. Це створює для зловмисників можливість перехоплення інформації з подальшим її дешифруванням та використанням у зловмисних цілях. Таке перехоплення може нанести непоправну шкоду тим, хто користується КФС й не тільки, оскільки вийшовши з ладу КФС може завдати не лише фінансових збитків власнику КФС, а нанести екологічних збитків та шкоду життю людини і т.д. А отже збільшується необхідність підвищувати захист даних, які передаються. Це стає можливим із використанням нових підходів щодо забезпечення криптостійкості системи захисту передачі даних бездротовим зв'язком.

Результати досліджень

На відміну від дротових мереж, в бездротових мережах можуть використовуватись зовсім інші види хакерських нападів. Тому і механізм захисту повинні відрізнятись.

Виділяють чотири основні категорії атак [1]:

1) атаки доступу;

2) атаки модифікації;

3) атаки на відмову в обслуговуванні;

4) атаки на відмову від зобов'язань.

Атака доступу – направлена зловмисником на отримання доступу до даних, до перегляду яких у нього немає прав, з метою порушення конфіденційності інформації.

Атака модифікації – має на меті зміну даних, тобто на порушення її цілісності. Виділяють три види модифікації:

1. заміна даних – направлена на заміну даних як у секретній так і загальнодоступній частині інформації;
2. додавання даних – додавання нових даних;
3. зміна даних – переміщення існуючих даних.

Найбільш розповсюджена загроза у бездротових мережах – перехоплення сигналу та його дешифрування. При навіть можливому перехваті повідомлень, потрібно забезпечити підвищення надійності захисту, який стосується неможливості прочитати інформацію – тобто забезпечити шифрування даних.

Таким алгоритмом може бути RC4 [2]. Він широко використовується в комп'ютерних мережах в системах захисту даних. Може використовуватись, наприклад, в протоколах SSL та TLS, алгоритмах забезпечення безпеки бездротових мереж WEP, WPA [3, 4].

Запропоновано удосконалення методу шифрування із використанням словникового колування (стиснення) перед проведенням шифрування. Це дозволить змінити статистичні характеристики вхідної послідовності символів: ентропію та ймовірність появи символів, довжину повідомлення, його об'єм.

В алгоритмі формування потокового шифру із використанням словникового методу стиснення інформації можна виділити наступні кроки:

- отримання вхідного текстового потоку $G = g_1 g_2 g_3 \ldots g_m$, що створений з деякого алфавіту $\Sigma = \{a_1, a_2, a_3, \ldots, a_q\}$. на основі отриманих даних від елементів кіберфізичної системи;

- формування вихідної текстової послідовності у вигляді двійкових символів використавши для перетворення таблицю ASCII;

- кодування (стиснення) отриманої вхідної комбінації із використанням словникового методу стиснення LZW [5] – отримання послідовності 9-бітних послідовностей $c(G) = p_1 p_2 p_3 \ldots p_v$;

- об'єднання отриманої послідовності $c(G) = p_1 p_2 p_3 \ldots p_v$ у нерозривну послідовність біт довжиною $l_{c(S)}$;

- розділення бітової послідовності довжиною $l_{c(S)}$ на блоки по $l_{s[k]}$, кількість яких становить k_{blocks}. За необхідності, у випадку якщо $l_{c(S)}$ не є кратним $l_{s[k]}$, потрібно додати додаткові нульові біти на початку $c(G) = p_1 p_2 p_3 \ldots p_v$. Результатом є отримання послідовності $Y=y_0 y_1 \ldots y_{kblocks}$;

- формування підстановки - S-блоку;

- формування ключового потоку $Z=z_0 z_1 \ldots z_{n-1}$;

- шифрування послідовності $Y=y_0 y_1 \ldots y_{kblocks}$ ключовим потоком $Z=z_0 z_1 \ldots z_{n-1}$ із використанням операції сума за модулем 2.

Алгоритм методу захисту передачі даних зображено на рис. 1.

На рис. 1 із зображеної схему алгоритму методу захисту передачі даних у кіберфізичній системі із використанням словникового методу кодування LZW та шифрування даних RC4 можна виділити наступні основні етапи виконання:

- блоки 1-3 – введення вхідних даних, які необхідні для реалізації методу захисту передачі даних;
- блоки 4-6 – відповідають за кодування кодової послідовності та отримання закодованої послідовності $c(G) = p_1 p_2 p_3 ... p_v$;
- блоки 7-11 – шифрування даних
- блок 12 - отримання вихідної послідовності $H = h_1 h_2 .. h_{k_{length}}$.
- блок 13 – передача зашифрованого тексту H.

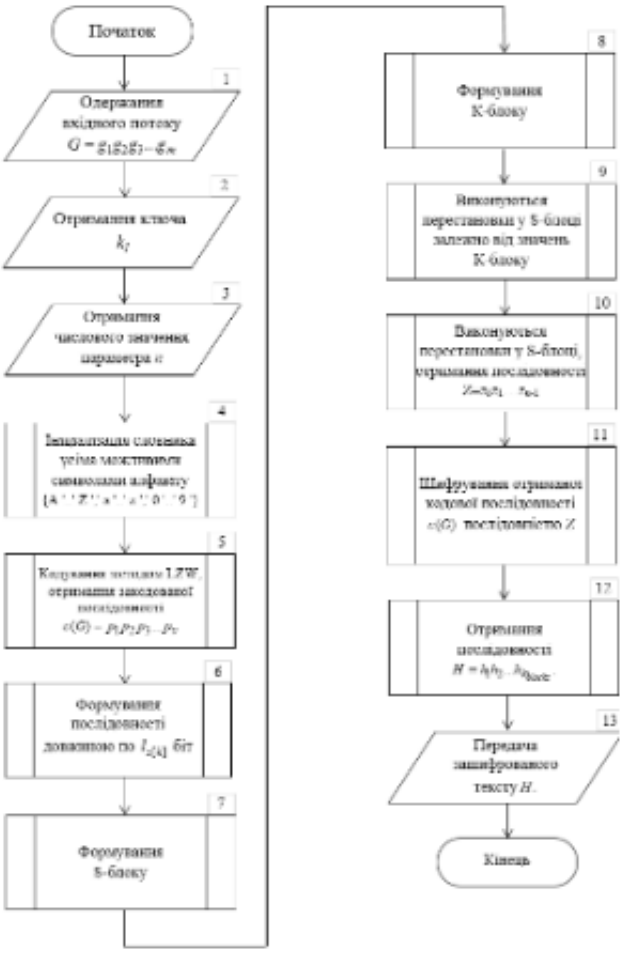


Рисунок 1 – Алгоритм методу захисту передачі даних

Апробуємо запропонований метод на вхідній послідовності даних кіберфізичної системи, яка містить, наприклад, дві кнопки, датчик температури, АЦП, і ці дані записані наступним чином bt0=0^bt1=0^t1=6A0E^adc1=30, де bt0, bt1 – умовні імена (змінні) першої та другої кнопок відповідно. Значення, що записуються під цими іменами можуть набувати значень 0 або 1 («ввімкнено», «вимкнено»). З виходу сенсору температури під ім'ям t1 буде записане відповідне значення, а із використанням АЦП – представлено у десятковому вигляді та записано під ім'ям adc1.

Отримані дані та ключ для шифрування вводяться у відповідні текстові поля програмних блоків програми, яка була створена для проведення апробації методу захисту передачі даних у кіберфізичній системі (рис. 2).

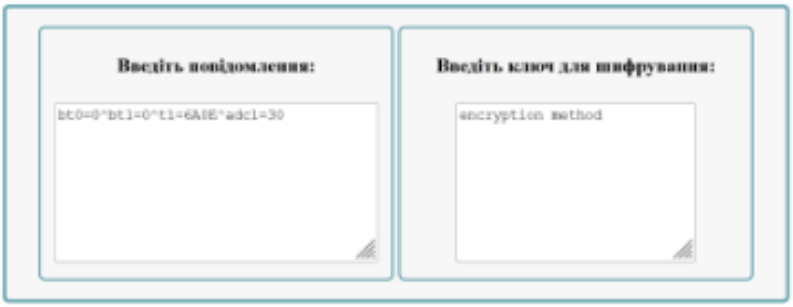


Рисунок 2 – Поля для введення вхідної послідовності даних кіберфізичної системи та ключа для проведення шифрування

Одночасно із введенням даних відбувається кодування (стиснення даних) (рис. 3)

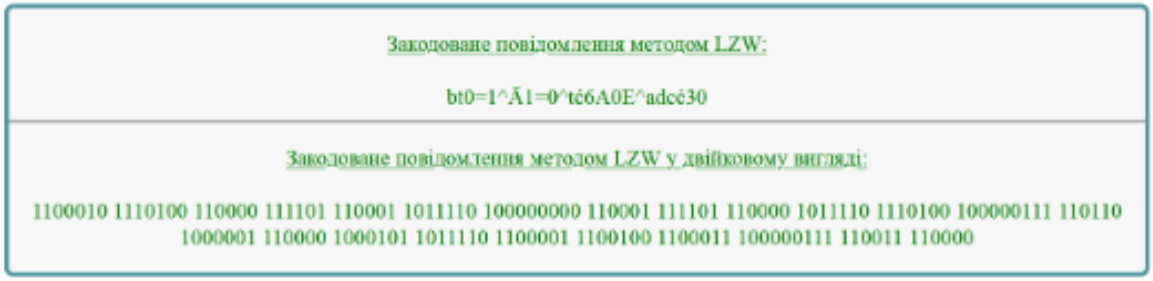


Рисунок 3 – Результат кодування (стиснення) даних кіберфізичної системи

Натиснувши клавішу Enter у полі «ключ для шифрування» отримаємо статистичні характеристики вихідних даних перед кодуванням та цих же даних після кодування (рис. 4), розділення символів на октети перед шифруванням (рис. 5) та результати шифрування закодованих даних (рис. 6):

Рисунок 4 – Результати кодування (стиснення) даних

На основі отриманих статистичних даних рис. 4 визначимо ентропію вхідних даних:

$$H_{вх} = \sum_{i=1}^{n} p_i \log_2 p_i = p_0 \log p_0 + p_1 \log p_1 + p_3 \log p_3 + p_6 \log p_6 + p_b \log p_b + p_t \log p_t +$$

$$+ p_- \log p_- + p_. \log p_. + p_A \log p_A + p_E \log p_E + p_a \log p_a + p_d \log p_d + p_c \log p_c = 0{,}185 \log_2 0{,}185 +$$

$$+ 3 \cdot 0{,}111 \log_2 0{,}111 + 7 \cdot 0{,}037 \log_2 0{,}037 + 0{,}0741 \log_2 0{,}0741 + 0{,}148 \log_2 0{,}148 = 3{,}42475 \text{ (біт/симв.)}$$

Підготовка до шифрування - розбиття символів на групи по 8 біт

Довжина повідомлення: 26 символи.
Об'єм даних: 208 біт

символ	у двійковій системі числення	у десятковій системі числення	кількість	ймовірність появи
	10000010	130	1	0.0385
	10000011	131	1	0.0385
	10001100	140	1	0.0385
	10001110	142	1	0.0385
	10010111	151	1	0.0385
½	10111101	189	1	0.0385
À	11000000	192	2	0.0769
Ç	11000111	199	1	0.0385
Ë	11001011	203	1	0.0385
Ø	11011000	216	1	0.0385
è	11101000	232	1	0.0385
	00011000	24	2	0.0769
	00000001	1	1	0.0385
	00000000	0	1	0.0385
H	01001000	72	1	0.0385
8	00111000	56	1	0.0385
0	00110000	48	2	0.0769
"	00100010	34	1	0.0385
&	00100110	38	1	0.0385
C	01000011	67	1	0.0385
	00011100	28	2	0.0769
f	01100110	102	1	0.0385

Рисунок 5 – Розділення даних на октети перед шифруванням

Результати шифрування даних
ключ: encryption method

Зашифроване повідомлення методом RC4:
,°úMx>3}0¦ ÃÏtÓ-x67F

символ	у двійковій системі числення	у десятковій системі числення
,	00101100	44
°	10110000	176
	10000110	134
ú	11111010	250
M	01001101	77
	10010101	149
x	01111000	120
>	00111110	62
3	00110011	51
}	01111101	125
0	00110000	48
¦	10100110	166
	00001010	10
Ã	11000011	195
Ï	11001111	207
t	01110100	116
¨	10101000	168
Ó	11010011	211
þ	11111110	254
-	00101101	45
x	01111000	120
	10000000	128
	00010010	18
õ	11110101	245
7	00110111	55
F	01000110	70

Рисунок 6 – Результат шифрування даних

Аналогічним чином було визначено ентропію появи символів за статистичним даними на рис. 4 закодованого (стисненого) потоку даних $H_{LZW} = 3{,}4282$ біт/симв. та ентропію появи символів за статистичним даними на рис. 5. потоку октетів, на які розділено перед шифруванням закодовану послідовність символів $H_{окт} = 4{,}3948$ біт/симв.

Різне значення ентропій показує, що частотний розподіл символів у вхідній послідовності та у закодованій послідовності значно різняться, а також відрізняється і їх кількість та об'єм.

Нижче наведено гістограми розподілу ймовірностей появи символів у вхідній послідовності символів (рис. 7) та гістограми розподілу ймовірностей появи символів створених октетів для шифрування (рис. 8).

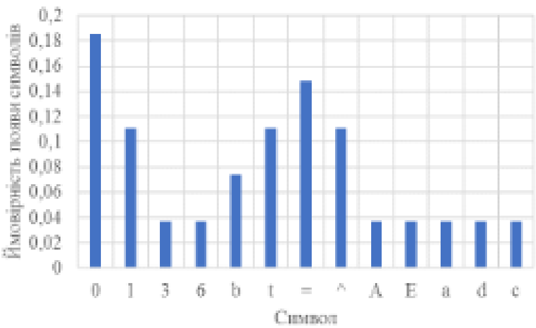


Рисунок 7 – Гістограма розподілу ймовірностей появи символів у вхідній послідовності символів

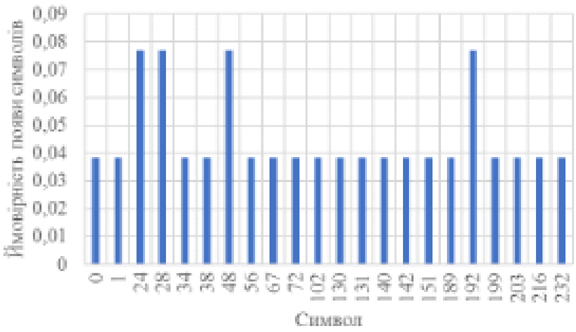


Рисунок 8 – Гістограма розподілу ймовірностей появи символів послідовності октетів, на які поділено закодовану послідовність

Усі отримані статистичні дані запишемо у зведеній таблиці 1.

Таблиця 1 – Зведений статистичний аналіз символів у вхідній та у закодованій послідовності

	Вхідна послідовність	Закодована послідовність	Октети, на які розподілено закодовану послідовність
Кількість символів алфавіту, що зустрічаються в послідовності	13	16	22
Ентропія, біт/симв.	3,42475	3,4282	4,3948
Довжина, симв.	27	23	26
Об'єм, біт	216	207	208

З отриманих у таблиці 1 даних можна відмітити наступне. У всіх трьох послідовностях (перша послідовність – вхідна послідовність, друга – послідовність закодованих символів, третя – послідовність октетів, на які розділено закодовану послідовність символів) різні ентропії, і до того ж значення ентропії постійно збільшуються починаючи від вхідної послідовності до закодованої і завершуючи коли вся послідовність розбивається на октети, а ймовірності появи символів навпаки зменшуються. Тобто передбачення кодової послідовності шляхом частотного аналізу шифрованої послідовності значно ускладнює процес дешифрування даних зловмисником. Об'єм вхідних даних та даних, що отримані після кодування, та розбиття закодованої послідовності на октети, відрізняється від початкового значення, що унеможливлює дешифрування тестової послідовності, яку надсилає зловмисник.

Висновки

У даній роботі було розглянуто удосконалення методу захисту передачі даних у кіберфізичній системі із використанням методу словникового стиснення та потокового шифрування. Використання методу кодування для даних кіберфізичної системи перед початком шифрування надає змогу змінити статистичні характеристики вхідної послідовності елементів (довжину повідомлення, об'єм даних, ентропію, ймовірності появи символів). Це саме ті статистичні характеристики, які потрібні зловмиснику для дешифрування даних. Надіславши тестову послідовність, зловмисник сподівається у результаті дешифрування цієї послідовності (виконується зворотна операція сума за модулем 2 до зашифрованої послідовності), отримати ключову послідовність, яка використовувалась під час шифрування. Маючу кодову послідовність та перехопивши наступні блоки інформації дешифрування їх не складе ніяких складнощів. Використання кодування перед шифруванням не дає змогу виділити вхідну тестову послідовність та отримати ключ.

Література

1. Смірнов О.А., Стасев Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

2. Гапак О.М. Захист інформації в комп'ютерних системах. Підручник / О.М. Гапак, С.І. Балога .- Ужгород: видавництво ПП «АУТДОР-ШАРК», 2021. – 184 с.

3. Гетьман, І. А. Технології захисту інформації: посібник для студентів вищих навчальних закладів спеціальності «Комп'ютерні науки» / І. А. Гетьман, О. В. Алтухов. – Краматорськ : ДДМА, 2018. – 120 с.

4. Беспроводная безопасность — шифрование [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/izuchite-besprovodnuiu-bezopasnost/besprovodnaia-bezopasnost-shifrovanie.

5. Ковтун Л.О. Основні ідеї та алгоритми адаптивного словникового кодування методом Лемпеля-Зіва [Текст] / Л. О. Ковтун, Д. М. Медзатий // Вісник Хмельницького національного університету. Технічні науки. – 2017. – № 4. – С. 129-136.

References

1. Smirnov O.A., Stasiev Yu.V. Barannik V.V. Zakhyst informatsii v avtomatyzovanykh systemakh upravlinnia. Navchalnyi posibnyk – Kharkiv: KhUPS, 2015. – 264 s.

2. Hapak O.M. Zakhyst informatsii v kompiuternykh systemakh. Pidruchnyk / O.M. Hapak, S.I. Baloha .- Uzhhorod: vydavnytstvo PP «AUTDOR-ShARK», 2021. – 184 s.

3. Hetman, I. A. Tekhnolohii zakhystu informatsii: posibnyk dlia studentiv vyshchykh navchalnykh zakladiv spetsialnosti «Kompiuterni nauky» / I. A. Hetman, O. V. Altukhov. – Kramatorsk : DDMA, 2018. – 120 s.

4. Besprovodnaya bezopasnost — shifrovanie [Elektronnij resurs]. – 2018. – Rezhim dostupu do resursu: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/izuchite-besprovodnuiu-bezopasnost/besprovodnaia-bezopasnost-shifrovanie.

5. Kovtun L.O. Osnovni idei ta alhorytmy adaptyvnoho slovnykovoho koduvannia metodom Lempelia-Ziva [Tekst] / L. O. Kovtun, D. M. Medzatyi // Visnyk Khmelnytskoho natsionalnoho universytetu. Tekhnichni nauky. – 2017. – № 4. – S. 129-136.

ДОДАТОК В

(обов’язковий)

Копії презентації:

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КІБЕРБЕЗПЕКИ

Корецька Людмила Олександрівна

Метод захисту передачі даних у кіберфізичних системах

Науковий керівник: к.т.н., доц. Кльоц Ю.П.

- **Метою магістерського дослідження** є підвищення захисту передачі даних у кіберфізчній системі бездротовим зв’язком за рахунок використання кодування вхідної послідовності символів, яка дозволяє змінювати статистичні властивості цієї послідовності, з подальшим її шифруванням потоковим шифром.
- **Об’єкт досліджень**: процес кодування та шифрування даних у кіберфізичній системі із використанням словникового алгоритму кодування та потокового шифру.
- **Предмет досліджень**: методи та алгоритми підвищення захисту передачі даних у кіберфізичній системі на основі потокових шифрів.
- Для досягнення мети поставлені наступні **завдання**:
  - провести аналіз існуючих методів кодування та шифрування даних, виділено їх недоліки та особливості, провести аналіз можливих атак при передачі даних бездротовим зв’язком у кіберфізичній системі;
  - навести математичну модель методу та провести її апробацію;
  - розробити вимоги до системи захисту, алгоритм методу захисту передачі даних та алгоритм оцінки ефективності методу захисту передачі даних;
  - провести моделювання запропонованого методу захисту передачі даних на даних різного об’єму.
- **Наукова новизна** отриманих результатів: удосконалено метод потокового шифрування даних, який полягає у використанні перед шифруванням алгоритмів кодування, що відрізняється від відомих зміною статистичного розподілу елементів вхідного повідомлення для підвищення захисту передачі даних у кіберфізичній системі.
- **Практична цінність**: проведене моделювання запропонованого методу захисту передачі даних у кіберфізичній системі показало їх ефективність і переваги у використанні порівняно із методом потокового шифрування.

Математична модель методу:

Математична модель методу захисту передачі даних описує два процеси: кодування та шифрування:

1) кодування:

$G = g_1 g_2 g_3 \dots g_m$ - текст (або рядок), що створений з деякого алфавіту $\Sigma = \{a_1, a_2, a_3, \dots, a_q\}$.

$c(G) = p_1 p_2 p_3 \dots p_v$, - закодований (стиснений) текст.

2) шифрування:

$l_{s[k]} = \log_2 n$ - довжина символів S-блоку ($l_{s[k]}$ - довжина i-ого елемента блоку підстановки S-блоку, $s[k]$ - елемент, який знаходиться на k-ому місці S-блоку).

$k_{blocks} = \frac{l_{c(S)}}{l_{s[k]}}$ - кількість блоків (розмір S-блоку)

$Y = y_0 y_1 \dots y_{k_{blocks}}$ послідовність октетів, на які ділиться закодована послідовність символів

S-блок – матриця з n елементів, в якій однакова кількість рядків та стовпців.

$$S = \begin{bmatrix} 0 & \dots & \sqrt{n}-1 \\ \cdot & & \cdot \\ \cdot & \dots & \cdot \\ \cdot & & \cdot \\ n-\sqrt{n} & \dots & n-1 \end{bmatrix}.$$

K-блок, де k_l, де l=0,1,…,L- ключ

$$K = \begin{bmatrix} k_0 & k_1 & \dots & k_9 & k_0 & \dots & k_5 \\ k_6 & & & \dots & & & k_1 \\ \cdot & & & \cdot & & & \cdot \\ \cdot & & & \cdot & & & \cdot \\ \cdot & & & \cdot & & & \cdot \\ \cdot & & & \cdot & & & \cdot \\ k_0 & k_1 & \dots & k_9 & k_0 & \dots & k_5 \end{bmatrix}.$$

Формування кожного стану схеми системи захисту відбувається за функцією переходів станів F

$$F = \begin{cases} j_t = (j_t + s[\sqrt{n}(j_t+1) + i_t] + k_l(i_t \bmod L)) \bmod n, \\ i_t = (i_t + 1) \bmod n, \\ \text{перестановка : } s[i_t] \text{ т } s[j_t]. \end{cases}$$

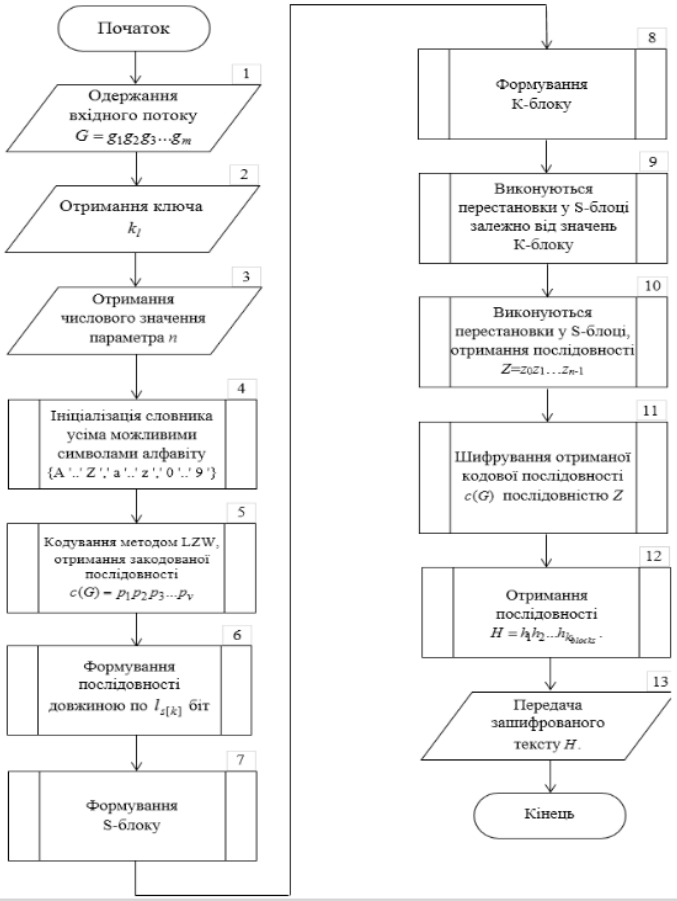
А функція виходу f дорівнює

$z_t = s_t[(s_t[i_t] + s_t[j_t]) \bmod n]$.

У результаті сформується послідовність $Z = z_0 z_1 \dots z_{n-1}$

Шифрування: $h_i = z_i \oplus y_i$. Результатом є послідовність $H = h_1 h_2 .. h_{k_{blocks}}$.

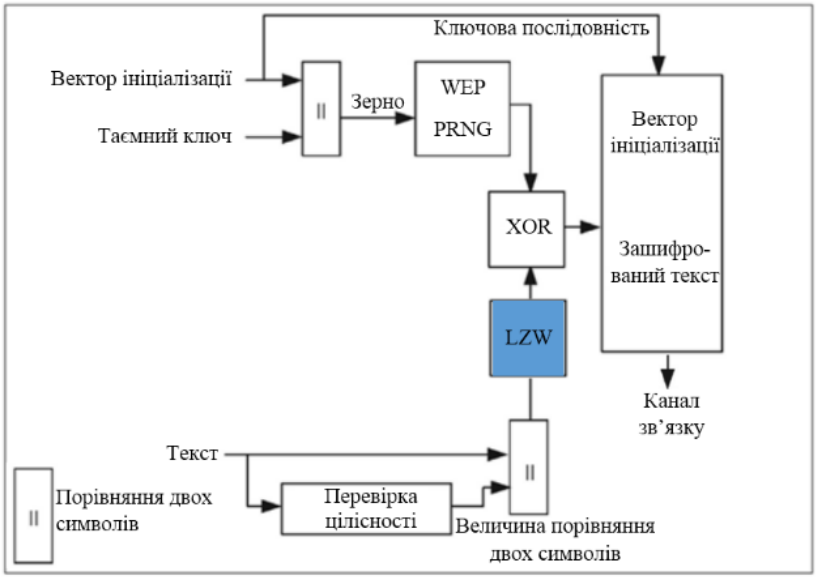
Алгоритм методу:



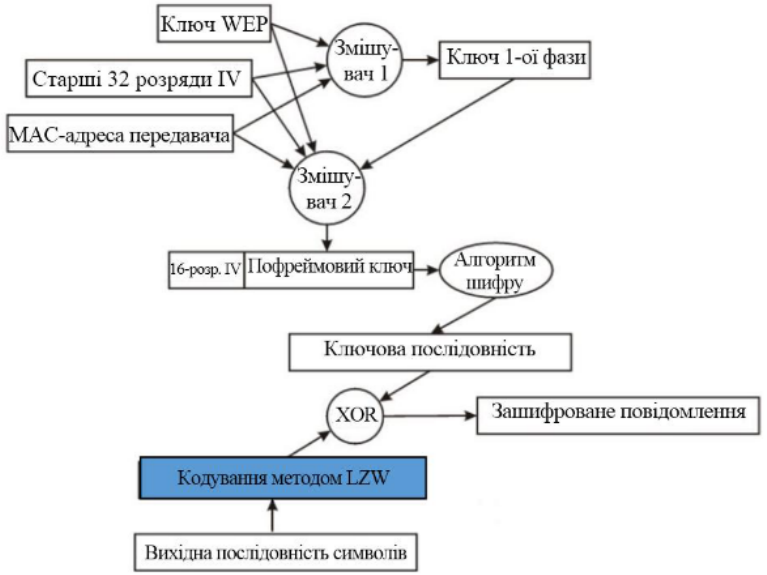
- блоки 1-3 – введення вхідних даних, які необхідні для реалізації методу захисту передачі даних;
- блоки 4-6 – відповідають за кодування кодової послідовності та отримання закодованої послідовності $c(G) = p_1 p_2 p_3 \ldots p_v$
- блоки 7-11 – шифрування даних
- блок 12 - отримання вихідної послідовності $H = h_1 h_2 .. h_{k_{blocks}}$.
- блок 13 – передача зашифрованого тексту H.

Запропонована реалізація методу у алгоритмах захисту даних:

WEP



WPA



Апробація запропонованого методу:

Введіть повідомлення:

bt0=0^bt1=0^t1=6A0E^adcc1=30

Введіть ключ для шифрування:

encryption method

Закодоване повідомлення методом LZW:

bt0=1^Á1=0^tć6A0E^adcć30

Закодоване повідомлення методом LZW у двійковому вигляді:

1100010 1110100 110000 111101 110001 1011110 100000000 110001 111101 110000 1011110 1110100 100000111 110110 1000001 110000 1000101 1011110 1100001 1100100 1100011 100000111 110011 110000

Результати кодування (стиснення) даних

Статистика символів до кодування:

Довжина повідомлення: 27 символи.
Об'єм даних: 216 біт

символ ASCII	символ ASCII у двійковому вигляді	позиція символу у таблиці ASCII	кількість	ймовірність появи
0	00110000	48	5	0.185
1	00110001	49	3	0.111
3	00110011	51	1	0.0370
6	00110110	54	1	0.0370
b	01100010	98	2	0.0741
t	01110100	116	3	0.111
=	00111101	61	4	0.148
^	01011110	94	3	0.111
A	01000001	65	1	0.0370
E	01000101	69	1	0.0370
a	01100001	97	1	0.0370
d	01100100	100	1	0.0370
c	01100011	99	1	0.0370

Символи у словнику LZW:

Довжина повідомлення: 23 символи.
Об'єм даних: 207 біт

символ ASCII	символ ASCII у двійковому вигляді	позиція символу у таблиці ASCII	кількість
100000000	256	1	0.0370
100000011	259	1	0.0370
100000111	263	2	0.0741
001100010	98	1	0.0370
001110100	116	2	0.0741
000110000	48	4	0.148
000111101	61	1	0.0370
001011110	94	3	0.111
000110001	49	1	0.0370
000110110	54	1	0.0370
001000001	65	1	0.0370
001000101	69	1	0.0370
001100001	97	1	0.0370
001100100	100	1	0.0370
001100011	99	1	0.0370
000110011	51	1	0.0370

Підготовка до шифрування - розбиття символів на групи по 8 біт

Довжина повідомлення: 26 символи.
Об'єм даних: 208 біт

символ	у двійковій системі числення	у десятковій системі числення	кількість	ймовірність появи
	10000010	130	1	0.0385
	10000011	131	1	0.0385
	10001100	140	1	0.0385
	10001110	142	1	0.0385
	10010111	151	1	0.0385
½	10111101	189	1	0.0385
Á	11000000	192	2	0.0769
Ç	11000111	199	1	0.0385
É	11001011	203	1	0.0385
Ø	11011000	216	1	0.0385
á	11101000	232	1	0.0385
	00011000	24	2	0.0769
	00000001	1	1	0.0385
	00000000	0	1	0.0385
H	01001000	72	1	0.0385
8	00111000	56	1	0.0385
0	00110000	48	2	0.0769
"	00100010	34	1	0.0385
&	00100110	38	1	0.0385
C	01000011	67	1	0.0385
	00011100	28	2	0.0769
f	01100110	102	1	0.0385

Результати шифрування даних
ключ: encryption method

Зашифроване повідомлення методом RC4:

,°ûMx>3}0¦ ÁÎr¨Ôi-xö7F

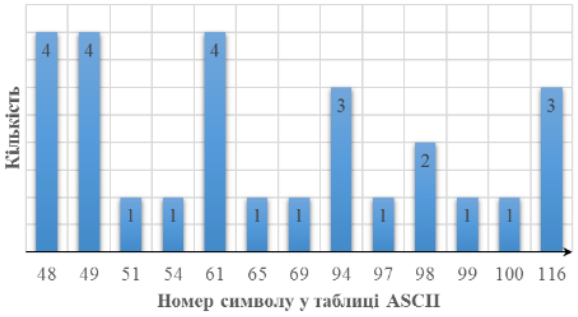
символ	у двійковій системі числення	у десятковій системі числення
,	00101100	44
°	10110000	176
	10000110	134
ú	11111010	250
M	01001101	77
	10010101	149
x	01111000	120
>	00111110	62
3	00110011	51
}	01111101	125
0	00110000	48
¦	10100110	166
	00001010	10
Á	11000001	193
Î	11001111	207
t	01110100	116
¨	10101000	168
Ô	11010011	211
î	11101110	238
-	00101101	45
x	01111000	120
	10000000	128
	00010010	18
õ	11110101	245
7	00110111	55
F	01000110	70

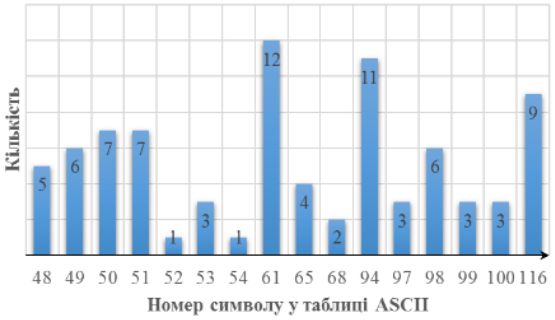
Зведені дані статистичних параметрів даних у кіберфізичній системі перед шифруванням:

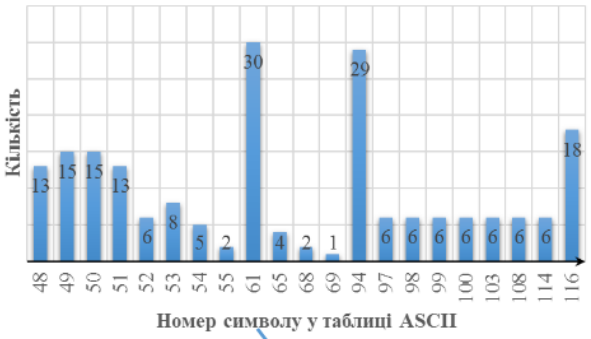
Файл	Параметр	Кількість символів алфавіту, що зустрічаються	Ентропія, біт/симв.	Довжина, симв.	Об'єм, біт
KPS1.txt	Вхідна послідовність	13	3,42475	27	216
	Закодована послідовність	16	3,4282	23	207
	Октети, на які розподілено закодовану послідовність	22	4,3948	26	208
KPS2.txt	Вхідна послідовність	16	3,717	83	664
	Закодована послідовність	36	3,64722	54	486
	Октети, на які розподілено закодовану послідовність	49	5,49375	61	488
KPS3.txt	Вхідна послідовність	21	3,98084	203	1624
	Закодована послідовність	60	3,5934	112	1008
	Октети, на які розподілено закодовану послідовність	89	6,34787	126	1008

Гістограми статистичних властивостей вхідного та закодованого тексту

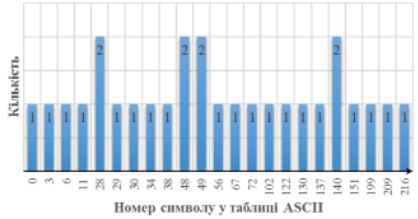
Вхідні послідовності символів різного об'єму:

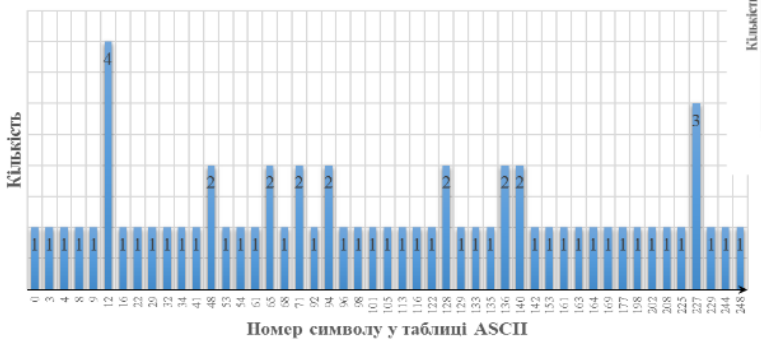


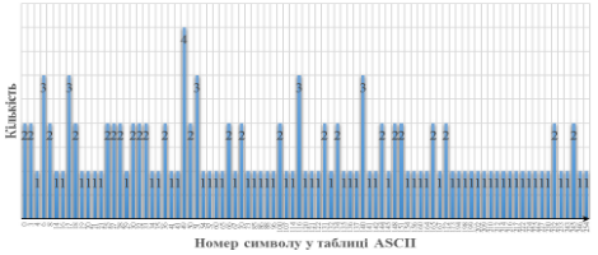




Закодовані послідовності символів різного об'єму:







Висновки:

- У кваліфікаційній роботі магістра було розроблено метод захисту передачі даних у кіберфізичній системі бездротовим зв’язком із сумісним використанням словникового методу кодування даних та потокового шифрування даних, який використовується у технологіях WEP та WPA:
- проведено аналіз існуючих методів кодування та шифрування даних, виділено їх недоліки та особливості, проведено аналіз можливих атак при передачі даних бездротовим зв’язком у кіберфізичній системі;
- наведено математичну модель методу та проведено її апробацію;
- розроблено вимоги до системи захисту, алгоритм методу захисту передачі даних та алгоритм оцінки ефективності методу захисту передачі даних;
- проведене моделювання за допомогою розробленого програмного коду методу захисту передачі даних на даних різного об’єму довело ефективність при його використанні.

ДОДАТОК Г

Статистичні властивості послідовностей символів різного об’єму

Таблиця Г.1 – Статистичні властивості вхідної послідовності у текстовому файлі KPS2.txt

символ ASCII	**символ ASCII у двійковому вигляді**	**позиція символу у таблиці ASCII**	**кількість**	**ймовірність появи**
0	00110000	48	5	0.0602
1	00110001	49	6	0.0723
2	00110010	50	7	0.0843
3	00110011	51	7	0.0843
4	00110100	52	1	0.0120
5	00110101	53	3	0.0361
6	00110110	54	1	0.0120
b	01100010	98	6	0.0723
t	01110100	116	9	0.108
=	00111101	61	12	0.145
^	01011110	94	11	0.133
A	01000001	65	4	0.0482
D	01000100	68	2	0.0241
a	01100001	97	3	0.0361
d	01100100	100	3	0.0361
c	01100011	99	3	0.0361

Таблиця Г.2 – Статистичні властивості закодованої послідовності, яка була отримана з текстового файлу KPS2.txt

символ ASCII у двійковому вигляді	**символ ASCII**	**кількість**	**ймовірність появи**
100000000	256	2	0.0241
100000011	259	1	0.0120
100000101	261	1	0.0120
100000111	263	1	0.0120
100001000	264	1	0.0120
100001010	266	2	0.0241
100001011	267	1	0.0120
100001100	268	1	0.0120
100001101	269	1	0.0120
100001111	271	1	0.0120
100010000	272	1	0.0120
100011000	280	1	0.0120
100011001	281	1	0.0120
100011011	283	1	0.0120
100011101	285	1	0.0120
100011111	287	1	0.0120
100100010	290	1	0.0120

Завершення табл. Г2

100101000	296	1	0.0120
100101110	302	1	0.0120
100101111	303	1	0.0120
001100010	98	1	0.0120
001110100	116	4	0.0482
000110000	48	3	0.0361
000111101	61	3	0.0361
000110001	49	2	0.0241
001011110	94	3	0.0361
000110010	50	3	0.0361
000110011	51	2	0.0241
000110100	52	1	0.0120
000110101	53	2	0.0241
001000001	65	2	0.0241
001000100	68	1	0.0120
001100001	97	1	0.0120
001100100	100	1	0.0120
001100011	99	2	0.0241
000110110	54	1	0.0120

Таблиця Г.3 – Статистичні властивості вхідної послідовності у текстовому файлі KPS3.txt

символ ASCII	**символ ASCII у двійковому вигляді**	**позиція символу у таблиці ASCII**	**кількість**	**ймовірність появи**
0	00110000	48	13	0.0640
1	00110001	49	15	0.0739
2	00110010	50	15	0.0739
3	00110011	51	13	0.0640
4	00110100	52	6	0.0296
5	00110101	53	8	0.0394
6	00110110	54	5	0.0246
7	00110111	55	2	0.00985
b	01100010	98	6	0.0296
t	01110100	116	18	0.0887
=	00111101	61	30	0.148
^	01011110	94	29	0.143
A	01000001	65	4	0.0197
D	01000100	68	2	0.00985
a	01100001	97	6	0.0296
d	01100100	100	6	0.0296
c	01100011	99	6	0.0296
E	01000101	69	1	0.00493
r	01110010	114	6	0.0296
l	01101100	108	6	0.0296
g	01100111	103	6	0.0296

Таблиця Г.4 – Статистичні властивості закодованої послідовності, яка була отримана з текстового файлу KPS3.txt

символ ASCII у двійковому вигляді	**символ ASCII**	**кількість**	**ймовірність появи**
100000000	256	2	0.00985
100000011	259	2	0.00985
100000100	260	2	0.00985
100000101	261	1	0.00493
100000111	263	3	0.0148
100001000	264	1	0.00493
100001001	265	2	0.00985
100001010	266	2	0.00985
100001011	267	1	0.00493
100001100	268	3	0.0148
100001101	269	1	0.00493
100001111	271	2	0.00985
100010000	272	1	0.00493
100010010	274	3	0.0148
100010101	277	3	0.0148
100011000	280	1	0.00493
100011001	281	1	0.00493
100011011	283	1	0.00493
100011101	285	1	0.00493
100011110	286	1	0.00493
100011111	287	1	0.00493
100100010	290	1	0.00493
100100101	293	1	0.00493
100101000	296	1	0.00493
100101110	302	1	0.00493
100101111	303	2	0.00985
100110001	305	1	0.00493
100110010	306	2	0.00985
100110101	309	2	0.00985
100110111	311	1	0.00493
100111111	319	1	0.00493
101000011	323	1	0.00493
101001000	328	2	0.00985
101010000	336	4	0.0197
101011000	344	1	0.00493
101011111	351	1	0.00493
101100000	352	2	0.00985
101100011	355	1	0.00493
101101100	364	1	0.00493
001100010	98	1	0.00493
001110100	116	7	0.0345
000110000	48	2	0.00985
000111101	61	3	0.0148
000110001	49	3	0.0148

Завершення табл. Г4

001011110	94	4	0.0197
000110010	50	7	0.0345
000110011	51	3	0.0148
000110100	52	2	0.00985
000110101	53	2	0.00985
001000001	65	2	0.00985
001000100	68	1	0.00493
001100001	97	1	0.00493
001100100	100	1	0.00493
001100011	99	3	0.0148
000110110	54	2	0.00985
001000101	69	1	0.00493
000110111	55	1	0.00493
001110010	114	2	0.00985
001101100	108	1	0.00493
001100111	103	2	0.00985

ДОДАТОК Д

Апробація методу кодування даних кіберфізичної системи методом LZSS

Для апробації методу захисту передачі даних в кіберфізичних системах було запропоновано використати декілька файлів із даними, які надходять від елементів кіберфізичної системи. Такими елементами можуть бути різноманітні кнопки, сенсори температури, сенсори рівня, сенсори тиску і т.д. Передбачається, що дані від сенсорів записуються у деякий файл. Опитування проводиться із деякою періодичністю залежно від вимог, що ставлять до певної кіберфізичної системи.

Відомо, що для того, щоб стиснення стало можливим, то в тексті повинні бути однакові елементи.

Проведемо дослідження для трьох різних файлів. Перший файл міститиме дані, що надходять від кіберфізичної системи, що містить в собі невелику кількість елементів. Наприклад, сенсор температури, дві кнопки, АЦП. Ці дані записані у файл KPS1.txt у наступному вигляді: bt0=1^bt1=0^t1=6A0E^adc1=30, де bt0, bt1 – умовні імена (змінні) першої та другої кнопок відповідно. Значення, що записуються під цими іменами можуть набувати значень 0 або 1 («ввімкнено», «вимкнено»). З виходу сенсору температури під ім’ям t1 буде записане відповідне значення, а із використанням АЦП – представлено у десятковому вигляді та записано під ім’ям adc.

Проведемо стиснення даних використавши компресор на основі методу LZSS. Результати наведено на рис. 4.1.

```
Input file size      : 27
Compressed           : 100% (-33%)
Compression time     : 0.03s (including fileIO)
Output file size     : 36
Compression ratio    : -33.33%
Compression ratio    : 10.667 bpb
```

Рисунок 4.1 – Результати стиску даних з файлу KPS1.txt

Аналіз отриманих даних показав, що використання методу LZSS для даних цього файлу показав протилежний результат: до стиску було 27 символів, після стиску – 36 символів, тобто на 33,33% більше. І це не дивно, оскільки для стиску потрібно мати однакові символів та їх комбінації.

Розглянемо кіберфізичну систему із більшою кількістю елементів. Наприклад, з шести кнопок, трьох сенсорів температури та АЦП, з виходу якого формуються дані про температуру та записуються у файл. Значення, що отримані з цих елементів, записані у файл KPS2.txt і виглядають наступним чином: bt0=1^bt1=0^bt2=1^bt3=0^bt4=1^bt5=0^t1=2AAD^adc1=25^t2=2AAD^ adc2=25^t3=3336^adc3=30. Візуально вже можна оцінити, що коефіцієнт стиску буде більший для цього файлу, оскільки багато однакових символів та груп символів. Проведемо стиск даних (рис. 4.2):

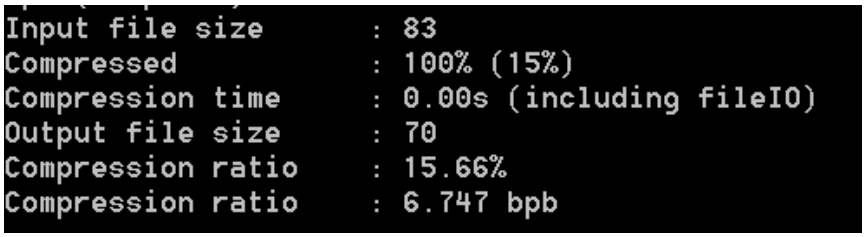


Рисунок 4.2 – Результати стиску даних з файлу KPS2.txt

Отриманий результат показує, що стиск відбувся на 15, 66%.

Проаналізуємо кіберфізичну систему ще з більшою кількість кнопок, сенсорів та відповідних даних АЦП. Наприклад, додатково введемо ще сенсори руху, сенсори світла. Отримані дані записані у файл KPS3.txt:

bt0=1^bt1=0^bt2=1^bt3=0^bt4=1^bt5=0^t1=2AAD^adc1=25^t2=2AAD^adc2 =25^t3=3336^adc3=30^t4=3336^adc4=30^t5=2E17^adc5=27^t6=2224^adc6=20^tr1 =0^tr2=1^tr3=0^tr4=0^tr5=0^tr6=1^lg1=1^lg2=0^lg3=0^lg4=1^lg5=1^lg5=1

Проведемо стиск отриманого файлу. Результати стиску на рис. 4.3

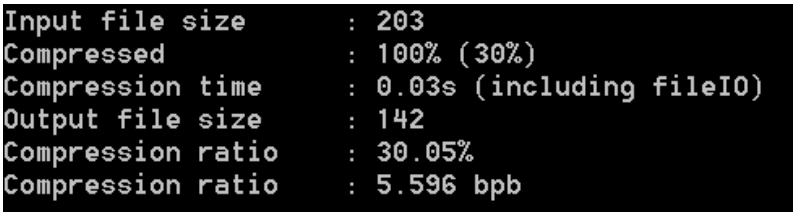


Рисунок 4.3 – Результати стиску даних з файлу KPS3.txt

Результати стиску зведемо до таблиці 4.1.

Таблиця 4.1 – Результати аналізу коефіцієнтів стиску даних різних кіберфізичних систем

Назва файлу	Розмір файлу до стиску, байт	Розмір файлу після стиску, байт	Коефієнт стиску, %	Час стиску, сек
KPS1.txt	27	36	-33,33	0,03
KPS2.txt	83	70	15,66	<0,01
KPS3.txt	203	142	30,05	0,03

Слід зазначити, що коефіцієнт стиску залежить не лише від об’єму даних, а й від їх повторюваності.

ДОДАТОК Е

Результати шифрування даних

Таблиця Е.1 – Зашифроване повідомлення посимвольно з файлу KPS3.txt

символ	у двійковій системі числення	у десятковій системі числення
8	00111000	56
y	01111001	121
D	01000100	68
{	01111011	123
Q	01010001	81
7	00110111	55
	10011011	155
¾	10111110	190
'	00100111	39
ú	11111010	250
X	01011000	88
i	01101001	105
k	01101011	107
)	00101001	41
r	01110010	114
õ	11110101	245
	00011101	29
ý	11111101	253
	10011000	152
±	10110001	177
ÿ	11111111	255
g	01100111	103
8	00111000	56
é	11101001	233
d	01100100	100
þ	11111110	254
	10000101	133
(	00101000	40
	00001100	12
Z	01011010	90
g	01100111	103
â	11100010	226
´	10110100	180
K	01001011	75
E	01000101	69
	00010001	17
+	00101011	43
^	01011110	94
\	01011100	92
ì	11101100	236
Á	11000001	193

Завершення табл. Е.1

Þ	11011110	222
□	10001011	139
	00010010	18
Õ	11010101	213
Ú	11011010	218
A	01000001	65
÷	11110111	247
i	01101001	105
	0	0
}	01111101	125
X	01011000	88
ë	11101011	235
5	00110101	53
¾	10111110	190
¢	10100010	162
¨	10101000	168
½	10111101	189
!	00100001	33
□	10011101	157
X	01011000	88

Таблиця Е.2 – Зашифроване повідомлення посимвольно з файлу KPS3.txt

символ	**у двійковій системі числення**	**у десятковій системі числення**
	00000101	5
#	00100011	35
	00000011	3
ø	11111000	248
t	01110100	116
Ü	11011100	220
¿	10111111	191
>	00111110	62
	00011010	26
£	10100011	163
w	01110111	119
A	01000001	65
å	11100101	229
¸	10111000	184
□	10000011	131
ø	11111000	248
□	10001100	140
Â	11000010	194
ý	11111101	253
ã	11100011	227
Ø	11011000	216

Продовження табл. Е.2

S	01010011	83
Ö	11010110	214
é	11101001	233
□	10000111	135
V	01010110	86
ì	11101100	236
□	10001011	139
c	01100011	99
ð	11110000	240
J	01001010	74
ô	11110100	244
	00011101	29
m	01101101	109
·	10110111	183
ì	11101100	236
	00010110	22
é	11101001	233
Û	11011011	219
à	11100000	224
S	01010011	83
□	10010000	144
@	01000000	64
	10101101	173
	00001000	8
·	10110111	183
7	00110111	55
î	11101110	238
□	10001010	138
ã	11100011	227
(	00101000	40
þ	11111110	254
r	01110010	114
□	10001010	138
Ì	11001100	204
Ì	11001100	204
d	01100100	100
ú	11111010	250
□	10010110	150
µ	10110101	181
L	01001100	76
°	10110000	176
□	10010101	149
Î	11001110	206
§	10100111	167
µ	10110101	181
q	01110001	113
1	00110001	49
Á	11000001	193

Продовження табл. Е.2

	00010111	23
□	10011101	157
□	10000010	130
-	00011110	30
É	11001001	201
Î	11001110	206
à	11100000	224
7	00110111	55
©	10101001	169
c	01100011	99
Á	11000001	193
è	11101000	232
Í	11001101	205
j	01101010	106
`	01100000	96
E	01000101	69
	00011100	28
Û	11011011	219
á	11100001	225
¬	10101100	172
□	10010000	144
□	10011110	158
□	10010111	151
Í	11001101	205
²	10110010	178
%	00100101	37
Ç	11000111	199
□	10001011	139
î	11101110	238
□	10000010	130
	00000111	7
	00010101	21
	00000011	3
æ	11100110	230
	00010011	19
,	00101100	44
¤	10100100	164
N	01001110	78
:	00111010	58
}	01111101	125
"	00100010	34
□	10011110	158
g	01100111	103
å	11100101	229
k	01101011	107
H	01001000	72
h	01101000	104
;	00111011	59

Завершення табл. Е.2

	00001110	14
Õ	11010101	213
¤	10100100	164
□	10010111	151
□	10010001	145
Ï	11001111	207
¦	10100110	166
	00010011	19
□	10011111	159

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «магістр»

Магістр Корецька Людмила Олександрівна
Тема Метод захисту передачі даних у кіберфізичних системах
Спеціальність 123 – Комп'ютерна інженерія

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «магістр»:

кількість листів креслень 9; кількість сторінок записки 89
1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі розроблено метод підвищення захисту передачі даних у кіберфізичній системі із використанням словникового методу стиску та потокового шифрування

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині роботи

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі подана загальна характеристика поставленої задачі, чітко визначено об'єкт, предмет та методи дослідження, сформульована актуальність. Визначені задачі, які необхідно вирішити для досягнення поставленої мети, практична цінність отриманих результатів, їхня новизна та наведені відомості про публікації. У першому розділі проведено огляд та аналіз відомих методів кодування та шифрування, визначено їх недоліки та переваги, обґрунтовано актуальність теми дослідження і виконана постановка задачі. В другому розділі виконано побудову математичної моделі запропонованого методу та проведено її апробацію. В третьому розділі визначено основні положення методу та розроблено алгоритм методу. Четвертий розділ присвячено апробації методу та моделюванню процесу захисту передачі даних кіберфізичних систем.

4. Позитивні сторони роботи Кваліфікаційна робота має комплексну наукову і практичну цінність. Наукова цінність полягає у розробці методу захисту передачі даних у кіберфізичній системі із використанням словникового методу стиснення та потокового шифрування. Практична цінність результатів дослідження полягає в обґрунтуванні вибору методу словникового кодування для руйнування статистичних характеристик вхідної послідовності символів та проведеному моделюванні запропонованого методу захисту передачі даних у кіберфізичній системі, що показало їх ефективність і переваги у використанні порівняно із методом потокового шифрування .

5. Негативні сторони роботи В роботі в недостатній мірі наведено деталізований опис програмних або програмно-апаратних рішень існуючих методів кодування та шифрування.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно, пояснювальна записка відповідає нормам щодо її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої мети та вирішення поставлених задач.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку відмінно

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
д.т.н., проф., зав. каф. автоматизації та комп'ютерно-інтегрованих технологій
Хмельницького національного університету Мартинюк Валерій Володимирович

« 06 » грудня 2021р. (підпис)

ХНУ. Зам. 54, тир. 6000, 2007.

UNICHECK

Ім'я користувача:
Кафедра кібербезпеки

ID перевірки:
1009635356

Дата перевірки:
10.12.2021 12:03:33 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.12.2021 12:05:15 EET

ID користувача:
100008300

Назва документа: **Записка Корецька - антиплагіат**

Кількість сторінок: **90** Кількість слів: **15365** Кількість символів: **108421** Розмір файлу: **3.14 MB** ID файлу: **1009636879**

6.86%
Схожість

Найбільша схожість: **3.19%** з Інтернет-джерелом (https://www.garrettcountyschools.org/search?page=273&term=2015-20

6.58% Джерела з Інтернету	185	Сторінка 92
2.54% Джерела з Бібліотеки	76	Сторінка 94

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	62

Fri Dec 10 11:11:32 EET 2021, Мостовий Сергій Володимирович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 10%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 98734 Название: Метод захисту передачі даних у кіберфізичних системах Добавлено в БД: 2021-12-10 Авторы: Корецька Л.О. Руководители: Кльоц Ю.П. Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	82890	822	835 (1%)	17 (2%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы