НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-науковою програмою «Комп'ютерні системи та мережі» зі спеціальності 123 «Комп'ютерна інженерія»
на тему: «Метод захищеної реалізації дискретного перетворення Фур'є в хмарах для ІОТ»

Виконала:
студентка ІІ курсу, групи ІО-21мн
Халіл Хана Ала Ель-Дін _____

Керівник:
доц. каф ОТ, к.т.н., доцент
Марковський Олександр Петрович _____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., професор
Кулаков Юрій Олексійович _____

Рецензент:
декан ФПМ, д.т.н., професор
Дичка Іван Андрійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність *123. Комп'ютерна інженерія*
(код і назва)

Освітньо-наукова програма *Комп'ютерні системи та мережі*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
«_____» _____ 2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Халіл Хані Ала Ель-Дін
(прізвище, ім'я, по батькові)

1. Тема дисертації Метод захищеної реалізації дискретного перетворення Фур'є в хмарах для ІОТ
Науковий керівник дисертації Марковський Олександр Петрович, доцент кафедри ОТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «25» березня 2024 р. № 1445-с

2. Строк подання студентом дисертації 28.05.2024

3. Об'єкт дослідження розподілена реалізація дискретного перетворення Фур'є в реальному часі з захищеним залученням віддалених комп'ютерних систем.

4. Предмет дослідження методи захищеної реалізації дискретного перетворення Фур'є в хмарах

5. Перелік завдань, які потрібно розробити: Огляд процедур перетворення Фур'є та аналіз перспектив їх захищеної реалізації. Розробка методу захищеної реалізації дискретного перетворення Фур'є з використанням попередніх сигналів. Розробка методу захищеної реалізації дискретного перетворення Фур'є з динамічною зміною кодів маски. Розробка програмних засобів моделювання запропонованого методу захищеної реалізації ДПФ.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	Кулаков Ю. О., професор		

7. Дата видачі завдання 13.10.22 _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	Затвердження теми дослідження.	13.10.22-15.12.22	
2.	Вивчення та аналіз літературних джерел.	16.12.22-21.03.23	
3.	Розробка методу реалізації захищеного дискретного перетворення Фур'є з використанням попередніх сигналів.	22.03.23-14.09.23	
	Розробка методу реалізації захищеного дискретного перетворення Фур'є з динамічною зміною кодів маски.	15.06.23-28.09.23	
4.	Теоретична оцінка ефективності запропонованих методів	29.09.23-18.11.23	
5.	Програмна реалізація запропонованих методів.	17.11.23-11.01.24	
6.	Тестування моделі та аналіз її ефективності.	12.01.24-20.03.24	
7.	Оформлення пояснювальної записки.	21.03.24-27.05.24	
8.	Передзахист.	27.05.24	
9.	Захист	12.06.24	

Студент _____Хана ХАЛІЛ_____ _____
(підпис)

Науковий керівник дисертації ___Олександр МАРКОВСЬКИЙ__ _____
(підпис)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Метод захищеної реалізації дискретного перетворення Фур'є в хмарах для IOT

студенткою: Халіл Ханою Ала Ель-Дін

Робота складається із вступу та 4 розділів. Загальний об’єм роботи: 94 аркушів основного тексту, 1 ілюстрація, 29 таблиць та 1 додатку. Для виконання магістерської дисертації було використано інформацію з 75 літературних джерел.

Актуальність.

Стрімкий розвиток хмарних технологій спричинив значні зміни в організації комп’ютерної обробки інформації. Оскільки застосовуючи хмарні системи користувачі отримують можливість використовувати потужні обчислювальні ресурси сучасних суперкомп’ютерів.

Для вирішення значної кількості прикладних задач, які пов’язані з потребою реалізації інтерфейсу між зовнішнім світом і комп'ютерною системою, використовуються методи цифрової обробки, а саме дискретне перетворення Фур’є. Велика кількість задач реалізації інтерфейсу повинна виконуватися в реальному часі, що потребує швидкої реалізації дискретного перетворення Фур’є.

Проблема пришвидшення реалізації дискретного перетворення Фур’є вирішується завдяки залученню віддалених комп’ютерних систем, використовуючи хмарні технології. Але, враховуючи той факт, що при використанні хмарних технологій не може бути забезпечена повна захищеність даних, що є важливим для деякої інформації, їх використання не набуває значного рівня.

Виходячи з переліченого вище вирішення задачі, яка полягає в створенні нового методу для реалізації захищеного дискретного перетворення Фур’є з

використанням віддалених комп’ютерних систем, стає актуальним та важливим на сучасному етапі розвитку технологій.

Мета і завдання дослідження. Мета досліджень полягає в підвищенні ефективності залучення хмарних ресурсів для дискретного перетворення Фур'є за рахунок збільшення рівня захищеності гомоморфного шифрування даних, що надсилаються на віддалені обчислювальні потужності.

Для досягнення поставленої мети було поставлено та вирішено такі завдання:

- аналіз існуючих методів та технологій гомоморфного шифрування сигналів, над якими виконуються перетворення Фур'є з залученням віддалених обчислювальних потужностей.
- теоретичне обґрунтування, розробка та дослідження нового методу для ефективної реалізації захищеного дискретного перетворення Фур’є у хмарах.
- моделювання роботи запропонованого методу на мові C++.
- проведення експериментальних досліджень для оцінки ефективності розробленого методу.
- порівняння теоретичних та експериментальних результатів розробленого методу реалізації захищеного дискретного перетворення Фур’є у хмарах з існуючими.

Об’єкт дослідження – розподілена реалізація дискретного перетворення Фур'є в реальному часі з захищеним залученням віддалених комп’ютерних систем.

Предмет дослідження – методи захищеної реалізації дискретного перетворення Фур’є в хмарах.

Методи досліджень. Для досягнення поставлених в магістерській роботі задач, використано методи теорії обчислень, теорії ймовірностей, статистичного аналізу, а також методи імітаційного моделювання.

Наукова новизна одержаних результатів роботи полягає у наступному:

Метод захищеної реалізації дискретного перетворення Фур'є в хмарах для IOT, який відрізняється організацією динамічної зміни кодів маски, завдяки чому досягається збільшення захищеності даних від спроби їх незаконної реконструкції на віддаленій комп’ютерній системі під час обробки.

Особистий внесок здобувача. Магістерське дослідження представляє собою самостійно проведену роботу, де відображено власний авторський підхід та отримані особисто теоретичні та практичні результати, що стосуються захищеної реалізації дискретного перетворення Фур'є в хмарах для IOT.

Практична цінність отриманих в результатів полягає в тому, що використання запропонованого методу гомоморфного шифрування сигналів при виконанні над ними дискретного перетворення Фур’є на віддалених комп’ютерних системах демонструє підвищення рівня захищеності даних.

Апробація результатів дисертації

Основні результати магістерської дисертації доповідались та обговорювались на міжнародній науково-технічній конференції Intelligence: proceedings of the International Conference ICSFTI2023, Kyiv, Ukraine, June 29-30 2023. м. Київ.

Публікації

За темою дисертації опубліковано статтю в фаховому виданні України.

Ключові слова

Дискретне перетворення Фур’є, хмарні комп’ютерні системи, віддалена обробка сигналів, гомоморфне шифрування.

ABSTRACT

for the master's thesis

«Method for Protected Implementation of Discrete Fourier Transform in Clouds for IoT»

author: Khalil Hana Ala El-Din

The thesis consists of an introduction and four chapters. The total volume of the work includes 94 pages of main text, 1 illustration, 29 table, and 1 appendix. Information from 75 literary sources was used to complete the master's thesis.

Relevance. The rapid development of cloud technologies has caused significant changes in the organization of computer data processing. Using cloud systems allows users to leverage the powerful computational resources of modern supercomputers. To address a significant number of applied tasks that involve the need to interface between the external world and the computer system, digital processing methods, specifically the discrete Fourier transform, are used. Many interface implementation tasks must be performed in real time, which requires fast implementation of the discrete Fourier transform.

The problem of accelerating the implementation of the discrete Fourier transform is addressed by involving remote computer systems using cloud technologies. However, since cloud technologies cannot ensure complete data security, which is crucial for some information, their use does not gain significant traction.

Given the above, solving the task of creating a new method for implementing a secure discrete Fourier transform using remote computer systems becomes relevant and important at the current stage of technological development.

Aim and Objectives of the Research. The aim of the master's thesis is to enhance the efficiency of leveraging cloud resources for discrete Fourier transform by increasing the level of security in homomorphic encryption of data sent to the remote computing systems.

To achieve this goal, the following tasks were set and resolved:

- Analysis of existing methods and technologies for homomorphic encryption of signals on which Fourier transforms are performed using remote computing systems.
- Theoretical justification, development, and investigation of a new method for the efficient implementation of secure discrete Fourier transform in the cloud.
- Model the proposed method in C++.
- Conduct experimental studies to evaluate the effectiveness of the developed method.
- Compare the theoretical and experimental results of the developed method for the secure implementation of the discrete Fourier transform in the cloud with existing methods.

Object of Research – distributed implementation of the discrete Fourier transform in real time with secure involvement of remote computer systems.

Subject of Research – methods of secure implementation of the discrete Fourier transform in the cloud.

Research Methods. To achieve the objectives set in the master's thesis, methods of computation theory, probability theory, statistical analysis, and simulation modeling were used.

Scientific Novelty of the Results. The method for protected implementation of discrete Fourier transform in clouds for IoT, which features the organization of dynamic changes in mask codes, thereby increasing data security against unauthorized reconstruction attempts on the remote computer system during processing.

Personal Contribution of the Author. The master's research represents an independently conducted work that reflects the author's own approach and personally obtained theoretical and practical results related to the protected implementation of discrete Fourier transform in clouds for IoT.

Practical Value of the Obtained Results. The use of the proposed method of homomorphic encryption of signals during discrete Fourier transform on remote computer systems demonstrates an increase in data security.

Approbation of the Dissertation Results. The main results of the master's thesis were reported and discussed at the international scientific and technical conference "Intelligence: Proceedings of the International Conference ICSFTI2023, Kyiv, Ukraine, June 29-30 2023," Kyiv.

Publications. An article on the thesis topic has been published in a professional journal of Ukraine.

Keywords. Discrete Fourier transform, cloud computer systems, remote signal processing, homomorphic encryption.

Пояснювальна записка

до магістерської дисертації

на тему: «Метод захищеної реалізації дискретного перетворення Фур'є в хмарах для IOT»

Київ – 2024

ЗМІСТ

ВСТУП

Стрімкий розвиток хмарних технологій привів до значних змін у способах організації комп’ютерної обробки інформації. Оскільки застосовуючи хмарні системи користувачі отримують можливість використовувати потужні обчислювальні ресурси сучасних суперкомп’ютерів. Завдяки появі такої можливості з’явилися передумови до вирішення більшої кількості задач, оскільки збільшилася швидкість обробки задач з більшою складністю.

Для вирішення значної кількості прикладних задач, які пов’язані з потребою реалізації інтерфейсу між зовнішнім світом і комп'ютерною системою, використовуються методи цифрової обробки, а саме дискретне перетворення Фур’є. Оскільки цей метод цифрової оброки забезпечує можливість аналізу вхідної інформації системою. Велика кількість задач реалізації інтерфейсу повинна виконуватися в реальному часі, що потребує швидкої реалізації дискретного перетворення Фур’є.

Проблема пришвидшення реалізації дискретного перетворення Фур’є вирішується завдяки залученню віддалених комп’ютерних систем, з використанням хмарних технологій. Не зважаючи на значне пришвидшення обробки інформації із залученням хмарних технологій, такий тип перетворення не використовується на значному рівні. Це відбувається через те, що певні типи інформації повинні бути конфіденційними, що не може бути забезпечено, оскільки використання хмарних технологій передбачає пересилку даних по відкритих каналах зв’язку.

Виходячи з переліченого вище вирішення задачі, яка полягає в створенні нового методу для реалізації захищеного дискретного перетворення Фур’є з використанням віддалених комп’ютерних систем, стає актуальним та важливим на сучасному етапі розвитку технологій.

РОЗДІЛ 1
ОГЛЯД ПРОЦЕДУР ПЕРЕТВОРЕННЯ ФУР'Є ТА АНАЛІЗ ПЕРСПЕКТИВ ЇХ ЗАХИЩЕНОЇ РЕАЛІЗАЦІЇ

На сучасному етапі розвитку комп'ютерних технологій реалізація інтерфейсу між комп'ютером та навколишнім середовищем є однією з найбільш пріоритетних задач [1]. Для забезпечення інтерфейсу між комп'ютерними системами та реальним світом використовується цифрова обробка сигналів Для більшості прикладних задач дуже важливо, щоб реалізація цифрової обробки сигналів відбувалася в реальному часі. З цього випливає, що значно зростають вимоги до швидкості виконання обчислень, які відбуваються під час перетворення вхідних сигналів. Для пришвидшення виконання операцій цифрової обробки сигналів, зокрема дискретного перетворення Фур'є, здійснюється розпаралелювання їх виконання [2]. Це стало можливим завдяки створення нових мікросхем, на яких відбувається розпаралелювання обчислень на апаратному рівні [3]. Відомі фірми створюють та випускають такі апаратні засоби, які було описано вище. Однією з таких компаній є Texas Instruments Inc, яка здійснює реалізацію декількох серій процесорів, які призначенні для здійснення швидкого перетворення Фур'є. Крім того для програмної реалізації основних операцій цифрової обробки віртуальних сигналів використовуються також і можливості потужностей призначених для цього графічних процесорів [4].

Завдяки виникненню хмарних технологій забезпечило широке коло користувачів доступом до практично усіх існуючих обчислювальних потужностей, що може бути корисно при вирішенні проблем, таких як наприклад, обробка цифрових сигналів та дискретні перетворення Фур'є [5]. Однак для повноцінного використання віддалених комп'ютерних систем користувачі повинні бути впевнені в повній конфіденційності даних, які вони відправлені за допомогою використання хмарних технологій. Нажаль без використання додаткових засобів захисту ця задача не може бути вирішена, оскільки дані, що

надсилаються на віддалені комп’ютерні системи відправляються по відритих каналах передачі інформації.

1.1. Аналіз особливостей використання перетворення Фур’є в сучасних системах комп’ютерного моніторингу стану об’єктів реального світу

Як вже було зазначено раніше, одним із найбільш важливих і фундаментальних методів при цифровій обробці сигналів є дискретне перетворення Фур’є. Це обумовлено тим фактом, що дискретне перетворення Фур’є зазвичай використовують для обробки таких типів даних як зображення та звуки, які представляють собою цифрові послідовності сигналів. Згідно з теоремою Фур’є будь-який сигнал, що існує в реальному світі можливо представити у вигляді набору його амплітуд та фаз синусоїд, які при сумуванні дають вхідний сигнал [3]. Для того щоб представити сигнал у вигляді набору синусоїд та амплітуд здійснюють виміри його значень через однакові проміжки часу, саме цей набір даних підлягає перетворенню.

Найбільш ефективним способом значного пришвидшення обчислень дискретного перетворення Фур’є є застосування хмарних технологій для доступу до більш потужних обчислювальних можливостей, завдяки отриманню доступу до глобальних мереж. Завдяки наявності такої можливості вирішується великий спектр і інших задач з великою обчислювальною складністю. Але крім переваг при використанні хмарних технологій є й свої недоліки, до яких відноситься неможливість забезпечення повної захищеності даних під час їх пересилки [6]. З чого випливає те, що будь-який зловмисник при наявності певних ресурсів може отримати несанкціонований доступ до даних, що відправляються.

Після проведення аналізу стало зрозуміло, що є потреба в захищеній відправці даних, що оброблюються на віддалених комп’ютерних системах з використанням дискретного перетворення Фур’є.

Згідно з такими потребами сучасних комп’ютерних технологій підтверджується актуальність вирішення такої проблеми як захищене перетворення Фур’є на віддалених комп’ютерних платформах.

В якості основних методів для здійснення обробки та аналізу зображень і сигналів частіше за все використовуються дискретне та швидке перетворення Фур'є [7]. З чого випливає, що такі типи процедур повинні бути забезпечені методами криптографічного захисту від несанкціонованого доступу до даних, якими вони оперують. Після проведення досліджень та аналізу відомих розробок зроблено висновки щодо неефективності використання методу [6], який полягає в інтегральному шифруванні, для здійснення організації захищеної фільтрації на віддалених комп'ютерних системах. З цього випливає, що для захищеної реалізації цих математичних операцій повинні бути організовані свої унікальні методи шифрування.

Крім того, велика кількість актуальних досліджень [8], яка присвячена вирішенню проблеми обробки сигналів з використанням операції захищеного перетворення Фур'є, вказує на той факт, що розробка нового унікального методу для захисту даних при реалізації цього методу є дійсно важливою [9].

Як вже було зазначено раніше, будь-який сигнал з фізичного світу можливо математично описати, якщо здійснити його виміри через фіксовані проміжки часу і представити їх у вигляді синусоїд. Кожна синусоїда в такому випадку може бути описана набором частот [8], які записуються у наступному вигляді $i \times w$, де i - це номер виміру $i = \{1, 2, 3, ...\}$, з амплітудами A_i і є зміщені по фазам j_i. Для збільшення якості аналізу та обробки вхідних даних зменшують часові проміжку, через які здійснюється вимір сигналу, відповідно до цього збільшується кількість числових значення, що призводить до збільшення кількості синусоїд. Саме тому збільшується кількість розрахунків, таким чином для більшої якості аналізу та обробки вхідних даних необхідно використовувати більш потужні обчислювальні потужності, інакше - сильно збільшується час, за який відбувається перетворення.

Згідно з характером обчислень [9], які відбуваються під час використання методу дискретного перетворення Фур'є вхідний сигнал зазвичай представляють у вигляді масиву із наступними значеннями $X = \{x_0, x_1, x_2, ... x_{n-1}\}$, де x_i - його

виміри через фіксовані проміжки часу, а n позначає їх кількість. Після здійснення перетворення Фур'є над вхідним масивом X утворюється масив складних чисел $Z = \{z_0, z_1, z_2, ..., z_{n-1}\}$, які складаються з синусоїдальних амплітуди та фази та які при здійсненні над ними операції сумування дають початковий сигнал. Описані вище дії виконуються згідно наступної формули [1.1]:

$$\forall l \in \{0,1,\dots,n-1\}: z_l = \sum_{i=0}^{n-1} x_i \cdot W_n^{iq}, \quad (1.1)$$

$$\text{де } W_n = e^{-j\cdot\frac{2\pi}{n}}$$

Із формули (1.1), яка описана вище можливо виділити компоненти синусоїди для кожного сигналу, у яких будуть частоти ω. Крім того, для більш зручної програмної реалізації складне число розділяють на уявну та реальну частини. Формула для обчислення реальної складової:

$$\forall i \in \{0,1,\dots,n-1\}: r_i = \frac{1}{n} \cdot \sum_{l=0}^{n-1} x_l \cdot \cos\frac{2\pi \cdot i \cdot l}{n}. \quad (1.2)$$

Формула для обчислення уявної складової:

$$\forall i \in \{0,1,\dots,n-1\}: y_i = \frac{1}{n} \cdot \sum_{l=0}^{n-1} x_l \cdot \sin\frac{2\pi \cdot i \cdot l}{n}. \quad (1.3)$$

Фактично, характеристики вхідного сигналу можливо виразити через його реальну та уявну складову, оскільки всі ці значення описують цей сигнал до та після перетворення [10]. З цього випливає, що якщо позначити через A_i амплітуду вхідного сигналу, а через φ_i його фазу, залежність характеристик сигналу від складових його спектрального представлення можливо записати у вигляді двох наступних формул.

Формула залежності амплітуди вхідного сигналу від реальної та уявної складової його спектрального представлення:

$$\forall i \in \{0,1,\dots,n-1\}: A_i = \sqrt{r_i^2 + y_i^2}. \quad (1.4)$$

Формула, яка виражає залежність фазового зсуву сигналу від реальної та уявної складової його спектрального представлення.

$$\forall i \in \{0,1,\dots,n-1\}: \varphi_i = \tan^{-1}\frac{y_i}{r_i} \tag{1.5}$$

При перетворенні сигналу найбільше ресурсів займають операції, які відбуваються у формулах (1.2) і (1.3), згідно із дослідженнями [10]. Для обчислення реальної або уявної компоненти сигналу за формулами (1.2) та (1.3), необхідно здійснити $2n^2$ операцій додавання та множення. Це твердження випливає з того факту, що необхідно здійснити $2n$ операцій множення та додавання для n відліків кожного сигналу. Це твердження доводить той факт, що при збільшенні кількості відліків, тобто значення n значно зростає складність перетворення сигналу, оскільки збільшується кількість операцій, які необхідно зробити. Для того щоб уникнути значного росту складності обчислень використовуються віддалені обчислювальні потужності, на яких пропонується виконати обрахунок формули (1.1). Оскільки обчислення значення кожного окремого відліку не залежить одне від одного [11], то можливо розпаралелити виконання перетворення сигналу, таким чином, щоб на віддаленій комп'ютерній системі воно виконувалося на n різних процесорах, що також впливає на збільшення швидкості виконання методу.

З формул (1.2) та (1.3) видно, що значення синусів та косинусів ніяк не залежать від відліків вхідного сигналу. З цього випливає, що обчислення цих значень можливо провести заздалегідь, оскілки вони залежать від даних, що можуть бути обчислені перед початком обробки вхідного сигналу. Через це, значення синусів та косинусів зазвичай обчислюють перед початком основної обробки сигналу у вигляді коефіцієнтів, розрахунок яких здійснюється за наступними формулами:

$$\forall i,l \in \{0,1,\dots,n-1\}: a_{il} = \frac{1}{n}\cdot\cos\frac{2\pi\cdot i\cdot l}{n}, \tag{1.6}$$

$$\forall i,l \in \{0,1,\dots,n-1\}: c_{il} = \frac{1}{n}\cdot\sin\frac{2\pi\cdot i\cdot l}{n}.$$

Виходячи із особливостей формули (1.6) стає цілком очевидно, що існує можливість для скорочення кількості здійснених операцій. Цей висновок зроблено на основі того факту, що коефіцієнти тригонометричних функцій, які лежать в основі формули (1.1) і знаходяться зліва від головної діагоналі мають коефіцієнти, будуть рівні симетричним елементам, які лежать справа від головної діагоналі. Цей факт є основною ідеєю, яка використовується в швидкому перетворенні Фур'є [12]. На відміну від дискретного перетворення Фур'є, використання швидкого перетворення Фур'є з урахуванням симетричності тригонометричних коефіцієнтів дозволяє зменшити кількість операцій додавання та віднімання з плаваючою комою до n^2. З чого випливає, що загальна кількість операцій, яка використовується для перетворення вхідного сигналу в числове представлення зменшилася в 2 рази [13].

Крім цього, якщо повертатися до ідеї застосування віддалених обчислювальних потужностей, швидке перетворення Фур'є виконується довше. Оскільки воно не може бути розпаралелене через використання властивості симетрії тригонометричних функцій, на відміну від дискретного перетворення Фур'є [14]. Таким чином, при віддаленій обробці даних використання методу дискретного перетворення Фур'є є більш доречним, тому що забезпечує більшу швидкість перетворення сигналу.

Якщо замість тригонометричних коефіцієнтів в основні формули (1.1) підставити значення з формули (1.6), їх можливо представити наступним чином:

$$r_i = \frac{1}{n} \cdot \sum_{l=0}^{n-1} x_l \cdot a_{il}, \tag{1.7}$$

$$y_i = \frac{1}{n} \cdot \sum_{l=0}^{n-1} x_l \cdot c_{il}.$$

Як було зазначено раніше у розділі існує необхідність організації зашифрованої реалізації дискретного перетворення Фур'є у хмарах, що означає передачу зашифрованого сигналу на віддалені комп'ютерні системи. Але при організації зашифрованої обробки сигналу необхідно враховувати один

важливий нюанс, мікроконтролер повинен мати можливість розшифрувати отримані значення реальної та уявної компонент спектру [15].

Виходячи із усього описаного раніше стає цілком очевидно, що використання віддалених комп'ютерних систем із залученням хмарних технологій для відправки інформації є надзвичайно важливим у процесі пришвидшення виконання дискретного перетворення Фур'є над вхідним сигналом [15]. Якщо припустити, що на віддаленій комп'ютерній платформі використовується незліченна кількість процесорів, і операції виконуються паралельно, то загальний час виконання операції перетворення сигналу T_{all} обчислюється за наступною формулою:

$$T_{all} = t_{mul} + \log_2 n \times t_{add}, \tag{1.8}$$

де t_{mul} - час виконання однієї операції множення значення з плаваючою комою, t_{add} - час виконання однієї операції додавання значення з плаваючою комою, а $\log_2 n$ - час за який відбувається завантаження результатів множення для подальшого виконання додавання.

Зазвичай системи, які потребують того, щоб організувати інтерфейс між реальним та віртуальним світом передбачають те, що цей зв'язок буде відбуватися в реальному часі [16]. Як приклади таких систем можливо навести системи, які здійснюють цілодобове спостереження за різноманітними об'єктами та місцями, і "очі" для роботів [16]. Виходячи із потреби здійснення обчислень в реальному часі постає необхідність пришвидшення здійснення процедур дискретного та швидкого перетворення Фур'є.

Через те, що вирішення цієї задачі є досить важливим для розвитку технологій на сучасному етапі, то почалися розробки мікрочіпів, які спеціалізуються на виконанні дискретного перетворення Фур'є і дозволяють прискорити реалізацію цього методу [17]. Ця задача є досить важливою, тому її вирішенням займається багато відомих компаній, наприклад Texas Instruments Inc, під керівництвом якої випускаються різні мікропроцесори, які є досить функціональними і дозволяють виконувати обчислення досить швидко.

Як вже було зазначено раніше одним із способів, завдяки якому можливо пришвидшити обробку вхідного сигналу є використання неосяжних можливостей віддалених комп’ютерних систем. Фактично, майже кожен користувач може отримати доступ до різноманітних віддалених обчислювальних потужностей [18] скориставшись хмарними технологіями, які реалізують зв’язок між різними пристроями. Фактично майже кожен сучасний пристрій, оснащений радіо модемом, отримує доступ до мережі Інтернет [19], і як наслідок до віддалених потужних обчислювальних можливостей .

Врешті решт, у сучасному технологічному світі, у якому існує необхідність реалізації інтерфейсу між навколишнім та віртуальним світом в режимі реального часу постає необхідність використання більш потужних систем для реалізації цифрових методів обробки сигналу [20]. Що робить відмову від використання віддалених потужних обчислювальних ресурсів вкрай нераціональною.

Крім того, що існує потреба в збільшенні ефективності за рахунок пришвидшення обчислень існує ще одна актуальна задача, яке полягає в використанні дискретного перетворення Фур’є в рамках технології IoT [21]. Оскільки передбачається, що вбудовані елементи управління в межах системи IoT можуть мати невеликий розмір і оснащені радіомодемами. З цього випливає, що існує потреба впровадження хмарних технологій для IoT, що дасть можливість зменшити площу пристроїв, завдяки залученню обчислювальних можливостей віддалених комп'ютерних систем.

Але крім значних переваг застосовування хмарних технологій для отримання доступу до значних обчислювальних потужностей віддалених комп’ютерів, у них є один значний недолік. Цей недолік полягає в використанні відкритих каналів передачі даних [22]. Які не можуть повноцінно забезпечити захист даних, тому зловмисник із достатнім рівнем бажання може отримати доступ до даних, що відправляються на віддалені комп’ютерні системи.

1.2. Аналіз вимог до реалізації спектральних перетворень в системах комп'ютерного моніторингу стану об'єктів з використанням хмарних технологій

Як вже було зазначено раніше головною проблемою використання віддалених комп'ютерних систем і хмарних технологій є передача даних по відкритих каналах зв'язку, внаслідок цього зловмисники можуть несанкціоновано втручатися в процес передачі даних або отримувати доступ до них. Актуальність цієї проблеми підтверджується великою кількістю випущених за останній час досліджень, які присвячено цій темі [23].

Минуло багато часу до того моменту як стала можлива реалізація захищеної обробки сигналів. Першим хто запропонував модель системи, яка була гомоморфною від початку і до кінця, став К. Джантрі [24]. Робота Джантрі, яка була опублікована у 2009 році, довела реальність створення систем шифрування, які були б повністю гомоморфними.

Основною ідеєю гомоморфного шифрування є можливість здійснення над зашифрованими даними певні математичні маніпуляції, і це не буде впливати на кінцевий результат при дешифруванні. Тобто, шифрування є гомоморфним за умови, що при здійсненні певних математичних операцій над цими даними, ці дії будуть еквівалентні до тих, що здійсненні над незашифрованими даними [25].

Гомоморфне шифрування зазвичай розглядають як логічне продовження ідеї про такі відомі криптографічні шифрування як симетричне та несиметричне. Гомоморфізм використовує такі перетворення, які не руйнують усі відносини між частинами даних. Гомоморфне шифрування поділяють на повністю гомоморфне, частково гомоморфне і так само як гомоморфне шифрування, залежно від типів операцій, які можна виконувати над зашифрованими даними [26].

Реалізація повністю гомоморфного шифрування стала можливою завдяки тому факту, що самі математичні операції такі як додавання і множення є гомоморфними [27]. Виходячи з того факту, що в гомоморфному шифруванні можливо виконати математичні операції над зашифрованими даними, потім

розшифрувати і отримати правильний результат, гомоморфним шифрування вважається за наступних умов:

$$D\big(E(b) + E(d)\big) = b + d, \qquad (1.9)$$
$$D(E(b) \times E(d)) = b \times d,$$

де $E(\times)$ – операції, які виконуються для здійснення шифрування, $D(\times)$ – операції, які виконуються для здійснення дешифрування, b та d – дані, над якими проводяться математичні операції.

Основними характеристиками повністю гомоморфного шифрування є можливість здійснення обчислення та оцінки над довільними схемами обмеженої та необмеженої глибини [28] відповідно. Ці характеристики є надзвичайно вагомою перевагою при використанні гомоморфного шифрування [29].

Частково гомоморфним типом називають такий тип шифрування, при якому справджується одна із тих умов, яку наведено у формулі (1.9). Як приклади схем, які є частково гомоморфними можливо навести шифрування RSA та схему Ель-Гамаля. Обидві схеми є схемами, які є гомоморфними до математичної операції множення [30].

З концепцією, яку створив Джантрі [31], якщо представити вхідні дані у вигляді числа d, над якими виконується операція шифрування, то для кожного такого числа буде існувати своя особиста матриця S. Ця матриця являє собою таку матрицю, яка при множенні на невідомий нікому крім особи яка здійснює шифрування ключ, що представляє собою вектор W. Описане твердження записується наступним чином:

$$d \times W = S \times W, \qquad (1.10)$$

З чого виходить, що необхідно знайти таку матрицю S, яка при множенні на вектор W дасть той самий результат, що і значення d помножене на W. Для того щоб знайти таку матрицю S необхідно просто розв’язати систему рівнянь для вектору S. За схожою схемою можливо знайти матрицю P для числа c, з використанням того ж самого вектору для шифрування W. Для отриманих значень доведено, що будуть справджуватися умови, які викладено у формулі

(1.9). З цього випливає що за схемою Джантрі будуть справедливими наступні значення для зашифрованих тверджень:

$$W \times (S + P) = d + c, \qquad (1.11)$$
$$W \times (S \times P) = d \times c,$$

Цілком очевидно, що для виконання дешифрування результату арифметичних операцій над зашифрованими даними з використанням вектору W, необхідно спочатку помножити на нього матрицю, після чого поділити на нього відповідні додатки [32]. Що призводить до того, що дані які шифруються з використанням вектору W, отримують свою матрицю, і саме шифрування може бути виконане на непідконтрольній користувачу системі.

Для того щоб оцінити складність зламу алгоритму гомоморфного шифрування зазвичай використовують критерій складності повного обчислення матриці [33].

Але разом із цим високу обчислювальну складність відносять до суттєвих недоліків розглянутої схеми. В принципі, як показують дослідження [34] висока обчислювальна складність притаманна більшості схем, які відносять до класу гомоморфних.

Аналіз загальної схеми гомоморфного шифрування дозволяє однозначно стверджувати, що в цьому практично ніякого сенсу [35]. Цей висновок було зроблено на основі аналізу можливостей цього шифрування, який показав, що дана схема є дуже громіздкою в плані проведення обчислювальних операцій. З чого випливає, що існує необхідність у розгляді можливості використання спеціалізованих схем гомоморфного шифрування, які можуть більше підходити для шифрування над якими здійснюється дискретне перетворення Фур'є.

1.3. Аналіз відомих схем гомоморфного шифрування перетворення Фур'є при їх обробці на віддалених комп'ютерних системах

Аналіз, який було проведено у попередніх частинах розділу доводить, що більшість досліджень, які спрямовані на вирішення проблеми забезпечення захищеної обробки даних розрахована на певні класи [36]. Як приклад можливо

навести матричні обчислення, обробку зображень, лінійну алгебру і т. ін. Для виконання операції модульного експоненціювання, яке являє собою спосіб, що є основним для більшості протоколів, що забезпечують захист даних, проводилася велика кількість досліджень [37]. Основною задачею цих досліджень було збільшення ефективності на облегшення виконання операції модульного експоненціювання. Спеціалізоване гомоморфне шифрування для виконання операцій обробки сигналів на віддалених комп'ютерних системах також стало популярним для дослідницьких робіт останніх років [38]. Отримання точних результатів після застосування методу спеціалізованого гомоморфного шифрування є наріжним каменем для всіх досліджень [39], які стосуються теми захищеного перетворення Фур'є на непідконтрольних користувачеві обчислювальних потужностях. З наданої вище інформації стає цілком зрозуміло, що існує потреба в створенні нових спеціалізованих методів для операцій обробки сигналів, оскільки вона продовжує бути досить популярною у сучасному науковому дискурсі [40].

У якості стратегій, які передбачають реалізацію захищених операцій з масованої обробки зображень пропонується використовувати математику та медіану ціллю яких є виконання фільтрації вхідних даних [41]. Описані методи базуються на інтервальному шифруванні інформації, яка передбачає точкову передачу їх перед відправкою на непідконтрольні системі обчислювальні потужності, де над ними буде виконуватися обробка. Даний підхід базується на блокуванні доступу до реального зображення шляхом виконання попередньої середньої фільтрації [42].

Враховуючи специфіку описаного методу стає цілком очевидно, що він не може бути ефективно використаний для реалізації дискретного перетворення Фур'є [43], саме тому необхідно розробити новий альтернативний метод виконання.

Враховуючи усі наведені вище факти стає цілком зрозуміло, що найбільш доречно організовувати дискретне перетворення Фур'є таким чином, щоб

реалізація цього методу складалася з двох частин. Ці дві частини пропонується поділити на меншу частину, реалізація якої буде відбуватися на користувацькій платформі і більшу, обчислення якої будуть здійсненні на непідконтрольних користувачу системах [44].

Для того щоб здійснити розробку найбільш ефективного за своєю реалізацією методу необхідно щоб він відповідав наступним найбільш стандартним характеристикам оцінювання [45]:

– Захищеність, яку забезпечує запропонований метод. Для того щоб з'ясувати яку захищеність забезпечує запропонований метол необхідно визначити кількість ресурсів, яку повинен витратити зловмисник для того щоб дізнатися дані, що пересилаються на непідконтрольні користувачу віддалені обчислювальні потужності [46];
– Складність обчислювального методу. Для того щоб оцінити складність запропонованого методу необхідно з'ясувати скільки операцій необхідно здійснити на платформі користувача для того щоб виконати дискретне перетворення Фур'є [47]. Значення цього критерію є дуже важливим, оскільки він є основою для того, щоб оцінити на скільки теоретично можливо збільшити швидкодії виконання процедури дискретного перетворення Фур'є з використанням цього методу.

Як вже було зазначено раніше збільшення ефективності виконання дискретного перетворення Фур'є є дуже важливим в сучасному науковому дискурсі, тому з кожним роком робіт, які стосуються цієї тематики стає все більше [48, 49].

В одній із робіт [50], яка присвячена вирішенню цього питання пропонується використання технології, основна ідея якої полягає у виконанні шифрування сигналів на платформі користувача, а перетворення Фур'є буде виконуватися на віддалених комп'ютерних потужностях. В якості основного механізму шифрування у даному методі використовується метод модулярного експоненціювання [51]. Характеристики, якими наділено даний метод

дозволяють досить суттєво змінювати рівень безпеки, який забезпечується описаним алгоритмом шифрування. Також цей метод забезпечує надзвичайно точні результати виконання дискретного перетворення Фур'є на непідконтрольних користувачеві обчислювальних потужностях. Але як би це не було прикро у всіх є свої недоліки, це твердження є справедливим і для цього методу, який через те що в нього закладено модулярне експоненціювання як метод шифрування, є досить громіздким в обчислювальному плані [52]. Основна складність описаного методу полягає в тому, що сам процес утворення ключів для модулярного експоненціювання є досить складним. Використання цього методу вважається недоцільним, оскільки виконання дискретного перетворення Фур'є відповідно до формул (1.1) і (1.2) буде набагато швидше ніж виконання модулярного експоненціювання для шифрування даних [53]. Теоретично можливо використовувати одні і ті самі ключі багаторазово, але на практиці це не має сенсу, оскільки тоді цей метод стає набагато менш захищеним.

В підсумку основним недоліком даного методу є надмірна складність його реалізації, або при спрощенні обчислень мала захищеність [54]. З цього випливає, що метод, що описано вище не відповідає одночасно двом стандартним характеристикам ефективності.

Також значною популярністю користується метод виконання захищеного перетворення Фур'є на віддалених комп'ютерних системах [55], який полягає в застосуванні методу, який спирається на адитивне маскування сигналів.

Для того щоб з'ясувати чи є описаний метод дійсно ефективним необхідно провести оцінку того факту, чи він забезпечує виконання наступних стандартних характеристик:

– Збільшення рівня захищеності, яку забезпечує метод;
– Значне зменшення складності обчислень при використанні описаного методу.

Зазначений метод дозволяє значно зменшити кількість операцій, яку потрібно виконати на платформі користувача [56]. Це твердження доводить той факт, що для того щоб виконати дискретне перетворення Фур'є з використанням

цього методу на платформі користувача здійснюється всього $2\times n^2$ операцій суми та добутку чисел з плаваючою комою. Відповідно до зменшення кількості операцій зростає швидкість з якою виконується дискретне перетворення Фур'є в чотири рази [57]. Крім того після проведення практичної перевірки цього методу стало зрозуміло, що його використання дозволяє пришвидшити виконання дискретного перетворення Фур'є приблизно в $6\cdot10^3$ рази [58]. З чого випливає, що застосування цього методу призводить до значного підвищення продуктивності з точки зору швидкості обробки даних, причому важливу роль в цьому процесі відіграє використання віддалених обчислювальних потужностей із залученням хмарних технологій для пересилки даних.

Як вже було зазначено вище для того щоб з'ясувати яку захищеність забезпечує запропонований метод [59] необхідно визначити кількість ресурсів, яку повинен витратити зловмисник для того щоб дізнатися дані, що пересилаються на непідконтрольні користувачу віддалені обчислювальні потужності. Крім того необхідно з'ясувати чи є у методу, який підлягає перевірці якихось недоліків, які суттєво впливають на можливість отримання зловмисником даних, які підлягають обробці на непідконтрольних користувачу системах. Як виявилося цей метод має такий недолік, який полягає в тому, що зловмисник може здійснити несанкціоноване отримання доступу до інформації обробці в процесі відправки даних на віддалену комп'ютерну систему для подальшого виконання дискретного перетворення Фур'є над отриманими значеннями [60]. Після перехоплення даних зловмисник з легкістю може відновити дані до того стану поки вони не були зашифровані. Це відбувається через те, що здійснивши перехоплення двох масивів даних, які ідуть послідовно, можливо відновити зашифровану інформацію . Зважаючи на особливість метода можливо розшифрувати дані виконуючі прості математичні операції.

В підсумку основним недоліком даного методу є недостатня захищеність при його реалізації [61]. З цього випливає, що метод, який описано вище

відповідає лише одному з двох зазначених стандартних характеристик ефективності.

Для більш повного аналізу ситуації, в якій зараз знаходиться рівень розробки методів для більш ефективної реалізації дискретного перетворення Фур'є у представленій роботі розглянуто ще один метод шифрування. Цей метод шифрування базується на виконанні потокового шифрування інформації, яка надходить на віддалені комп'ютерні системи з мікроконтролеру, який використовує хмарні технології, як середовище передачі даних [62]. Даний метод можливо охарактеризувати значною простотою обчислень, яка забезпечується шляхом значного спрощення процесів шифрування та дешифрування порівняно з іншими відомими методами. В результаті цього забезпечується значне пришвидшення часу який витрачається на те щоб здійснити дискретне перетворення Фур'є для вхідного масиву даних [63].

Але у цього методу є значний недолік, який випливає з того факту, що він, через свої особливості може бути застосований лише для певного класу задач, що значно обмежує область у якій може бути використано описаний метод [64].

В підсумку основним недоліком даного методу є невелика область застосування [65]. З цього випливає, що незважаючи на те що метод, який описано вище відповідає двом зазначеним стандартним характеристикам ефективності, він є досить обмеженим у використанні.

Огляд відомих методів реалізації захищеного дискретного перетворення Фур'є показує, що жоден із них не забезпечують належного рівня ефективності [66]. Оскільки два перші методи не відповідають одразу двом найбільш стандартним характеристикам оцінювання, а останній можливо використовувати лише для певних типів задач.

Виходячи з усього описаного вище стає зрозуміло, що дійсно існує потреба в створенні нового методу, що забезпечить захищену реалізацію Фур'є на непідконтрольних користувачу платформах і буде більш ефективним, тобто

відповідатиме стандартним характеристикам оцінювання та буде більш універсальним.

Висновки до розділу 1

Після огляду процедур перетворення Фур'є та аналізу перспектив реалізації цього методу обробки сигналів на віддалених обчислювальних потужностях з використанням хмарних технологій можуть бути сформульовані наступні висновки:

1. Аналіз сучасних тенденцій у розвитку комп'ютерних технологій показав, що зростання популярності Інтернету речей робить пріоритетною задачу реалізації інтерфейсу між комп'ютером та навколишнім середовищем. Для реалізації такої взаємодії широко використовується цифрова обробка сигналів, заснована на методах перетворення Фур'є. Існує значна потреба у підвищенні якості аналізу та обробки вхідних даних, що вимагає збільшення кількості відліків для кожного сигналу, призводячи до значного збільшення часу на обчислення. Крім того, існують жорсткі вимоги щодо часу перетворення сигналу, оскільки більшість термінальних мікроконтролерів працюють у реальному часі та обробляють велику кількість даних, що потребує високої швидкості обробки. Для підвищення швидкості виконання дискретного перетворення Фур'є використовуються можливості віддалених комп'ютерних потужностей, доступ до яких забезпечується завдяки хмарним технологіям. Важливо зазначити, що мікроконтролери, які складають основу систем Інтернету речей, часто мають доступ у мережу, що дозволяє їм взаємодіяти з хмарними ресурсами. Проте, існує проблема використання непідконтрольних користувачу віддалених комп'ютерних систем, що вимагає додаткових заходів безпеки.

2. Проаналізовано вимоги, що висуваються до реалізації спектральних перетворень в системах комп'ютерного моніторингу стану об'єктів з використанням хмарних технологій. Внаслідок аналізу цих потреб сформовано дві характеристики для оцінки ефективності запропонованого варіанту реалізації дискретного перетворення Фур'є. В якості першої характеристики використовується рівень захищеності, яку забезпечує метод. В якості другої

характеристики оцінки ефективності використовується складність обчислювального методу.

3. Здійснено аналіз відомих схем гомоморфного шифрування перетворення Фур’є при їх обробці на віддалених комп’ютерних системах. В результаті огляду сучасних схем захищеної реалізації дискретного перетворення Фур’є виявлено, що жоден із них не забезпечує належного рівня ефективності через невідповідність вимогам, які висуває сучасний технологічний світ. Це стосується і потреб Інтернету речей, де швидкість обробки даних та їх захищеність є вирішальними факторами. Тому розробка нових методів, які враховують специфіку Інтернету речей, є важливим кроком у забезпеченні безпеки та ефективності обробки даних у цій сфері.

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ ЗАХИЩЕНОЇ РЕАЛІЗАЦІЇ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ФУР'Є З ВИКОРИСТАННЯМ ПОПЕРЕДНІХ СИГНАЛІВ

2.1. Розробка методу захищеної реалізації дискретного перетворення Фур'є з використанням попередніх сигналів

Запропонований метод базується на адитивному маскуванні відліків сигналів. Основна ідея запропонованого методу, яка різнить його від відомих, полягає в використанні інших сигналів що надходять на систему для шифрування поточного. Відповідно, розроблений метод орієнтований на застосування до сукупності послідовних сигналів $X_1, X_2, X_3, \dots$, причому кожен сигнал складається з n відліків, тобто $X_j = \{x_{j1}, x_{j2}, x_{j3}, \dots, x_{jn}\}$. Тому далі використовуються індекси j та i для позначення відповідно номеру сигналу та відліку. Крім інших сигналів, що надходять на систему, для шифрування у запропонованому методу використовується сукупність з p маскувальних сигналів $M_1, M_2, M_3, \dots, M_p$. Причому кожен маскувальний сигнал M_k складається з n відліків, тобто $M_k = \{m_{k1}, m_{k2}, m_{k3}, \dots, m_{kn}\}$, які є випадково згенерованими значеннями. Для реалізації шифрування, яке передбачає застосування інших сигналів, використовується пул H у якому вони зберігаються. Цей пул є матрицею, яка містить t рядків і n стовпців та постійно оновлюється шляхом запису до неї вхідних сигналів.

Крім того, у запропонованому методі використовується змінна f, яка набуває бінарних значень, для випадкового вибору даних якими шифрується вхідні значення, тобто маскою або іншим сигналом.

Відповідно до запропонованого методу, кожен відлік замаскованого сигналу Q_j, такого, що $Q_j = \{q_{j1}, q_{j2}, q_{j3}, \dots, q_{jn}\}$ визначається наступним чином:

$$j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}, k \in \{1,2,3,\dots,p\}, d \in \{1,2,3,\dots,p\}:$$

$$q_{ji} = x_{ji} + (1 - f) \cdot m_{ki} + f \cdot h_{di}.$$

Для реалізації захищеного перетворення Фур'є з використанням хмарних технологій запропонований метод передбачає виконання двох етапів:

1. Генерація масок та обчислення їх компонент спектрального представлення.
2. Гомоморфне шифрування сигналу, яке передбачає використання маскувального сигналу або одного з попередніх сигналів, перед посилкою його для обробки в хмару, а також дешифрування прийнятого з віддаленої комп'ютерної системи результату перетворення Фур'є, тобто спектрального представлення сигналу.

Перший етап:

1. Формується набір маскувальних сигналів. Перед початком обробки вхідних сигналів запропонований метод передбачає формування набору із p масок M_k для $\forall\ k = \{1, 2, 3, ..., p\}$. Оскільки кожна маска M_k складається з n відліків, тобто $M_k = \{m_{k1}, m_{k2}, m_{k3}, ..., m_{kn}\}$, то для формування кожного маскувального сигналу ініціалізується робота генератора випадкових чисел, з використанням якого утворюється послідовність із n випадкових чисел.
2. Обчислюються значення реальної та уявної компонент спектрального представлення маскувальних сигналів. Над p маскувальними сигналами здійснюється дискретне перетворення Фур'є, за наступними формулами:

$$i \in \{1,2,3, \dots, n\},\ s_i = \sum_{l=0}^{n} a_{li} \cdot m_i,\ b_i = \sum_{l=0}^{n} c_{li} \cdot m_i,$$

де s_i та b_i - значення реальної та уявної компоненти спектру відповідно.

Другий етап:

1. Формується пул попередніх сигналів. Запропонований метод передбачає запис перших t значень, що надходять на систему, до пулу попередніх сигналів. Значення записані до пулу попередніх сигналів зберігаються у відповідному масиві для шифрування наступних сигналів.
2. Визначення типу даних, які використовуються для шифрування. Генерується випадкове бінарне значення і записується у змінну f, якщо $f = 0$, то для шифрування використовуються відліки масок, а якщо $f = 1$, то - значення попередніх сигналів. Для шифрування перших t значень з пулу

попередніх сигналів використовуються лише відліки масок, тобто змінна $f = 0$, за умови що $j < t$.

3. Визначення номеру маскувального або попереднього сигналу, які використовуються для шифрування:
   - Якщо $f = 0$, то генерується випадкове значення k таке, що $k \in \{1, 2, 3, ..., p\}$.
   - Якщо $f = 1$, то генерується випадкове значення d таке, що $d \in \{1, 2, 3, ..., t\}$.
4. Гомоморфне шифрування сигналу. До відліків X_j додаються значення з набору масок або пулу попередніх сигналів у відповідності із згенерованими значенням у пунктах 2-3. В результаті обчислень формується зашифрований сигнал Q_j.
5. Дискретне перетворення Фур'є. Зашифрований сигнал Q_j відправляється на віддалену комп'ютерну систему, на якій над ним здійснюється дискретне перетворення Фур'є, згідно з наступними формулами:

$$\forall\, l \in \{1,2,3,\dots,n\}, i \in \{1,2,3,\dots,n\}, w_i = \sum_{l=1}^{n} a_{il} \cdot q_l,\ v_i = \sum_{l=1}^{n} c_{il} \cdot q_l, \quad (2.1)$$

де w_i та v_i - отримані від віддаленої комп'ютерної системи обчислені значення реальної та уявної компоненти спектру відповідно.

6. Дешифрування отриманих результатів. Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур'є. Дешифрування відбувається шляхом віднімання від кожного відліку компонент спектру відповідних значень масок або попередніх вхідних сигналів, які використані для шифрування згідно зі згенерованими значенням у пунктах 2-3. Таким чином дешифрування отриманих результатів відбувається відповідно до наступних формул:

$$r_{ji} = w_{ji} - (1 - f) \cdot s_{ki} - f \cdot r_{di},\ y_{ji} = v_{ji} - (1 - f) \cdot b_{ki} - f \cdot y_{di}. \quad (2.2)$$

7. Запис даних. Після здійснення дешифрування попередній сигнал, який використовувався для шифрування, і обчислені для нього компоненти

спектрального представлення видаляються з пам’яті, а значення отримані після дешифрування записуються до пам’яті.

Функціонування запропонованого методу може бути проілюстровано за допомогою наступного прикладу. Нехай система передбачає перетворення сигналів, що складається із 8 відліків (n = 8), кількість масок, що входять до маскувального набору дорівнює 3 (k = 3), а у пулі попередніх сигналів зберігається 3 значення (t =3). Відповідно до першого етапу запропонованого методу перед початком обробки сигналів здійснюється генерація масок та обчислення їх спектрального представлення. Результат генерації маскувальних сигналів, кількість відліків яких відповідає умовам наведеним вище представлено у таблиці 2.1.

Таблиця 2.1. - Приклад значень маскувальних сигналів та їх відліків

Маски	Відліки							
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
M_1	2,36	0,34	1,28	0,22	1,27	3,05	1,91	3,09
M_2	3,21	2,67	0,07	0,32	2,63	1,08	0,11	3,06
M_3	0,22	0,33	1,37	1,65	2,31	1,72	3,04	0,83

Після генерації масок здійснюється обчислення їх реальної та уявної компонент спектрального представлення. Результати обчислень представлено у таблиці 2.2.

Відповідно до запропонованого методу до перших вхідних сигналів додаються випадкові маскувальні значення. Приклад перших трьох сигналів наведено у таблиці 2.3. Формується випадкове значення k таке, що k=2. Відповідно до цього до відліків вхідного сигналу X_1 додаються значення маски M_2. В результаті обчислень формується зашифрований сигнал Q_1, який представлено у таблиці 2.4.

Таблиця 2.2. - Значення спектрального представлення реальних та уявних компонент для маскувальних сигналів

Компоненти	Відліки							
	1	2	3	4	5	6	7	8
S_1	13,52	1,20	0,44	0,98	0,12	0,98	0,44	1,20
B_1	0,00	4,58	-0,08	3,32	-0,00	-3,32	0,08	-4,58
S_2	13,15	3,64	5,66	-2,48	-1,11	-2,48	5,66	3,64
B_2	0,00	0,85	-0,37	0,77	-0,00	-0,77	0,37	-0,85
S_3	11,47	-3,65	-1,88	-0,53	2,41	-0,53	-1,88	-3,65
B_3	0,00	2,07	0,43	-1,27	0,00	1,27	-0,43	-2,07

Таблиця 2.3. - Приклад перших 3 отриманих системою сигналів та їх відліків

Сигнали	Відліки							
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
X_1	2,03	0,07	2,54	1,63	0,92	2,72	2,44	0,13
X_2	3,24	0,16	2,97	0,56	1,80	1,03	1,65	1,25
X_3	1,70	0,73	3,27	1,16	2,18	2,57	0,59	1,58

Таблиця 2.4. - Замаскований перший сигнал та його відліки

Сигнал	Відліки							
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Q_1	5,24	2,74	2,61	1,95	3,55	3,80	2,55	3,19

Зашифрований сигнал Q_1 відправляється на віддалену комп'ютерну систему, на якій над ним здійснюється дискретне перетворення Фур'є, згідно з формулами (1.2 та 1.3). Отримані від віддаленої комп'ютерної системи обчислені значення реальної та уявної компоненти спектру записані у таблиці 2.5. Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур'є. Дешифрування відбувається

шляхом віднімання від кожного відліку W_1 та V_1 значень компонент спектру S_2 та B_2. Результат дешифрування отриманих зафіксовано у таблиці 2.5.

Таблиця 2.5. - Зашифровані та дешифровані компоненти спектрального представлення сигналу X_1

Компоненти	Відліки							
	1	2	3	4	5	6	7	8
W_1	25,65	1,81	3,63	1,57	2,28	1,57	3,63	1,81
V_1	0,00	1,57	-1,40	1,69	-0,00	-1,69	1,40	-1,57
R_1	12,50	-1,83	-2,03	4,05	3,39	4,05	-2,03	-1,83
Y_1	0,00	0,71	-1,03	0,92	0,00	-0,92	1,03	-0,71

Формується випадкове значення k таке, що k=2. Відповідно до цього до відліків вхідного сигналу X_2 додаються значення маски M_2. В результаті обчислень формується зашифрований сигнал Q_2, який представлено у таблиці 2.6.

Таблиця 2.6. - Замаскований другий сигнал та його відліки

Сигнал	Відліки							
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Q_2	6,45	2,83	3,04	0,88	4,43	2,11	1,76	4,31

Зашифрований сигнал Q_2 відправляється на віддалену комп'ютерну систему, на якій над ним здійснюється дискретне перетворення Фур'є, згідно з формулами (1.2) та (1.3). Отримані від віддаленої комп'ютерної системи обчислені значення реальної та уявної компоненти спектру записані у таблиці 2.7.

Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур'є. Дешифрування

відбувається шляхом віднімання від кожного відліку W_2 та V_2 значень компонент спектру S_2 та B_2. Результат дешифрування отриманих зафіксовано у таблиці 2.7.

Таблиця 2.7. - Зашифровані та дешифровані компоненти спектрального представлення сигналу X_2

Компоненти	Відліки							
	1	2	3	4	5	6	7	8
W_2	25,81	4,95	6,09	-0,91	5,55	-0,91	6,09	4,95
V_2	0,00	0,65	0,25	3,20	-0,00	-3,20	-0,25	-0,65
R_2	12,66	1,30	0,43	1,57	6,66	1,57	0,43	1,30
Y_2	0,00	-0,20	0,62	2,42	0,00	-2,42	-0,62	0,20

Формується випадкове значення k таке, що k=1. Відповідно до цього до відліків вхідного сигналу X_3 додаються значення маски M_1. В результаті обчислень формується зашифрований сигнал Q_3, який представлено у таблиці 2.8

Таблиця 2.8. - Замаскований третій сигнал та його відліки

Сигнал	Відліки							
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Q_3	4,06	1,07	4,55	1,38	3,45	5,62	2,50	4,67

Зашифрований сигнал Q_3 відправляється на віддалену комп'ютерну систему, на якій над ним здійснюється дискретне перетворення Фур'є, згідно з формулами (1.2 та 1.3). Отримані від віддаленої комп'ютерної системи обчислені значення реальної та уявної компоненти спектру записані у таблиці 2.9.

Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур'є. Дешифрування відбувається шляхом віднімання від кожного відліку W_3 та V_3 значень компонент спектру S_1 та B_1. Результат дешифрування отриманих зафіксовано у таблиці 2.9.

Таблиця 2.9. - Зашифровані та дешифровані компоненти спектрального представлення сигналу X_3

Компоненти	Відліки							
	1	2	3	4	5	6	7	8
W_3	27,30	-0,29	0,45	1,50	1,82	1,50	0,45	-0,29
V_3	0,00	3,50	-0,64	7,59	-0,00	-7,59	0,64	-3,50
R_3	13,78	-1,50	0,01	0,52	1,70	0,52	0,01	-1,50
Y_3	0,00	-1,07	-0,56	4,28	-0,00	-4,28	0,56	1,07

Після обробки перших трьох сигналів на систему надходить четвертий сигнал. Приклад четвертого отриманого системою сигналу наведено у таблиці 2.10. Формуються випадкові значення f та k таке, що k=2. Відповідно до цього до відліків вхідного сигналу X_4 додаються значення маски M_2. В результаті обчислень формується зашифрований сигнал Q_4, який представлено у таблиці 2.10.

Зашифрований сигнал Q_4 відправляється на віддалену комп'ютерну систему, на якій над ним здійснюється дискретне перетворення Фур'є, згідно з формулами (1.2 та 1.3). Отримані від віддаленої комп'ютерної системи обчислені значення реальної та уявної компоненти спектру записані у таблиці 2.11.

Таблиця 2.10. - Четвертий сигнал до та після шифрування

Сигнали	Відліки							
	1	2	3	4	5	6	7	8
X_4	0,52	3,11	2,20	3,04	0,87	0,85	2,41	0,01
Q_4	3,73	5,78	2,27	3,36	3,50	1,93	2,52	3,07

Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур'є. Дешифрування

відбувається шляхом віднімання від кожного відліку W_4 та V_4 значень компонент спектру S_2 та B_2. Результат дешифрування отриманих зафіксовано у таблиці 2.11.

На систему надходить п'ятий сигнал. Приклад п'ятого отриманого системою сигналу наведено у таблиці 2.12. Формуються випадкові значення f та k таке, що f=3. Відповідно до цього до відліків вхідного сигналу X_5 додаються значення сигналу X_1. В результаті обчислень формується зашифрований сигнал Q_5, який представлено у таблиці 2.12.

Таблиця 2.11. - Зашифровані та дешифровані компоненти спектрального представлення сигналу X_4

Компоненти	Відліки							
	1	2	3	4	5	6	7	8
W_4	26,14	2,73	2,44	-2,28	-2,13	-2,28	2,44	2,73
V_4	0,00	-2,67	-1,28	-3,18	-0,00	3,18	1,28	2,67
R_4	12,99	-0,91	-3,22	0,20	-1,02	0,20	-3,22	-0,91
Y_4	0,00	-3,53	-0,91	-3,95	0,00	3,95	0,91	3,53

Таблиця 2.12. - П’ятий сигнал до та після шифрування

Сигнали	Відліки							
	1	2	3	4	5	6	7	8
X_5	1,12	1,65	2,83	2,41	0,94	1,71	1,56	0,52
Q_5	3,15	1,71	5,38	4,04	1,87	4,43	4,01	0,65

Зашифрований сигнал Q_5 відправляється на віддалену комп’ютерну систему, на якій над ним здійснюється дискретне перетворення Фур’є, згідно з формулами (1.2 та 1.3). Отримані від віддаленої комп’ютерної системи обчислені значення реальної та уявної компоненти спектру записані у таблиці 2.13.

Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур’є. Дешифрування

відбувається шляхом віднімання від кожного відліку W_5 та V_5 значень компонент спектру S_1 та B_1. Результат дешифрування отриманих зафіксовано у таблиці 2.13.

Таблиця 2.13. - Зашифровані та дешифровані компоненти спектрального представлення сигналу X_5

Компоненти	Відліки							
	1	2	3	4	5	6	7	8
W_5	25,24	-3,04	-4,36	5,61	3,57	5,61	-4,36	-3,04
V_5	0,00	-1,85	-1,45	0,89	0,00	-0,89	1,45	1,85
R_5	25,24	-3,04	-4,36	5,61	3,57	5,61	-4,36	-3,04
Y_5	0,00	-1,85	-1,45	0,89	0,00	-0,89	1,45	1,85

2.2. Доведення конструктивності методу захищеної реалізації дискретного перетворення Фур'є

Конструктивність запропонованого методу може бути продемонстрована шляхом встановлення того факту що після всіх перетворень відліки спектрального представлення сигналу обчислюються за формулами (1.2) та (1.3).

Відповідно до формули (2.1) отримані від віддаленої комп'ютерної системи обчислені значення реальної w_{ji} та уявної v_{ji} компоненти спектру обраховуються наступним чином:

- формула для обчислення реальної компоненти спектру має наступний вигляд:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$w_{ji} = \sum_{l=1}^{n} a_{il} \cdot q_{jl} = \sum_{l=1}^{n} a_{il} \cdot \left(x_{jl} + (1-f) \cdot m_{kl} + f \cdot h_{dl}\right).$$

- формула для обчислення уявної компоненти спектру подібна до реальної і має наступний вигляд:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$v_{ji} = \sum_{l=1}^{n} c_{il} \cdot q_{jl} = \sum_{l=1}^{n} c_{il} \cdot \left(x_{jl} + (1-f) \cdot m_{kl} + f \cdot h_{dl}\right).$$

Після розкриття дужок формули для обчислення значень компонент спектрального представлення сигналу отриманих від віддаленої комп'ютерної системи зводиться до наступного вигляду:

- формула для обчислення реальної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$w_{ji} = \sum_{l=1}^{n} a_{il} \cdot x_{jl} + \sum_{l=1}^{n} a_{il} \cdot \left((1-f) \cdot m_{kl} + f \cdot h_{dl}\right) =$$

$$= \sum_{l=1}^{n} a_{il} \cdot x_{jl} + (1-f) \cdot s_{ki} + f \cdot r_{di}.$$

- формула для обчислення уявної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$v_{ji} = \sum_{l=1}^{n} c_{il} \cdot x_{jl} + \sum_{l=1}^{n} c_{il} \cdot \left((1-f) \cdot m_{kl} + f \cdot h_{dl}\right) =$$

$$= \sum_{l=1}^{n} c_{il} \cdot x_{jl} + (1-f) \cdot b_{ki} + f \cdot y_{di}.$$

Після підстановки до формули (2.2) за якою відбувається дешифрування даних, значень отриманих від віддаленої комп'ютерної системи з формули (2.1) результат виконання дискретного перетворення Фур'є обраховується наступним чином:

- формула для обчислення реальної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$r_{ji} = \sum_{l=1}^{n} a_{il} \cdot x_{jl} + (1-f) \cdot s_{ki} + f \cdot r_{di} - (1-f) \cdot s_{ki} - f \cdot r_{di} =$$

$$= \sum_{l=1}^{n} a_{il} \cdot x_{jl}. \qquad (2.4)$$

- формула для обчислення уявної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$y_{ji} = \sum_{l=1}^{n} c_{il} \cdot x_{jl} + (1-f) \cdot b_{ki} + f \cdot y_{di} - (1-f) \cdot b_{ki} - f \cdot y_{di} =$$

$$= \sum_{l=1}^{n} c_{il} \cdot x_{jl} \qquad (2.5)$$

Результат формул (2.4) та (2.5) співпадає із результатом формул (1.2) та (1.3), виходячи з цього доведено конструктивність запропонованого методу, оскільки після його застосування до вхідних даних реальна та уявна компоненти дійсно відповідають дискретному перетворенню Фур'є.

2.3. Обґрунтування ефективності запропонованого методу

Важливим аспектом захищеної реалізації дискретного перетворення Фур'є виступає оцінка його ефективності [67]. Виходячи з поставленої мети оцінка ефективності запропонованого методу може бути проведена за двома критеріями: рівень захищеності даних, який досягається з використанням запропонованого алгоритму, та коефіцієнт прискорення обчислень.

Оскільки для прискорення обробки передбачено використання віддалених комп'ютерних систем, то необхідно, щоб запропонований метод забезпечував високий рівень захищеності даних. Порівняно з відомим [68], метод, що запропоновано у цьому розділі, дозволяє згенерувати більшу кількість маскувальних сигналів. Це твердження доводиться шляхом встановлення того факту, що кількість можливих значень, що використовується для шифрування сигналу більша або рівна $p+M$ (M - загальна кількість сигналів, що надійшла на систему). При цьому, цілком очевидно, що $p+M > p$. При цьому порівнюючи з відомим методом [69], рівень пам'яті, використаної на мікроконтролері є однаковим за умови, що попередні сигнали будуть розміщені у пам'яті за рахунок зменшення кількості масок.

Відповідно до мети викладеної роботи, кількість часу, який витрачається на реалізацію дискретного перетворення Фур'є, повинна зменшитися. Для визначення того на скільки розроблений обчислювальний метод є швидшим необхідно обчислити коефіцієнт прискорення V, який характеризується відношенням часу, за який виконується перетворення Фур'є на мікроконтролері - T_{DFT} до часу, за який відбувається перетворення Фур'є з використанням запропонованого методу, тобто шифрування та дешифрування даних - T_S. Виходячи з попереднього твердження коефіцієнт можна визначити за наступною формулою:

$$V = \frac{T_{DFT}}{T_S}.$$

Цілком очевидно, що час за який виконується дискретне перетворення Фур'є характеризується кількістю операцій, які необхідно зробити для

отримання результату. Відповідно для визначення часу, який потрібен на виконання дискретного перетворення Фур'є необхідно обчислити кількість додавань та множень. Таким чином, кількість множень дорівнює $2 \bullet n^2$, а кількість додавань рівна $2 \bullet n \bullet (n\text{-}1)$. Якщо позначити через τ_a час за який виконується операція додавання, а через τ_m час, за який виконується операція множення, то визначення часу, за який виконується дискретне перетворення Фур'є можливо записати наступним чином:

$$T_{DFT} = 2 \cdot n^2 \cdot \tau_m + 2 \cdot n \cdot (n-1) \cdot \tau_a \qquad (2.3).$$

Виходячи з того факту, що сучасні процесори виконують операцію множення з плаваючою комою приблизно в 30 разів швидше за операцію додавання, то формула (2.3) може бути записана у наступному вигляді:

$$T_{DFT} = 2 \cdot n \cdot \tau_a \cdot (31 \cdot n - 1).$$

Для розрахунку T_S необхідно обчислити час, який витрачається на кожному етапі розробленого методу:

1. Шифрування даних.
2. Відправка даних на віддалену комп'ютерну систему.
3. Обчислення операції дискретного перетворення Фур'є на віддаленій комп'ютерній системі.
4. Отримання результатів дискретного перетворення Фур'є з віддаленої комп'ютерної системи на мікроконтролер.
5. Дешифрування даних для отримання коректного результату.

Для шифрування сигналу необхідно додати до нього n відліків одного маскувального або попереднього сигналу. З цього випливає, що час, за який відбувається шифрування даних становить $n \bullet \tau_a$.

Для того щоб здійснити дешифрування необхідно від всіх відліків кожної компоненти спектрального представлення сигналу відняти відповідні їм значення, які зберігаються у пам'яті системи. Для дешифрування двох компонент спектрального представлення необхідно обчислити різницю з $2 \bullet n$ значеннями. З

цього випливає, що час, за який відбувається дешифрування даних становить $2 \bullet n \bullet \tau_a$.

Цілком очевидно, що загальний час пересилок даних T_t значно менший порівняно з усіма операціями, що відбуваються на мікроконтролері [70], з чого випливає, що значення T_t можливо не враховувати під час аналізу загального часу виконання запропонованого методу. Таким чином, формула, за якою розраховується час дискретного перетворення Фур'є з використанням запропонованого методу визначається наступним чином:

$$T_S = 3 \cdot n \cdot \tau_a.$$

Відповідно до цього, значення коефіцієнту прискорення розраховується за наступною формулою:

$$V = \frac{T_{DFT}}{T_S} = \frac{2 \cdot n \cdot \tau_a \cdot (31 \cdot n - 1)}{3 \cdot n \cdot \tau_a} \approx 20 \cdot n.$$

2.4. Функціональне моделювання методу захищеної реалізації дискретного перетворення Фур'є

Функціональне моделювання розробленого методу передбачає створення алгоритму, в якому чітко визначені усі кроки для реалізації захищеного перетворення Фур'є над сигналами, що надійшли на систему [71].

Для виконання захищеного перетворення Фур'є з використанням хмарних технологій запропонований метод передбачає реалізацію двох етапів. Виходячи з цього для опису реалізації запропонованого методу використовується два наступні алгоритми:

1. Підготовчий етап формування масок. На цьому етапі виконується генерація масок та часткове обчислення їх спектрального представлення.
2. Захищене перетворення Фур'є над сигналом. На цьому етапі виконується гомоморфне шифрування сигналу з використанням маскувального сигналу або одного з попередніх сигналів, перед посилкою його для обробки в хмару, а також дешифрування прийнятого з віддаленої комп'ютерної системи результату перетворення Фур'є, тобто спектрального представлення сигналу.

Перший блок першого алгоритму виводить користувачу інформацію про можливі способи введення масок. Тобто користувач обирає яким із наступних способів будуть утворюватися маскувальні сигнали:

1) Система генерує випадкові значення відліків маскувальних сигналів.
2) Користувач вводить маскувальні сигнали самостійно.

У другому блоці система зчитує відповідь користувача. В межах третього блоку відбувається перевірка введеного значення. Якщо користувач обрав другий метод, то здійснюється перехід, до блоку сім, інакше система виконує наступний блок. У четвертому блоці початкове значення індексу k встановлюється в 0. У наступному, п’ятому блоці алгоритму відбувається ініціалізація генератору випадкових чисел, з використанням якого формується k-та маска та значення k збільшується на 1. У шостому блоці алгоритму відбувається перевірка, значення індексу k, якщо він менший за значення p, то здійснюється перехід до блоку під номером п’ять, інакше - відбувається перехід до дев’ятого блоку алгоритму. У наступному, сьомому блоці алгоритму система очікує введення користувачем маскувальних сигналів. У восьмому блоці система зчитує введені дані і записує їх у пам’ять. У межах виконання дев’ятого блоку алгоритму відбувається формування матриць з обчисленими значеннями спектрального представлення маскувальних сигналів. Для p масок обчислюється результат виконання дискретного перетворення Фур’є над ними, в результаті формуються значення реальної та уявної компонент. Таким чином для кожної маски у системі формується набір із двох матриць для обчислених значень реальної та уявної компоненти.

Перший блок другого алгоритму передбачає введення відліків сигналу. У другому блоці початкове значення індексу j встановлюється в 0. В межах третього блоку відбувається перевірка, значення індексу j, якщо він менший за значення t, то значення індексу f встановлюється в одиницю та здійснюється перехід до блоку під номером п’ять, інакше - відбувається перехід до наступного блоку алгоритму. В межах четвертого блоку відбувається ініціалізація генератору

випадкових чисел, з використанням якого генерується випадкове бінарне значення і записується у змінну *f*. У п’ятому блоці алгоритму відбувається перевірка, значення індексу *f*, якщо він дорівнює одиниці, то здійснюється перехід до блоку під номером вісім, інакше - відбувається перехід до наступного блоку алгоритму. В межах шостого блоку відбувається ініціалізація генератору випадкових чисел, з використанням якого генерується випадкове значення *k* таке, що $k \in \{1, 2, 3, ..., p\}$. У наступному, сьомому блоці до відліків вхідного сигналу додаються відповідні їм відліки маски із номером *k*. Після виконання цього блоку алгоритму здійснюється перехід до блоку під номером одинадцять. В межах восьмого блоку відбувається ініціалізація генератору випадкових чисел, з використанням якого генерується випадкове значення *d* таке, що $d \in \{1, 2, 3, ..., t\}$. У наступному, дев’ятому блоці до відліків вхідного сигналу додаються відповідні їм відліки сигналу із пулу попередніх із номером *d*. Десятий блок передбачає запис сигналу, який надійшов на систему, на місце *d* у пул попередніх сигналів.

Після формування зашифрованого сигналу в межах блоків 4-10 у одинадцятому блоці відбувається дискретне перетворення Фур’є. У дванадцятому блоці система отримує зашифровані значення реальної та уявної компонент спектру. В межах тринадцятого блоку відбувається перевірка, значення індексу *f*, якщо він дорівнює одиниці, то здійснюється перехід до блоку під номером шістнадцять, інакше - відбувається перехід до наступного блоку алгоритму. У наступному, чотирнадцятому блоці від зашифрованих відліків реальної компоненти спектру віднімаються відповідні їм значення рядка із номером *k* з масиву для зберігання реальних компонент маскувальних сигналів. У п'ятнадцятому блоці від зашифрованих відліків уявної компоненти спектру віднімаються відповідні їм значення рядка із номером *k* з масиву для зберігання уявних компонент маскувальних сигналів. Після виконання цього блоку алгоритму здійснюється перехід до блоку під номером дев’ятнадцять. У наступному, шістнадцятому блоці від зашифрованих відліків реальної

компоненти спектру віднімаються відповідні їм значення рядка із номером *d* з масиву для зберігання реальних компонент пулу попередніх сигналів. У сімнадцятому блоці від зашифрованих відліків уявної компоненти спектру віднімаються відповідні їм значення рядка із номером *d* з масиву для зберігання уявних компонент пулу попередніх сигналів. Вісімнадцятий блок передбачає запис реальної компоненти спектрального представлення сигналу, на місце *d* у масив для зберігання реальних компонент пулу попередніх сигналів. Дев’ятнадцятий блок передбачає запис уявної компоненти спектрального представлення сигналу, на місце *d* у масив для зберігання уявних компонент пулу попередніх сигналів. У двадцятому блоці значення індексу *j* збільшується на 1.

Висновки до розділу 2

Результатом виконання теоретичних та практичних досліджень у другому розділі є розробка нового методу шифрування дискретного перетворення Фур'є та його оцінка. Висновки до другого розділу можуть бути сформульовані наступним чином:

1. Здійснено розробку методу захищеної реалізації дискретного перетворення Фур'є, який базується на адитивному маскуванні груп сигналів. Характерною особливістю запропонованого методу є здійснення шифрування з використанням випадково обраного попереднього або маскувального сигналу, який адитивно накладається на вхідні дані. Відповідно до цього для дешифрування отриманих сигналів реальної та уявної компоненти від них віднімаються відповідні обчислені значення, які зберігаються в пам'яті мікроконтролера.
2. Здійснено оцінку ефективності запропонованого методу відповідно до стандартних характеристик. Розроблений метод демонструє підвищення рівня захищеності на два порядки, завдяки використанню необмеженої кількості значень для шифрування. Також варто зазначити, що запропонований метод дозволяє зменшити складність обчислень дискретного перетворення Фур'є завдяки можливостей віддалених комп'ютерних систем більше ніж в 20 разів.
3. Теоретично та практично доведено конструктивність розробленого методу. Детальний аналіз показав, що після його застосування до вхідних даних реальна та уявна компоненти спектрального представлення дійсно відповідають дискретному перетворенню Фур'є. Це підтверджено шляхом виведення математичних формул, які точно описують процес перетворення та демонструють його правильність. Крім цього, у представленій роботі наведено приклад застосування розробленого методу, який демонструє коректність отриманих результатів для вхідного сигналу. Цей приклад включає покроковий опис процесу обробки сигналу, починаючи від підготовки даних для шифрування і закінчуючи дешифруванням спектральних компонент. Таким чином, проведене

дослідження не тільки підтвердило теоретичну обґрунтованість розробленого методу, але й продемонструвало його практичну цінність.

4. Здійснено функціональне моделювання розробленого методу захищеної реалізації дискретного перетворення Фур’є на віддалених обчислювальних потужностях. Алгоритм розроблено для подальшої реалізації програмного забезпечення. Основна мета полягає в забезпеченні швидкого та безпечного виконання дискретного перетворення Фур’є, зокрема, для обробки великих обсягів даних у режимі реального часу.

РОЗДІЛ 3
РОЗРОБКА МЕТОДУ ЗАХИЩЕНОЇ РЕАЛІЗАЦІЇ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ФУР'Є З ДИНАМІЧНОЮ ЗМІНОЮ КОДІВ МАСКИ

3.1. Розробка методу захищеної реалізації дискретного перетворення Фур'є з динамічною зміною кодів маски

Запропонований метод базується на адитивному маскуванні відліків сигналів. Основна ідея запропонованого методу, яка різнить його від відомих, полягає в динамічній зміні кодів маски при обробці послідовності вхідних сигналів. Відповідно, розроблений метод орієнтований на застосування до сукупності послідовних сигналів $X_1, X_2, X_3, \dots$, причому кожен сигнал складається з n відліків, тобто $X_j = \{x_{j1}, x_{j2}, x_{j3}, \dots, x_{jn}\}$. Тому далі використовуються індекси j та i для позначення відповідно номеру сигналу та відліку. Для шифрування у запропонованому методі використовується сукупність з p маскувальних сигналів $M_1, M_2, M_3, \dots, M_p$. Причому кожен маскувальний сигнал M_k складається з n відліків, тобто $M_k = \{m_{k1}, m_{k2}, m_{k3}, \dots, m_{kn}\}$, які є випадково згенерованими значеннями. Динамічна зміна кодів маски досягається шляхом використання маскувальної матриці T_j. Ця матриця містить p рядків і n стовпців та заповнюється бінарними значеннями:

$$k \in \{1,2,3,\dots,p\}, i \in \{1,2,3,\dots,n\}: T[k][i] \in \{0,1\}.$$

Відповідно до запропонованого методу, кожен відлік замаскованого сигналу Q_j, такого, що $Q_j = \{q_{j1}, q_{j2}, q_{j3}, \dots, q_{jn}\}$ визначається наступним чином:

$$j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}: q_{ji} = x_{ji} + \sum_{k=1}^{p} T_j[k][i] \cdot m_{ki} \qquad (3.1)$$

Для реалізації захищеного перетворення Фур'є з використанням хмарних технологій запропонований метод передбачає виконання двох етапів:

1. Генерація масок та часткове обчислення їх спектрального представлення.
2. Гомоморфне шифрування сигналу з використанням маскувальної матриці перед посилкою його для обробки в хмару, а також дешифрування прийнятого з віддаленої комп'ютерної системи результату перетворення Фур'є, тобто спектрального представлення сигналу.

Перший етап:

1. Формується набір маскувальних сигналів. Перед початком обробки вхідних сигналів запропонований метод передбачає формування набору із p масок M_k для $\forall\ k = \{1, 2, 3, ..., p\}$. Оскільки кожна маска M_k складається з n відліків, тобто $M_k = \{m_{k1}, m_{k2}, m_{k3}, ..., m_{kn}\}$, то для формування кожного маскувального сигналу ініціалізується робота генератора випадкових чисел, з використанням якого утворюється послідовність із n випадкових чисел.
2. Формуються матриці з частково обчисленими значеннями спектрального представлення маскувальних сигналів. Для p масок обчислюється добуток відліків маскувального сигналу та коефіцієнтів реальної та уявної компонент, за наступними формулами:

$$\forall\ l \in \{1,2,3,\dots,n\}, i \in \{1,2,3,\dots,n\},\ s_{li} = a_{li} \cdot m_i,\ b_{li} = c_{li} \cdot m_i,$$

де s_{li} та b_{li} - частково обчислені значення реальної та уявної компоненти спектру відповідно.

Таким чином, після здійснення розрахунків, для кожної маски M_k формується набір із двох матриць S_k та B_k для частково обчислених значень реальної та уявної компоненти (рис. 3.1)

s_{11}	s_{12}	s_{13}	…	s_{1n}
s_{21}	s_{22}	s_{23}	…	s_{2n}
s_{31}	s_{32}	s_{33}	…	s_{3n}
⋮	⋮	⋮		⋮
s_{n1}	s_{n2}	s_{n3}	…	s_{nn}

b_{11}	b_{12}	b_{13}	…	b_{1n}
b_{21}	b_{22}	b_{23}	…	b_{2n}
b_{31}	b_{32}	b_{33}	…	b_{3n}
⋮	⋮	⋮		⋮
b_{n1}	b_{n2}	b_{n3}	…	b_{nn}

Рисунок 3.1 – Частково обчислені значення компонент спектрального представлення.

Другий етап:

1. Генерується маскувальна матриця T_j для вхідного сигналу X_j. Для кожного сигналу X_j формується матриця T_j, яка містить p рядків і n стовпців та заповнюється бінарними значеннями, тобто 0 та 1, випадковим чином.
2. Гомоморфне шифрування сигналу. До відліків вхідного сигналу X_j додаються значення з набору масок у відповідності із унікальною маскувальною матрицею T_j, згідно з формулою (3.1). В результаті обчислень формується зашифрований сигнал Q_j.
3. Дискретне перетворення Фур'є. Зашифрований сигнал Q_j відправляється на віддалену комп'ютерну систему, на якій над ним здійснюється дискретне перетворення Фур'є, згідно з наступними формулами:

$$\forall\, l \in \{1,2,3,\dots,n\}, i \in \{1,2,3,\dots,n\}, w_i = \sum_{l=1}^{n} a_{il} \cdot q_l,\ v_i = \sum_{l=1}^{n} c_{il} \cdot q_l \qquad (3.2)$$

де w_i та v_i - отримані від віддаленої комп'ютерної системи обчислені значення реальної та уявної компоненти спектру відповідно.

4. Дешифрування отриманих результатів. Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур'є. Дешифрування відбувається шляхом віднімання від кожного відліку компонент спектру сум значень масок, які використані для шифрування згідно з маскувальною матрицею. Таким чином дешифрування отриманих результатів відбувається відповідно до наступних формул:

$$r_{ji} = w_{ji} - \sum_{k=1}^{p}\sum_{l=1}^{n} T_j[k][l] \cdot S_k[i][l],\ y_{ji} = v_{ji} - \sum_{k=1}^{p}\sum_{l=1}^{n} T_j[k][l] \cdot B_k[i][l] \qquad (3.3)$$

Функціонування запропонованого методу може бути проілюстровано за допомогою наступного прикладу. Нехай система передбачає перетворення сигналів, що складається із 8 відліків (n = 8), а кількість масок, що входять до маскувального набору дорівнює 3 (k = 3). Відповідно до першого етапу запропонованого методу перед початком обробки сигналів здійснюється генерація масок та часткове обчислення їх спектрального представлення.

Результат генерації маскувальних сигналів, кількість відліків яких відповідає умовам наведеним вище представлено у таблиці 3.1.

Таблиця 3.1. - Приклад значень маскувальних сигналів та їх відліків

Сигнали	Відліки							
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
M_1	2,36	0,34	1,28	0,22	1,27	3,05	1,91	3,09
M_2	3,21	2,67	0,07	0,32	2,63	1,08	0,11	3,06
M_3	0,22	0,33	1,37	1,65	2,31	1,72	3,04	0,83

Після генерації масок здійснюється часткове обчислення їх спектрального представлення. Результат обчислення представлено у таблицях 3.2-3.7.

Таблиця 3.2. - Частково обчислені компоненти реального спектрального представлення для маски M_1

Номер компоненти	Номери відліків							
	0	1	2	3	4	5	6	7
0	2,36	0,34	1,28	0,22	1,27	3,05	1,91	3,09
1	2,36	0,24	0,00	-0,16	-1,27	-2,16	0,00	2,18
2	2,36	0,00	-1,28	0,00	1,27	0,00	-1,91	0,00
3	2,36	-0,24	0,00	0,16	-1,27	2,16	0,00	-2,18
4	2,36	-0,34	1,28	-0,22	1,27	-3,05	1,91	-3,09
5	2,36	-0,24	0,00	0,16	-1,27	2,16	0,00	-2,18
6	2,36	0,00	-1,28	0,00	1,27	0,00	-1,91	0,00
7	2,36	0,24	0,00	-0,16	-1,27	-2,16	0,00	2,18

Таблиця 3.3. - Частково обчислені значення спектрального представлення уявної компоненти для маски M_1

Номери відліків	Номери відліків							
	0	1	2	3	4	5	6	7
0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
1	0,00	-0,24	-1,28	-0,16	0,00	2,16	1,91	2,18
2	0,00	-0,34	0,00	0,22	0,00	-3,05	0,00	3,09
3	0,00	-0,24	1,28	-0,16	0,00	2,16	-1,91	2,18
4	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,00	0,24	-1,28	0,16	0,00	-2,16	1,91	-2,18
6	0,00	0,34	0,00	-0,22	0,00	3,05	0,00	-3,09
7	0,00	0,24	1,28	0,16	0,00	-2,16	-1,91	-2,18

Таблиця 3.4. - Частково обчислені значення спектрального представлення реальної компоненти для маски M_2

Номери відліків	Номери відліків							
	0	1	2	3	4	5	6	7
0	3,21	2,67	0,07	0,32	2,63	1,08	0,11	3,06
1	3,21	1,89	0,00	-0,23	-2,63	-0,76	0,00	2,16
2	3,21	0,00	-0,07	0,00	2,63	0,00	-0,11	0,00
3	3,21	-1,89	0,00	0,23	-2,63	0,76	0,00	-2,16
4	3,21	-2,67	0,07	-0,32	2,63	-1,08	0,11	-3,06
5	3,21	-1,89	0,00	0,23	-2,63	0,76	0,00	-2,16
6	3,21	0,00	-0,07	0,00	2,63	0,00	-0,11	0,00
7	3,21	1,89	0,00	-0,23	-2,63	-0,76	0,00	2,16

Таблиця 3.5. - Частково обчислені значення спектрального представлення уявної компоненти для маски M_2

Номери відліків	Номери відліків							
	0	1	2	3	4	5	6	7
0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
1	0,00	-1,89	-0,07	-0,23	0,00	0,76	0,11	2,16
2	0,00	-2,67	0,00	0,32	0,00	-1,08	0,00	3,06
3	0,00	-1,89	0,07	-0,23	0,00	0,76	-0,11	2,16
4	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,00	1,89	-0,07	0,23	0,00	-0,76	0,11	-2,16
6	0,00	2,67	0,00	-0,32	0,00	1,08	0,00	-3,06
7	0,00	1,89	0,07	0,23	0,00	-0,76	-0,11	-2,16

Таблиця 3.6. - Частково обчислені значення спектрального представлення реальної компоненти для маски M_3

Номери відліків	Номери відліків							
	0	1	2	3	4	5	6	7
0	0,22	0,33	1,37	1,65	2,31	1,72	3,04	0,83
1	0,22	0,23	0,00	-1,17	-2,31	-1,22	0,00	0,59
2	0,22	0,00	-1,37	0,00	2,31	0,00	-3,04	0,00
3	0,22	-0,23	0,00	1,17	-2,31	1,22	0,00	-0,59
4	0,22	-0,33	1,37	-1,65	2,31	-1,72	3,04	-0,83
5	0,22	-0,23	0,00	1,17	-2,31	1,22	0,00	-0,59
6	0,22	0,00	-1,37	0,00	2,31	0,00	-3,04	0,00
7	0,22	0,23	0,00	-1,17	-2,31	-1,22	0,00	0,59

Таблиця 3.7. - Частково обчислені значення спектрального представлення уявної компоненти для маски M_3

Номери відліків	Номери відліків							
	0	1	2	3	4	5	6	7
0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
1	0,00	-0,23	-1,37	-1,17	0,00	1,22	3,04	0,59
2	0,00	-0,33	0,00	1,65	0,00	-1,72	0,00	0,83
3	0,00	-0,23	1,37	-1,17	0,00	1,22	-3,04	0,59
4	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,00	0,23	-1,37	1,17	0,00	-1,22	3,04	-0,59
6	0,00	0,33	0,00	-1,65	0,00	1,72	0,00	-0,83
7	0,00	0,23	1,37	1,17	0,00	-1,22	-3,04	-0,59

Другий етап запропонованого методу передбачає надходження на мікроконтролер сигналу для обробки. Приклад першого отриманого системою сигналу наведено у таблиці 3.8.

Таблиця 3.8. - Приклад першого сигналу та його відліків

Сигнал	Відліки							
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
X_1	55,00	15,00	86,00	24,00	66,00	245,00	76,00	26,00

Для демонстрації коректності отриманих результатів після дешифрування вхідного сигналу у таблицях 3.9-3.10. представлений результат виконання дискретного перетворення Фур'є над даними з таблиці 3.8.

Після отримання сигналу у системі генерується маскувальна матриця T_1 для першого вхідного сигналу. Результат генерації T_1, яка полягає у випадковому заповненні матриці 0 та 1 представлений у таблиці 3.11.

Таблиця 3.9. - Значення спектрального представлення реальної компоненти для вхідного сигналу

Компонента	Відліки							
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
R_1	593,00	-172,22	-41,00	150,22	-27,00	150,22	-41,00	-172,22

Таблиця 3.10. - Значення спектрального представлення уявної компоненти для вхідного сигналу

Компонента	Відліки							
	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
Y_1	0,00	154,05	-210,00	174,05	0,00	-174,05	210,00	-154,05

Таблиця 3.11. - Приклад маскувальної матриці

Номери масок	Номери відліків							
	1	2	3	4	5	6	7	8
1	1,00	0,00	1,00	0,00	1,00	1,00	0,00	0,00
2	0,00	0,00	0,00	0,00	0,00	1,00	1,00	0,00
3	1,00	1,00	0,00	1,00	1,00	1,00	0,00	1,00

Після генерації маскувальної матриці відбувається гомоморфне шифрування сигналу. До відліків вхідного сигналу X_1 додаються значення з набору масок у відповідності із маскувальною матрицею T_1, згідно з формулою (3.1). В результаті обчислень формується зашифрований сигнал Q_1, який представлено у таблиці 3.12.

Таблиця 3.12. - Замаскований перший сигнал та його відліки

Зашифрований сигнал	Відліки							
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Q_1	57,58	15,33	87,28	25,65	69,58	250,85	76,11	26,83

Зашифрований сигнал Q_1 відправляється на віддалену комп'ютерну систему, на якій над ним здійснюється дискретне перетворення Фур'є, згідно з формулами (1.2) та (1.3). Отримані від віддаленої комп'ютерної системи обчислені значення реальної та уявної компоненти спектру записані у таблицях 3.13 та 3.14.

Таблиця 3.13. - Значення спектрального представлення реальної компоненти для замаскованого сигналу W_1

Компонента	Відліки							
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
W_1	609,21	-177,70	-36,23	153,70	-28,11	153,70	-36,23	-177,70

Таблиця 3.14. - Значення спектрального представлення уявної компоненти для замаскованого сигналу Q_1

Компонента	Відліки							
	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
V_1	0,00	156,20	-213,70	178,54	0,00	-178,54	213,70	-156,20

Після отримання обрахованих значень мікроконтролер, здійснює дешифрування результатів дискретного перетворення Фур'є. Дешифрування відбувається шляхом віднімання від кожного відліку компонент спектру сум значень масок, які використані для шифрування згідно з маскувальною матрицею. Таким чином дешифрування отриманих результатів відбувається відповідно до формул (3.3). Результат дешифрування отриманих зафіксовано у таблицях 3.15-3.16.

Результатом роботи запропонованого методу є отримання на мікроконтролері реальної та уявної компонент спектрального представлення для сигналу X_1.

Таблиця 3.15. - Значення спектрального представлення реальної компоненти для вхідного сигналу після дешифрування

Компонента	Відліки							
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
R_1	593,00	-172,22	-41,00	150,22	-27,00	150,22	-41,00	-172,22

Таблиця 3.16. - Значення спектрального представлення уявної компоненти для вхідного сигналу після дешифрування

Компонента	Відліки							
	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
Y_1	0,00	154,05	-210,00	174,05	0,00	-174,05	210,00	-154,05

Результат виконання перетворення Фур'є над вхідними даними, який продемонстровано у таблиці 3.13 та 3.14 ідентичний до значень реальної та уявної компонент, отриманих в результаті застосування розробленого методу до вхідного сигналу із таблиці 3.15-3.16.

3.2. Теоретичний аналіз розробленого методу

Конструктивність запропонованого методу може бути продемонстрована шляхом встановлення того факту що після всіх перетворень відліки спектрального представлення сигналу обчислюються за формулою (1.2 та 1.3).

Відповідно до формули (3.2) отримані від віддаленої комп'ютерної системи обчислені значення реальної w_{ji} та уявної v_{ji} компоненти спектру обраховуються наступним чином:

- формула для обчислення реальної компоненти спектру має наступний вигляд:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$w_{ji} = \sum_{l=1}^{n} a_{il} \cdot q_{jl} = \sum_{l=1}^{n} a_{il} \cdot \left(x_{jl} + \sum_{k=1}^{p} T_j[k][l] \cdot m_{kl}\right).$$

- формула для обчислення уявної компоненти спектру подібна до реальної і має наступний вигляд:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}:$$

$$v_{ji} = \sum_{l=1}^{n} c_{il} \cdot q_{jl} = \sum_{l=1}^{n} c_{il} \cdot \left(x_{jl} + \sum_{k=1}^{p} T_j[k][l] \cdot m_{kl}\right).$$

Після розкриття дужок формули для обчислення значень компонент спектрального представлення сигналу отриманих від віддаленої комп'ютерної системи зводиться до наступного вигляду:

- формула для обчислення реальної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}: w_{ji} = \sum_{l=1}^{n} a_{il} \cdot x_{jl} + \sum_{l=1}^{n}\sum_{k=1}^{p} T_j[k][l] \cdot a_{il} \cdot m_{kl}$$

- формула для обчислення уявної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}: v_{ji} = \sum_{l=1}^{n} c_{il} \cdot x_{jl} + \sum_{l=1}^{n}\sum_{k=1}^{p} T_j[k][l] \cdot c_{il} \cdot m_{kl}$$

Після підстановки до формули (3.3) за якою відбувається дешифрування даних, значень отриманих від віддаленої комп'ютерної системи з формулами (3.2) результат виконання дискретного перетворення Фур'є обраховується наступним чином:

- формула для обчислення реальної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}: r_{ji} = \sum_{l=1}^{n} a_{il} \cdot x_{jl} +$$

$$+ \sum_{l=1}^{n}\sum_{k=1}^{p} T_j[k][l] \cdot a_{il} \cdot m_{kl} - \sum_{k=1}^{p}\sum_{l=1}^{n} T_j[k][l] \cdot a_{il} \cdot m_{kl} = \sum_{l=1}^{n} a_{il} \cdot x_{jl} \qquad (3.7)$$

- формула для обчислення уявної компоненти спектру:

$$\forall j \in \{1,2,3,\dots\}, i \in \{1,2,3,\dots,n\}: y_{ji} = \sum_{l=1}^{n} c_{il} \cdot x_{jl} +$$

$$+ \sum_{l=1}^{n}\sum_{k=1}^{p} T_j[k][l] \cdot c_{il} \cdot m_{kl} - \sum_{k=1}^{p}\sum_{l=1}^{n} T_j[k][l] \cdot c_{il} \cdot m_{kl} = \sum_{l=1}^{n} c_{il} \cdot x_{jl} \qquad (3.8)$$

Результат формули (3.7) та формули (3.8) співпадає із результатом формули (1.2) та (1.3), виходячи з цього доведено конструктивність запропонованого

методу, оскільки після його застосування до вхідних даних реальна та уявна компоненти дійсно відповідають дискретному перетворенню Фур'є.

3.3. Оцінка ефективності запропонованого методу

Важливим аспектом захищеної реалізації дискретного перетворення Фур'є виступає оцінка його ефективності [72]. Виходячи з поставленої мети оцінка ефективності запропонованого методу може бути проведена за двома критеріями: рівень захищеності даних, який досягається з використанням запропонованого алгоритму, та коефіцієнт прискорення обчислень.

Оскільки для прискорення обробки передбачено використання віддалених комп'ютерних систем, то необхідно, щоб запропонований метод забезпечував високий рівень захищеності даних. Порівняно з відомим [73], метод, що запропоновано у цьому розділі, дозволяє згенерувати більшу кількість маскувальних сигналів, тобто $2^{p \cdot n}$. Це твердження доводиться шляхом встановлення того факту, що кількість можливих варіацій заповнення маскувальної матриці T_j, рівна $2^{p \cdot n}$, оскільки її розмір рівний $p \times n$ і вона заповнюється бінарними значеннями. Цілком очевидно, що $2^{p \cdot n} > p$. При цьому порівнюючи з методом, що запропонований у другому розділі, рівень пам'яті, використаної на мікроконтролері є більшим при однаковій кількості маскувальних сигналів. Оскільки у представленому у другому розділі методі кількість пам'яті, що використовується рівна $3 \cdot n \cdot p$, оскільки для кожної з p масок необхідно зберігати $2 \cdot n$ значень компонент спектру. У запропонованому методі необхідно зберігати дані про відліки p маскувальних сигналів і про частково обчислені значення їх спектрального представлення. Через те, що частково обчисленні значення компонент спектру являють собою 2 матриці розмірності $n \times n$, кількість зайнятої пам'яті при використанні розробленого методу рівна $n \cdot p + 2 \cdot n^2 \cdot p$. Виходячи з цього для збереження ефективності необхідно, щоб рівень пам'яті був однаковий у цьому та попередньому методі був однаковий. Однакова кількість зайнятої пам'яті може бути досягнута шляхом використання меншої

кількості маскувальних сигналів. Це може бути проілюстровано наступними розрахунками:

1. Нехай кількість маскувальних сигналів використаних у першому методі рівна $p`$, а у другому p.
2. Для досягнення однакової кількості зайнятої пам'яті необхідно, щоб значення $3 \bullet n \bullet p`$ було рівно $n \bullet p + 2 \bullet n^2 \bullet p$. Таким чином кількість маскувальних сигналів розробленого методу для досягнення однакового рівня зайнятої пам’яті розраховується за наступною формулою:

$$p = \frac{3 \cdot n \cdot p}{n + n^2 \cdot p} = \frac{3 \cdot p`}{1 + n} \qquad (3.4).$$

Згідно з формулою (3.4) за умови досягнення однакового рівня зайнятої пам’яті під час використання кожного з методів, кількість маскувальних сигналів, що буде утворена за умови використання другого обчислюється за наступною формулою:

$$p = 2^{\frac{3 \cdot p`}{1+n} \cdot n}. \qquad (3.5)$$

Оскільки значення $n/(n+1)$ прямує до 1, можливо переписати формулу (3.5), як $2^{3 \bullet p}$, що значно перевищує кількість варіацій маскувальних сигналів для першого методу, оскільки $2^{3 \bullet p} > p$, для всіх значень p.

Відповідно до мети викладеної роботи, кількість часу, який витрачається на реалізацію дискретного перетворення Фур’є, повинна зменшитися. Для визначення того на скільки розроблений обчислювальний метод є швидшим необхідно обчислити коефіцієнт прискорення V, який характеризується відношенням часу, за який виконується перетворення Фур’є на мікроконтролері - T_{DFT} до часу, за який відбувається перетворення Фур’є з використанням запропонованого методу, тобто шифрування та дешифрування даних - T_S. Виходячи з попереднього твердження коефіцієнт можна визначити за наступною формулою:

$$V = \frac{T_{DFT}}{T_S}.$$

Цілком очевидно, що час за який виконується дискретне перетворення Фур'є характеризується кількістю операцій, які необхідно зробити для отримання результату. Відповідно для визначення часу, який потрібен на виконання дискретного перетворення Фур'є необхідно обчислити кількість додавань та множень. Таким чином, кількість множень дорівнює $2 \bullet n^2$, а кількість додавань рівна $2 \bullet n \bullet (n-1)$. Якщо позначити через τ_a час за який виконується операція додавання, а через τ_m час, за який виконується операція множення, то визначення часу, за який виконується дискретне перетворення Фур'є можливо записати наступним чином:

$$T_{DFT} = 2 \cdot n^2 \cdot \tau_m + 2 \cdot n \cdot (n-1) \cdot \tau_a. \quad (3.6)$$

Виходячи з того факту, що сучасні процесори виконують операцію множення з плаваючою комою приблизно в 30 разів швидше за операцію додавання, то формула (3.6) може бути записана у наступному вигляді:

$$T_{DFT} = 2 \cdot n \cdot \tau_a \cdot (31 \cdot n - 1).$$

Для розрахунку T_S необхідно обчислити час, який витрачається на кожному етапі розробленого методу:

1. Шифрування даних.
2. Відправка даних на віддалену комп'ютерну систему.
3. Обчислення операції дискретного перетворення Фур'є на віддаленій комп'ютерній системі.
4. Отримання результатів дискретного перетворення Фур'є з віддаленої комп'ютерної системи на мікроконтролер.
5. Дешифрування даних для отримання коректного результату.

Для шифрування сигналу необхідно додати до нього n відліків p маскувальних сигналів, тобто $p \bullet n$ значень. З цього випливає, що час, за який відбувається шифрування даних становить $p \bullet n \bullet \tau_a$.

Для того щоб здійснити дешифрування необхідно від всіх відліків кожної компоненти спектрального представлення сигналу відняти суми відповідних елементів з матриць частково обчислених значень. Для кожного з p сигналів для

дешифрування необхідно обчислити суми з 2•n•(n-1) значень. Оскільки розроблений метод передбачає обрахування сум елементів з матриць частково обчислених значень, які не залежать від отриманих з системи даних, то підготовку до операції дешифрування можливо розпочати під час етапів відправки та отримання даних.

Цілком очевидно, що загальний час пересилок даних T_t значно менший порівняно з усіма операціями, що відбуваються на мікроконтролері [74], з чого випливає, що значення T_t можливо не враховувати під час аналізу загального часу виконання запропонованого методу.

Таким чином, формула, за якою розраховується час дискретного перетворення Фур'є з використанням запропонованого методу визначається наступним чином:

$$T_S = p \cdot n \cdot \tau_a \cdot (2 \cdot n - 1).$$

Відповідно до цього, значення коефіцієнту прискорення розраховується за наступною формулою:

$$V = \frac{T_{DFT}}{T_S} = \frac{2 \cdot n \cdot \tau_a \cdot (31 \cdot n - 1)}{p \cdot n \cdot \tau_a \cdot (2 \cdot n - 1)} \approx \frac{31}{p}.$$

3.4. Розробка засобів програмної реалізації розробленого методу

Функціональне моделювання розробленого методу передбачає створення алгоритму, в якому чітко визначені усі кроки для реалізації захищеного перетворення Фур'є над сигналами, що надійшли на систему [75].

Для виконання захищеного перетворення Фур'є з використанням хмарних технологій запропонований метод передбачає реалізацію двох етапів. Виходячи з цього для опису реалізації запропонованого методу використовується два наступні алгоритми:

1. Підготовчий етап формування масок. На цьому етапі виконується генерація масок та часткове обчислення їх спектрального представлення.
2. Захищене перетворення Фур'є над сигналом. На цьому етапі виконується гомоморфне шифрування сигналу з використанням маскувальної матриці перед посилкою його для обробки в хмару, а також дешифрування

прийнятого з віддаленої комп’ютерної системи результату перетворення Фур’є, тобто спектрального представлення сигналу.

Перший блок першого алгоритму виводить користувачу інформацію про можливі способи введення масок. Тобто користувач обирає яким із наступних способів будуть утворюватися маскувальні сигнали:

1) Система генерує випадкові значення відліків маскувальних сигналів.
2) Користувач вводить маскувальні сигнали самостійно.

У другому блоці система зчитує відповідь користувача. В межах третього блоку відбувається перевірка введеного значення. Якщо користувач обрав другий метод, то здійснюється перехід, до блоку сім, інакше система виконує наступний блок. У четвертому блоці початкове значення індексу k встановлюється в 0. У наступному, п’ятому блоці алгоритму відбувається ініціалізація генератору випадкових чисел, з використанням якого формується k-та маска та значення k збільшується на 1. У шостому блоці алгоритму відбувається перевірка, значення індексу k, якщо він менший за значення p, то здійснюється перехід до блоку під номером п’ять, інакше - відбувається перехід до дев’ятого блоку алгоритму. У наступному, сьомому блоці алгоритму система очікує введення користувачем маскувальних сигналів. У восьмому блоці система зчитує введені дані і записує їх у пам’ять. У межах виконання дев’ятого блоку алгоритму відбувається формування матриць з частково обчисленими значеннями спектрального представлення маскувальних сигналів. Для p масок обчислюється добуток відліків маскувального сигналу та коефіцієнтів реальної та уявної компонент. Таким чином для кожної маски у системі формується набір із двох матриць для частково обчислених значень реальної та уявної компоненти.

Перший блок другого алгоритму передбачає введення відліків сигналу. У другому блоці початкове значення індексу j встановлюється в 0. В межах третього блоку відбувається ініціалізація генератору випадкових чисел, з використанням якого формується j-ий рядок маскувальної матриці. Сформований рядок маскувальної матриці заповнюється бінарними значеннями, тобто 0 та 1,

випадковим чином і визначає які відліки *j*-ої маски використовуються для шифрування вхідного сигналу. У четвертому блоці значення *j* збільшується на 1. У наступному, п’ятому блоці алгоритму відбувається перевірка, значення індексу *j*, якщо він менший за значення *p*, то здійснюється перехід до блоку під номером три, інакше - відбувається перехід до наступного блоку алгоритму. Результатом виконання блоків 2-5 є формування маскувальної матриці для вхідного сигналу. У шостому блоці початкове значення індексу *j* встановлюється в 0. У сьомому блоці початкове значення індексу *i* встановлюється в 0. В межах восьмого блоку відбувається перевірка значення *i*-го стовпця в *j*-му рядку маскувальної матриці , якщо воно рівне 0, то здійснюється перехід до десятого блоку алгоритму, а якщо воно рівне 1, то здійснюється перехід, до наступного блоку. У дев’ятому блоці алгоритму відбувається операція суми між *i*-им відліком *j*-ої маски та *i*-им відліком сигналу. У десятому блоці значення індексу *i* збільшується на 1. У наступному, одинадцятому блоці алгоритму відбувається перевірка, значення індексу *i*, якщо він менший за значення *n*, то здійснюється перехід до блоку під номером вісім, інакше - відбувається перехід до наступного блоку алгоритму. У дванадцятому блоці значення індексу *j* збільшується на 1. У наступному, тринадцятому блоці алгоритму відбувається перевірка, значення індексу *j*, якщо він менший за значення *p*, то здійснюється перехід до блоку під номером сім, інакше - відбувається перехід до наступного блоку алгоритму. Після формування зашифрованого сигналу в межах блоків 6-13 у чотирнадцятому блоці відбувається дискретне перетворення Фур’є. У п’ятнадцятому блоці система отримує зашифровані значення реальної та уявної компонент спектру.

У шістнадцятому блоці початкове значення індексу *j* встановлюється в 0. У сімнадцятому блоці початкове значення індексу *i* встановлюється в 0. В межах вісімнадцятому блоку відбувається перевірка значення *i*-го стовпця в *j*-му рядку маскувальної матриці, якщо воно рівне 0, то здійснюється перехід до двадцять першого блоку алгоритму, а якщо воно рівне 1, то здійснюється перехід, до наступного блоку. У дев’ятнадцятого блоці алгоритму відбувається операція

віднімання від *i*-го відліку зашифрованої реальної компоненти суми значень *i*-го стовпця *j*-ої відповідної матриці. У двадцятому блоці алгоритму відбувається операція віднімання від *i*-го відліку зашифрованої уявної компоненти суми значень *i*-го стовпця *j*-ої відповідної матриці. У двадцять першому блоці значення індексу *i* збільшується на 1. У наступному, двадцять другому блоці алгоритму відбувається перевірка, значення індексу *i*, якщо він менший за значення *n*, то здійснюється перехід до блоку під номером вісімнадцять, інакше - відбувається перехід до наступного блоку алгоритму. У двадцять третьому блоці значення індексу *j* збільшується на 1. У наступному, двадцять четвертому блоці алгоритму відбувається перевірка, значення індексу *j*, якщо він менший за значення *p*, то здійснюється перехід до блоку під номером сім, інакше - відбувається перехід до наступного блоку алгоритму. Результатом всіх операцій в межах блоків 16-24 є дешифровані значення реальної та уявної компонент спектру.

Висновки до розділу 3

В результаті виконання теоретичних та практичних досліджень в межах третього розділу сформульовано наступні висновки:

1. Розроблено новий метод захищеної реалізації дискретного перетворення Фур'є, основою якого є використання адитивного маскуванні груп сигналів. Характерна особливість розробленого методу полягає в динамічній зміні кодів маски, що досягається шляхом використання маскувальних матриць. Для шифрування в запропонованому методі відліки масок адитивно накладаються на вхідний сигнал, відповідно до маскувальної матриці. Відповідно до цього для того щоб здійснити дешифрування отриманих реальної та уявної компонент від них віднімаються обчислені значення, які зберігаються в пам'яті мікроконтролера.

2. Проведено оцінку ефективності запропонованого методу згідно зі стандартними характеристиками. Розроблений метод показує значне підвищення рівня захищеності завдяки динамічній зміні кодів маски, що використовується для шифрування. Також слід зазначити, що запропонований метод дозволяє знизити обчислювальну складність дискретного перетворення Фур'є, завдяки використанню віддалених обчислювальних потужностей.

3. Теоретично та експериментально доведено конструктивність запропонованого методу. Ретельний аналіз показав, що після застосування методу до вхідних даних, реальні та уявні компоненти спектрального представлення дійсно відповідають дискретному перетворенню Фур'є. Це підтверджено шляхом виведення математичних формул, які точно описують процес шифрування та дешифрування і демонструють його коректність. Крім того, у розділі наведено приклад використання розробленого методу, який демонструє правильність отриманих результатів для вхідних даних. Цей приклад включає детальний опис процесу обробки вхідних даних. Таким чином, проведене дослідження обґрунтувало коректність запропонованого методу з теоретичної та практичної точки зору.

4. Виконано функціональне моделювання запропонованого методу захищеної реалізації дискретного перетворення Фур’є на віддалених комп’ютерних системах. Результати моделювання показали високу ефективність методу в умовах реального часу. Моделювання підтвердило здатність методу забезпечувати належний рівень безпеки і ефективності при обробці великих обсягів даних, що є важливим для розширення можливостей систем Інтернету речей і підвищення їх надійності. Таким чином, рівень захищеності зростає в експоненційно, що забезпечує значно вищу безпеку даних у порівнянні з відомими методами, а швидкість обробки даних збільшується на один порядок.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ МОДЕЛЮВАННЯ ЗАПРОПОНОВАНОГО МЕТОДУ ЗАХИЩЕНОЇ РЕАЛІЗАЦІЇ ДПФ

В межах четвертого розділу викладеної роботи описано результати розробки програмних засобів для реалізації двох запропонованих методів захищеної реалізації дискретного перетворення Фур'є в хмарах. Розроблене програмне забезпечення реалізує імітаційне моделювання двох представлених у роботі методів. Розроблене програмне забезпечення реалізує наступні задачі для кожного з описаних у другому та третьому розділах методів:

- Введення вхідних сигналів та параметрів для подальшої обробки сигналів та генерації маскувальних значень.
- Генерація маскувальних значень та обчислення їх реальної та уявної компоненти спектрального представлення. Відповідно до обраного методу компоненти спектру можуть обчислення повністю або частково. Генерація маскувальних значень та обчисленні їх компонент спектрального представлення відбувається відповідно до введених користувачем параметрів.
- Гомоморфне шифрування сигналу, яке передбачає використання значень для маскування згідно з алгоритмами розроблених методів та відповідно до введених користувачем параметрів.
- Імітація відправки зашифрованого сигналу на віддалену комп’ютерну систему з використанням хмарних технологій для здійснення дискретного перетворення Фур’є.
- Дешифрування прийнятого з віддаленої комп’ютерної системи результату дискретного перетворення Фур’є, тобто спектрального представлення сигналу, згідно з алгоритмами розроблених методів та відповідно до введених користувачем параметрів.

Мовою програмування для реалізації розроблених методів захищеного перетворення Фур’є було обрано C++. Даний вибір обґрунтовано тим, що ця мова програмування в повній мірі надає можливості використання апаратних потужностей обчислювальної платформи.

4.1. Проектування структурної організації програмного забезпечення

Як вже було зазначено вище запропоноване програмне забезпечення здійснює реалізацію двох запропонованих вище методів ефективної реалізації дискретного перетворення Фур'є.

Структура першого методу складається із наступних модулів:

1. Модуль генерації маскувальних значень;
2. Модуль зчитування вхідного сигналу;
3. Модуль шифрування;
4. Модуль реалізації дискретного перетворення Фур'є;
5. Модуль дешифрування;
6. Модуль запису.

Розроблений метод передбачає генерацію маскувальних значень перед початком роботи системи. Дана задача виконується модулем генерації масок, у якому можливо здійснити запис маскувальних сигналів в два можливі способи. В першому способі користувач сам записує маскувальні значення в пам'ять на рівні програми. У другому способі відбувається виклик генератору випадкових чисел, який формує маскувальні сигнали. Значення маскувальних сигналів записуються до двовимірного масиву чисел з плаваючою комою masks.

Значення з масиву плаваючою комою masks надходять на модуль дискретного перетворення Фур'є, який виконує обчислення значень реальної та уявної компонент спектру для кожного маскувального сигналу. Отримані значення всіх реальних компонент спектру записуються у двовимірний масив з плаваючою точкою mask_prev_real. Отримані значення всіх уявних компонент спектру записуються у двовимірний масив з плаваючою комою mask_prev_imaginary.

Після закінчення роботи другого модуля перший етап цього методу вважається завершеним. Далі починається виконання модуля зчитування вхідних сигналів. У цьому модулі відбувається імітація надходження сигналу на систему. Спочатку у цей модуль виконує зчитування з консолі кількості відліків на які

поділяється сигнал. Далі модуль зчитування може виконати одну із наступних операцій: генерація випадкового сигналу, для демонстрації прикладу, і зчитування значень які користувач вводить у консоль. Отримані значення відліків вхідного сигналу записуються у одновимірний масив з плаваючою комою input.

Після завершення роботи модуля, який реалізує зчитування сигналу, дані отримані після його виконання, тобто масив input надходить до модуля шифрування. Основною задачею даного модуля є шифрування відліків вхідного сигналу, шляхом додавання до них відліків маскувальних значень або попередніх сигналів, за умови що вони є. Вибір між маскувальними значеннями та попередніми сигналами відбувається випадковим чином. Отримані значення відліків зашифрованого вхідного сигналу записуються у одновимірний масив з плаваючою комою output_Q.

Значення з масиву плаваючою комою output_Q надходять на модуль дискретного перетворення Фур'є, який виконує обчислення значень реальної та уявної компонент спектру для кожного вхідного сигналу. Отримані значення реальної компоненти спектру записуються у одновимірний масив з плаваючою точкою output_a. Отримані значення уявної компоненти спектру записуються у одновимірний масив з плаваючою комою output_b.

Зашифровані значення компонент спектру з модуля дискретного перетворення Фур'є, тобто масиви output_a та output_b, надходять до модуля дешифрування. Основною задачею даного модуля є дешифрування відліків реальної та уявної компонент спектру. Для дешифрування реальної компоненти спектрального представлення необхідно від її відліків відняти значення реальної компоненти маскувальних значень або попередніх сигналів. Для дешифрування уявної компоненти спектрального представлення необхідно від її відліків відняти значення уявної компоненти маскувальних значень або попередніх сигналів. Вибір між маскувальними значеннями та попередніми сигналами відбувається згідно того, що було обрано в модулі шифрування. Отримані значення розшифрованих реальних компонент спектру записуються у одновимірний масив

з плаваючою точкою output_a. Отримані значення всіх уявних компонент спектру записуються у одновимірний масив з плаваючою комою output_b.

Після виконання модуля дешифрування запускається модуль запису отриманих даних. Цей модуль передбачає виконання запису даних, які можуть бути використані для подальших обчислень. Модуль запису запускається лише якщо виконується одна з наступних умов: в масиви попередніх сигналів недостатня кількість записів, або в модулі шифрування було використано попередній сигнал. Якщо виконується перша умова, то цей модуль виконує наступні операції:

- Вхідний сигнал додається до пулу попередніх сигналів, який представляє собою двовимірний масив prev_signals.
- Розшифрована реальна компонента вхідного сигналу додається до пулу реальних компонент попередніх сигналів, тобто до двовимірного масиву signal_prev_r.
- Розшифрована уявна компонента вхідного сигналу додається до пулу уявних компонент попередніх сигналів, тобто до двовимірного масиву signal_prev_im.

Якщо виконується друга умова, то цей модуль виконує наступні операції:

- Сигнал, який було використано для шифрування, видаляється з пулу попередніх, а на його місце записується вхідний сигнал. Описані дії вкинуються стосовно двовимірного масиву prev_signals.
- Реальна компонента сигналу, який було використано для шифрування, видаляється з пулу реальних компонент попередніх сигналів. На її місце записується реальна компонента вхідного сигналу. Описані дії вкинуються стосовно двовимірного масиву signal_prev_r.
- Уявна компонента сигналу, який було використано для шифрування, видаляється з пулу уявних компонент попередніх сигналів. На її місце записується уявна компонента вхідного сигналу. Описані дії вкинуються стосовно двовимірного масиву signal_prev_im.

Структура другого методу складається із наступних модулів:

1. Модуль генерації маскувальних значень;
2. Модуль часткового обчислення дискретного перетворення Фур’є для маскувальних сигналів;
3. Модуль зчитування вхідного сигналу;
4. Модуль створення маскувальної матриці;
5. Модуль шифрування;
6. Модуль реалізації дискретного перетворення Фур’є;
7. Модуль дешифрування.

Розроблений метод передбачає генерацію маскувальних значень перед початком роботи системи. Дана задача виконується модулем генерації масок, у якому можливо здійснити запис маскувальних сигналів в два можливі способи. В першому способі користувач сам записує маскувальні значення в пам’ять на рівні програми. У другому способі відбувається виклик генератору випадкових чисел, який формує маскувальні сигнали. Значення маскувальних сигналів записуються до двовимірного масиву чисел з плаваючою комою masks.

Значення з масиву плаваючою комою masks надходять на модуль часткового обчислення маскувальних сигналів, який виконує частини обчислень, які необхідні для розрахунку значень реальної та уявної компонент спектру для кожного маскувального сигналу за стандартними формулами. Таким чином даний модуль передбачає лише виконання операції множення без операції суми, яка присутня у стандартному методі. Отримані частково обчислені значення реальної компоненти спектру записуються у двовимірний масив з плаваючою точкою output_q. Отримані значення уявної компоненти спектру записуються у двовимірний масив з плаваючою комою output_w.

Після закінчення роботи другого модуля перший етап цього методу вважається завершеним. Далі починається виконання модуля зчитування вхідних сигналів. У цьому модулі відбувається імітація надходження сигналу на систему. Спочатку у цей модуль виконує зчитування з консолі кількості відліків на які поділяється сигнал. Далі модуль зчитування може виконати одну із наступних

операцій: генерація випадкового сигналу, для демонстрації прикладу, і зчитування значень які користувач вводить у консоль. Отримані значення відліків вхідного сигналу записуються у одновимірний масив з плаваючою комою input.

Після завершення роботи модуля, який реалізує зчитування сигналу, розмір введеного сигналу, тобто масиву input надходить до модуля створення маскувальної матриці. Основною задачею даного модуля є створення маскувальної матриці для кожного сигналу, яка потрібна для того, щоб здійснювати динамічну зміну кодів маски. Заповнення маскувальної матриці для сигналу відбувається бінарними значеннями випадковим чином. Згенерована маскувальна матриця записується у двовимірний масив з плаваючою комою mask_mtrx.

Після завершення роботи модуля створення маскувальної матриці дані отримані в результаті його виконання, тобто двовимірний масив mask_mtrx надходить до модуля шифрування. Основною задачею даного модуля є шифрування відліків вхідного сигналу, шляхом додавання до них відліків маскувальних значень, відповідно до того як заповнена маскувальна матриця. Отримані значення відліків зашифрованого вхідного сигналу записуються у одновимірний масив з плаваючою комою output_Q.

Значення з масиву плаваючою комою output_Q надходять на модуль дискретного перетворення Фур’є, який виконує обчислення значень реальної та уявної компонент спектру для вхідного сигналу. Отримані значення реальної компоненти спектру записуються у одновимірний масив з плаваючою точкою output_a. Отримані значення уявної компоненти спектру записуються у одновимірний масив з плаваючою комою output_b.

Зашифровані значення компонент спектру з модуля дискретного перетворення Фур’є, тобто масиви output_a та output_b, надходять до модуля дешифрування. Основною задачею даного модуля є дешифрування відліків реальної та уявної компонент спектру. Для дешифрування реальної компоненти спектрального представлення необхідно від її відліків відняти суму відліків

значень частково обчислених реальних компонент маскувальних значень з масиву mask_real згідно із маскувальною матрицею. Для дешифрування уявної компоненти спектрального представлення необхідно від її відліків відняти суму відліків значень частково обчислених уявних компонент маскувальних значень з масиву mask_imaginary згідно із маскувальною матрицею. Отримані значення розшифрованих реальних компонент спектру записуються у одновимірний масив з плаваючою точкою output_a. Отримані значення всіх уявних компонент спектру записуються у одновимірний масив з плаваючою комою output_b.

4.2. Функціональна архітектура програмного забезпечення

Задачею функції Input(vector<double>& in, int length, int mode) є імітація надходження сигналу на платформу користувача. У якості параметрів функція приймає вектор in та дві змінні length, та mode. Вектор in призначенний для зберігання отриманих даних. Змінна length визначає розмір сигналу, який отримує система. Змінна mode використовується для визначення способу у який буде вводитися сигнал. В рамках функції організовано два різні цикли for із змінною i, який саме цикл буде виконано залежить від змінної mode. У першому циклі виконується випадкова генерація чисел з плаваючою комою, які додаються до вектору in. У другому циклі система зчитує дані введені у консоль користувачем і додає їх до вектору in.

Задачею функції Output(string name, vector<double>& in) є вивід одновимірних масивів. У якості параметрів функція приймає рядок name та вектор in. Рядок name використовується у функції для зберігання рядку, який позначає назву масиву. Вектор in використовується для передачі у функцію масиву чисел з плаваючою комою, який потрібно вивести. В рамках функції організовано один цикл for із змінною i, кількість ітерацій залежить від розміру вхідного вектору in. Перед початком виконання циклу у консоль виводиться назва масиву, який передано у функцію з використанням рядка name. В межах циклу виконується вивід у консоль значень з плаваючою комою з точністю два знаки після коми.

Задачею функції Output_Mas(string name, vector<vector<double>>& in) є вивід двовимірних масивів. У якості параметрів функція приймає рядок name та масив векторів in. Рядок name використовується у функції для зберігання рядку, який позначає назву масиву. Масив векторів in використовується для передачі у функцію значень, які потрібно вивести. В рамках функції організовано один вкладений цикл for. У зовнішньому циклі for використовується змінна i, кількість ітерацій залежить від кількості векторів в масиві in. У внутрішньому циклі for використовується змінна l, кількість ітерацій залежить від розміру вектору, який зберігається на i-му місці в масиві in. Перед початком виконання циклу у консоль виводиться назва масиву, який передано у функцію з використанням рядка name. В межах циклів виконується вивід у консоль значень з плаваючою комою з точністю два знаки після коми, причому кожен новий вектор виводиться з нового рядка.

Задачею функції DFT(vector<double>& in, vector<double>& out_r, vector<double>& out_y) є обчислення компонент реальної та уявної складової спектрального представлення потрібного сигналу. У якості параметрів функція приймає вектор in, вектор out_r та вектор out_y. Вектор in використовується для передачі у функцію значень, над якими виконується дискретне перетворення Фур’є. Вектор out_r використовується для збереження обчислених значень реальної складової спектрального представлення сигналу in. Вектор out_y використовується для збереження обчислених значень уявної складової спектрального представлення сигналу in. В рамках функції організовано один вкладений цикл for. У зовнішньому циклі for використовується змінна i, кількість ітерацій залежить від кількості відліків в масиві in. У внутрішньому циклі for використовується змінна l, кількість ітерацій залежить в від кількості відліків в масиві in. В межах внутрішнього циклу for до змінних r та y додаються значення обраховані за формулами (1.2) та (1.3). В межах зовнішнього циклу for обчислені значення r та y додаються до векторів out_r та out_y відповідно. В результаті

виконання усіх операцій в межах цієї функції є заповнені масиви реальної та уявної складової спектрального представлення сигналу.

Задачею функції DFT_partly(vector<double> & in, vector<vector<double>>& out_r, vector<vector<double>> & out_y) є часткове обчислення значень компонент реальної та уявної складової спектрального представлення потрібного сигналу. У якості параметрів функція приймає вектор in, вектор векторів out_r та вектор векторів out_y. Вектор in використовується для передачі у функцію значень, над якими виконується дискретне перетворення Фур'є. Вектор векторів out_r використовується для збереження частково обчислених значень реальної складової спектрального представлення сигналу in. Вектор out_y використовується для збереження частково обчислених значень уявної складової спектрального представлення сигналу in. В рамках функції організовано один вкладений цикл for. У зовнішньому циклі for використовується змінна i, кількість ітерацій залежить від кількості відліків в масиві in. У внутрішньому циклі for використовується змінна l, кількість ітерацій залежить в від кількості відліків в масиві in. В межах внутрішнього циклу for обчислюються значення r та y за формулами (1.2 та 1.3) і додаються до векторів vector_r та vector_y. В межах зовнішнього циклу for сформовані вектори vector_r та vector_y додаються до out_r та out_y відповідно. В результаті виконання усіх операцій в межах цієї функції є заповнені масиви частково обчислених компонент реальної та уявної складової спектрального представлення сигналу.

Задачею функції void Masks(int mask_amount, int samples_amount, vector<vector<double>>& mask_mtrx, int mode) є запис до масиву маскувальних сигналів масок. У якості параметрів функція приймає змінну mask_amount, змінну samples_amount вектор векторів mask_mtrx та змінну mode. Змінна mask_amount визначає кількість масок, які згенерує система для подальшого шифрування. Змінна samples_amount визначає кількість відліків в одному маскувальному сигналі, який система генерує. Вектор векторів mask_mtrx застосовується для передачі у функцію масиву, в який після відпрацювання

функції будуть записані значення маскувальних сигналів. Змінна mode використовується для визначення режиму, в якому буде здійснена генерація масок. На початку виконання функція перевіряє значення змінної mode, оскілки є можливість згенерувати маскувальні значення або розглянути демонстраційний приклад, оскільки для демонстрації роботи запропонованого методу використовується заздалегідь сформований масив маскувальних значень. Для генерації маскувальних значень випадковим чином у межах функції організовано один вкладений цикл. У зовнішньому циклі for використовується змінна i, кількість ітерацій залежить від кількості маскувальних сигналів mask_amount. У внутрішньому циклі for використовується змінна j, кількість ітерацій залежить від кількості відліків для маскувальних сигналів, тобто samples_amount. В межах внутрішнього циклу for генерується випадкове число з плаваючою комою і додається до вектору vector_row. В межах зовнішнього циклу for сформований вектор vector_row додається до результуючої матриці mask_mtrx. В результаті виконання усіх операцій в межах цієї функції заповнюється масив маскувальних сигналів, які потім використовуються для виконання шифрування.

Задачею функції Create_Mask_Matrix(int mask_amount, int samples_amount, vector<vector<double>>& mask_mtrx, int mode) є генерація маскувальної матриці, яка заповнюється бінарними значеннями і забезпечує динамічну зміну маскувальних кодів. У якості параметрів функція приймає змінну mask_amount, змінну samples_amount вектор векторів mask_mtrx та змінну mode. Змінна mask_amount визначає кількість масок, які використовуються для шифрування сигналу. Змінна samples_amount визначає кількість відліків в сигналі, який система шифру. Вектор векторів mask_mtrx застосовується для передачі у функцію масиву, в який після відпрацювання функції буде записана маскувальна матриця. Змінна mode використовується для визначення режиму, в якому буде здійснена генерація маскувальної матриці. На початку виконання функція перевіряє значення змінної mode, оскілки є можливість згенерувати маскувальну матрицю випадковим чином або розглянути значення, які записані в системі для

демонстраційного прикладу. Для генерації маскувальної матриці випадковим чином у межах функції організовано один вкладений цикл. У зовнішньому циклі for використовується змінна i, кількість ітерацій залежить від кількості маскувальних сигналів, які використовуються для шифрування - mask_amount. У внутрішньому циклі for використовується змінна j, кількість ітерацій залежить від кількості відліків на які поділяється кожен вхідний сигнал, тобто samples_amount. В межах внутрішнього циклу for генерується випадкове бінарне значення (одиниця або нуль) і додається до вектору vector_row. В межах зовнішнього циклу for сформований вектор vector_row додається до результуючої матриці mask_mtrx. В результаті виконання усіх операцій в межах цієї функції заповнюється маскувальна матриця, яка використовується для визначення того які відліки кожної маски шифрують вхідний сигнал.

Задачею функції void Encryption_1(vector<double>& mask, vector<double>& signal_q, vector<vector<double>>& other_signals, int signal_num, int flag) є здійснення шифрування, яке представлене у першому розробленому методі. У якості параметрів функція приймає вектор mask, вектор signal_q, вектор векторів other_signals, змінну signal_num та змінну flag. Вектор mask застосовується для передачі у функцію маскувального сигналу, який використовується для шифрування. Вектор signal_q застосовується для зберігання значень вхідного сигналу, над яким у функції відбувається шифрування. Вектор векторів other_signals застосовується для передачі у функцію масиву попередніх сигналів, які використовуються для шифрування. Змінна signal_num використовується для визначення номеру попереднього сигналу, який використовується для шифрування. Змінна flag використовується для визначення того що використовується для шифрування вхідного сигналу (маскувальні або інші сигнали). В рамках функції організовано один цикл for із змінною j, кількість ітерацій залежить від розміру вектору signal_q. В межах циклу відбувається додавання до signal_q, у якому на початку зберігається вхідний сигнал, відліків попередніх сигналів або маски в залежності від значення змінної flag. В

результаті виконання усіх операцій в межах цієї функції відбувається шифрування вхідного сигналу.

Задачею функції void Decryption_1(vector<double>& signal_r, vector<double>& signal_im, vector<double>& mask_r, vector<double>& mask_im, vector<vector<double>>& signal_prev_r, vector<vector<double>>& signal_prev_im, int signal_num, int flag) є здійснення дешифрування, яке представлене у першому розробленому методі. У якості параметрів функція приймає вектор signal_r, вектор signal_im, вектор mask_r, вектор mask_ im, вектор векторів signal_prev_r, вектор векторів signal_prev_im, змінну signal_num та змінну flag. Вектор signal_r застосовується для зберігання значень зашифрованої реальної компоненти, яка буде розшифрована в межах цієї функції. Вектор signal_im застосовується для зберігання значень зашифрованої уявної компоненти, яка буде розшифрована в межах цієї функції. Вектор mask_r застосовується для передачі у функцію реальної компоненти маскувального сигналу, який використано для шифрування. Вектор mask_im застосовується для передачі у функцію уявної компоненти маскувального сигналу , який використано для шифрування. Вектор векторів signal_prev_r застосовується для передачі у функцію масиву реальних компонент попередніх сигналів, які використовуються для дешифрування. Вектор векторів signal_prev_im застосовується для передачі у функцію масиву уявних компонент попередніх сигналів, які використовуються для дешифрування. Змінна signal_num використовується для визначення номеру попереднього сигналу, який використовується для дешифрування. Змінна flag використовується для визначення того що використовується для дешифрування вхідного сигналу (маскувальні або інші сигнали). В рамках функції організовано один цикл for із змінною j, кількість ітерацій залежить від розміру вектору signal_r. В межах циклу відбувається віднімання від зашифрованих компонент спектру signal_r та signal_im, відповідних відліків компонент попередніх сигналів або маски в залежності від значення змінної flag. В результаті виконання усіх операцій в

межах цієї функції відбувається дешифрування компонент спектрального представлення вхідного сигналу.

Задачею функції void Encryption_2(vector<double>& mask, vector<double>& signal_q, vector<double>& mask_mtrx) є здійснення шифрування, яке представлене у другому розробленому методі. У якості параметрів функція приймає вектор mask, вектор signal_q та вектор mask_mtrx. Вектор mask застосовується для передачі у функцію маскувального сигналу, який використовується для шифрування. Вектор signal_q застосовується для зберігання значень вхідного сигналу, над яким у функції відбувається шифрування. Вектор mask_mtrx використовується для передачі у функцію рядку маскувальної матриці, відповідно до якої відбувається зміна кодів маски. В рамках функції організовано один цикл for із змінною j, кількість ітерацій залежить від розміру вектору signal_q. В межах циклу відбувається додавання до signal_q, у якому на початку зберігається вхідний сигнал, відліків маски відповідно до значень в маскувальній матриці. В результаті виконання усіх операцій в межах цієї функції відбувається шифрування вхідного сигналу.

Задачею функції void Decryption_2(vector<double>& real, vector<double>& imagine, vector<double>& mask_mtrx, vector<vector<double>>& mask_real, vector<vector<double>>& mask_imagine) є здійснення дешифрування, яке представлене у другому розробленому методі. У якості параметрів функція приймає вектор real, вектор imagine, вектор mask_mtrx, вектор векторів mask_real, вектор векторів mask_imagine. Вектор real застосовується для зберігання значень зашифрованої реальної компоненти, яка буде розшифрована в межах цієї функції. Вектор imagine застосовується для зберігання значень зашифрованої уявної компоненти, яка буде розшифрована в межах цієї функції. Вектор векторів mask_real застосовується для передачі у функцію масиву частково обчислених відліків реальної компоненти маскувального сигналу, який використано для шифрування. Вектор векторів mask_imagine застосовується для передачі у функцію масиву частково обчислених відліків уявної компоненти маскувального

сигналу, який використано для шифрування. Вектор mask_mtrx використовується для передачі у функцію рядку маскувальної матриці, відповідно до якої відбувається зміна кодів маски. Для реалізації дешифрування в рамках функції організовано один вкладений цикл. У зовнішньому циклі for використовується змінна j, кількість ітерацій залежить від кількості маскувальних сигналів, які використовуються для шифрування. У внутрішньому циклі for використовується змінна k, кількість ітерацій залежить від кількості відліків на які поділяється кожен вхідний сигнал. В межах внутрішнього циклу відбувається віднімання від зашифрованих компонент спектру real та imagine, відповідних відліків компонент маски в залежності від значень в маскувальній матриці. В результаті виконання усіх операцій в межах цієї функції відбувається дешифрування компонент спектрального представлення вхідного сигналу.

4.3. Структурна організація взаємодії в програмному забезпеченні

У програмі, яка реалізує виконання двох розроблених методів для захищеного перетворення Фур'є взаємодія між файлами організована наступним чином:

Основний файл програми, ConsoleApplication3.cpp, містить точку входу, тобто функцію main, яка ініціює виконання програми. У цьому файлі здійснюється початкова ініціалізація, а також викликаються функції, що знаходяться в інших файлах програми.

Файл InOut.h є заголовковим файлом, який містить декларації функцій для введення та виведення даних. Ці функції реалізовані у файлі InOut.cpp. Заголовковий файл оголошує функції, роблячи їх доступними для використання в інших частинах програми.

У файлі InOut.cpp реалізовані наступні функції для введення та виведення даних, які були оголошені в InOut.h:

- void Input – функція для введення до пам'яті значень відліків вхідного сигналу;
- void Output – функція для виведення у консоль значень одновимірних масивів;

- void Output_Mas - функція для виведення у консоль значень двовимірних масивів.

Файл Source.h є ще одним заголовковим файлом, який містить декларації функцій для обробки даних. Ці функції реалізовані у файлі Source.cpp. Як і у випадку з InOut.h, заголовковий файл Source.h оголошує функції, роблячи їх доступними для інших частин програми.

У файлі Source.cpp реалізовані наступні функції для обробки вхідних даних, оголошені у файлі Source.h:

- void DFT – функція для реалізації дискретного перетворення Фур'є;
- void DFT_partly – функція для реалізації часткового дискретного перетворення Фур'є;
- void Masks – функція для створення маскувальних значень;
- void Create_Mask_Matrix – функція для реалізації створення маскувальної матриці;
- void Encryption_1 – функція для реалізації шифрування над вхідним сигналом за першим запропонованим методом;
- void Decryption_1 – функція для реалізації дешифрування над вхідним сигналом за першим запропонованим методом;
- void Encryption_2 – функція для реалізації шифрування над вхідним сигналом за другим запропонованим методом;
- void Decryption_2 – функція для реалізації дешифрування над вхідним сигналом за другим запропонованим методом.

У загальному, ConsoleApplication3.cpp включає заголовкові файли InOut.h та Source.h, щоб мати доступ до оголошених у них функцій. Це дозволяє основному файлу викликати функції, реалізовані в InOut.cpp та Source.cpp.

Ця структура дозволяє розподілити код на логічні модулі, що полегшує розробку, тестування та підтримку програми. Заголовкові файли містять декларації функцій, які реалізовані у відповідних файлах реалізації. Така

організація коду сприяє зрозумілості та модульності програми, забезпечуючи легкий доступ до функцій з різних частин програми.

4.4. Структура організації даних в програмному забезпеченні

В основному файлі програми, що розробляється, дані організовані за допомогою різних типів векторів. Ця структура забезпечує зберігання та обробку числових даних у програмі. У даній програмі вектори використовуються для зберігання вхідних даних, результатів обробки, масок, попередніх сигналів та інших числових даних. Вони забезпечують гнучкість, ефективність і безпеку роботи з динамічними наборами даних, що є ключовими для успішної реалізації алгоритмів дискретного перетворення Фур'є та їхньої захищеної реалізації на віддалених обчислювальних потужностях. Нижче наведено опис даних, які використовуються у програмі.

Для зберігання відліків вхідного сигналу у програмі використовується вектор input призначений для зберігання даних у вигляді чисел з плаваючою точкою. Для збереження результатів обробки вхідного сигналу використовуються вектори output_a та output_b, які являють собою реальну та уявну компоненти спектрального представлення сигналу. Вектор output_Q використовується для зберігання відліків зашифрованого сигналу. Двовимірні вектори, де кожен елемент є вектором чисел з плаваючою точкою, output_q та output_w зберігають матрицю результатів часткового дискретного перетворення Фур'є для маскувальних сигналів. Двовимірний вектор masks зберігає множину масок, що використовуються при обробці даних. Двовимірний вектор mask_mtrx призначений для зберігання матриці масок. Двовимірний вектор prev_signals зберігає попередні сигнали для подальшого використання при шифруванні, яке передбачає перший метод. Вектор prev_signals_nums зберігає цілі числа, що, відповідають номерам доданих до списку попередніх сигналів. Тривимірні вектори mask_real та mask_imaginary зберігають частково обчислені реальні та уявні компоненти масок, які використовуються у шифруванні. Двовимірні вектори signal_prev_r та signal_prev_im зберігають інформацію про реальну та

уявну компоненти спектрального представлення попередніх сигналів. Двовимірні вектори mask_prev_r та mask_prev_im зберігають інформацію про реальну та уявну комопненти спектрального представлення маскувальних сигналів. Змінна mask_number встановлює кількість маскувальних сигналів, які використовуються для шифрування. Змінна prev_signals_num встановлює кількість попередніх сигналів, які зберігаються у пам'яті програми для подальшого використання при шифруванні. Змінна signals_num зберігає поточну кількість сигналів. Змінна signal_num зберігає індекс попереднього сигналу, який використовується для шифрування. Змінна mask_num зберігає індекс маски, яка використовується для шифрування. Змінна flag використовується для визначення типу даних, які використовується для шифрування з першим методом. Змінна prog_mode використовується для того, щоб користувач міг вводити необмежену кількість сигналів. Змінна length зберігає довжину оброблюваного сигналу. Змінна method зберігає вибраний метод шифрування.

4.5. Інструкція користувачеві для роботи з програмним забезпеченням

Для успішного запуску та використання програми, необхідно слідувати певним інструкціям. Спочатку користувачу необхідно запустити файл ConsoleApplication3.exe подвійним клацанням або через командний рядок. Після запуску програми у консолі будуть виведені інструкції, які потрібно виконувати.

Коли програма запитає введення значень сигналу або маски, їх необхідно вводити через кому, а після запису натискати клавішу Enter. Наприклад, якщо потрібно ввести значення маски або сигналу, їх треба записувати у наступному форматі «12.5» і натиснути Enter. Всі інші типи даних, що потребуються програмою, є цілими числами. Треба записати необхідне значення у консоль і натикнути Enter після кожного введення. Наприклад, якщо потрібно ввести ціле число, треба ввести «3» і натиснути Enter.

Під час виконання програма ConsoleApplication3.exe виводить проміжні значення обчислень у консоль користувача.

Висновки до розділу 4

В результаті розробки програмного забезпечення, яке виконує імітацію захищеної реалізації дискретного перетворення Фур'є на віддалених комп'ютерних системах з застосуванням розробленого методу сформульовано наступні висновки:

1. Було розроблено комплексну структуру програмного забезпечення, яка забезпечує зручність у використанні, підтримці та розширюваності. Структура враховує розподіл завдань між різними компонентами програми, що дозволяє легко інтегрувати нові функції та методи без значних змін в існуючому коді. Такий підхід сприяє зменшенню кількості помилок і полегшує налагодження.
2. Створено функціональні компоненти програмного забезпечення, які відповідають за шифрування, обробку даних та їх дешифрування. Архітектура побудована на модульному принципі, що забезпечує високу гнучкість та можливість масштабування. Модульний підхід дозволяє проводити незалежне тестування кожного компонента, що покращує загальну надійність та стабільність програми.
3. Описано детальну взаємодію між основними файлами програми. ConsoleApplication3.cpp виконує роль основного файлу, який координує роботу програми, викликаючи функції з InOut.cpp та Source.cpp. Загальна організація взаємодії забезпечує ефективний обмін даними між компонентами, мінімізуючи залежності та підвищуючи зручність у підтримці коду.
4. Детально описано структуру організації даних в програмному забезпеченні. Пояснена доцільність використання кожного використаного типу даних. використання векторів для зберігання вхідних та вихідних даних, а також проміжних результатів обчислень.
5. Розроблено детальну інструкцію для користувача, що описує процес запуску розробленої програми ConsoleApplication3.exe та подальші кроки взаємодії з програмою. Користувачеві необхідно слідувати інструкціям, що

виводяться в консолі, та вводити значення сигналів або масок у зазначеному форматі.

6. Результати роботи розробленого програмного забезпечення довели конструктивність запропонованого методу, оскільки обчислення стандартного перетворення Фур’є співпадають з даними отриманими при застосуванні розробленої програми. Крім того експериментально підтверджено зростання рівня захищеності даних більше ніж в 20 разів та збільшення швидкості обробки вхідного сигналу на 2 порядки.

ВИСНОВКИ

Результати проведених теоретичних та експериментальних досліджень в межах представленої роботи, основною метою якої є розробка методу захищеної реалізації дискретного перетворення Фур'є можуть бути сформульовані наступним чином:

1. Аналіз сучасних тенденцій у розвитку комп'ютерних технологій показав, що зростання популярності Інтернету речей робить пріоритетною задачу реалізації інтерфейсу між комп'ютером та навколишнім середовищем. Для реалізації такої взаємодії широко використовується цифрова обробка сигналів, заснована на методах перетворення Фур'є. Існує значна потреба у підвищенні якості аналізу та обробки вхідних даних, що вимагає збільшення кількості відліків для кожного сигналу, призводячи до значного збільшення часу на обчислення. Крім того, існують жорсткі вимоги щодо часу перетворення сигналу, оскільки більшість термінальних мікроконтролерів працюють у реальному часі та обробляють велику кількість даних, що потребує високої швидкості обробки. Для підвищення швидкості виконання дискретного перетворення Фур'є використовуються можливості віддалених комп'ютерних потужностей, доступ до яких забезпечується завдяки хмарним технологіям. Важливо зазначити, що мікроконтролери, які складають основу систем Інтернету речей, часто мають доступ у мережу, що дозволяє їм взаємодіяти з хмарними ресурсами. Проте, існує проблема використання непідконтрольних користувачу віддалених комп'ютерних систем, що вимагає додаткових заходів безпеки.

2. Проаналізовано вимоги, що висуваються до реалізації спектральних перетворень в системах комп'ютерного моніторингу стану об'єктів з використанням хмарних технологій. Внаслідок аналізу цих потреб сформовано дві характеристики для оцінки ефективності запропонованого варіанту реалізації дискретного перетворення Фур'є. В якості першої характеристики використовується рівень захищеності, яку забезпечує метод. В якості другої

характеристики оцінки ефективності використовується складність обчислювального методу.

3. Здійснено аналіз відомих схем гомоморфного шифрування перетворення Фур'є при їх обробці на віддалених комп'ютерних системах. В результаті огляду сучасних схем захищеної реалізації дискретного перетворення Фур'є виявлено, що жоден із них не забезпечує належного рівня ефективності через невідповідність вимогам, які висуває сучасний технологічний світ. Це стосується і потреб Інтернету речей, де швидкість обробки даних та їх захищеність є вирішальними факторами. Тому розробка нових методів, які враховують специфіку Інтернету речей, є важливим кроком у забезпеченні безпеки та ефективності обробки даних у цій сфері. Здійснено розробку методу захищеної реалізації дискретного перетворення Фур'є, який базується на адитивному маскуванні груп сигналів. Характерною особливістю запропонованого методу є здійснення шифрування з використанням випадково обраного попереднього або маскувального сигналу, який адитивно накладається на вхідні дані. Відповідно до цього для дешифрування отриманих сигналів реальної та уявної компоненти від них віднімаються відповідні обчислені значення, які зберігаються в пам'яті мікроконтролера.

4. Здійснено розробку методу захищеної реалізації дискретного перетворення Фур'є, який базується на адитивному маскуванні груп сигналів. Характерною особливістю запропонованого методу є здійснення шифрування з використанням випадково обраного попереднього або маскувального сигналу, який адитивно накладається на вхідні дані. Відповідно до цього для дешифрування отриманих сигналів реальної та уявної компоненти від них віднімаються відповідні обчислені значення, які зберігаються в пам'яті мікроконтролера.

5. Здійснено оцінку ефективності запропонованого методу відповідно до стандартних характеристик. Розроблений метод демонструє підвищення рівня захищеності на два порядки, завдяки використанню необмеженої кількості значень для шифрування. Також варто зазначити, що запропонований метод

дозволяє зменшити складність обчислень дискретного перетворення Фур'є завдяки можливостей віддалених комп'ютерних систем більше ніж в 20 разів.

6. Теоретично та практично доведено конструктивність розробленого методу. Детальний аналіз показав, що після його застосування до вхідних даних реальна та уявна компоненти спектрального представлення дійсно відповідають дискретному перетворенню Фур'є. Це підтверджено шляхом виведення математичних формул, які точно описують процес перетворення та демонструють його правильність. Крім цього, у представленій роботі наведено приклад застосування розробленого методу, який демонструє коректність отриманих результатів для вхідного сигналу. Цей приклад включає покроковий опис процесу обробки сигналу, починаючи від підготовки даних для шифрування і закінчуючи дешифруванням спектральних компонент. Таким чином, проведене дослідження не тільки підтвердило теоретичну обґрунтованість розробленого методу, але й продемонструвало його практичну цінність.

7. Здійснено функціональне моделювання розробленого методу захищеної реалізації дискретного перетворення Фур'є на віддалених обчислювальних потужностях. Алгоритм розроблено для подальшої реалізації програмного забезпечення. Основна мета полягає в забезпеченні швидкого та безпечного виконання дискретного перетворення Фур'є, зокрема, для обробки великих обсягів даних у режимі реального часу.

8. Розроблено новий метод захищеної реалізації дискретного перетворення Фур'є, основою якого є використання адитивного маскуванні груп сигналів. Характерна особливість розробленого методу полягає в динамічній зміні кодів маски, що досягається шляхом використання маскувальних матриць. Для шифрування в запропонованому методі відліки масок адитивно накладаються на вхідний сигнал, відповідно до маскувальної матриці. Відповідно до цього для того щоб здійснити дешифрування отриманих реальної та уявної компонент від них віднімаються обчислені значення, які зберігаються в пам'яті мікроконтролера.

9. Проведено оцінку ефективності запропонованого методу згідно зі стандартними характеристиками. Розроблений метод показує значне підвищення рівня захищеності завдяки динамічній зміні кодів маски, що використовується для шифрування. Також слід зазначити, що запропонований метод дозволяє знизити обчислювальну складність дискретного перетворення Фур'є, завдяки використанню віддалених обчислювальних потужностей.

10. Теоретично та експериментально доведено конструктивність запропонованого методу. Ретельний аналіз показав, що після застосування методу до вхідних даних, реальні та уявні компоненти спектрального представлення дійсно відповідають дискретному перетворенню Фур'є. Це підтверджено шляхом виведення математичних формул, які точно описують процес шифрування та дешифрування і демонструють його коректність. Крім того, у розділі наведено приклад використання розробленого методу, який демонструє правильність отриманих результатів для вхідних даних. Цей приклад включає детальний опис процесу обробки вхідних даних. Таким чином, проведене дослідження обґрунтувало коректність запропонованого методу з теоретичної та практичної точки зору.

11. Виконано функціональне моделювання запропонованого методу захищеної реалізації дискретного перетворення Фур'є на віддалених комп'ютерних системах. Результати моделювання показали високу ефективність методу в умовах реального часу. Моделювання підтвердило здатність методу забезпечувати належний рівень безпеки і ефективності при обробці великих обсягів даних, що є важливим для розширення можливостей систем Інтернету речей і підвищення їх надійності. Таким чином, рівень захищеності зростає в експоненційно, що забезпечує значно вищу безпеку даних у порівнянні з відомими методами, а швидкість обробки даних збільшується на один порядок.

12. Було розроблено комплексну структуру програмного забезпечення, яка забезпечує зручність у використанні, підтримці та розширюваності. Структура враховує розподіл завдань між різними компонентами програми, що дозволяє

легко інтегрувати нові функції та методи без значних змін в існуючому коді. Такий підхід сприяє зменшенню кількості помилок і полегшує налагодження.

13. Створено функціональні компоненти програмного забезпечення, які відповідають за шифрування, обробку даних та їх дешифрування. Архітектура побудована на модульному принципі, що забезпечує високу гнучкість та можливість масштабування. Модульний підхід дозволяє проводити незалежне тестування кожного компонента, що покращує загальну надійність та стабільність програми.

14. Описано детальну взаємодію між основними файлами програми. ConsoleApplication3.cpp виконує роль основного файлу, який координує роботу програми, викликаючи функції з InOut.cpp та Source.cpp. Загальна організація взаємодії забезпечує ефективний обмін даними між компонентами, мінімізуючи залежності та підвищуючи зручність у підтримці коду.

15. Детально описано структуру організації даних в програмному забезпеченні. Пояснена доцільність використання кожного використаного типу даних. використання векторів для зберігання вхідних та вихідних даних, а також проміжних результатів обчислень.

16. Розроблено детальну інструкцію для користувача, що описує процес запуску розробленої програми ConsoleApplication3.exe та подальші кроки взаємодії з програмою. Користувачеві необхідно слідувати інструкціям, що виводяться в консолі, та вводити значення сигналів або масок у зазначеному форматі.

17. Результати роботи розробленого програмного забезпечення довели конструктивність запропонованого методу, оскільки обчислення стандартного перетворення Фур’є співпадають з даними отриманими при застосуванні розробленої програми. Крім того експериментально підтверджено зростання рівня захищеності даних більше ніж в 20 разів та збільшення швидкості обробки вхідного сигналу на 2 порядки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Khalil H. Protected Discrete Fourier Transform Implementation On Remote Computer Systems / Hana Khalil, Olexander Markovskyi, Alireza Miratаei // Information, Computing And Intelligent Systems. – 2020. – No. 1. – P. 26–33.

2. Халіл, Х. А. Е.-Д., Селіванов, В., Саницький, А., Сергійчук, Н. Ротаційно-компенсаційний метод побудови магічних квадратів для криптографічних застосувань // «Наука і техніка сьогодні. – 2023. – № 13. – С. 829–842.

3. Халіл, Х. А. Е.-Д., Марковський, О. П., Аліреза, М., Павло, С. Метод гомоморфного шифрування даних при виконанні над ними перетворень фур'є в хмарах // The International Conference on Security, Fault Tolerance, Intelligence" (ICSFTI2023), Київ, 29–30 черв. 2023 р. – Київ, 2024. – С. 102–111.

4. Oleksandr, M., Doukas, N., Alireza, M., Bardis, N. Method of protecting data processed by the discrete Fourier transform in remote computer systems // 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 9–11 December 2022. – [S. l.], 2022. – P. 1–5.

5. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M. A view of cloud computing // Communications of the ACM. – 2010. – Vol. 53, no. 4. – P. 50–58.

6. Bianchi T. On the Implementation of the Discrete Fourier Transform in the Encrypted Domain / T. Bianchi, A. Piva, M. Barni // IEEE Transactions on Information Forensics and Security. – 2009. – Vol. 4, no. 1. – P. 86–97.

7. Nakonechny A. Signal processing using modern cloud technologies / A. Nakonechny, P. Pazan // Visnik of the National University "Lviv Polytechnic" series Automation, measurement and control. – 2015. – Vol. 821. – P. 8–16.

8. Instruments T. TMS320F2812 Digital Signal Processor. Implementation Tutorial / Texas Instruments. – [S. l.] : Texas Instruments, 2013. – 122 p.

9. Markovskyi O. Secure Modular Exponentiation in Cloud Systems / O. Markovskyi, N. Bardis, S. Kirilenko // Proceeding of the Congress on Information

Technology. Computationnal and Experimental Physics (CITCEP 2015), 18–20 December 2015. – Krakow. – P. 266–269.

10. Boroujerdi N. Cloud Computing: Changing Cogitation about Computing / N. Boroujerdi, S. Nazem // IJCSI International Journal of Computer Science Issues. – 2012. – Vol. 9, no. 4. – P. 169–180.

11. Sandeep G. Radix 4 Fast Fourier Transform Using New Distributive Arithmetic / Guduguntla Sandeep, S.P.V.Subba Rao // International Journal of Recent Technology and Engineering. – 2019. – Vol. 8. – P. 11–15.

12. Xia, Z., Zhu, Y., Sun, X., Qin, Z., Ren, K. Towards Privacy-Preserving Content-Based Image Retrieval in Cloud Computing // IEEE Transactions on Cloud Computing. – 2018. – Vol. 6, no. 1. – P. 276–286.

13. Hassen H. Distributed Fast Fourier Transform (DFFT) on MapReduce Model for Arabic Handwriting Feature Extraction Technique via Cloud Computing Technologies / Hamdi Hassen, Maher Khemakhem // IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Tunisia, 1 January 2014. – [S. l.], 2014. – P. 33–39.

14. Liu, L., Wang, L., Shi, Y.-Q., Chang, C.-C. Separable Data-Hiding Scheme for Encrypted Image to Protect Privacy of User in Cloud // Symmetry. – 2019. – Vol. 11, no. 1. – P. 82.

15. Markovskyi, O., Gymenuk, I., Mirataei, A., Turoshanko, J., Voloshuk, M. The method of accelerated secure image filtering on remote computer systems // Telecommunication and information technology. – 2019. – Vol. 65, no. 4. – P. 99–110.

16. Bujbarova M. Method for protected Fourier transforms on remote distributed computer systems / M. Bujbarova, Y. Vynogradov, V. Priymak // Visnik of National Technical University of Ukraine "KPI" Informatics, Control and Computer Engineering. – 2016. – Vol. 65. – P. 64–71.

17. Russ J. C. Image Processing Handbook / John C. Russ, F. Brent Neal. – [S. l.] : Taylor & Francis Group, 2018. – 1035 p.

18. Sathish V. Cloud-based Image Processing With Data Priority Distribution Mechanism / V. Sathish, T. A. Sangeetha // Journal of Computer Applications. – 2013. – Vol. 6, no. 1. – P. 6–8.

19. Boroujerdi N. Cloud Computing: Changing Cogitation about Computing/ / N. Boroujerdi, S. Nazem // IJCSI International Journal of Computer Science Issues. – 2012. – Vol. 9, no. 3. – P. 169–180.

20. Markovskyi, O. P., Humeniuk, I. O., Mirataei, A., Toroshanko, Y. I. Method For Speed Up Protected Image Filtration In Clouds // Telecommunication and information technologies. – 2019. – No. 4. – P. 99–110.

21. Xia, Z., Zhu, Y., Sun, X., Qin, Z., Ren, K. Towards Privacy-Preserving Content-Based Image Retrieval in Cloud Computing // IEEE Transactions on Cloud Computing. – 2018. – Vol. 6, no. 1. – P. 276–286.

22. Markovskyi, O., Gymenuk, I., Mirataei, A., Turoshanko, J., Voloshuk, M. The method of accelerated secure image filtering on remote computer systems // Telecommunication and information technology. – 2019. – Vol. 65, no. 4. – P. 99–110.

23. Pratt W. K. Digital image processing: PIKS inside / William K. Pratt. – 3rd ed. – New York : Wiley, 2001. – 735 p.

24. Марковський О. П. Захищена реалізація фільтрації зображень в GRID- системах / О. П. Марковський, М. О. Невдащенко, А. М. Білашевська // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка. – 2014. – № 61. – С. 105–109.

25. Bardis N. Secure Implementation of Modular Exponentiation on Cloud Computing Resources / N. Bardis, O. Markovskyi // Computational Science and Systems Engineering : Proceeding of International Conference Applied Mathematics, Athens, 6 October 2017. – [S. l.]. – P. 90–96.

26. Forsyth D. A. Computer Vision: A Modern Approach / David A. Forsyth, Jean Ponce. – [S. l.] : Pearson Education, Limited, 2015. – 793 p.

27. Shapiro L. Computer Vision and Image Processing / Linda Shapiro. – [S. l.] : Elsevier Science & Technology Books, 1992. – 638 p.

28. Буйбарова М. Метод захищеної реалізації перетворень Фур'є на віддалених розподілених комп'ютерних системах / М. Буйбарова, Ю. Виноградов, В. Приймак // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка. – 2019. – № 64. – С. 64–71.

29. Анісімов А. В. Алгоритмічна теорія великих чисел : модулярна арифметика велик. чисел / А. В. Анісімов. – Київ : Академперіодика, 2001. – 153 с.

30. Singmaster D. A Concise Introduction to the Theory of Numbers / David Singmaster, A. Baker // The Mathematical Gazette. – 1985. – Vol. 69, no. 450. – P. 318.

31. Бриль В. Сучасні методи та засоби криптографічного перетворення інформації з метою її захисту / В. Бриль. – Київ : НТУУ "КПІ", 1999. – 87 с.

32. Boroujerdi N. Cloud Computing: Changing Cogitation about Computing / N. Boroujerdi, S. Nazem // IJCSI International Journal of Computer Science Issues. – 2012. – Vol. 9, no. 3. – P. 169–180.

33. Гамаюн В. П. Method compressing data binary on base coding nonzero bit / В. П. Гамаюн // Problems of Informatization and Management. – 2014. – Vol. 3, no. 47. – P. 13–17.

34. Xiang C. Verifiable and Secure Outsourcing Schemes of Modular Exponentiations Using One Untrusted Cloud Server and Their Application / Can Xiang // IACR Cryptology ePrint Archive. – 2014. – P. 500–510.

35. Ahmed M. Cloud Computing and Security Issues in the Cloud / Monjur Ahmed, Mohammad Ashraf Hossain // International Journal of Network Security & Its Applications. – 2014. – Vol. 6, no. 1. – P. 25–36.

36. Mclvor C. Fast Montgomery modular multiplication and RSA cryptographic processor architectures / C. Mclvor, M. McLoone, J. V. McCanny //

Conference Record of the 37th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA. – [S. l.]. – P. 379–384.

37. Wang, P., Wang, J., Chen, Y., Ni, G. Rapid processing of remote sensing images based on cloud computing // Future Generation Computer Systems. – 2013. – Vol. 29, no. 8. – P. 1963–1968.

38. Bardis N. Secure, Green Implementation of Modular Arithmetic Operations for IoT and Cloud Applications / Nikolaos Bardis // Green IT Engineering: Components, Networks and Systems Implementation. – Cham, 2017. – P. 43–64.

39. Menezes A. J. Overview of Cryptography / Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone // Handbook of Applied Cryptography. – [S. l.], 2018. – P. 1–48.

40. Elliott C. Quantum cryptography / C. Elliott // IEEE Security and Privacy Magazine. – 2004. – Vol. 2, no. 4. – P. 57–61.

41. Welschenbach M. Cryptography in C and C++ / Michael Welschenbach. – 2nd ed. – Berkeley, CA : Apress, 2001. – 432 p.

42. Гамаюн В. П. Algorithm of the fast realization the sequential calculation / В. П. Гамаюн // Problems of Informatization and Management. – 2007. – Vol. 1, no. 19. – P. 32–36.

43. ДСТУ 3396.1-96. Захист інформації. Технічний захист інформації. Основні положення. – Чинний від 1997-01-01. – Вид. офіц. – Київ : Держстандарт України, 1997. – 8 с.

44. ДСТУ 3396.2-96. Захист інформації. Технічний захист інформації. Терміни та визначення. – Чинний від 1997-01-01. – Вид. офіц. – Київ : Держстандарт України, 1997. – 11 с.

45. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу : НД ТЗІ від 28.04.1999 р. № 22 : станом на 28 груд. 2012 р.

46. Reyhani-Masoleh A. Fast normal basis multiplication using general purpose processors / A. Reyhani-Masoleh, M. A. Hasan // IEEE Transactions on Computers. – 2003. – Vol. 52, no. 11. – P. 1379–1390.

47. Yen S.-M. The fast cascade exponentiation algorithm and its applications on cryptography / Sung-Ming Yen, Chi-Sung Laih // Advances in Cryptology – AUSCRYPT '92. – Berlin, Heidelberg, 1993. – P. 447–456.

48. Yen S. A note on multi-exponention / S. Yen, C. Laih, A. Lenstra // IEEE Proceeding, Computer and Digital Techniques. – 2004. – Vol. 141, no. 5. – P. 122–131.

49. Задірака В. Комп'ютерна криптологія : Підручник / В. Задірака, О. Олексюк. – Київ : Вища шк., 2002. – 511 с.

50. Taylor R. R. A High-Performance Flexible Architecture for Cryptography / R. Reed Taylor, Seth Copen Goldstein // Lecture Notes in Computer Science. – 1717th ed. – Berlin, Heidelberg, 2000. – Vol. 1717 : Proceedings of 1-th International Workshop "Cryptographic Hard¬ware and Embedded Systems". – P. 231–245.

51. Tenca A. F. High-Radix Design of a Scalable Modular Multiplier / Alexandre F. Tenca, Georgi Todorov, Çetin K. Koç // Lecture Notes in Computer Science. – Berlin, Heidelberg, 2001. – Vol. 2162 : Cryptographic Hardware and Embedded Systems – CHES 2001. – P. 185–196.

52. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W. New Algorithms for Secure Outsourcing of Modular Exponentiations // Lecture Notes in Computer Science. – Berlin, Heidelberg, 2012. – Vol. 7459 : Computer Security – ESORICS 2012. – P. 541–556.

53. Марковський, О. П., Русанова, О. В., Олієвський, А. А., Черевик, В. М. Метод прискореного модулярного експоненціювання з використанням передобчислень // Телекомунікаційні та інформаційні технології. – 2018. – Т. 58, № 1. – С. 73–81.

54. Koblitz N. Introduction to Elliptic Curves and Modular Forms / Neal Koblitz. – 2nd ed. – New York, NY : Springer New York, 1993. – 252 p.

55. Coutinho S. C. The Mathematics of Ciphers: Number Theory and RSA Cryptography / S. C. Coutinho. – [S. l.] : AK Peters, Ltd., 1998. – 196 p.

56. Стіренко С. Г. Забезпечення безперервного відтворення потокового відео в однорангових мережах з використанням erasures кодів. / С. Г. Стіренко, А. Габінет, Ю. В. Костенко // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка: збірник наукових праць. – 2015. – № 65. – С. 105–110.

57. Stallings W. Cryptography and network security: Principles and practice / William Stallings. – 5th ed. – Boston : Prentice Hall, 2011. – 719 p.

58. Brickell, E. F., Gordon, D. M., McCurley, K. S., Wilson, D. B. Fast Exponentiation with Precomputation // Lecture Notes in Computer Science. – Berlin, Heidelberg, 2012. – Vol. 2059 : Advances in Cryptology – EUROCRYPT' 92. – P. 200–207.

59. Kawamura S.-i. A Fast Modular Exponentiation Algorithm / Shin-ichi Kawamura, Kyoko Takabayashi, Atsushi Shimbo // Ieice Transactions On Fundamentals Of Electronics, Communications And Computer Sciences. – 1991. – Vol. 74, no. 8. – P. 2136–2142.

60. Schreier H. Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts,Theory and Applications / Hubert Schreier, Michael A. Sutton, Jean-Jose Orteu. – New York : Springer London, Limited, 2009. – 363 p.

61. Malgouyres F. Noise selection approach to image restoration / Francois Malgouyres // Applications in signal and image processing IX. – 2001. – Vol. 4478. – P. 34–41.

62. Задірака В. К. Комп'ютерна арифметика багато розрядних чисел : Підручник / В. К. Задірака, О. С. Олексюк. – Київ : Вища шк., 2002. – 264 с.

63. Tanenbaum A. S. Computer Networks, Global Edition / Andrew S. Tanenbaum, David Wetherall. – 6th ed. – [S. l.] : Pearson Education, Limited, 2021. – 943 p.

64. Технічний захист інформації на програмно-керованих АТС загального користування : НД ТЗІ від 28.05.1999 р. № 1 : станом на 1 лип. 1999 р.

65. Knuth D. Art of Computer Programming / Donald Knuth. – [S. l.] : Addison-Wesley Longman, Incorporated, 2014. – Vol. 2 : The Seminumerical Algorithms. – 784 p.

66. McConnell J. J. Analysis of algorithms: An active learning approach / Jeffrey J. McConnell. – 2nd ed. – Sudbury, Mass : Jones and Bartlett Publishers, 2008. – 451 p.

67. Haches G. Montgomery multiplication with no final subtraction / G. Haches, J. J. Quisquater // Lecture Notes in Computer Science. – Berlin, Heidelberg, 2013. – Vol. 2465 : Cryptographic Hardware and Embedded Systems – CHES 2013. – P. 293–301.

68. Марковський, О. П., Іванов, Д. Г., Великий, М. М., Невдащенко, М. В. Метод резервування та прискореного відновлення даних в системах їх віддаленого зберігання // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. – 2015. – № 63. – С. 46–54.

69. Koc C. K. Montgomery reduction with even modulus / C. K. Koc // IEE Proceedings - Computers and Digital Techniques. – 2004. – Vol. 141, no. 5. – P. 314–316.

70. Jutla C. S. On finding small solutions of modular multivariate polynomial equations / Charanjit S. Jutla // Lecture Notes in Computer Science. – Berlin, Heidelberg, 1998. – Vol. 658 : Advances in cryptography. Proceeding of EUROCRYPT’98. – P. 221–235.

71. Bos J. Addition Chain Heuristics / Jurjen Bos, Matthijs Coster // Lecture Notes in Computer Science. – New York, NY, 2014. – Vol. 2116 : Advances in Cryptology – CRYPTO’ 89 Proceedings. – P. 400–407.

72. Savage J. E. The complexity of computing / John E. Savage. – [S. l.] : John Wiley & Sons, 1976. – 391 p.

73. Hamacher V. C. Computer organization / V. Carl Hamacher, Zvonko Vranesic, Safwat Zaky. – 5th ed. – Boston : McGraw-Hill, 2002. – 805 p.

74. Brey B. B. The Intel Microprocessors / Barry B. Brey. – 8th ed. – Upper Saddle River, NJ : Pearson Prentice Hall, 2008. – 944 p.

75. Hars L. Long Modular Multiplication for Cryptographic Applications / Laszlo Hars // Lecture Notes in Computer Science. – Berlin, Heidelberg, 2004. – Vol. 3156 : Cryptographic Hardware and Embedded System- CHES'2004. – P. 45–61.

ДОДАТОК 1

Текст програмного коду

до магістерської дисертації

на тему: «Метод захищеної реалізації дискретного перетворення Фур'є в хмарах для IOT»

Київ – 2024

ConsoleApplication3.cpp

```
#include <iostream>
#include <iomanip>
#include "Source.h"
#include "InOut.h"
#include <stdio.h>
#include <string.h>
#include <time.h>

int main()
{

    vector<double> input;

    vector<double> output_a;
    vector<double> output_b;

    vector<double> output_Q;

    vector<vector<double>> output_q;
    vector<vector<double>> output_w;

    vector<vector<double>> masks;
    vector<vector<double>> mask_mtrx;

    vector<vector<double>> prev_signals;
    vector<int> prev_signals_nums;

    vector<vector<vector<double>>> mask_real;
    vector<vector<vector<double>>> mask_imaginary;

    vector<vector<double>> mask_prev_real;
    vector<vector<double>> mask_prev_imaginary;

    vector<vector<double>> signal_prev_r;
    vector<vector<double>> signal_prev_im;

    int mask_number = 3;
    int prev_signals_num = 3;
    int signals_num = 0;
    int signal_num = 0;
    int mask_num;
    int flag;
    int prog_mode = 1;
    int length;
    int method;

    srand(time(0));

    cout << "Insert signal length:\nThe length of the example signal is eight" << endl;
    cin >> length;


    //METHOD SELECTION

    cout << "Select the method by which the signal is encrypted:\nFirst - 1\nSecond -
2\n" << endl;
    cin >> method;
```

```
    while (prog_mode == 1)
    {

        //  INPUT SIGNAL
        Input(input, length);
        Output("Signal:", input);

        DFT(input, output_a, output_b);

        Output("Real:", output_a);
        Output("Imaginary:", output_b);

        //  GENERATION OF MASKS

        Masks(mask_number, input.size(), masks);
        Output_Mas("Masks:", masks);

        if (method == 1)
        {
            //  DISCRETE FOURIER TRANSFORM FOR MASKS

            for (int i = 0; i < mask_number; i++)
            {
                DFT(masks[i], output_a, output_b);

                mask_prev_real.push_back(output_a);
                mask_prev_imaginary.push_back(output_b);

//                Output_Mas("Masks real:", mask_prev_real);
//                Output_Mas("Masks imaginary:", mask_prev_imaginary);
            }


            //  ENCRYPTION BLOCK

            mask_num = rand() % mask_number;

            cout << "Mask number\n" << mask_num << endl;

            output_Q = input;

            if (signals_num < prev_signals_num)
            {
                flag = 0;

                prev_signals.push_back(input);
                prev_signals_nums.push_back(signals_num);

                Encryption_1(masks[mask_num], output_Q, prev_signals, signal_num);
            }
            else
            {
                flag = rand() % 2;
                signal_num = rand() % prev_signals_num;

                if (flag == 1)
                    cout << "Using signal #\n" << signal_num<< endl;
                if (flag == 0)
                    cout << "Using mask #\n" << mask_num << endl;
```

```
                Encryption_1(masks[mask_num], output_Q, prev_signals, signal_num,
flag);
            }

            Output("Encrypted signal:", output_Q);

            //  DISCRETE FOURIER TRANSFORM FOR ENCRYPTED SIGNAL

            DFT(output_Q, output_a, output_b);

            Output("Real before:", output_a);
            Output("Imaginary before:", output_b);

            //  DECRYPTION BLOCK

            vector<double> vector_r(length, 0.0);
            vector<double> vector_y(length, 0.0);

            signal_prev_r.push_back(vector_r);
            signal_prev_im.push_back(vector_y);

            Decryption_1(output_a, output_b, mask_prev_real[mask_num],
mask_prev_imaginary[mask_num], signal_prev_r, signal_prev_im, signal_num, flag);

            Output("Real:", output_a);
            Output("Imaginary:", output_b);


            //  WRITING BLOCK

            if (flag)
            {
                prev_signals.erase(prev_signals.begin() + signal_num);
                prev_signals.push_back(input);

                signal_prev_r.erase(signal_prev_r.begin() + signal_num);
                signal_prev_r.push_back(output_a);

                signal_prev_im.erase(signal_prev_im.begin() + signal_num);
                signal_prev_im.push_back(output_b);

                prev_signals_nums.erase(prev_signals_nums.begin() + signal_num);
                prev_signals_nums.push_back(signals_num);
            }

            signals_num++;
        }

        else
        {
            //  PARTLY DISCRETE FOURIER TRANSFORM FOR MASKS

            for (int i = 0; i < mask_number; i++)
            {
                DFT_partly(masks[i], output_q, output_w);
                mask_real.push_back(output_q);
                mask_imaginary.push_back(output_w);
            }
```

```
            //  MASKING MATRIX

            Create_Mask_Matrix(mask_number, input.size(), mask_mtrx);
            Output_Mas("Mask Matrix:", mask_mtrx);

            //  ENCRYPTION BLOCK

            output_Q = input;
            for (int i = 0; i < mask_number; i++)
            {
                Encryption_2(masks[i], output_Q, mask_mtrx[i]);
            }

            Output("Encrypted signal:", output_Q);

            //  DISCRETE FOURIER TRANSFORM FOR ENCRYPTED SIGNAL

            DFT(output_Q, output_a, output_b);

            Output("Real before:", output_a);
            Output("Imaginary before:", output_b);

            //  DECRYPTION BLOCK

            for (int i = 0; i < mask_number; i++)
            {
                Decryption_2(output_a, output_b, mask_mtrx[i], mask_real[i],
mask_imaginary[i]);
            }

            Output("Real:", output_a);
            Output("Imaginary:", output_b);

        }

        cout << "Do you want to enter another signal?\nYes - 1\nNo - 0\n";
        cin >> prog_mode;
    }

}
```

InOut.cpp

```
#include <iostream>
#include <iomanip>
#include <vector>
#include "InOut.h"

void Input(vector<double>& in, int length, int mode)
{
    double var;
    in.clear();

    if (mode == 0)
    {
        for (int i = 0; i < length; i++)
        {
            in.push_back(rand() / 10000.0);
```

```
        }
    }
    else
    {
        cout << "Insert signal:" << endl;

        for (int i = 0; i < length; i++)
        {
            cin >> var;
            in.push_back(var);
        }
    }

}

void Output(string name, vector<double>& in)
{
    int N = in.size();

    cout << name << endl;

    for (unsigned int k = 0; k < N; k++)
    {
        cout << fixed;
        cout.precision(2);

        cout << setw(7) << in[k] << "\t";
     }
    cout << endl << endl;
}

void Output_Mas(string name, vector<vector<double>>& in)
{
    int N = in.size();

    cout << name << endl;

    for (int i = 0; i < N; i++)
    {
        for (auto l = in[i].begin(); l != in[i].end(); l++)
        {
            cout << fixed;
            cout.precision(2);

            cout << setw(7) << *l << "\t";
        }
        cout << endl;
    }
    cout << endl;
}
```

InOut.h

```
#pragma once
#include <vector>
using namespace std;

#ifndef INOUT_H
#define INOUT_H
```

```
void Input(vector<double>& in, int length, int mode = 0);

void Output(string a, vector<double>& in);

void Output_Int(string name, vector<int>& in);

void Output_Mas(string name, vector<vector<double>>& in);

#endif
```

Source.cpp

```
#include <vector>
#include <iostream>
#include <complex>
#define _USE_MATH_DEFINES
#include <math.h>
#include <time.h>
#include "Source.h"

using namespace std;

void DFT(vector<double>& in, vector<double>& out_r, vector<double>& out_y)
{

    out_r.clear();
    out_y.clear();

    int N = in.size();

    for (int i = 0; i < N; i++)
    {
        double r = 0;
        double y = 0;

        for (int l = 0; l < N; l++)
        {
            r += cos((2 * M_PI * i * l) / N) *(in.at(l));
            y += -sin((2 * M_PI * i * l) / N) * (in.at(l));
        }

        out_r.push_back(r);
        out_y.push_back(y);
    }
    return;
}

void DFT_partly(vector<double> & in, vector<vector<double>>& out_r,
vector<vector<double>> & out_y)
{
    int N = in.size();

    out_r.clear();
    out_y.clear();

    for (int i = 0; i < N; i++)
    {
        double r = 0;
```

```
        double y = 0;

        vector<double> vector_r;
        vector<double> vector_y;

        vector_r.clear();
        vector_y.clear();

        for (int l = 0; l < N; l++)
        {
            r = cos((2 * M_PI * i * l) / N) * (in.at(l));
            vector_r.push_back(r);

            y = -sin((2 * M_PI * i * l) / N) * (in.at(l));
            vector_y.push_back(y);
        }

        out_r.push_back(vector_r);
        out_y.push_back(vector_y);
    }
    return;
}

void Masks(int mask_amount, int samples_amount, vector<vector<double>>& mask_mtrx, int
mode)
{
    vector<double> vector_row;

    srand(time(0));

    mask_mtrx.clear();

    if (mode == 0)
    {
        mask_mtrx = { {2.36, 0.34, 1.28, 0.22, 1.27, 3.05, 1.91, 3.09},
                      {3.21, 2.67, 0.07, 0.32, 2.63, 1.08, 0.11, 3.06},
                      {0.22, 0.33, 1.37, 1.65, 2.31, 1.72, 3.04, 0.83} };
    }
    else
    {
        for (int i = 0; i < mask_amount; i++)
        {
            vector_row.clear();

            for (int j = 0; j < samples_amount; j++)
            {
                vector_row.push_back(rand()/10000.0);
            }
            mask_mtrx.push_back(vector_row);
        }
    }
}

void Create_Mask_Matrix(int mask_amount, int samples_amount, vector<vector<double>>&
mask_mtrx, int mode)
{
    vector<double> vector_row;

    srand(time(0));
```

```
    mask_mtrx.clear();

    if (mode == 0)
    {
        mask_mtrx = { {1.00, 0.00, 1.00, 0.00, 1.00, 1.00, 0.00, 0.00},
                      {0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 1.00, 0.00},
                      {1.00, 1.00, 0.00, 1.00, 1.00, 1.00, 0.00, 1.00} };
    }
    else
    {
        for (int i = 0; i < mask_amount; i++)
        {
            vector_row.clear();

            for (int j = 0; j < samples_amount; j++)
            {
                vector_row.push_back(rand()%2);
            }
            mask_mtrx.push_back(vector_row);
        }
    }
}

void Encryption_1(vector<double>& mask, vector<double>& signal_q,
vector<vector<double>>& other_signals, int signal_num, int flag)
{
    for (int j = 0; j < signal_q.size(); j++)
    {
        signal_q[j] += (1 - flag) * mask[j] + flag * other_signals[signal_num][j];
    }
}

void Decryption_1(vector<double>& signal_r, vector<double>& signal_im, vector<double>&
mask_r, vector<double>& mask_im, vector<vector<double>>& signal_prev_r,
vector<vector<double>>& signal_prev_im, int signal_num, int flag)
{
    vector<double> vector_r;
    vector<double> vector_y;

    for (int j = 0; j < signal_r.size(); j++)
    {

        signal_r[j] -= (1 - flag) * mask_r[j] + flag * signal_prev_r[signal_num][j];
        signal_im[j] -= (1 - flag) * mask_im[j] + flag * signal_prev_im[signal_num][j];
    }
}

void Encryption_2(vector<double>& mask, vector<double>& signal_q, vector<double>&
mask_mtrx)
{
    for (int j = 0; j < signal_q.size(); j++)
    {
        signal_q[j] += mask_mtrx[j] * mask[j];
    }
}

void Decryption_2(vector<double>& real, vector<double>& imagine, vector<double>&
mask_mtrx, vector<vector<double>>& mask_real, vector<vector<double>>& mask_imagine)
{
    for (int j = 0; j < mask_mtrx.size(); j++)
```

```
    {
        for (int k = 0; k < mask_mtrx.size(); k++)
        {
            real[j] -= mask_real[j][k] * mask_mtrx[k];
            imagine[j] -= mask_imagine[j][k] * mask_mtrx[k];
        }
    }
}
```

Source.h

```
#pragma once
#include <vector>
using namespace std;

#ifndef SOURCE_H
#define SOURCE_H

void DFT(vector<double>& in, vector<double>& out_r, vector<double>& out_b);

void DFT_partly(vector<double>& in, vector<vector<double>>& out_r,
vector<vector<double>>& out_y);

void Masks(int mask_amount, int samples_amount, vector<vector<double>>& mask_mtrx, int
mode = 0);

void Create_Mask_Matrix(int mask_amount, int samples_amount, vector<vector<double>>&
mask_mtrx, int mode = 0);

void Encryption_1(vector<double>& mask, vector<double>& signal_q,
vector<vector<double>>& other_signals, int signal_num, int flag = 0);

void Decryption_1(vector<double>& signal_r, vector<double>& signal_im, vector<double>&
mask_r, vector<double>& mask_im, vector<vector<double>>& signal_prev_r,
vector<vector<double>>& signal_prev_im, int signal_num, int flag);
//void Decryption_1(vector<double>& signal_r, vector<double>& signal_im,
vector<double>& mask_r, vector<double>& mask_im, vector<double>& signal_prev_r,
vector<double>& signal_prev_im, int signal_num, int flag);

void Encryption_2(vector<double>& mask, vector<double>& signal_q, vector<double>&
mask_mtrx);

void Decryption_2(vector<double>& real, vector<double>& imagine, vector<double>&
mask_mtrx, vector<vector<double>>& mask_real, vector<vector<double>>& mask_imagine);

       #endif
```