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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«МЕТОД ТА ЗАСІБ ГЕШУВАННЯ ДАНИХ»

Виконав: студент 2-го курсу,групи БС-21м
спеціальності 125Кібербезпека
Нога І.В.
Керівник: д.т.н., проф., зав. кафедри ЗІ
Лужецький В.А.
«22» 12 2022 р.

Опонент: к.т.н., доцент кафедри ПЗ
Катєльніков Д.І.
«22» 12 2022 р.

Допущено до захисту

Завідувач кафедри ЗІ, д.т.н, проф.

Лужецький В.А.

22. 12 2022р.

Вінниця ВНТУ – 2022 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма – «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д. т. н., проф.
_____ В. А. Лужецький
15 09 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Нозі Іллі Вікторовичу

1. Тема роботи: «Метод та засіб гешування даних»

керівник роботи: Лужецький Володимир Андрійович, зав. кафедри ЗІ, д.т.н., проф, затверджені наказом ВНТУ №203 від 14.09.2022.

2. Строк подання студентом роботи – 19 грудня 2022 року

3. Вихідні дані до роботи:

— Обсяг даних, що підлягають гешуванню - не більше 2^{64} біт.

— Розрядність геш-значення — 256.

— Метод гешування даних — без використання ітераційної процедури.

4. Зміст текстової частини: Вступ. 1. Аналіз підходів до побудови геш-функцій. 2. Розробка методу гешування. 3. Програмна реалізація методу гешування. 4. Економічне обґрунтування. Висновки. Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу.

Ітераційний метод хешування (плакат, А4). Підхід до побудови геш-функцій на основі алгоритмів блокового шифрування (плакат, А4). Підхід до побудови геш-функцій на основі складнообчислювальної математичної задачі (плакат, А4). Підхід до побудови геш-функцій з нуля (плакат, А4). Метод гешування (плакат, А4). Алгоритми кроків гешування (плакат, А4). Результати тестування (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдан прийн
1	Лужецький В.А., д.т.н., проф. зав. кафедри ЗІ		
2	Лужецький В.А., д.т.н., проф. зав. кафедри ЗІ		
3	Лужецький В.А., д.т.н., проф. зав. кафедри ЗІ		
4	Лесько О.Й., к.е.н., проф., зав. кафедри ЕПВМ		

7. Дата видачі завдання – 1 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Прим
1	Аналіз завдання. Вступ	01.09.2022 – 06.09.2022	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	07.09.2022 – 17.09.2022	
3	Розробка рішень реалізації	18.09.2022 – 25.09.2022	
4	Практична реалізація, моделювання, експериментування, результати	26.10.2022 – 17.11.2022	
5	Розробка розділу економічного обгрунтування доцільності розробки	18.11.2022 – 26.11.2022	
6	Аналіз виконання ТЗ, висновки	27.11.2022 – 29.11.2022	
7	Оформлення пояснювальної записки	30.11.2022 – 06.12.2022	
8	Попередній захист МКР та доопрацювання МКР	07.12.2022 – 19.12.2022	
9	Представлення МКР до захисту, рецензування	20.12.2022 – 21.12.2022	
10	Захист МКР	22.12.2022 – 26.12.2022	

Студент _____ (підпис) Ног

Керівник роботи _____ (підпис) Лужецький

АНОТАЦІЯ

УДК 004.056

Нога І.В. Метод та засіб гешування даних. Магістерська кваліфікаційна робота зі спеціальності 125 – кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2022. 76 с.

Бібліогр.: 15 назв; рис.: 27; табл.: 12.

Магістерська кваліфікаційна робота присвячена розробці методу гешування даних. В рамках цієї роботи було проведено аналіз існуючих підходів щодо побудови функцій гешування. Розроблено метод гешування даних без використання ітеративної процедури та алгоритми, що реалізують цей метод. Реалізовано програмний засіб для хешування даних. Оцінено витрати на розробку та обґрунтовано економічну доцільність використання засобу.

Ключові слова: геш-функція, дайджест повідомлення, метод гешування.

ABSTRACT

Noga I.V. The leg of I.V. Method and means of data hashing. Master's qualification work on specialty 125 - cyber security, educational program - Security of information and communication systems. Vinnytsia: VNTU, 2022. 76 p.

Bibliography: 15 titles; Fig.: 27; tab.: 12.

The master's thesis is devoted to the development of a data hashing method. As part of this work, an analysis of existing approaches to the construction of hashing functions was carried out. A method of data hashing without the use of an iterative procedure and algorithms implementing this method have been developed. A software tool for data hashing has been implemented. Development costs were estimated and the economic feasibility of using the tool was substantiated.

Keywords: hash function, message digest, hashing method.

ЗМІСТ

ВСТУП

Сьогодні надзвичайно актуальною проблемою є забезпечення інформаційної безпеки. З урахуванням постійної зміни сучасних вимог до інформаційно-телекомунікаційних систем вирішення цієї проблеми постійно потребує нових методів та засобів, що будуть забезпечувати її вирішення. Серед важливих задач інформаційної безпеки особливе місце займають забезпечення цілісності, достовірності та можливості підтвердження авторства [6].

Вище згадані задачі сьогодні ефективно вирішуються з використанням методів цифрового підписування. Невід'ємною частиною таких методів є функції гешування. Крім того, вони знаходять широке застосування і для вирішення ряду інших питань, пов'язаних із забезпеченням захисту потоків даних, наприклад для гешування паролів користувачів з метою подальшого їх шифрування і зберігання в базі даних [6].

Однією з найважливіших характеристик геш-функцій, що обумовили їх широке впровадження в практику, виявилася здатність отримувати з відкритого тексту великої довжини, геш-код – текст фіксованої довжини. Крім того, застосування геш-функцій дозволяє усунути надмірність відкритого тексту, що при подальшому криптографічному перетворенні геш-коду відкритого тексту позитивно позначається на криптографічних властивостях зашифрованого повідомлення [6].

В зв’язку з цим, сьогодні актуальним є питання дослідження та аналізу сучасних геш-функції, їх недоліків та методів побудови, а також розробка ефективних програмних засобів для прикладного застосування в різних додатках.

Об’єктом дослідження є процес гешування даних.

Предмет дослідження — метод та засіб для гешування даних.

Мета — підвищення швидкості гешування даних шляхом розробки методу та засобу гешування даних.

Для досягнення поставленої мети необхідно реалізувати низку задач, а

саме:

— проаналізувати відомі підходи до побудови геш-функцій;

— розробити метод для гешування даних, без використання ітераційної процедури;

— розробити програмний засіб, що реалізує запропонований метод гешування.

Наукова новизна полягає у запропонованому методі гешування даних, який на відміну від відомих використовує неітераційну процедуру гешування, що забезпечує пришвидшення процесу гешування.

1. АНАЛІЗ ПІДХОДІВ ДО ПОБУДОВИ ГЕШ-ФУНКЦІЙ

1.1 Вимоги, що висуваються до функції гешування

Геш-функція - це математична функція, яка перетворює числове вхідне значення в інше стисле числове значення. Вхідні дані для геш-функції мають довільну довжину, але вихідні дані завжди мають фіксовану довжину. Значення, що повертаються геш-функцією, називаються дайджестом повідомлення або просто геш-значеннями [1].

Особливості геш-функцій [1].

Вихід фіксованої довжини (геш-значення).

Геш-функція охоплює дані довільної довжини до фіксованої довжини. Цей процес часто називають гешуванням даних.

Загалом, геш набагато менше, ніж вхідні дані, тому геш-функції іноді називають функціями стиснення. Оскільки геш представляє собою менше уявлення великих даних, його також називають дайджестом.

Геш-функція з *n*-бітовим виходом називається *n*-бітної геш-функцією. Популярні геш-функції генерують значення від 160 до 512 біт.

Ефективність.

Зазвичай для будь-якої геш-функції *h* з входом x обчислення *h* (x) є швидкої операцією.

Обчислювальні геш-функції набагато швидше, ніж симетричне шифрування.

Щоб бути ефективним криптографічним інструментом, бажано, щоб геш-функція мала такі властивості [1]:

Стійкість до першого образу.

Ця властивість означає, що в обчислювальному відношенні має бути важко перевернути геш-функцію. Іншими словами, якщо геш-функція h створила геш-значення k, то буде важко знайти будь-який вхідний значення x, яке гешує до k. Ця властивість захищає від зловмисника, який має тільки геш-значення і намагається знайти вхідні дані [1].

Слабка опірність колізії.

Ця властивість означає, що задані вхідні дані і їх геш, має бути важко знайти інший вихідний файл з тим же гешем. Іншими словами, якщо геш-функція h для входу x створює геш-значення *h* (x), то повинно бути важко знайти будь-яке інше вхідне значення y таке, що *h* (y) = *h* (x). Ця властивість геш-функції захищає від зловмисника, у якого є вхідний значення і його геш, і він хоче замінити інше значення допустимим значенням замість початкового вхідного значення [1].

Сильна опірність колізії.

Це властивість означає, що має бути важко знайти два різних входа будь-якої довжини, які призводять до одного і того ж гешу. Це властивість також називається геш-функцією без зіткнень. Іншими словами, для геш-функції h важко знайти будь-які два різних входу x і y, для яких h (x) = h (y). Оскільки геш-функція є функцією стиснення з фіксованою довжиною геш-функції, неможливо, щоб геш-функція не мала колізій [2]. Це властивість відсутності зіткнень тільки підтверджує, що ці зіткнення важко знайти. Ця властивість дуже заважає зловмисникові знайти два вхідних значення з однаковим гешем. Крім того, якщо геш-функція стійка до зіткнень, вона є другою стійкою до зображень [2].

В основі гешування лежить математична функція, яка працює з двома блоками даних фіксованого розміру для створення геш-коду. Ця геш-функція є частиною алгоритму гешування.

Алгоритм гешування включає в себе раунди вищевказаної геш-функції, такі як блоковий шифр. Кожен раунд вимагає введення фіксованого розміру, зазвичай це комбінація самого останнього блоку повідомлення і результату останнього раунду. Цей процес повторюється стільки раз, скільки потрібно для хешування всього повідомлення [1].

Оскільки значення геш-функції першого блоку повідомлень стає входом для другої операції гешування, вихід якої змінює результат третьої операції, і так далі. Цей ефект відомий як лавинний ефект перемішування.

Лавинний ефект призводить до істотно розрізняються значенням геш-функції для двох повідомлень, які відрізняються навіть одним бітом даних.

Треба розуміти різницю між геш-функцією і алгоритмом. Геш-функція генерує геш-код, працюючи з двома блоками даних фіксованої довжини. Алгоритм гешування - це процес використання геш-функції, що визначає, як повідомлення буде розбито і як результати попередніх блоків повідомлень об'єднуються воєдино [1].

Розмір кожного блоку даних варіюється в залежності від алгоритму. Зазвичай розміри блоків сягають від 128 до 512 біт (рис.1.1).

Рисунок 1.1 - Структура розширеного повідомлення:

Вимоги, що висуваються до функції хешування [4]:

Значення функції гешування генерується функцією H виду h = H(**M**), де **M** - повідомлення довільної довжини (прообраз геш-функції), H(**M**) - значення функції гешування фіксованої довжини.

Функція гешування H повинна мати такі властивості:

1. Бути застосовної до блоку даних будь-якої довжини.
2. Давати на виході значення фіксованої довжини.
3. Значення H(x) повинне обчислюватися відносно легкодля будь-якого заданого x, а алгоритм обчисленняповинний бути практичним з погляду як апаратної, так іпрограмної реалізації.
4. Для будь-якого даного коду h повинне бути практично неможливо

обчислити x, для якого H(x) = *h*. Таку властивість називають односторонність або стійкість до пошуку першого прообразу (рис.1.2).

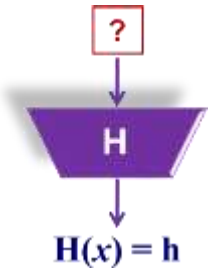


Рисунок 1.2 - Стійкість до пошуку прообразу

5. Для будь-якого даного блоку x повинне бути практичнонеможливо обчислити y, для якого H(x) = H(y). Таку властивість називають слабка опірність колізіям абостійкість до пошуку першого прообразу (колізій першого роду) (рис.1.3).

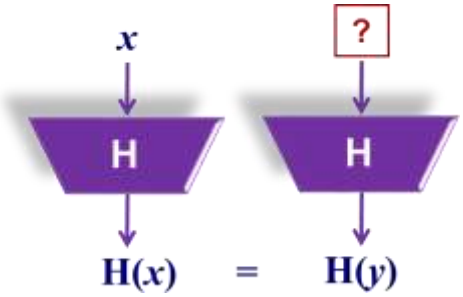


Рисунок 1.3. Слабка опірність колізіям

6. Повинно бути практично неможливо обчислити будь-яку пару різних значень x і y, для яких H(x) = H(y). Таку властивість називають сильна опірність колізіям або стійкість до колізій (колізій другого роду) (рис.1.4.).

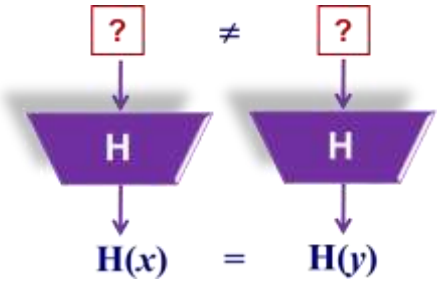


Рисунок 1.4. Сильна опірність колізіям

Ефективність геш-функції - це імовірність (P) того, що з появою

помилки вданих значення функції гешування залишиться колишнім. Якщо значення функції гешування *n*-розрядне, то $P = 2^{-n}$ для довільного виду даних і $P < 2^{-n}$ для форматованих даних.

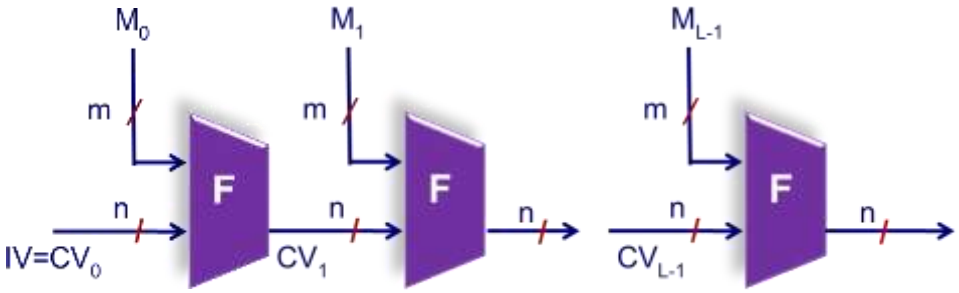


Рисунок 1.5. Ітераційна функція гешування

IV - початкове значення; CV_i – проміжне геш-значення; M_i - *i*-й блок, що вводиться;F - алгоритм ущільнення; L – кількість блоків, що вводяться;*n* – довжина геш-коду; *m* - довжина блоку, що вводиться.

1.2 Побудова геш-функцій на основі алгоритмів блокового шифрування

Існує декілька підходів до побудови геш-функцій (рис.1.6.), одним із яких і є підхід на основі алгоритмів блокового шифрування [5].

Рисунок 1.6. Підходи до побудови геш-функцій

Головний недолік абсолютно криптостійкого шифру одноразових блокнотів – велика довжина ключа, яка має бути не меншою від довжини повідомлення. Через це використовувати його на практиці складно [1]. Для побудови стійкого шифру, зручного для практичного використання, можна запропонувати такі підходи:

1. Блоковість. Вибираємо довжину ключа меншою за довжину повідомлення, розбиваємо повідомлення на окремі блоки і шифруємо кожний з них (крім, можливо, останнього) шляхом сумування з ключем по модулю два [1].

2. Режими шифрування. Для збільшення стійкості блокового шифрування можна забезпечити використання при шифруванні кожного наступного блоку результатів шифрування попереднього блоку (наприклад, в якості нового ключа шифрування для блоку). Тоді зловмисник не зможе розшифрувати блок криптотексту, доки не розшифрує всі попередні блоки [1].

3. Багатораундовість. Додатково збільшити стійкість блокового шифрування можна з рахунок багаторазового виконання шифрування кожного блоку за умови, що функція шифрування є нелінійним перетворенням [1].

В блокових алгоритмах вхідна послідовність розбивається на блоки – ділянки певної довжини (найчастіше, по 64 біти) (рис.1.7.).

Рисунок 1.7. Узагальнена схема ітерації геш-функції

Якщо довжина відкритого тексту виявляється некратною довжині блоку, застосовується операція доповнення (padding) останнього блоку до необхідної довжини, яка полягає у дописуванні необхідної кількості нулів або випадкового набору символів.

Криптографічне перетворення в блокових алгоритмах шифрування здійснюється над кожним блоком окремо. Його сутність полягає у застосуванні до блока багаторазово математичного перетворення. Внаслідок цього результуюче перетворення виявляється криптографічно більш сильним, ніж перетворення над окремо взятим блоком (рис.1.8.).

$$h_i = E_{h_{i-1}}(M_i) \oplus M_i$$

Рисунок 1.8. Схема ітерації безпечної геш-функції

Метою таких перетворень є створення залежності кожного біту блоку шифротексту від кожного біту ключа і кожного біту відкритого тексту:

Зашифрований блок може використовуватися для шифрування наступного, в результаті чого кожний блок отримає контекст, що властивий всьому повідомленню. Такі механізми шифрування використовуються для уникнення від деяких атак, заснованих на стиранні або вставки блоків, і визначаються відповідним режимом шифрування [1].

Режим шифрування – метод застосування блокового шифру, в якому для забезпечення вищого рівня криптостійкісті шифрування блоків вхідного повідомлення здійснюється з використання блоків криптотексту. Розрізняють п’ять основних режимів шифрування [1]:

1. ECB (Electronic Codebook Mode) – режим електронної кодової книги.
2. CBC (Cipher Block Chaining Mode) – режим зціплення блоків по криптотексу.
3. CFB (Cipher Feedback Mode) – режим з оберненим зв’язком по криптотексу.
4. OFB (Output-Feedback Mode) – режим з оберненим зв’язком по виходу.
5. CTR (Coutner) – режим з лічильником.

Блокові алгоритми шифрування сьогодні є основним засобом криптографічного захисту інформації.

Основні переваги блокових алгоритмів шифрування [1].

- Висока швидкість шифрування/розшифрування.

-Висока гарантована стійкість, яка до того ж може бути доведена математично.

- Можливість ефективної програмної реалізації.

1.3 Побудова геш-функцій на основі складнообчислювальної математичної задачі

Повідомлення представляється як послідовність m-розрядних блоків M_0 , M_1 ,..., M_{L-1}.

Для обчислення геш-коду **G** використовується операція піднесення до степеня за модулем. H_0 = початкове значення. $G = H_{L-1}$.

Спосіб 1.

$$H_i = H_{i-1}^{M_{i-1}} \bmod p$$

Спосіб 2.

$$H_i = M_{i-1}^{H_{i-1}} \bmod p$$

Спосіб 3.

$$H_i = g^{H_{i-1} \oplus M_{i-1}} \bmod p$$

g – примітивний елемент за модулем p.

1.4 Побудова геш-функцій з “нуля”

Проста функція хешування.

Однієї з найпростіших функцій гешування є зв'язування всіх блоків операцією порозрядного виключного "АБО" (XOR).

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im},$$

де C_i - i-й біт геш-коду,

m - число n-бітових блоків вихідних даних,

b_{ij}- i-й біт у j-му блоці,

⊕ - операція XOR.

Удосконалена проста функція хешування.

Початкова ініціалізація *n*-бітового значення функції гешування нульовим значенням.

Послідовна обробка *n*-бітових блоків даних за таким правилом (рис.1.9.).

- Виконання циклічного зсуву проміжного значення функції гешування вліво на один біт.
- Додавання поточного блоку до проміжного значення функції гешування за допомогою операції XOR.

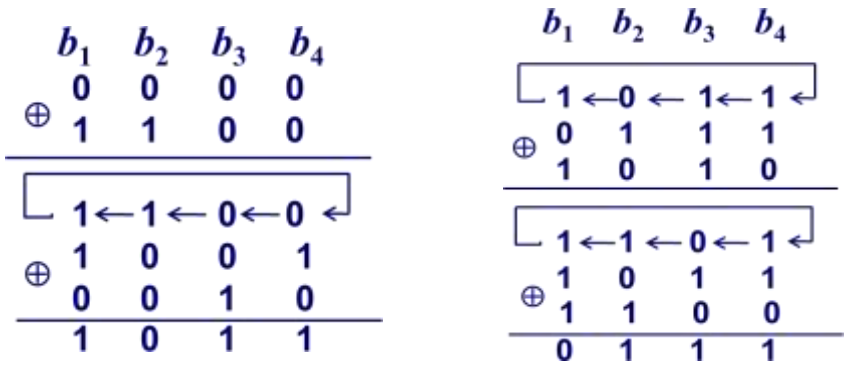


Рисунок 1.9. Удосконалена проста функція гешування

Розглянемо деякі популярні геш-функції [3]:

Дайджест повідомлення (MD):

MD5 був найпопулярнішою і широко використовуваною геш-функцією протягом декількох років (рис.1.10.).

Дайджести MD5 широко використовуються в світі програмного забезпечення для забезпечення цілісності переданого файлу [9] (рис.1.11.).

Наприклад, файлові сервери часто надають попередньо обчислену контрольну суму MD5 для файлів, щоб користувач міг порівняти з нею контрольну суму завантаженого файлу.

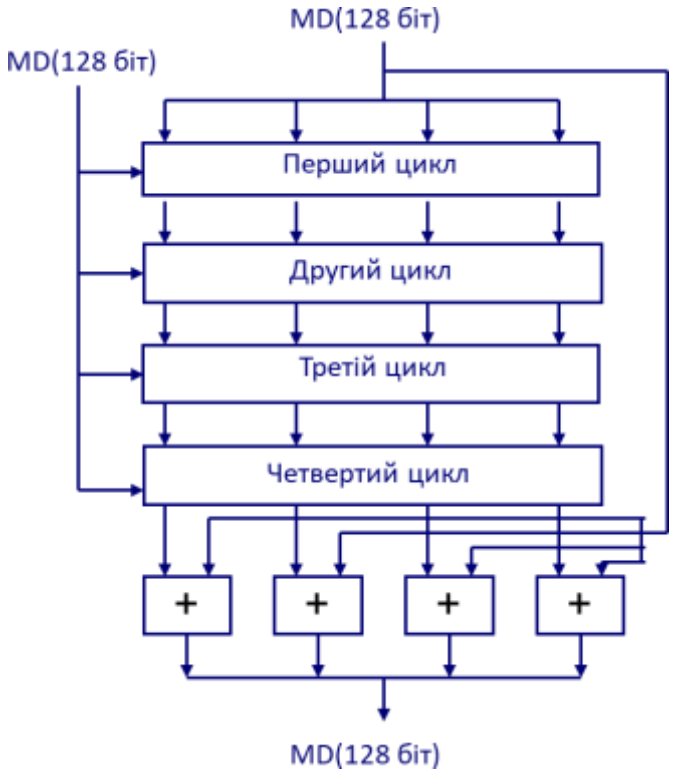


Рисунок 1.10. Обробка 512-бітного блоку в алгоритмі MD-5

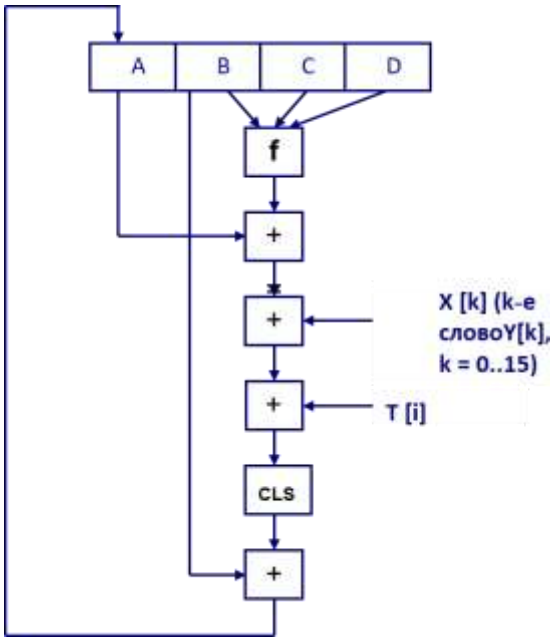


Рисунок 1.11. Логіка виконання окремого кроку MD-5

У 2004 році в MD5 були виявлені колізії. Повідомлялося, що аналітична атака була успішною тільки через годину з використанням

комп'ютерного кластера. Ця атака зіткнення привела до злому MD5 і, отже, більше не рекомендується для використання.

RIPEMD.

RIPEND - це абревіатура для дайджесту повідомлення оцінки примітивів RACE. Цей набір геш-функцій був розроблений відкритим дослідним спільнотою і широко відомий як сімейство європейських геш-функцій (рис. 1.12.).

У комплект входять RIPEND, RIPEMD-128 і RIPEMD-160. Також існують 256 і 320-бітові версії цього алгоритму [10].

Оригінальний RIPEMD (128 біт) заснований на принципах проектування, використовуваних в MD4 і забезпечує сумнівну безпеку. 128-розрядної версії RIPEMD прийшла в якості швидкої заміни для усунення вразливостей в оригінальній версії RIPEMD.

RIPEMD-160 є покращеною версією і найбільш широко використовуваної версією в сімействі. 256 і 320-бітові версії знижують ймовірність випадкового зіткнення, але не мають більш високого рівня безпеки в порівнянні з RIPEMD-128 і RIPEMD-160 відповідно (рис.1.13.).

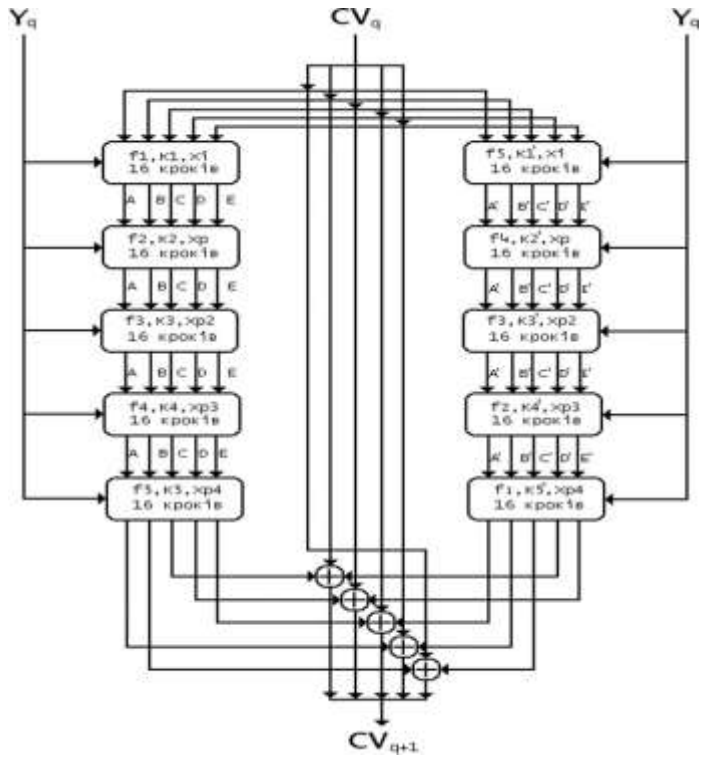


Рисунок 1.12. Обробка 512-бітного блоку алгоритму RIPEMD-160

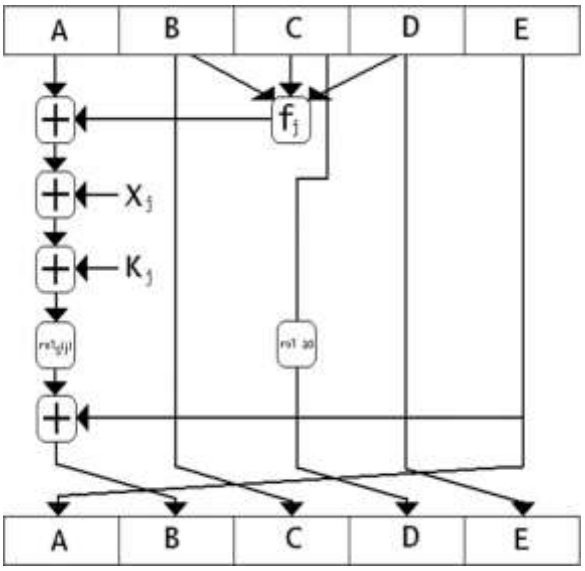


Рисунок 1.13. Логіка виконання окремого кроку RIPEMD-160

Безпечна геш-функція (SHA)

Сімейство SHA складається з чотирьох алгоритмів SHA; SHA-0, SHA-1, SHA-2 і SHA-3. Хоча з однієї сім'ї, є структурно різні [11].

Первісна версія - SHA-0, 160-бітна геш-функція, була опублікована Національним інститутом стандартів і технологій (NIST) в 1993 році . Вона мала кілька слабких місць і не стала дуже популярною. Пізніше, в 1995 році, SHA-1 був розроблений для виправлення передбачуваних недоліків SHA-0 [12]. SHA-1 є найбільш широко використовуваним з існуючих геш-функцій SHA. Він використовується в декількох широко використовуваних програмах і протоколах, включаючи протокол Secure Socket Layer (SSL) [11].

У 2005 році був знайдений метод для виявлення колізій для SHA-1 протягом практичного періоду часу, що робить довгострокову можливість використання SHA-1 сумнівною [13].

Сімейство SHA-2 (рис.1.14.) має ще чотири варіанти SHA: SHA-224, SHA-256, SHA-384 і SHA-512 в залежності від кількості біт в їх геш-значення. Про геш-функції SHA-2 поки не повідомлялося про успішні атаки.

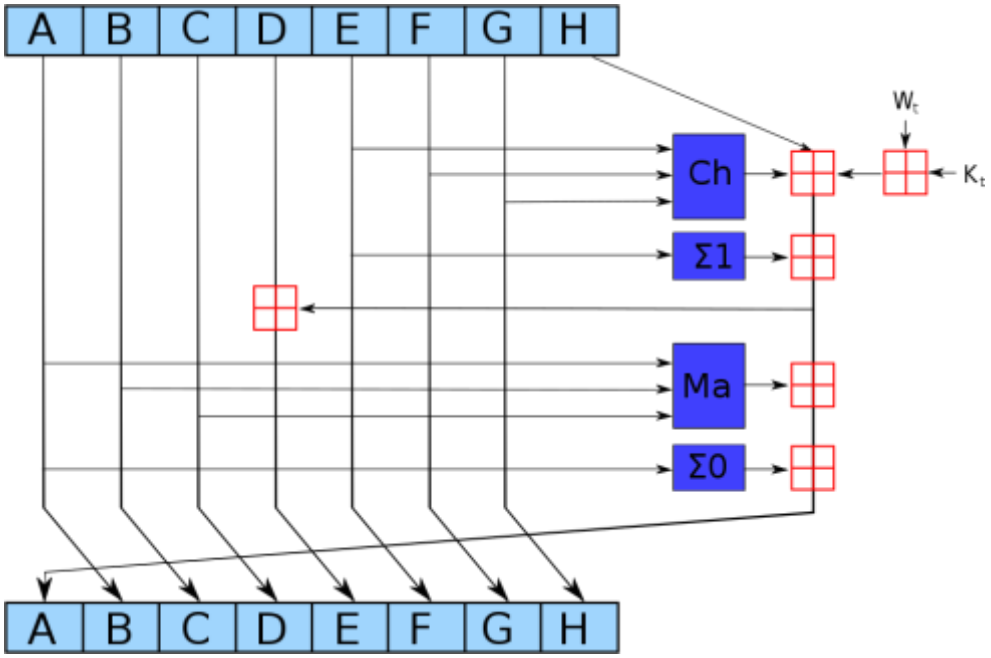


Рисунок 1.14. Схема однієї ітерації алгоритмів SHA-2

Хоча SHA-2 - сильна геш-функція. Хоча суттєво відрізняється, його базовий дизайн все ще слід за дизайном SHA-1. Отже, NIST закликав до створення нових конкурентоспроможних конструкцій геш-функцій.

У жовтні 2012 року NIST вибрав алгоритм Keccak в якості нового стандарту SHA-3. Keccak пропонує безліч переваг, таких як ефективна продуктивність і хороша стійкість до атак.

Висновок до розділу:

Аналіз відомих підходів щодо гешування даних показав, що всі вони використовують ітераційну процедуру, недоліком якої є те, що зламати геш-функцію можна шляхом послідовного просування від геш-значення останньої ітерації до попередніх ітерацій. Тому висуваються підвищені вимоги до стійкості кожної з ітерацій. Це ускладнює перетворення на кожній ітерації, і як наслідок, зменшення швидкості процесу гешування. Отже, актуальною є задача пришвидшення процесу гешування. Для розв’язання цієї задачі потрібен неітераційний підхід до гешування.

2. РОЗРОБКА МЕТОДУ ГЕШУВАННЯ

Кожне повідомлення можна охарактеризувати алфавітом, що використовується для побудови цього повідомлення. Як одну із характеристик повідомлення пропонується використовувати кількість входжень до повідомлення кожного окремого елемента алфавіту. Друга характеристика повідомлення - це номери позицій, в яких розташовані конкретні елементи алфавіту. З урахуванням цього пропонується метод, в якому спочатку визначаються такі дві характеристики, а потім – згортка отриманих результатів до коду, що має довжину геш-значення.

2.1 Метод неітераційного процесу хешування

Пропонується метод гешування, який передбачає виконання таких кроків:

1. Визначити кількість байтів певного змісту у файлі.
2. Визначити суму номерів позицій байтів певного змісту у файлі.
3. Визначити інтегральну оцінку для байтів певного змісту.
4. Обчислення значення геш-функції.

Тепер розглянемо кожен із кроків:

Крок 1.Визначити кількість байтів певного змісту у файлі.

Повідомлення **F**– файл, для якого обчислюється значення геш-функції представляється як сукупність байтів

$$\mathbf{F}=\{b_0, b_1, \ldots, b_{L-1}\}.$$

Кожному байту згідно таблиці ASCII кодів поставити у відповідність ціле число від 0 до 255.

Створити масив **K** з 256елементів, що мають значення (0).

$$\mathbf{K}=\{k_0, k_1, \ldots, k_{255}\}.$$

Послідовно зчитати байти файлу, якщо значення i-того байту $i=\{0,\ldots,i\text{-}1\}$ дорівнює $j=\{0,\ldots,255\}$, то елемент k_j збільшити на 1:

$$k_j=k_j+1.$$

Крок 2.Визначити суму номерів позицій байтів певного змісту у файлі.

Створити масив **S** з 256 елементів, що мають значення **(0):**

$$\mathbf{S}=\{S_0,S_1,\ldots,S_{255}\}, \text{ де } \mathbf{S}_l=0 \text{ для } l=\{0,...,255\}.$$

Послідовно зчитати байти файлу, якщо значення i-го байту $i=l$, то обчислити:

$$\mathbf{S}_l=(S_l+i) \bmod 2^{64}.$$

Крок 3. Визначити інтегральну оцінку для байтів певного змісту.

Сформувати інтегральний масив $\mathbf{M}=\{m_0,m_1,m_2\ldots,m_{255}\}$, де:

$$m_j = (k_j+s_j) \bmod 2^{64}.$$

Крок 4. Обчислення значення геш-функції.

Створити масив $\mathbf{M}_1$ з елементів типу bool:

$$\mathbf{M}_1=\{x_0,x_1,x_2,\ldots,x_{255}\}, \text{ де } x_j=0.$$

Створити масив $\mathbf{M}_2=\{y_0,y_1,\ldots,y_{255}\}$, де y_j – молодший біт m_j, масиву **M.**

Обчислити нові значення елементів масиву $\mathbf{M}_1$:

$$x_j= (x_j+y_j)\bmod 2, \quad j=(0,\ldots,255).$$

Перенумерувати елементи масиву $\mathbf{M}_1$, $x_j=x_{(L+4)}\text{mod}256$

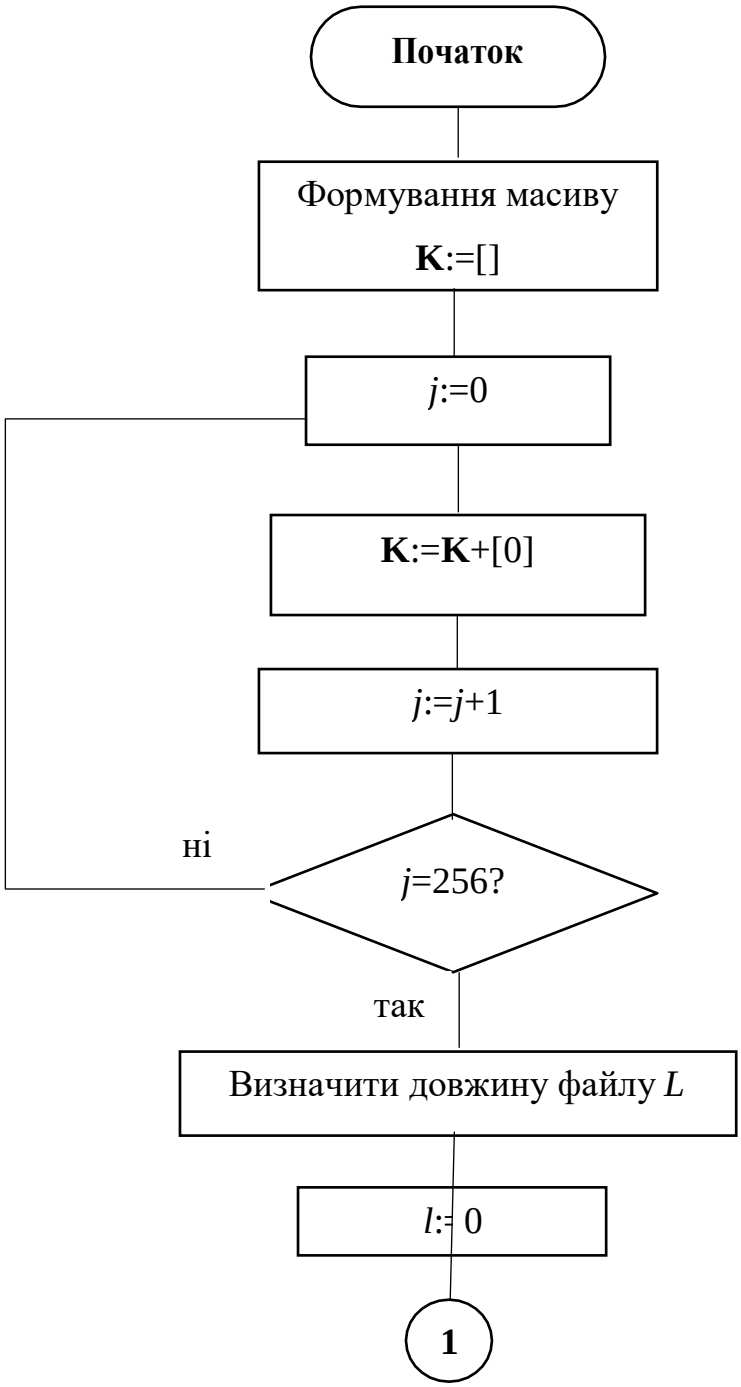
Сформувати масив **M**, та оновити елементи цього масиву:

$$x_j=\text{R1}x_0,\ x_j=x_j+y_j,\ j=(0,\ldots,255).$$

Описані дії повторити циклічно 63 рази.

2.2 Розробка алгоритму, що реалізує метод хешування

Алгоритм першого кроку.



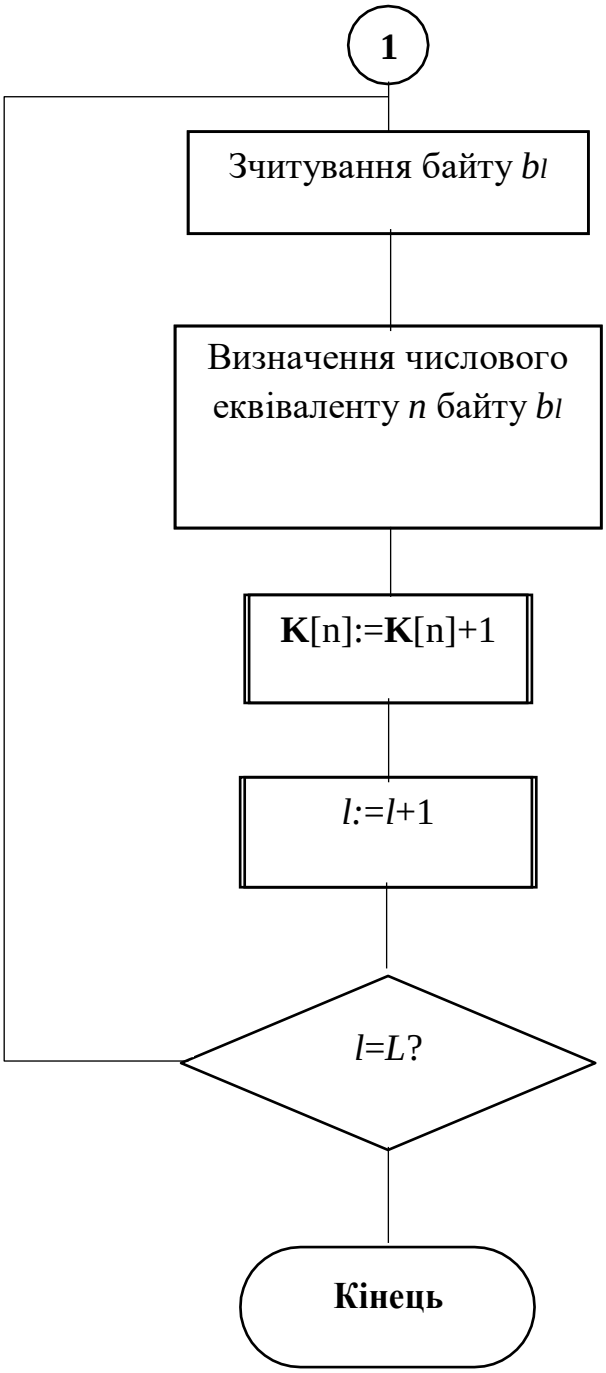
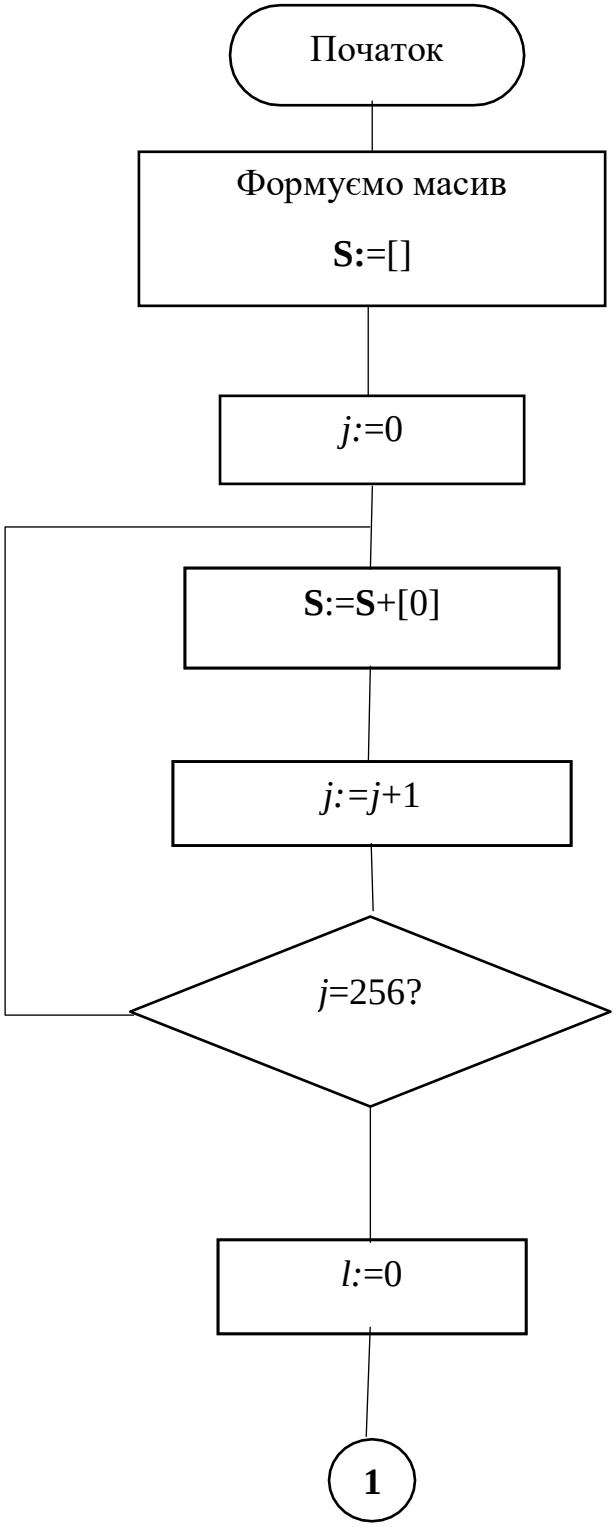


Рисунок 2.1 - Алгоритм першого кроку методу гешування

На першому кроці формуємо масив **K**={k_1,k_2,…,k_{256}}, що складається з елементів (0). Зчитуємо довжину повідомлення з файлу. Далі, створений масив **K** заповнюємо елементами, по позиціях згідно розширеної таблиці ASCII кодів, якщо код елементу за ASCII зустрічається в повідомленні, то елемент (0) на цій позиції збільшується на 1, **K**[n]=**K**[n]+1.

Алгоритм другого кроку.



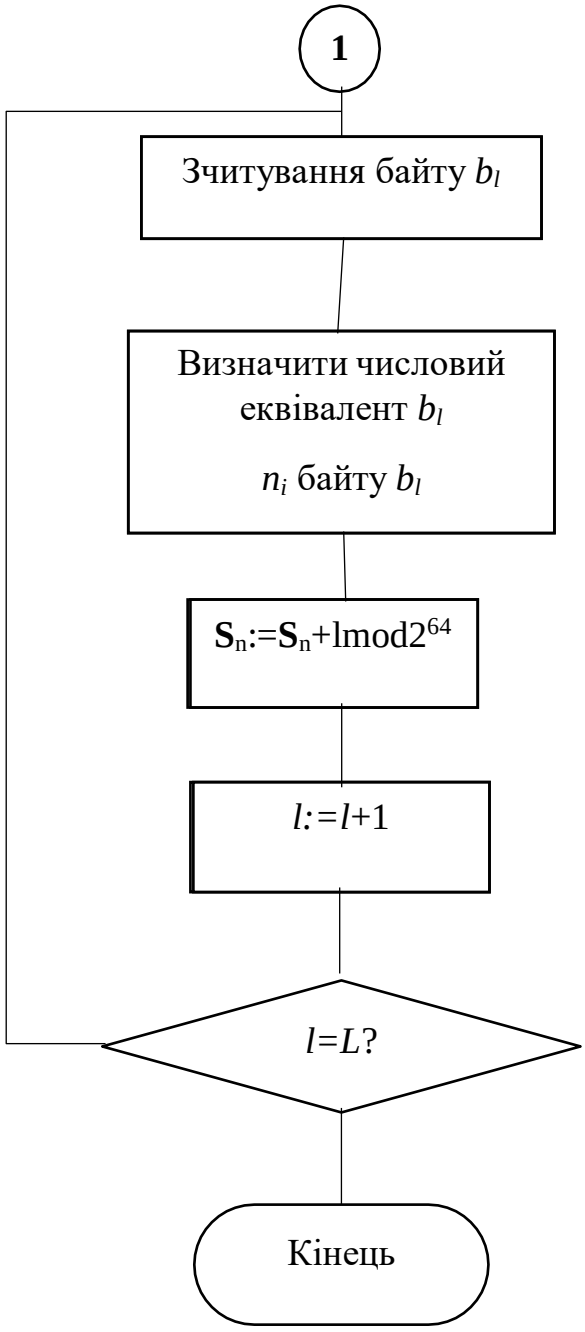


Рисунок 2.2 - Алгоритм другого кроку методу гешування

На другому кроці формуємо масив $\mathbf{S}=\{s_1,s_2,\ldots,s_{256}\}$, що складається з елементів (0). Зчитуємо довжину повідомлення з файлу. Далі, створений масив **S** заповнюємо елементами, по позиціях згідно розширенної таблиці ASCII, якщо код елементу за ASCII зустрічається в повідомленні, то номер позиції елемента збільшується на число позиції елементу в повідомленні, $\mathbf{S}_n=\mathbf{S}_{n+l} \bmod 264$.

Алгоритм третього кроку.

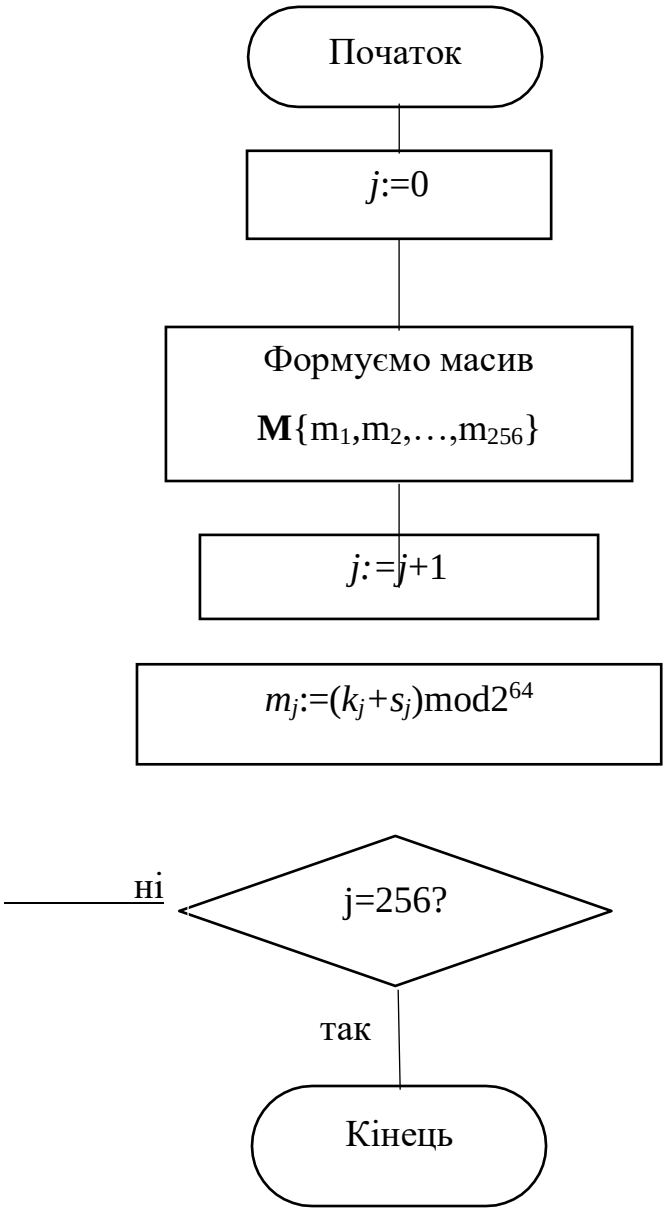
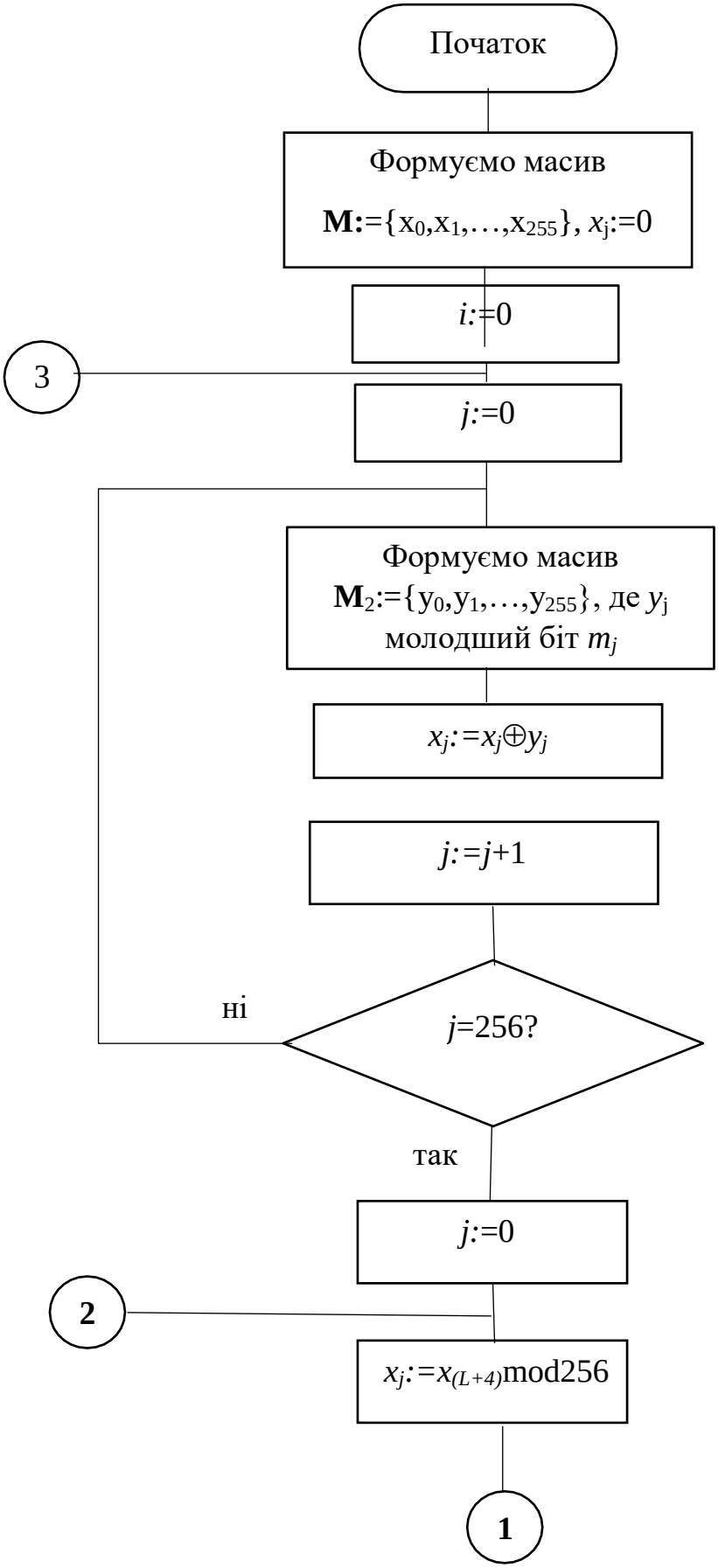


Рисунок 2.3 - Алгоритм третього кроку методу гешування

На третьому кроці формуємо масив **M**={m_0,m_1,…,m_{255}}, що складається з елементів (0). Далі обчислюємо значення $m_j:=(k_j+s_j)\mathrm{mod}2^{64}$, де m_jявляє собою поелементне додавання масивів **K** та **S**.

Алгоритм четвертого кроку.



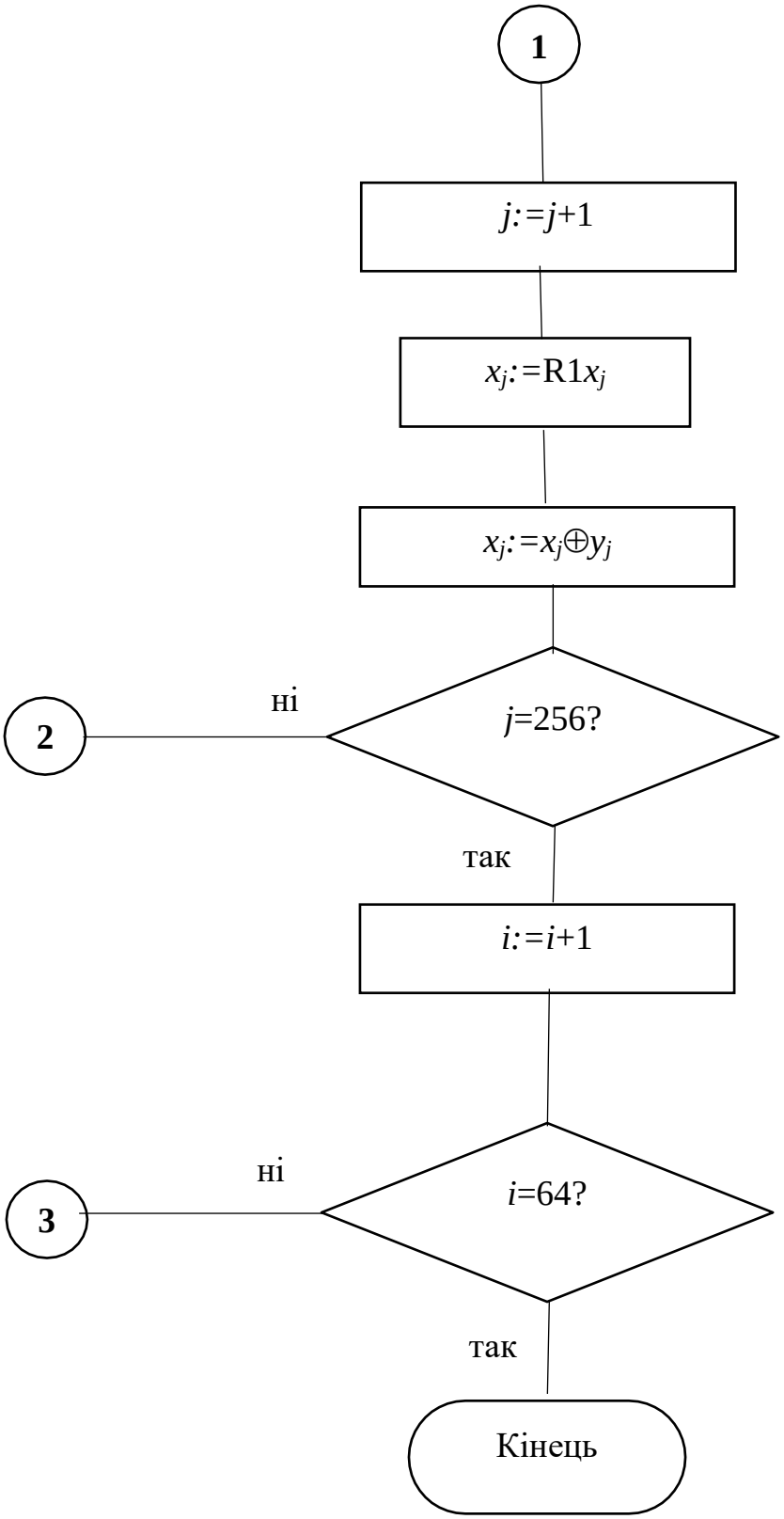


Рисунок 2.4 - Алгоритм четвертого кроку методу гешування

Висновок до розділу:

Розроблено метод гешування даних без використання ітераційної процедури. Здійснено покроковий аналіз методу та розроблено алгоритм для його реалізації.

3. РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ГЕШУВАННЯ

3.1 Реалізація методу хешування

Для реалізації методу гешування було створено консольний застосунок на мові програмування C++. Програма на вході отримує повідомлення у вигляді текстового файлу, який характеризується двома властивостями, що зберігаються в масиви. Програма перевіряє наявність файлу та можливість його відкриття та зберігає повідомлення у вигляді масиву символів.

```
    string filename("input.txt");
vector<char> bytes;
char byte = 0;
fstream input_file;
input_file.open(filename);
if (!input_file.is_open()) {
    cerr <<"Could not open the file - '" <<filename <<"'"
<<endl;
    return 1;
}
while (input_file.get(byte)) { bytes.push_back(byte); }
input_file.close();
```

Для збереження першої характеристики повідомлення використовується масив з 256 елементів цілих чисел з розрядністю 64– індекс кожної комірки масиву дорівнює значенню коду символу в розширеній таблиці ASCII. При входженні символу в повідомлення, елемент з відповідним індексом інкрементується.

Для збереження другої характеристики повідомлення використовується масив з 256 з елементів цілих чисел з розрядністю 64– індекс кожної комірки

масиву дорівнює значенню коду символу в розширеній таблиці ASCII. Для цього використовується лічильник, в якому рахується кількість елементів.

Збереження та додавання позицій елементів в масив виконується за допомогою лічильника, який на кожній ітерації циклу дорівнює позиції елементу зменшеній на одиницю.

```
    for (auto &symbol: bytes) {
    counter++;
elementCounts[int(symbol)]++;
sumOfIndexes[int(symbol)] += counter;
}
```

Наступним кроком для обчислення значення гешу є додавання двох створених масивів поелементно за модулем 2^{64}. Додавання виконується функцією, яка отримує два масиви та повертає масив з результатом.

```
    vector<unsigned long long
int>vectorsAddition(vector<unsigned long long int>&a,
vector<unsigned long long int>&b) {
vector<unsigned long long int> res(256);
    for (int i = 0; i <256; i++) {
        res[i] = (a[i] + b[i]) % ULONG_LONG_MAX;
}
return res;
}
```

Створюється масив в якому буде зберігатись сума при додаванні наймолодших бітів, який ініціюється нулями та тимчасовий масив для збереження наймолодших бітів.

```
    vector<bool> needToShiftArray(256, 0);
vector<bool> tmpArrayOfRightMostBits(256, 0);
```

На кожній ітерації в тимчасовий масив зберігається наймолодший біт кожного елемента масиву з результатом додавання за модулем 2^{64} в відповідну комірку. Наймолодший біт знаходиться за допомогою операції

ділення без остачі на 2.

До масиву з сумою додається тимчасовий масив за модулем 2.

Кожен елемент масиву з результатом додавання за модулем 2^{64} бітово зсувається на одну позицію праворуч для можливості циклічно за допомогою ділення без остачі на 2 виокремлювати наймолодший біт. Масив з результатом додавання бітів зсувається праворуч на 4 позиції.

```
    for(inti = 0; i<64; i++) {
for(intj = 0; j<256; j++) {
tmpArrayOfRightMostBits[j] = vectorOfSum[j] % 2;
}
for(intk = 0; k<256; k++) {
needToShiftArray[k] = (needToShiftArray[k] +
tmpArrayOfRightMostBits[k]) % 2;
vectorOfSum[k] = vectorOfSum[k] >>1;
}
needToShiftArray= rightShift4(needToShiftArray);
}
```

Функція для зсуву масиву на 4 позиції праворуч.

```
    vector<bool>rightShift4(vector<bool>&H) {
vector<bool> tmpVector(256);
    int tmp1 = H[256 - 1];
    int tmp2 = H[256 - 2];
    int tmp3 = H[256 - 3];
    int tmp4 = H[256 - 4];
    for (int i = 256 - 1; i >3; i--) {
        tmpVector[i] = H[i - 4];
}
    tmpVector[0] = tmp1;
tmpVector[1] = tmp2;
tmpVector[2] = tmp3;
```

```
tmpVector[3] = tmp4;

    return tmpVector;
}
```

Виокремлення наймолодшого біту, додавання та зсув повторюються 64 рази, щоб всі біти чисел з масиву додавання за модулем 2^{64} були задіяні. Після виконання всіх ітерацій результат з гешом знаходитиметься в масиві з сумами додавання за модулем 2.Результат зберігається в вихідний файл.

```
    ofstreamoutfile;
outfile.open("output.txt");
for(inti = 0; i<256; i++) {
outfile<<needToShiftArray[i];
}
outfile.close();
```

3.2 Тестування програмного засобу для хешування

Створено текстовий файл з змістом, що вказано на (рис.3.1).

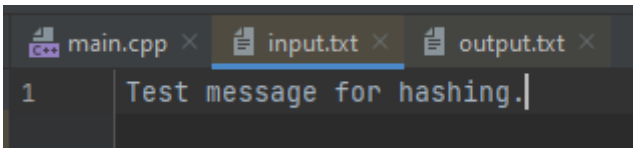


Рисунок 3.1 - Приклад вхідного файлу з повідомленням.

Після опрацювання програмним засобом тесту файлу, отримано дайджест повідомлення (рис.3.2).

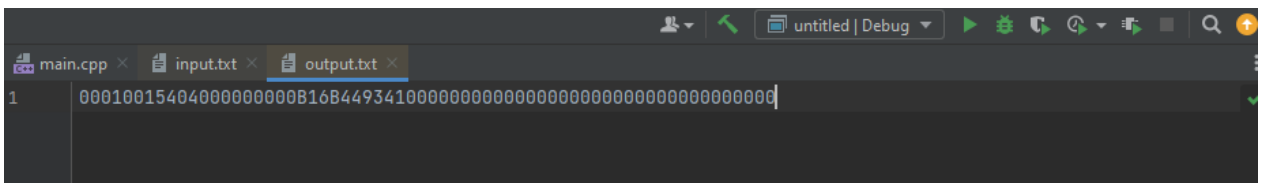


Рисунок 3.2 - Приклад вихідного файлу з геш-значенням.

Проведено невелику зміну в тексті повідомлення (рис.3.3).

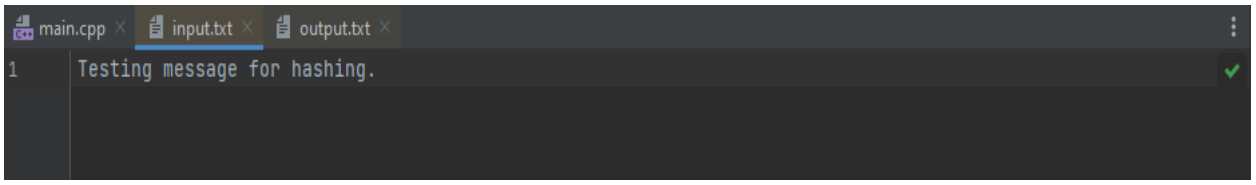


Рисунок 3.3. Приклад зміни вхідного файлу з повідомленням.

В результаті, отримано новий дайджест повідомлення, що не схожий на попередній (рис.3.4.).

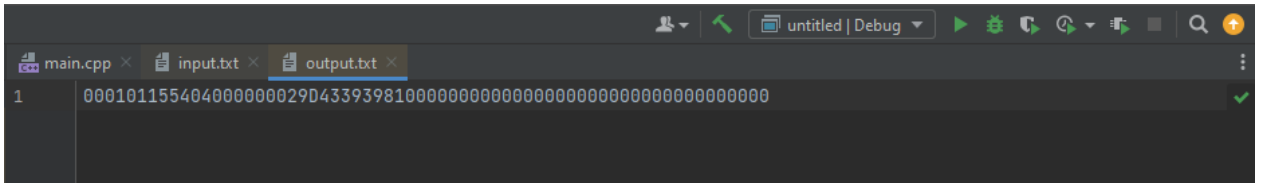


Рисунок 3.4 Приклад вихідного файлу з геш-значенням.

Наведено, ще один приклад повідомлення (рис.3.5)

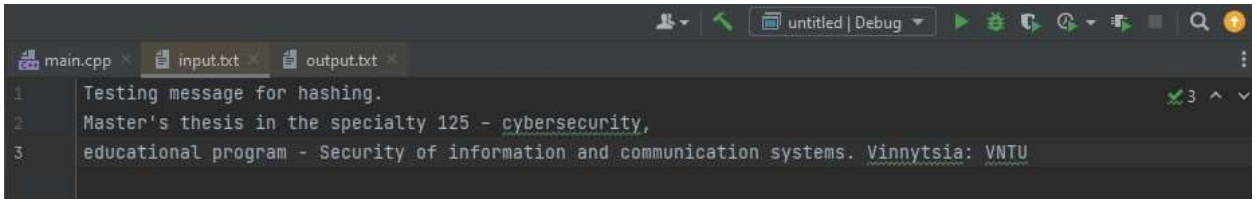


Рисунок 3.5 Приклад повідомлення

Та приклад дайджесту повідомлення (рис.3.6.)

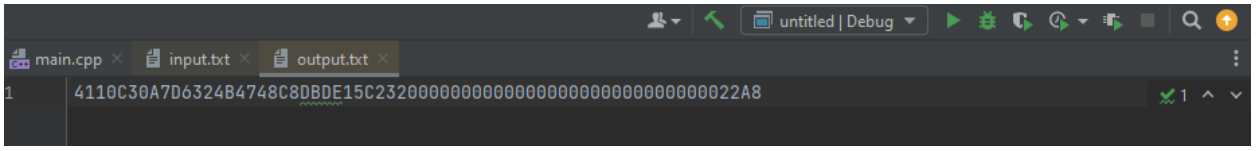


Рисунок 3.6. Геш-значення повідомлення

Знову проведемо невелику зміну повідомлення (рис.3.7.)

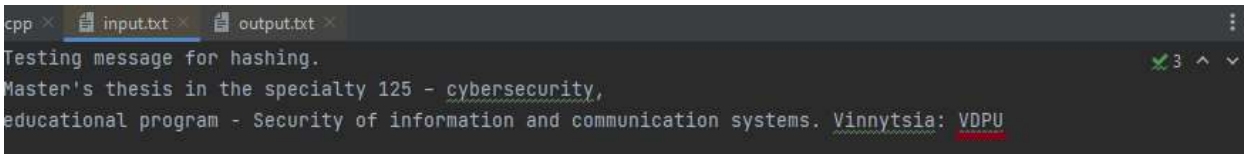


Рисунок 3.7. Приклад зміни повідомлення

Тепер переглянемо відмінності дайджесту повідомлення (рис.3.8.)

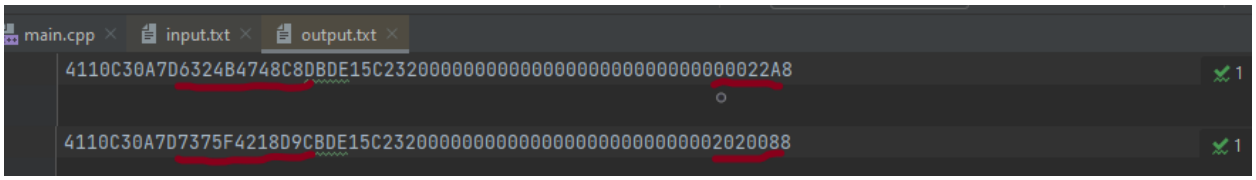


Рисунок 3.8. Відмінності дайджесту

Як показано, дайджест повідомлення набув зовсім нових значень навіть при зміні двох символі.

Висновки до розділу:

Створено консольний застосунок на мові програмування C++ та подано детальний розбір коду застосунку. Також розглянуті різні варіанти повідомлень та їх геш-значень, та доведено, що навіть при невеликій зміні повідомлення, геш-значення не збігається з дайджестом попереднього варіанту повідомлення.

4. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота «Метод та засіб гешування даних» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Метод та засіб гешування даних» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [14].

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
	2	3	4	5	6
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
2	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

За результатами розрахунків, наведених в табл. 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [14].

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод та засіб гешування даних» становить 41,0 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	5
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	3	2	2
5. Ринкові переваги (експлуатаційні витрати)	2	3	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	4	4	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	40	43	40
Середньоарифметична сума балів $СБ_с$	41,0		

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [15]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i , \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт α_iвизначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення β_iдля різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{аі}} , \tag{4.2}$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного *і*-го технічного показника якості відповідно для нової розробки та аналога;

для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{аі}}{I_{ні}} ; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі

отриманих наявних та проектних показників, а результати порівняння зведемо до табл. 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Швидкість виконання завдання	мс	2	1	2	0,35
Надійність системи	%	80	92	1,15	0,25
Точність гешування	%	75	90	1,2	0,15
Обчислювальна легкість	%	60	96	1,6	0,1
Доступність інтерфейсу	бал	7,5	9	1,2	0,15

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 2 \cdot 0{,}35 + 1{,}15 \cdot 0{,}25 + 1{,}2 \cdot 0{,}15 + 1{,}6 \cdot 0{,}1 + 1{,}2 \cdot 0{,}15 = 1{,}51.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,51 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Метод та засіб гешування даних», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій,

секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [14]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=24 дні.

$$З_о = 17770{,}00 \cdot 34 / 24 = 25174{,}17 \; грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17770,00	740,42	34	25174,17
Дослідник (інженер-розробник програмного забезпечення)	17420,00	725,83	28	20323,33
Науковий консультант (дослідник проблем гешування даних)	17500,00	729,17	8	5833,33
Технік	7100,00	295,83	15	4437,50
Всього				55768,33

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Метод та засіб гешування даних» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (4.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \quad (4.6)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i– коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду [14];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 24дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 6700,00 · 1,10 · 1,65 / *(24 · 8)* = 63,34 *грн.*

$З_{р1}$ = 63,34 · 5,50 = 348,35 *грн.*

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$= (55768,33 + 4568,77) · 11 / 100% = 6637,08 *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Розміщення комп'ютерного обладнання	5,50	2	1,10	63,34	348,35
Інсталяція програмного забезпечення дослідження методів та розробки засобу гешування даних	7,42	5	1,70	97,88	726,29
Підготовка робочого місця розробника програмного забезпечення	6,25	2	1,10	63,34	395,85
Компіляція програмних блоків	22,00	4	1,50	86,37	1900,08
Налагодження програмних блоків	4,30	5	1,70	97,88	420,90
Тестування програмного засобу	10,00	3	1,35	77,73	777,30
Всього					4568,77

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (55768,33 + 4568,77 + 6637,08) · 22 / 100% = 14734,32*грн.*

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Метод та засіб гешування даних».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j \; - \sum_{j=1}^{n} B_j \cdot Ц_{вj} \,, \qquad (4.9)$$

де H_j– норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = 1,1 … 1,15);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3,0 · 215,00 · 1,12 - 0 · 0 = 722,40 *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний	215,00	3,0	0	0	722,40
Папір для заміток	85,00	3,0	0	0	285,60
Начиння канцелярське	142,00	4,0	0	0	636,16
Органайзер офісний	162,00	3,0	0	0	544,32
Картридж для принтера	2560,00	1,0	0	0	2867,20
Диск оптичний	11,50	4,0	0	0	51,52
FLASH-пам’ять	179,00	1,0	0	0	200,48

Mini SD-card (64 Gb, class 10)	259,00	1,0	0	0	290,08
Всього					5597,76

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Метод та засіб гешування даних», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$= 1,1 … 1,15).

$К_в$= 1 · 2690,00 · 1,12 = 3012,80 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Відеокарта AMD Radeon ™ Graphics (2GB)	1	2690,00	3012,80
Відеокарта NVIDIA GeForce RTX 3060 (6 GB)	1	6250,00	7000,00
Всього			10012,80

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$В_{спец} = \sum_{i=1}^{k} Ц_i \cdot С_{пр.i} \cdot К_i \ , \qquad (4.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.і}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = 1,10…1,12*);*

k – кількість найменувань устаткування.

$B_{спец}$ = 44999,00 · 1 · 1,11 = 49948,89 *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Lenovo Legion 5 15ACH6H (процесор AMD Ryzen 5 5600Hz	1	44999,00	49948,89
Всього			49948,89

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = 1,10…1,12*);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = 5620,00 · 1 · 1,11 = 6238,20 *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Пакет забезпечення мови розробки	1	5620,00	6238,20
Середовище розробки WebStorm 2022.3	1	4460,00	4950,60
Абонентна плата доступу до мережі Internet	1	420,00	466,20
Всього			11655,00

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

Таблиця 4.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Обладнання доступу до мережі Internet	8210,00	2	2	684,17
Обладнання виводу інформації принтер HP LaserJet з Wi-Fi	6999,00	4	2	291,63
Робоче місце інженера-дослідника спеціалізоване	8750,00	5	2	291,67
Офісна оргтехніка	7900,00	5	2	263,33
Приміщення лабораторії досліджень	625000,00	20	2	5208,33
ОС Windows 11	8790,00	2	2	732,50
ПЗ OFFICE 2019	6430,00	2	2	535,83
Всього				8007,46

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = (8210,00 · 2) / (2 · 12) = 684,17 *грн.*

Проведені розрахунки зведемо до таблиці.

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot K_{вni}}{\eta_i}, \qquad (4.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i– тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 6,15

грн;

$K_{вп i}$ – коефіцієнт, що враховує використання потужності, $K_{вп i}<1$;

η_i– коефіцієнт корисної дії обладнання, $\eta_i<1$.

B_e = 0,36 · 270,0 · 6,15 · 0,95 / 0,97 = 597,78 *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ПК Lenovo Legion 5 15ACH6H	0,36	270,0	597,78
Обладнання доступу до мережі Internet	0,03	270,0	49,82
Обладнання виводу інформації принтер EPSON M102w з Wi-Fi (G3Q35A)	0,25	12,0	18,45
Робоче місце інженера-дослідника спеціалізоване	0,11	270,0	182,66
Офісна оргтехніка	0,62	2,5	9,53
Всього			858,23

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Метод та засіб гешування даних» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (4.15)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $H_{св}$ = 22%.

$B_{св}$ = (55768,33 + 4568,77) · 22 / 100% = 13274,16 *грн.*

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_{в} = (З_{о} + З_{р}) \cdot \frac{Н_{ів}}{100\%}, \quad (4.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 55%.

$I_{в}$ = (55768,33 + 4568,77) · 55 / 100% = 33185,40 *грн.*

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_{о} + З_{р}) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$= (55768,33 + 4568,77) · 100 / 100% = 60337,10 *грн.*

Витрати на проведення науково-дослідної роботи на тему «Метод та засіб гешування даних» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.18)$$

$B_{заг}$= 55768,33 +4568,77 +6637,08 +14734,32 +5597,76 +10012,80 +49948,89 + 11655,00 + 8007,46 +858,23 +13274,16 +0,00 +33185,40 +60337,10 = 274585,31 *грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,95.

ЗВ = 274585,31 / 0,95 = 289037,17 *грн.*

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Метод та засіб гешування даних» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	1000	2000	3000	1000

N – кількість споживачів які використовували аналогічний продукту році до впровадження результатів нової науково-технічної розробки, приймемо 11000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 12500,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо зростання на 743,75 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [14]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році ϑ =18%;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (743{,}75 \cdot 11000{,}0 + 13243{,}75 \cdot 1000) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 5832742{,}00$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (743{,}75 \cdot 11000{,}0 + 13243{,}75 \cdot 3000) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 13043699{,}0$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (743,75 \cdot 11000,0+13243,75 \cdot 6000) \cdot 0,83 \cdot 0,4 \cdot (1\text{-}0,18/100\%)=23860134,5$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 =(743,75 \cdot 11000,0+13243,75 \cdot 7000) \cdot 0,83 \cdot 0,4 \cdot (1\text{-}0,18/100\%)=27465613,0$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки,роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,29;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

ПП =5832742,00/(1+0,29)^1+13043699,00/(1+0,29)^2+23860134,50/(1+0,29)^3+
+27465613,00/(1+0,29)^4=4521505,43+7838290,37+11114853,85+9918145,04=
=33392794,69 грн.

Величина початкових інвестицій *PV* , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (4.22)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2,3;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 289037,17 грн.

$PV = k_{інв} \cdot ЗВ = 2{,}3 \cdot 289037{,}17 = 664785{,}48$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.23}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 33392794,69 грн;

PV – теперішня вартість початкових інвестицій, 664785,48 грн.

$E_{абс} = ПП - PV = 33392794{,}69 - 664785{,}48 = 32728009{,}21$ грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.24}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 32728009,21 грн;

PV – теперішня вартість початкових інвестицій, 664785,48 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+32728009{,}21/664785{,}48)^{1/4} = 1{,}66.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.25}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d =0,12;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,4.

$\tau_{мін}$ = 0,12+0,4 = 0,52<1,66 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Метод та засіб гешування даних» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.26}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,66 = 0,60 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу:

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод та засіб гешування даних» становить 41,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,51 рази.

Також термін окупності становить 0,60 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може

спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Метод та засіб гешування даних».

ВИСНОВКИ

Аналіз відомих підходів щодо гешування даних показав, що всі вони використовують ітераційну процедуру, недоліком якої є те, що зламати геш-функцію можна шляхом послідовного просування від геш-значення останньої ітерації до попередніх ітерацій. Тому висуваються підвищені вимоги до стійкості кожної з ітерацій. Це ускладнює перетворення на кожній ітерації, і як наслідок, зменшення швидкості процесу гешування. Отже, актуальною є задача пришвидшення процесу гешування.

У ході виконання магістерської кваліфікаційної роботи було розроблено метод та засіб для гешування даних без використання ітеративної процедури. Здійснено покроковий аналіз методу та розроблено алгоритм для його реалізації.

Створено консольний застосунок мовою програмування C++ та подано детальний розбір коду застосунку. Також розглянуті різні варіанти повідомлень та їх геш-значень, та доведено, що навіть при невеликій зміні повідомлення, геш-значення не збігається з дайджестом попереднього варіанту повідомлення.

У економічні частина магістерської кваліфікаційної роботи було оцінено комерційний потенціал и розробки, загальні витрати на впровадження та виконання результатів наукової роботи, розраховано чистий прибуток та ціну реалізації щодо результатів розробки, а також розраховано період окупності результатів розробки та наукової роботи. В ході результатів розрахунків було виявлено доцільність даної розробки, так як показник якості розробки показав, що даний метод є більш ефективним за альтернативи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технології захисту інформації. Ю. А. Тарнавський. 47-49 с. Режим доступу: https://ela.kpi.ua/bitstream/123456789/23896/1/TZI_book.pdf

2. R. Anderson. The classification of hash functions. Proc. of the IMA Conference on Cryptography and Coding, Cirencester, December 1993, Oxford University Press, 1995, pp. 83-95.

3. Cryptography tutorial. 2015 by Tutorials Point (I) Pvt. Ltd

4. The Cryptix Foundation Limited and David Hopwood. Standard Cryptographic Algorithm Naming. Version 1.0.20a - 22 October, 2002. Режим доступу: http://www.users.zetnet.co.uk/hopwood/crypto/scan/md.html

5.Gary C. Kessler. An Overview of Cryptography. 1998-2020 r. Режим доступу: https://www.garykessler.net/library/crypto.html

6. M. Bellare, R. Canetti, and H. Krawczyk. Keyed Hash Functions and Message Authentication // Proceedings of Crypto'96, LNCS 1109, pp. 1-15.

7. Meyer, S. and Matyas, S.M., Cryptography, New York Wiley, 1982.

8. B. Preneel and P. van Oorschot. Building fast MACs from hash functions // Advances in Cryptology — CRYPTO'95 Proceedings, Lecture Notes in Computer Science, Springer-Verlag Vol.963, 1995, pp. 1-14.

9. H. Dobbertin. The Status of MD5 After a Recent Attack // RSA Labs' CryptoBytes, Vol. 2 No. 2, Summer 1996.

10. H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A strengthened version of RIPEMD // Fast Software Encryption, LNCS Vol 1039, pp. 71-82.

11. Фергюсон Н., Шнайер Б.: Практична криптографія, 550 с.

12. FIPS 180-2: Secure Hash Standard (SHS): 6. Applicability, pp 54-63.

13. Somitra Kumar Sanadhya, Palash Sarkar. Deterministic Constructions of 21-Step Collisions for the SHA-2 Hash Family, pp. 10-12.

14. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й.

Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

15. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Метод та засіб гешування даних

Автор роботи: Нога Ілля Вікторович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 85.7%. Схожість – 14.3%.

Аналіз звіту подібності (відмітити потрібне):

☐ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

√ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Каплун В. А. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Нога І. В (прізвище, ініціали)

Керівник роботи _____ (підпис) Лужецький В. А. (прізвище, ініціали)

Додаток Б. Повний код програми

```
#include <bits/stdc++.h>


using namespace std;

vector<unsigned long long int> vectorsAddition(vector<unsigned long long
int>&a, vector<unsigned long long int>&b) {
    vector<unsigned long long int> res(256);
    for (int i = 0; i <256; i++) {
        res[i] = (a[i] + b[i]) % ULONG_LONG_MAX;
}
return res;
}

vector<bool> rightShift4(vector<bool>&H) {
    vector<bool> tmpVector(256);
    int tmp1 = H[256 - 1];
    int tmp2 = H[256 - 2];
    int tmp3 = H[256 - 3];
    int tmp4 = H[256 - 4];
    for (int i = 256 - 1; i >3; i--) {
        tmpVector[i] = H[i - 4];
}
    tmpVector[0] = tmp1;
tmpVector[1] = tmp2;
tmpVector[2] = tmp3;
tmpVector[3] = tmp4;

    return tmpVector;
}

void createMap(unordered_map<string, char> *um)
{
    (*um)["0000"] = '0';
(*um)["0001"] = '1';
(*um)["0010"] = '2';
(*um)["0011"] = '3';
(*um)["0100"] = '4';
(*um)["0101"] = '5';
(*um)["0110"] = '6';
(*um)["0111"] = '7';
(*um)["1000"] = '8';
(*um)["1001"] = '9';
(*um)["1010"] = 'A';
(*um)["1011"] = 'B';
(*um)["1100"] = 'C';
(*um)["1101"] = 'D';
(*um)["1110"] = 'E';
(*um)["1111"] = 'F';
}

// function to find hexadecimal
// equivalent of binary
string convertBinToHex(string bin)
{
```

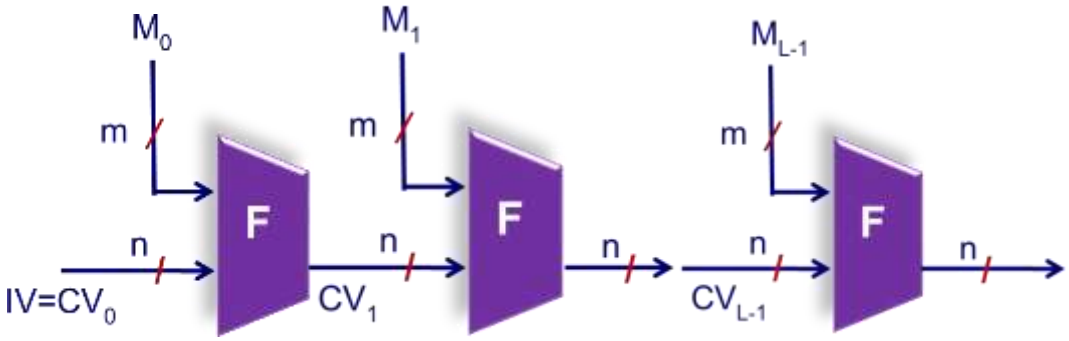
```
int l = bin.size();
    int t = bin.find_first_of('.');

// length of string before '.'
int len_left = t != -1 ? t : l;

// add min 0's in the beginning to make
    // left substring length divisible by 4
for (int i = 1; i <= (4 - len_left % 4) % 4; i++)
        bin = '0' + bin;

// if decimal point exists
if (t != -1)
    {
// length of string after '.'
int len_right = l - len_left - 1;

// add min 0's in the end to make right
        // substring length divisible by 4
for (int i = 1; i <= (4 - len_right % 4) % 4; i++)
            bin = bin + '0';
}

// create map between binary and its
    // equivalent hex code
unordered_map<string, char> bin_hex_map;
createMap(&bin_hex_map);

    int i = 0;
string hex = "";

    while (1)
    {
// one by one extract from left, substring
        // of size 4 and add its hex code
hex += bin_hex_map[bin.substr(i, 4)];
i += 4;
        if (i == bin.size())
break;

// if '.' is encountered add it
        // to result
if (bin.at(i) == '.')
        {
            hex += '.';
i++;
}
    }

// required hexadecimal number
return hex;
}

int main() {
//reading input file
string filename("input.txt");
vector<char> bytes;
    char byte = 0;
fstream input_file;
```

```
input_file.open(filename);
    if (!input_file.is_open()) {
        cerr <<"Could not open the file - '" << filename <<"'" << endl;
        return 1;
}
while (input_file.get(byte)) { bytes.push_back(byte); }
    input_file.close();

// count element in input message
unsigned long long int counter = 0;
// first - ascii code, second - count in message
vector<unsigned long long int> elementCounts(256, 0);
//vector storage summs of indexes of elements
vector<unsigned long long int> sumOfIndexes(256, 0);

// fill vector which storage elements count in input message
for (auto &symbol: bytes) {
        counter++;
elementCounts[int(symbol)]++;
sumOfIndexes[int(symbol)] += counter;
}
// cout << endl << "elementCounts: " << endl;
for (const auto &i: elementCounts) cout << i <<" ";
//cout << endl;
   // cout << endl << "sumOfIndexes: " << endl;
for (const auto &i: sumOfIndexes) cout << i <<" ";

// initialize array of sums;
vector<unsigned long long int> vectorOfSum(256, 0);
vectorOfSum = vectorsAddition(elementCounts, sumOfIndexes);


vector<bool> needToShiftArray(256, 0);
vector<bool> tmpArrayOfRightMostBits(256, 0);

    for (int i = 0; i <64; i++) {

for (int j = 0; j <256; j++) {
            tmpArrayOfRightMostBits[j] = vectorOfSum[j] % 2;
}
for (int k = 0; k <256; k++) {
            needToShiftArray[k] = (needToShiftArray[k] +
tmpArrayOfRightMostBits[k]) % 2;
vectorOfSum[k] = vectorOfSum[k] >>1;
}

        needToShiftArray = rightShift4(needToShiftArray);

}
    string bin = "";
    for (int i = 0; i <256; i++)
if (needToShiftArray[i])

bin.push_back('1');
        else
bin.push_back('0');
// cout << endl << bin;
cout << endl << convertBinToHex(bin);
```

```
ofstream outfile;
outfile.open("output.txt");
outfile << convertBinToHex(bin);
outfile.close();
    return 0;
}
```

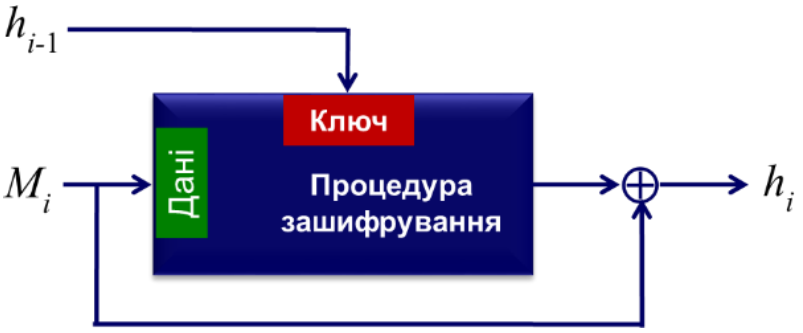
ІЛЮСТРАТИВНА ЧАСТИНА

ІТЕРАЦІЙНИЙ МЕТОД ГЕШУВАННЯ



ПІДХІД ДО ПОБУДОВИ ГЕШ-ФУНКЦІЙ НА ОСНОВІ АЛГОРИТМІВ БЛОКОВОГО ШИФРУВАННЯ

$$h_i = E_{h_{i-1}}(M_i) \oplus M_i$$



ПІДХІД ДО ПОБУДОВИ ГЕШ-ФУНКЦІЙ НА ОСНОВІ СКЛАДНООБЧИСЛЮВАЛЬНОЇ МАТЕМАТИЧНОЇ ЗАДАЧІ

Повідомлення представляється як послідовність m-розрядних блоків M_0 , M_1 , ..., M_{L-1}

Для обчислення геш-коду **G** використовується операція піднесення до степеня за модулем. H_0 = початкове значення.

$$G = H_{L-1}$$

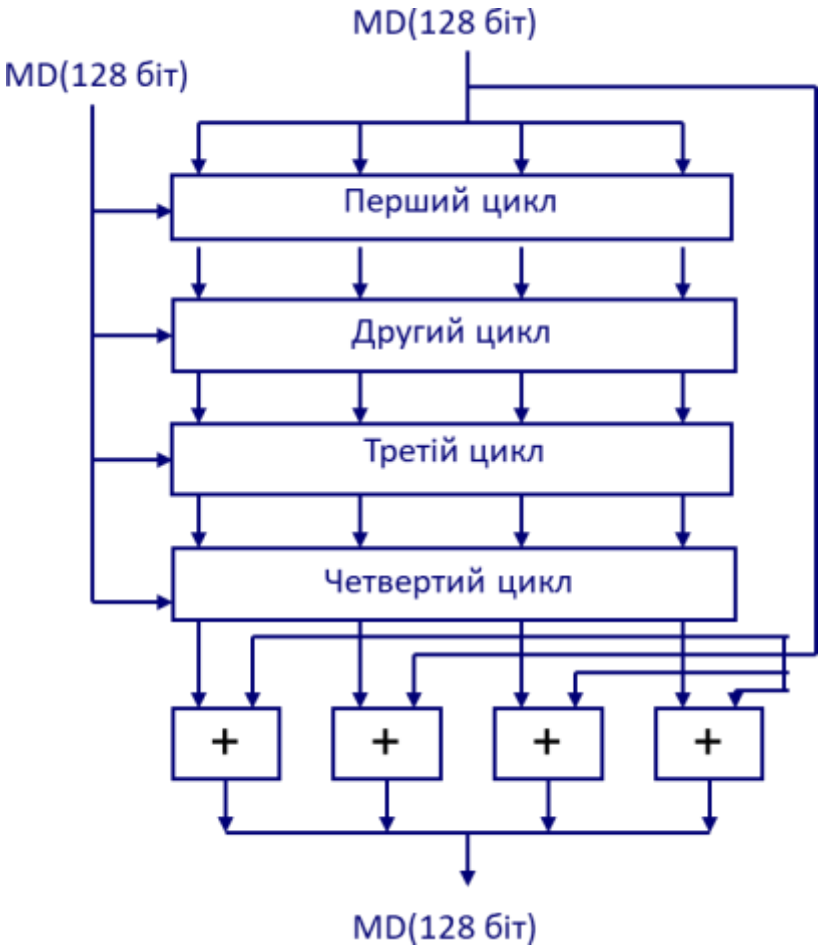
Спосіб 1. $H_i = M_{i-1}^{H_{i-1}} \bmod p$

Спосіб 2. $H_i = H_{i-1}^{M_{i-1}} \bmod p$

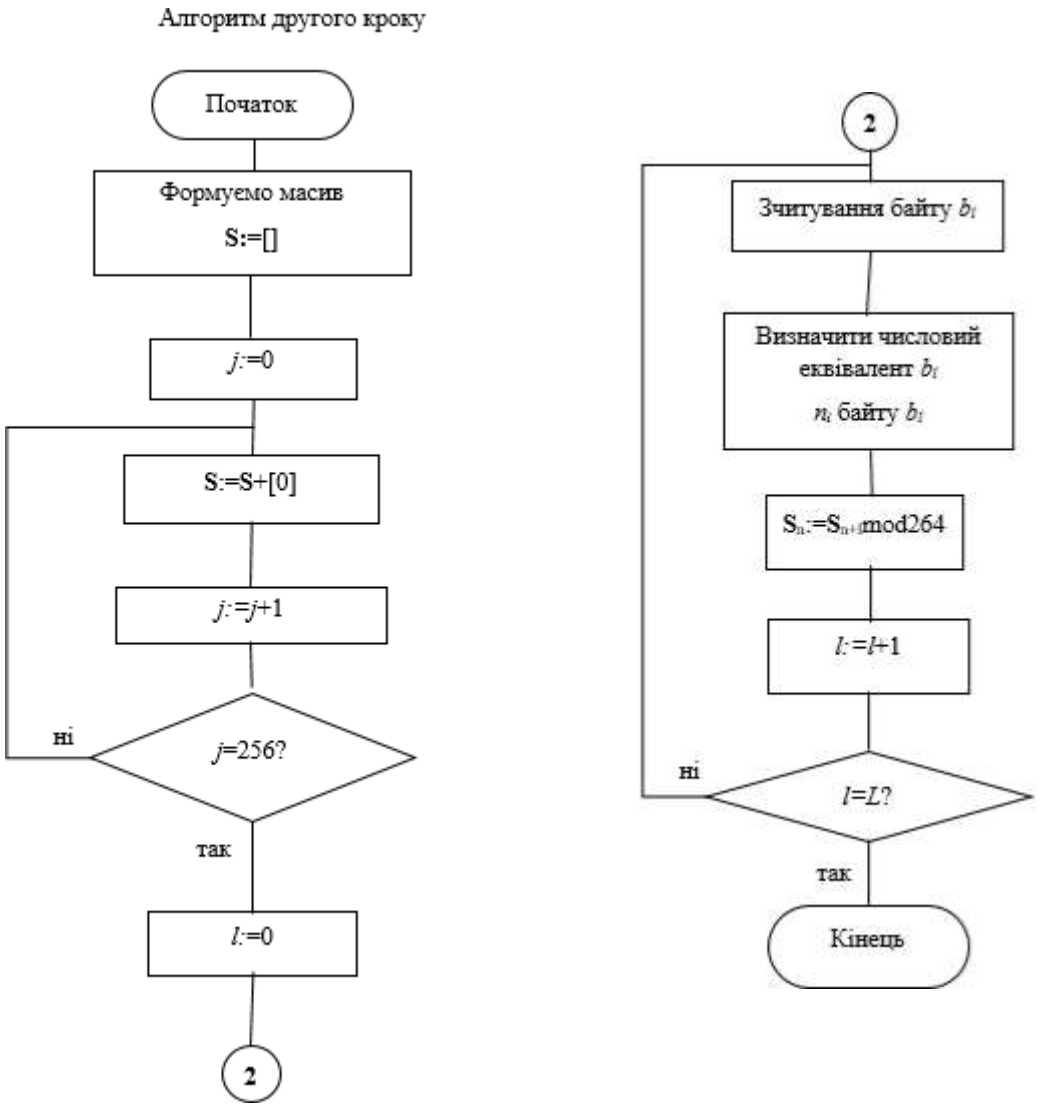
Спосіб 3. $H_i = g^{H_{i-1} \oplus M_{i-1}} \bmod p$

g – примітивний елемент за модулем p.

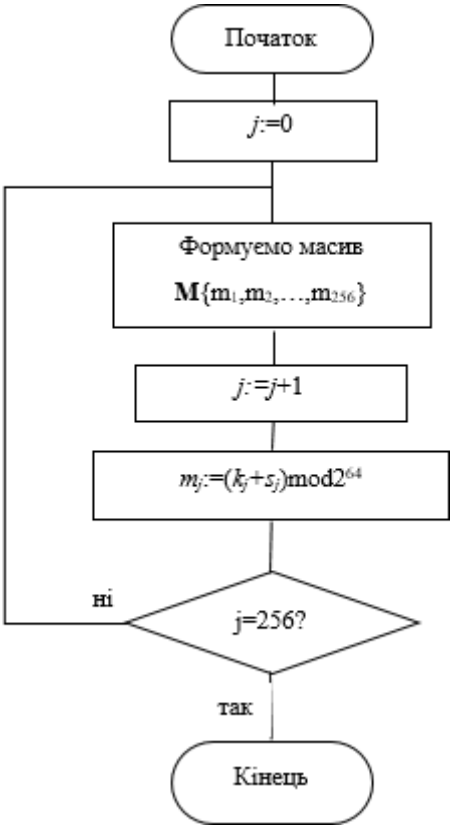
ПІДХІД ДО ПОБУДОВИ ГЕШ-ФУНКЦІЙ З НУЛЯ



АЛГОРИТМ ДРУГОГО КРОКУ ГЕШУВАННЯ



АЛГОРИТМ ТРЕТЬОГО КРОКУ ГЕШУВАННЯ



РЕЗУЛЬТАТИ ТЕСТУВАННЯ