МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова
праця на правах рукопису

Кисюк Дмитро Васильович

УДК [004.6+004.7].056(043.5)

ДИСЕРАТЦІЯ

Методи та засоби організації роботи з даними у приватних хмарних сервісах на основі байт-орієнтованого гешування
(назва дисертації)

123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

12 «Інформаційні технології»
(галузь знань)

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших акторів мають посилання на відповідне джерело

_____ Д. В. Кисюк

Наукові керівники
Лужецький Володимир Андрійович
доктор технічних наук, професор,
в.о. зав. каф. захисту інформації
Захарченко Сергій Михайлович
кандидат технічних наук, професор

Вінниця – 2026

АНОТАЦІЯ

Кисюк Д. В. Методи та засоби організації роботи з даними у приватних хмарних сервісах на основі байт-орієнтованого гешування. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії в галузі знань 12 Інформаційні технології за спеціальністю 123 «Комп'ютерна інженерія» – Вінницький національний технічний університет, Вінниця, 2026.

У дисертаційній роботі розв'язано актуальну науково-прикладну задачу комп'ютерної інженерії, що полягає у підвищенні ефективності організації роботи з даними у приватних хмарних сервісах за рахунок розроблення байт-орієнтованих методів гешування та спеціалізованих апаратно-програмних засобів їх реалізації в умовах обмежених обчислювальних і енергетичних ресурсів.

Актуальність теми дослідження зумовлена зростанням обсягів даних, що обробляються у приватних хмарних інфраструктурах, зокрема у відокремлених та ізольованих контурах, де висуваються підвищені вимоги до продуктивності, енергоефективності та надійності підсистем роботи з даними. У таких середовищах традиційні програмні підходи до реалізації гешування та контролю цілісності даних не завжди забезпечують необхідні експлуатаційні характеристики, що обумовлює доцільність застосування спеціалізованих методів обробки даних та апаратного прискорення.

Дисертаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків.

У **вступі** обґрунтовано актуальність теми дослідження та визначено поточний стан вирішення задачі організації роботи з даними у приватних хмарних сервісах з позицій комп'ютерної інженерії. Визначено об'єкт і предмет дослідження, сформульовано мету та основні завдання дисертаційної роботи, наведено методи дослідження, що використовувалися для досягнення поставленої мети, розкрито наукову новизну та практичне значення отриманих результатів. Наведено відомості щодо апробації результатів дослідження та публікацій автора.

У **першому розділі** проведено системний аналіз сучасних підходів до організації роботи з даними у хмарних сервісах з акцентом на приватні хмарні інфраструктури. Розглянуто архітектурні моделі хмарних сервісів, особливості ізольованих контурів обробки даних та обмеження, характерні для приватних хмарних середовищ, зокрема обмеження за продуктивністю, пропускною здатністю, енергоспоживанням і масштабованістю. Проаналізовано роль гешування як базового механізму забезпечення цілісності даних, оптимізації процесів зберігання, дедуплікації та резервування. Показано, що використання загальноприйнятих програмних реалізацій геш-функцій призводить до суттєвого навантаження на обчислювальні ресурси та системи та збільшення енергоспоживання, що є критичним фактором для ресурсно-обмежених вузлів приватних хмарних інфраструктур. На підставі проведеного аналізу сформульовано вимоги до методів гешування та засобів їх реалізації, орієнтованих на використання у таких середовищах.

У **другому розділі** розроблено байт-орієнтовані методи гешування даних, які забезпечують побудову ресурсно-обмежених пристроїв та спеціалізованих процесорів. Запропоновані методи базуються на обробці характеристичних ознак вхідного потоку даних та спрямовані на зменшення обчислювальної складності й енергоспоживання при формуванні геш-значень. Удосконалено методи організації роботи з даними у приватних хмарних сервісах шляхом інтеграції розроблених байт-орієнтованих методів гешування у процеси завантаження, зберігання, дедуплікації та резервування даних. Проведено порівняльний аналіз запропонованих методів з відомими методами за показниками продуктивності, апаратної складності реалізації та доцільності використання у складі апаратно-програмних підсистем обробки даних.

У **третьому розділі** запропоновано функціональну та структурну моделі спеціалізованого процесора байт-орієнтованого гешування, який реалізує обробку характеристичних ознак вхідного потоку даних відповідно до розроблених методів. Розроблено багатошарову апаратно-програмну архітектуру підсистеми організації роботи з даними у приватних хмарних сервісах, що забезпечує інтеграцію

спеціалізованого процесора гешування з серверними та клієнтськими компонентами інфраструктури. Запропонована архітектура дозволяє зменшити навантаження на центральні обчислювальні підсистеми, підвищити продуктивність та забезпечити енергоефективне функціонування підсистеми обробки даних.

У дисертаційній роботі проведено експериментальні дослідження розроблених методів та спеціалізованого процесора у складі прототипу підсистеми роботи з даними приватної хмарної інфраструктури. Результати експериментів підтвердили зростання швидкодії операцій гешування, зменшення енергоспоживання та підвищення ефективності організації процесів роботи з даними порівняно з відомими підходами.

У загальних **висновках** підсумовано основні результати дисертаційної роботи, підтверджено досягнення поставленої мети та визначено напрями подальших досліджень, пов’язані з розвитком енергоефективних апаратно-програмних підсистем обробки даних для приватних хмарних інфраструктур та спеціалізованих обчислювальних середовищ.

Ключові слова: хмарна безпека, хмарні обчислення, хмарні технології, приватні хмарні сервіси, моделі хмарних сервісів, критичні системи, кібербезпека, захист інформації, гешування, геш-функція, байт-орієнтоване гешування, криптопроцесор, перевірка цілісності, кіберфізичні системи, апаратно-програмна архітектура, edge-комп’ютинг, IoT, FPGA.

ABSTRACT

Kysiuk D. V. Methods and means of organizing data processing in private cloud services based on byte-oriented hashing. – Qualification scientific work, manuscript.

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the field of knowledge 12 Information Technologies, specialty 123 “Computer Engineering” – Vinnytsia National Technical University, Vinnytsia, 2026.

The dissertation addresses a relevant scientific and applied problem in the field of computer engineering, which consists in improving the efficiency of data organization and processing in private cloud services through the development of byte-oriented hashing methods and specialized hardware–software means for their implementation under conditions of limited computational and energy resources.

The relevance of the research topic is determined by the continuous growth of data volumes processed in private cloud infrastructures, particularly within isolated and segregated environments, where increased requirements are imposed on performance, energy efficiency, and reliability of data processing subsystems. In such environments, conventional software-based approaches to hashing and data integrity control do not always provide the required operational characteristics, which substantiates the feasibility of applying specialized data processing methods and hardware acceleration.

The dissertation consists of an introduction, three chapters, general conclusions, a list of references, and appendices.

In the **introduction**, the relevance of the research topic is justified and the current state of solving the problem of data organization in private cloud services is analyzed from the standpoint of computer engineering. The object and subject of the research are defined, the purpose and main objectives of the dissertation are formulated, the research methods used to achieve the stated goal are presented, and the scientific novelty and practical significance of the obtained results are disclosed. Information on the approbation of research results and the author’s publications is also provided.

The **first chapter** presents a systematic analysis of modern approaches to organizing data processing in cloud services, with an emphasis on private cloud infrastructures. Architectural models of cloud services, features of isolated data processing domains, and constraints inherent in private cloud environments are examined, including limitations related to performance, bandwidth, energy consumption, and scalability. The role of hashing as a fundamental engineering mechanism for ensuring data integrity and for optimizing storage, deduplication, and backup processes is analyzed. It is shown that the use of widely adopted software implementations of hash functions leads to significant computational load and increased energy consumption, which is a critical factor for resource-constrained nodes of private cloud infrastructures. Based on the conducted analysis, requirements for hashing methods and their implementation means oriented toward such environments are formulated.

In the **second chapter**, byte-oriented data hashing methods intended for implementation in resource-constrained devices and specialized processors are developed. The proposed methods are based on the processing of characteristic features of the input data stream and are aimed at reducing computational complexity and energy consumption during hash value generation. The methods of organizing data processing in private cloud services are improved by integrating the developed byte-oriented hashing methods into the processes of data ingestion, storage, deduplication, and backup. A comparative analysis of the proposed methods with known software implementations is carried out in terms of performance, hardware implementability, and feasibility of use within hardware–software data processing subsystems.

The **third chapter** proposes functional and structural models of a specialized byte-oriented hashing processor that performs the processing of characteristic features of the input data stream in accordance with the developed methods. A multilayer hardware–software architecture of a data organization subsystem for private cloud services is developed, ensuring the integration of the specialized hashing processor with server-side and client-side components of the infrastructure. The proposed architecture makes it possible to reduce the load on central processing resources, increase performance, and ensure energy-efficient operation of the data processing subsystem.

Within the scope of the dissertation, experimental studies of the developed methods and the specialized processor were carried out as part of a prototype of a data processing subsystem for a private cloud infrastructure. The experimental results confirm an increase in hashing operation throughput, a reduction in energy consumption, and an improvement in the efficiency of data organization processes compared to conventional software-based approaches.

In the general **conclusions**, the main results of the dissertation are summarized, the achievement of the stated research objective is confirmed, and directions for further research related to the development of energy-efficient hardware–software data processing subsystems for private cloud infrastructures and specialized computing environments are outlined.

Keywords: cloud security, cloud computing, cloud technologies, private cloud services, cloud service models, critical systems, cybersecurity, information protection, hashing, hash function, byte-oriented hashing, cryptoprocessor, integrity verification, cyber-physical systems, hardware-software architecture, edge computing, IoT, FPGA.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Кисюк Д. В. Аналітичний огляд постачальників хмарних послуг. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025, №3, С. 105–111. DOI: https://doi.org/10.31891/2219-9365-2025-83-14
2. Кисюк Д. В. Актуальність проблеми впровадження хмарних сервісів для організації роботи з даними. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2025, Т.356, № 5. DOI: https://doi.org/10.31891/2307-5732-2025-357-24
3. Кисюк Д. В., Захарченко С. М. Оптимізація застосування хмарних сервісів для розподіленої обробки даних в системах колективного доступу. *Наукові праці ВНТУ*. 2025, Вип. №2, С. 66–72. DOI: https://doi.org/10.31649/2307-5376-2025-2-66-72
4. Лужецький В. А., Кисюк Д. В. Новий підхід до побудови байт-орієнтованих криптографічних геш-функцій. *Вимірювальна техніка та метрологія*. 2026, Випуск 87(1), ISTCMTM, Львів, Номер 1, С. 51-58. DOI: https://doi.org/10.23939/istcmtm2026.01.051
5. Лужецький В. А., Захарченко С. М., Кисюк Д. В. Метод байт-орієнтованого гешування з підвищеним рівнем дифузії та спеціалізований МН-процесор для приватних хмарних інфраструктур. *Вимірювальна та обчислювальна техніка в технологічних процесах.* 2026, № 85(1), С. 134-141. DOI: https://doi.org/10.31891/2219-9365-2026-85-16
6. Лужецький В. А., Савицька Л. А., Кисюк Д. В. Спеціалізований процесор для ущільнення даних. *Тези доповідей П'ятої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації»*. 2016, Вінниця : ВНТУ, С.108–110. URL: http://ir.lib.vntu.edu.ua/handle/123456789/11399
7. Лужецький В. А., Кисюк Д. В., Савицька Л. А. Метод байт-орієнтованого хешування. *Тези доповідей V Міжнародної науково-практичної конференції*

«Методи та засоби кодування, захисту й ущільнення інформації». 2016, Вінниця–Немирів. URL: http://ir.lib.vntu.edu.ua/handle/123456789/9992

8. Лужецький В. А., Кисюк Д. В. Новий підхід до побудови криптографічних хеш-функцій. *Матеріали статей П'ятої Міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія»*. 2015, Івано-Франківськ : ФОП Голіней О. М., С.149–150. URL: http://ir.lib.vntu.edu.ua/handle/123456789/14457
9. Кисюк Д. В., Захарченко С. М. Аналіз переваг використання хмарних сервісів для обробки даних у сучасних системах управління базами даних. *Матеріали LII Науково-технічної конференції підрозділів ВНТУ.* 2023, Вінниця. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-023/paper/view/18183.
10. Кисюк Д. В., Лужецький В. А. Програмний засіб для побудови хеш-значення за допомогою байт-орієнтованої обробки даних для мобільних пристроїв. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)»*. 2023, Вінниця. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/18180
11. Мазур Б. С., Кисюк Д. В. Хмарні сервіси. Обробка та збереження даних за допомогою хмарних сховищ. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)»*. 2024, Вінниця. URL: https://iq.vntu.edu.ua/method/getfile.php?fname=153063.docx&x=1
12. Кисюк Д. В., Шулдик С. С. Хмарні сервіси як інструмент управління проєктами у системах забезпечення ресурсами підприємства SAP ERP. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»*. 2025, Вінниця. URL: https://iq.vntu.edu.ua/method/getfile.php?fname=179474.pdf&x=1
13. Лужецький В. А., Кисюк Д. В. Узагальнений метод хешування байтової форми представлення інформації. *Тези доповідей Четвертої Міжнародної*

науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія». 2014, Вінниця : ВНТУ, С. 184–186. URL: http://ir.lib.vntu.edu.ua/handle/123456789/14878.

14. Лужецький В. А., Кисюк Д. В. Метод хешування даних на основі їх характеристичних ознак. *Тези доповідей II Міжнародної науково-практичної конференції «Актуальні питання забезпечення кібербезпеки та захисту інформації»*. 2014, Київ : Європейський університет, С. 100–102. URL: http://ir.lib.vntu.edu.ua/handle/123456789/15351.
15. Лужецький В. А., Кисюк Д. В., Савицька Л. А. Метод байт-орієнтованого хешування. *Тези доповідей П'ятої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації»*. 2016, Вінниця : ВНТУ, С. 37–38. URL: http://ir.lib.vntu.edu.ua/handle/123456789/9992.
16. Лужецький В. А., Кисюк Д. В., Комаров А. О. Метод та спеціалізований процесор для байт-орієнтованого хешування даних. *Тези доповідей Міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерне моделювання (ІТСМ)»*. 2016, Івано-Франківськ : ПНУ ім. В. Стефаника, С. 123–125. URL: http://itcm.pnu.edu.ua/2016/docs/Luzhetskyi.pdf.
17. Кисюк Д. В., Заруцький М. В. Перспективи впровадження штучного інтелекту в хмарних обчисленнях. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»*. 2025, Вінниця, URL: https://iq.vntu.edu.ua/method/getfile.php?fname=169386.pdf&x=1.
18. Кисюк Д. В., Червінський В. О. Архітектура приватних хмарних сервісів для зберігання та обробки даних. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»*. 2025, Вінниця, URL: https://iq.vntu.edu.ua/method/getfile.php?fname=169386.pdf&x=1.
19. Кисюк Д. В., Сірак В. О. Методи підвищення ефективності зберігання та обробки великих обсягів даних у приватних хмарних сервісах. *Матеріали*

Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)». 2026, Вінниця, URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/viewFile/27079/22199.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ПЗ – Програмне забезпечення

АП – Апаратне прискорення

АПА – Апаратно-програмна архітектура

ХС – Хмарні сервіси

ПХС – Приватні хмарні сервіси

ХІ – Хмарна інфраструктура

ГФ – Геш-функція

БОГ – Байт-орієнтоване гешування

ГЗ – Геш-значення

КЦД – Контроль цілісності даних

ДД – Дедуплікація даних

СП – Спеціалізований процесор

МН-процесор – Процесор байт-орієнтованого гешування

CPU – Central Processing Unit

FPGA – Field-Programmable Gate Array

ASIC – Application-Specific Integrated Circuit

ОЗП – Оперативна пам’ять

ПЗП – Постійний запам’ятовуючий пристрій

ПС – Пропускна здатність

ЕЕ – Енергоефективність

РОП – Ресурсно-обмежені пристрої

HW – Hardware

SW – Software

HW/SW – Апаратно-програмна реалізація

I/O – Input / Output

DMA – Direct Memory Access

ВСТУП

Актуальність теми дослідження. Сучасний розвиток інформаційних технологій характеризується переходом від локальних обчислювальних систем до розподілених хмарних інфраструктур, які забезпечують зберігання, обробку та передачу великих обсягів даних. Концепція хмарних обчислень, що передбачає надання обчислювальних ресурсів як сервісу, стала основою для побудови сучасних інформаційних систем різного призначення — від корпоративних і державних до критично важливих та відомчих [1–4]. Формалізоване визначення хмарних обчислень, запропоноване NIST, підкреслює такі ключові характеристики, як еластичність ресурсів, мережевий доступ та вимірюваність сервісів, що безпосередньо впливають на архітектуру систем обробки даних [2].

Подальший розвиток хмарних технологій зумовив активне впровадження приватних хмарних середовищ, які дозволяють організаціям зберігати контроль над інфраструктурою та даними, одночасно використовуючи переваги хмарної моделі [3], [27]. У таких системах особливе значення має організація роботи з даними, оскільки ефективність, надійність і безпека обробки інформації визначають загальну продуктивність та стійкість інформаційної інфраструктури [30], [31].

З позицій комп’ютерної інженерії ключовим викликом є забезпечення цілісності та узгодженості даних у розподілених середовищах. Архітектура хмарних систем передбачає використання багатьох обчислювальних вузлів, сховищ і мережевих компонентів, що ускладнює контроль коректності даних на всіх етапах їх життєвого циклу [81], [82]. Порушення цілісності даних у таких системах може бути зумовлене як збоєм апаратних компонентів, так і помилками програмного забезпечення або несанкціонованими впливами [12], [13].

Одним із базових понять криптографії та інженерних механізмів забезпечення цілісності даних є використання геш-перетворень. Геш-функції застосовуються для контролю цілісності інформації, побудови структур даних у розподілених системах, реалізації основних підходів кібербезпеки, механізмів дедуплікації, журналювання та синхронізації станів [11], [14], [16]. У хмарних

середовищах гешування є фундаментальною операцією, що використовується як на програмному, так і на апаратному рівнях [59], [60], [83].

Водночас більшість стандартизованих криптографічних геш-функцій (MD5, SHA-1, SHA-2, SHA-3) були розроблені з урахуванням універсальних обчислювальних платформ і не орієнтовані на ефективну реалізацію в умовах обмежених апаратних ресурсів або спеціалізованих обчислювальних модулів [16], [18], [69]. Дослідження архітектур комп'ютерних систем показують, що оптимізація алгоритмів під конкретну архітектуру процесора або спеціалізованого прискорювача може суттєво підвищити продуктивність і зменшити енергоспоживання [20], [21].

Для приватних хмарних сервісів, особливо у відомчих та критичних інформаційних системах, характерним є використання ресурсно-обмежених обчислювальних вузлів, спеціалізованих серверів або апаратних модулів безпеки [32], [74], [76]. У таких умовах виникає необхідність застосування методів гешування, адаптованих до особливостей апаратної реалізації та потокової обробки даних.

Перспективним напрямом у цьому контексті є байт-орієнтоване гешування, яке передбачає обробку даних на рівні байтів без складних бітових перестановок. Такий підхід дозволяє спростити логіку обчислень, підвищити рівень паралелізму та забезпечити ефективну апаратну реалізацію геш-алгоритмів [39], [40], [42]. Праці, присвячені розробці байт-орієнтованих методів гешування, демонструють їхню придатність для використання у спеціалізованих процесорах та вбудованих системах [41], [44].

Важливою інженерною задачею є інтеграція методів байт-орієнтованого гешування у архітектуру приватних хмарних сервісів. Це вимагає узгодження алгоритмічних рішень з апаратно-програмною архітектурою системи, включаючи рівні зберігання даних, обчислювальні вузли та підсистеми, механізми взаємодії між ними [35], [36], [38]. Недостатня увага до архітектурних аспектів може знижувати ефективність навіть криптографічно стійких алгоритмів [54], [56].

Аналіз сучасних наукових досліджень свідчить, що значна частина робіт у галузі гешування зосереджена на формальних властивостях безпеки та криптостійкості [61–65], тоді як питання апаратної реалізації, оптимізації під конкретні архітектури та використання у приватних хмарних середовищах залишаються недостатньо опрацьованими. Це створює науково-технічний розрив між теоретичними криптографічними моделями та практичними інженерними рішеннями для систем обробки даних [87–90].

Таким чином, актуальність даного дослідження зумовлена необхідністю розробки методів організації роботи з даними у приватних хмарних сервісах, які базуються на байт-орієнтованому гешуванні та орієнтовані на ефективну апаратно-програмну реалізацію. Поєднання алгоритмічних методів гешування з архітектурними рішеннями комп'ютерної інженерії дозволяє підвищити продуктивність, надійність та масштабованість приватних хмарних систем, що визначає теоретичну і практичну значущість обраної теми дослідження.

Зв'язок з науковими програмами, планами, темами. Дисертація виконувалась відповідно до тематичних планів наукових досліджень кафедр обчислювальної техніки та захисту інформації Вінницького національного технічного університету.

Мета та задачі дослідження. Метою дослідження є підвищення ефективності організації роботи з даними у приватних хмарних сервісах шляхом розробки байт-орієнтованих методів гешування та засобів їх реалізації, а також моделей та методів апаратно-програмної інтеграції цих засобів у підсистеми зберігання й обробки даних.

Для досягнення поставленої мети необхідно розв'язати такі **задачі**:

- Проаналізувати сучасний стан методів і засобів організації роботи з даними у хмарних сервісах;
- розробити модель процесів роботи з файлами та об`єктами баз даних у приватній хмарній інфраструктурі;
- розробити методи гешування даних для приватних хмарних сервісів;

- розробити функціональну та структурну моделі спеціалізованого процесора байт-орієнтованого гешування з обмеженим енергоспоживанням;

- розробити архітектуру підсистеми організації роботи з даними у приватних хмарних сервісах;

- розробити методику комплексного оцінювання ефективності організації роботи з даними у приватних хмарних сервісах;

- провести експериментальні дослідження розроблених методів та програмно-апаратних засобів.

Об’єктом дослідження є процеси організації роботи з даними у приватних хмарних інформаційних системах, зокрема системах оборонного призначення, що функціонують в умовах обмежених обчислювальних та енергетичних ресурсів.

Предметом дослідження є моделі, методи, апаратно-програмні засоби байт-орієнтованого гешування даних та архітектурні рішення підсистем організації роботи з файлами та об`єктами баз даних у приватних хмарних сервісах.

Методи дослідження. Для вирішення поставлених задач застосовувалися методи: системного аналізу та узагальнення науково-технічних джерел — для дослідження сучасного стану та тенденцій розвитку хмарних обчислень і механізмів забезпечення цілісності даних у розподілених середовищах [1–4], [54], [58]; теорії розподілених систем і хмарних архітектур — для обґрунтування вимог до організації роботи з даними у приватних хмарних сервісах та формування цільової моделі їх функціонування [2], [3], [81], [82]; методи криптографічного аналізу та теорії криптографічних примітивів — для формалізації вимог до геш-перетворень, аналізу стійкості до колізій і атак на прообраз, а також визначення обмежень відомих підходів до побудови геш-функцій [14–16], [61–65], [68], [69]; методи математичного моделювання та алгоритмічного синтезу — для побудови та обґрунтування байт-орієнтованих методів гешування, визначення структури перетворень і параметрів алгоритмів [16], [19], [39], [40], [42], [43]; методи структурного та функціонального моделювання — для розроблення апаратно-програмної організації процесів гешування у складі приватних хмарних сервісів, а також для опису взаємодії модулів обробки, пам’яті та введення-виведення [20],

[81], [82]; методи комп’ютерної інженерії та проектування цифрових пристроїв — для формування архітектури спеціалізованого процесора (апаратного прискорювача) байт-орієнтованого гешування, вибору структурних елементів та оцінювання апаратних витрат [20–22], [41], [44]; методи експериментального дослідження та обчислювального експерименту — для перевірки працездатності запропонованих методів, порівняння їх продуктивності та ресурсних витрат із відомими рішеннями, а також для підтвердження ефективності у сценаріях хмарного зберігання й обробки даних [36–38], [45], [50], [59], [60]; методи аналізу стандартів та нормативних документів у сфері інформаційної безпеки — для узгодження вимог до цілісності та захисту даних у приватних хмарних середовищах і формування обмежень на практичну реалізацію рішень [23], [24], [76–79]; методи порівняльного аналізу та техніко-економічного оцінювання — для обґрунтування доцільності застосування спеціалізованих апаратних засобів гешування у складі приватних хмарних сервісів та визначення напрямів їх масштабування [3], [20], [54].

Наукова новизна отриманих результатів.

- вперше запропоновано метод неітеративного байт-орієнтованого гешування даних, який на відміну від відомих передбачає використання таких характеристичних ознак повідомлення, як кількість байтів певного значення та сума номерів їх позицій, кожна з яких ущільнюється окремо, а результат гешування отримується множенням ущільнених характеристичних ознак, що забезпечує пришвидшення процесу гешування та зменшення апаратної складності засобу гешування;

- вперше запропоновано метод неітеративного байт-орієнтованого гешування даних, який на відміну від відомих передбачає використання таких характеристичних ознак повідомлення, як кількість байтів певного значення та сума номерів їх позицій з додаванням криптографічної солі, які змішуються шляхом операції множення, а результат гешування отримується ущільненням добутку, що забезпечує підвищення рівня криптографічної стійкості та пришвидшення процесу гешування;

- вперше запропоновано структурну модель спеціалізованого процесора байт-орієнтованого гешування, яка на відміну від відомих містить структурні елементи, що формують характеристичні ознаки даних та їх ущільнення, що забезпечує можливість побудови малоресурсних апаратних засобів гешування;

- удосконалено метод організації роботи з даними у приватних хмарних сервісах шляхом інтеграції апаратно-програмних засобів байт-орієнтованого гешування в основні процеси життєвого циклу даних, що забезпечує контроль цілісності даних та підвищення ефективності використання обчислювальних ресурсів;

- удосконалено структурну модель багатошарової апаратно-програмної підсистеми організації роботи з даними у приватній хмарі, шляхом уведення шару сервісу гешування та додавання засобів на апаратний рівень та рівень операційної системи, що спрощує масштабування такої підсистеми без пропорційного збільшення енергоспоживання та апаратних витрат.

Практичне значення отриманих результатів:

- запропоновані моделі та методи можуть бути використанні у приватних хмарних середовищах оборонного призначення для забезпечення контролю цілісності даних, зменшення обчислювальних витрат при обробленні великих обсягів даних та функціонування таких систем в умовах обмежених ресурсів та підвищених вимог до надійності;

- запропоновані методики комплексної оцінки ефективності обчислювальних систем можуть бути використанні для формування рекомендацій щодо модифікації існуючих та побудови перспективних інформаційних систем;

- практичне застосування результатів дослідження можливе у складі сервісів керування даними, підсистем зберігання та маршрутизації інформаційних потоків, механізмів контролю цілісності та інформаційно-аналітичних систем військового призначення.

Результати роботи впроваджені на базі 9 Центру захисту інформації та кібербезпеки в інформаційно-телекомунікаційних системах, Центру масштабування технологічних рішень (військова частина А4753), Військова

частина А4717, а також у навчальний процес кафедри обчислювальної техніки, кафедри захисту інформації Вінницького національного технічного університету та кафедри РЕС КП ПС факультету АСУ та НЗПА Харківського національного університету повітряних сил.

Особистий внесок здобувача. Основні наукові результати та висновки дисертаційної роботи отримані автором самостійно та викладені у 5 наукових працях, що були написані у співавторстві. Зокрема, у працях, які написані у співавторстві, дисертантові належать такі напрацювання: [36] – аналіз сучасних підходів щодо організації роботи з даними у хмарних сервісах; [37] – аналіз актуальності проблеми впровадження хмарних сервісів для організації роботи з даними; [38] – запропоновано удосконалений метод організації роботи з даними у приватних хмарних сервісах та удосконалено структурну модель багатошарової апаратно-програмної підсистеми організації роботи з даними; [39] – запропоновано методи неітеративного байт-орієнтованого гешування даних; [108] – запропоновано структурну модель спеціалізованого процесора байт-орієнтованого гешування.

Постановка завдань та їх обговорення здійснено під керівництвом д.т.н., проф., зав. кафедри захисту інформації Лужецького В. А. та к.т.н., проф. кафедри обчислювальної техніки Захарченко С. М.

Апробація результатів дослідження.

Результати дисертаційного дослідження апробовано на міжнародних наукових та науково-практичних конференціях:

- Четверта Міжнародна науково-практична конференція «Інформаційні технології та комп'ютерна інженерія», м. Вінниця.
- П'ята Міжнародна науково-практична конференція «Інформаційні технології та комп'ютерна інженерія», м. Івано-Франківськ.
- П'ята Міжнародна науково-практична конференція «Методи та засоби кодування, захисту й ущільнення інформації», м. Вінниця–Немирів.
- Друга Міжнародна науково-практична конференція «Актуальні питання забезпечення кібербезпеки та захисту інформації», м. Київ.

- Міжнародна науково-практична конференція «Інформаційні технології та комп'ютерне моделювання (ІТСМ)», м. Івано-Франківськ.
- LII Науково-технічна конференція підрозділів Вінницького національного технічного університету, м. Вінниця.
- Всеукраїнська науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи», м. Вінниця.

Публікації. За матеріалами дисертаційної роботи опубліковано 19 наукових праць, з яких 5 статей у наукових фахових виданнях України; 14 – у матеріалах і тезах конференцій.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, основної частини, що містить 3 розділи, висновків, списку використаних джерел та додатків. Основний зміст викладено на 162 сторінках друкованого тексту, містить 32 рисунки, 21 таблицю. Загальний обсяг роботи - 179 сторінок.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ЩОДО ОРГАНІЗАЦІЇ РОБОТИ З ДАНИМИ У ХМАРНИХ СЕРВІСАХ

1.1 Огляд хмарних мереж та сервісів

1.1.1 Загальні види хмарних мереж та їх особливості

Хмарні обчислення як парадигма організації обчислювальних ресурсів сформувалися на перетині ідей розподілених систем, мережевих сервісів та віртуалізації. Ще у фундаментальних роботах з теорії розподілених систем підкреслювалася важливість абстрагування ресурсів і віддаленого доступу до них [82; 83], а згодом ці підходи були трансформовані у концепцію хмарної моделі [1, 2].

Відповідно до визначення NIST [2], хмарні обчислення характеризуються такими базовими властивостями: самообслуговування на вимогу, універсальний мережевий доступ, об’єднання ресурсів, еластичність та вимірюваність споживання. Залежно від способу розгортання та характеру доступу виділяють кілька основних видів хмарних сервісів.

Дослідження літературних джерел засвідчило, що тема розробки хмарних сервісів – це порівняно новий напрямок і потребує додаткових досліджень з метою розширення можливостей побудови хмарних сховищ та їхньої успішної практичної реалізації не лише в бізнесі, а й інших не менш важливих сферах людської діяльності. Хмарні технології являють собою сукупність апаратного та програмного забезпечення, що забезпечує обробку та виконання клієнтських заявок. Іншими словами, це певний метод зберігання і надання даних кінцевому користувачеві. В науковому апараті поряд з категорією хмарні технології, хмарні сервіси, хмарна інфраструктура, функціонує поняття хмарне сховище даних. **Хмарне сховище даних** (англ. cloud storage) – це модель онлайн-сховища, дані в якому зберігаються на множинних серверах, розподілених у мережі. Сервери надаються клієнтам, як правило, третьою стороною. Інформацію про кількість або

будь-які елементи внутрішньої структури серверів клієнту зазвичай не видно, на відміну від методу зберігання даних на власних виділених серверах, що купуються або орендуються спеціально для подібних цілей.

У зв'язку з розробкою інформаційних технологій, пріоритетним інструментом більшості компаній та організацій у боротьбі за своє місце є інформація. Оскільки майже неможливо прийняти адекватне, компетентне управлінське чи бізнес рішення, яке гарантує успіх без володіння інформацією з цього питання. Щодня в будь-якій компанії чи організації з'являється величезна кількість інформації, яка потребує використання для прийняття рішень, інформація має можливість накопичуватися в різних джерелах та послугах, що вже буде слугувати досвідом підприємства в майбутньому.

Тобто поступово збільшується обсяг інформації до таких розмірів, що існує нагальна потреба в покращенні обчислювальних можливостей для зберігання та роботи з великою кількістю даних. Це часто призводить до різних видів витрат, пов'язаних з грошовими, тимчасовими, людськими та іншими різноманітними ресурсами. У зв'язку з цим теоретики та практики комп'ютерної інженерії безперервно ведуть пошук альтернативних рішень для зменшення великої кількості витрат на зберігання та обробку великих даних.

Хмарні обчислення є базовою моделлю організації обчислювальних ресурсів, яка забезпечує еластичний доступ до інфраструктури, платформ і прикладних сервісів. Для задач загального призначення це насамперед дає економію капітальних витрат і масштабованість. Водночас у відомчих та військових інформаційних системах пріоритет зміщується від економічних переваг до керованості інфраструктури, контролю цілісності даних, доменної ізоляції та прогнозованої продуктивності.

Тобто, хмарні сервіси є альтернативою великій кількості процесів обробки інформаційних даних, які підприємства намагаються виконувати самостійно, що тягне за собою затратні ресурси. Взагалі, характеристики сьогоднішніх хмарних обчислень, визначаються в США стандарти наступним чином: *самостійне*

обслуговування на вимогу, універсальний доступ до мережі, поєднання ресурсів, еластичність та облік споживання.

Хмарні технології є базою сучасних інформаційних систем у державному та корпоративному секторах. [27, 99]. В наукових доробках В. Гончаренко, В. Семенюк, Н. Коршун, О. Марущак і О. Руденко, І. Кравченко, С. Пащенко, С. Дьяків проаналізовано впровадження хмар [100-105]. О. Мироненко присвятила свої наукові розвідки питанню цифрової трансформації адміністрації [96]. Ю. Зінченко дослідив використання хмар для місцевих громад та організацій [94].

Аналіз наукових праць показує, що використання хмарних сховищ можливе не тільки в бізнесі, а й в інших напрямках діяльності сучасного користувача Інтернет. На необхідності впровадження хмарних сервісів у медичне, освітнє а на даний момент і у військове середовище акцентують увагу вітчизняні та зарубіжні дослідники.

Специфіка використання кожної з моделей визначається способом використання, так [2, 93, 94]:

1) «приватна хмара», призначена для використання всередині окремо взятої організації. Ця модель характеризується тим, що всі витрати, пов’язані з адаптацією та обслуговуванням, організація бере на себе. Позитивним моментом є те, що для цієї моделі розгортання характерне значне зниження ризиків «витоку» даних;

2) «публічна хмара», використовується для розгортання серед безлічі організацій. Характерними особливостями цієї моделі є нижчі тарифні ціни на її використання порівняно з моделлю «приватна хмара», але водночас і невисока безпека даних, що зберігаються на такій хмарі;

3) «гібридна хмара» – модель розгортання, що є комбінацією приватної та публічної хмар, які мають між собою тісний взаємозв’язок, що дає змогу знайти баланс між захистом конфіденційних даних і вартістю утримання хмар

4) «хмари спільноти (community cloud)» - передбачає використання інфраструктури групою організацій із подібними вимогами до безпеки.

Наразі найперспективнішими моделями розгортання є публічна та гібридна моделі хмарних технологій з огляду на їхню більшу економічну доцільність, порівняно з приватною хмарою, і зменшуваний ризик втрати даних [99].

Проте для сектору безпеки й оборони ключовою є саме приватна модель, оскільки вона дозволяє реалізувати жорстке розмежування доменів довіри та ізоляцію інформаційних потоків відповідно до нормативних вимог [25, 26, 77]. У цьому контексті хмарна інфраструктура розглядається не лише як економічний інструмент, а як критичний компонент національної кіберстійкості [73, 74].

Для хмарних сервісів ключовими перевагами є централізація зберігання, підтримка резервування, масштабованість і спрощення спільної роботи з даними. Проте для приватних і військових хмар визначальними є не загальні користувацькі переваги, а можливість контролювати архітектуру, довірені контури, політики доступу та механізми контролю цілісності. [2, 3].

1.1.2 Моделі надання хмарних сервісів

Окрім класифікації за способом розгортання, хмарні сервіси поділяються за моделями надання функціональності. Базовими моделями є Software as a Service (SaaS), Platform as a Service (PaaS) та Infrastructure as a Service (IaaS) [3, 4].

Software as a Service (SaaS) – це модель, у якій користувач отримує доступ до готового програмного забезпечення через мережу Інтернет без необхідності встановлення та адміністрування програмних компонентів на власних ресурсах. Програмний продукт розміщується в інфраструктурі провайдера, а доступ до нього здійснюється через веб-інтерфейс або тонкий клієнт. У цій моделі відповідальність за оновлення, резервування та захист програмного забезпечення покладається на провайдера. Прикладами можуть бути системи електронної пошти, офісні пакети, системи управління проєктами та інші прикладні сервіси [3, 30].

Platform as a Service (PaaS) передбачає надання програмної платформи для розроблення, тестування та розгортання додатків. Користувач отримує доступ до середовища виконання, баз даних та інструментів розробки, проте не контролює фізичну інфраструктуру. Модель PaaS орієнтована переважно на розробників програмного забезпечення, які створюють додатки з використанням ресурсів

платформи. Прикладами такого програмного забезпечення є Feng Office Community Edition, Simple Groupware, Zarafa та інші [3].

Програмне забезпечення як послуга (**Infrastructure as a Service - IaaS)** – це модель, що забезпечує доступ до віртуалізованих обчислювальних ресурсів: серверів, сховищ даних, мережевих компонентів. Користувач має можливість самостійно конфігурувати віртуальні машини, встановлювати операційні системи та програмне забезпечення. У рамках IaaS реалізується найбільший рівень контролю над середовищем виконання серед базових моделей [1, 3].

Поряд із базовими моделями існують похідні або спеціалізовані моделі. Обладнання (обчислювальна потужність) як послуга **Hardware as a Service (HaaS)** передбачає надання обчислювальної потужності та апаратних ресурсів як сервісу, що особливо актуально для високопродуктивних обчислень. Послуги, як правило, пропонуються як еквівалент реальних обчислювальних систем, таких як сервери, суперкомп'ютери. Над програмою реалізації цієї ідеї повністю або частково працюють такі сервіси як OpenVZ, FreeVPS, Linux-Vserver, Apache Hama, Glusterfs Open Source Project, а також Moose File System (Moosefs).

Комунікація як служба (комунікації як послуга, **Communication as a Service - CaAs**) – це рішення комунікації, побудоване в хмарі для підприємства, яке надає мовленнєвий сигнал в Інтернеті або в будь яких інших IP-мережах (VoIP), обмін миттєвими повідомленнями (IM), відеоконференціями. Модель CAAS дозволяє діловим клієнтам вибірково розгортати комунікації та послуги на основі оплати послуг вчасно для послуг, що використовуються. Проекти Foss, такі як Ekiga, ILBC, Speex, тісно пов'язані з цією сферою використання хмарних послуг.

Моніторинг як сервіс (**Monitoring as a Service - MaaS**) – є хмарним програмним забезпеченням моніторингу та безпеки. Такими OpenSource-рішеннями сьогодні є Ganglia, Zabbix, Hyperic HQ

Інфраструктура як послуга (**Infrastructure as a Service** - IaaS) – це надання комп'ютерної інфраструктури (зазвичай у формі віртуалізації) як послуги, заснованої на концепції хмарних обчислень. В загальному розумінні, IaaS – це

комбінація SaaS, HaaS, тому що вона включає в себе як, як правило, в множині, і CaaS і іноді MaaS з метою об'єднання і моніторингу всієї системи, і тому в основному використовується підприємствами. Вільними реалізаціями цієї концепції є Eucalyptus, OpenNebula, OpenStack, Nimbus тощо.

Платформа як сервіс (**Platform as a Service - PaaS***)* – надання програмній платформі та інструментам певних характеристик, необхідних для розробки, тестування, розгортання, підтримки різних додатків. Вона також включає готові до використання хмарні сервіси, які разом утворюють програмну платформу. Яскравими прикладами зі світу OpenSource в даний час є Xen Cloud Platform, Cloud Foundry, Apache Hadoop, Apache Hive та інші.

Комп'ютер (virtual desktop) як послуга (**Desktop as a Service - DaaS**) – надання віртуального комп'ютера, який кожен користувач може індивідуально налаштувати під свої завдання. Таким чином, користувач, що приходить на роботу, просто вводить свої дані (зазвичай логін і пароль) і може працювати, використовуючи обчислювальні потужності стороннього сервера, завдяки технологіям віртуалізації, а не свого персонального комп'ютера.

Робочий простір як послуга (**Workspace as a Service - WaaS**) – це надання набору SaaS, призначеного для створення робочого середовища. На відміну від DaaS, у цьому випадку користувач отримує доступ лише до програмного забезпечення, тоді як усі розрахунки відбуваються безпосередньо на його персональному комп'ютері. Насправді ця категорія є гібридом SaaS і PaaS, оскільки, на відміну від останньої, це платформа, спрямована не на розробку та тестування програмного забезпечення, а на офісну роботу. На даний момент реалізація цієї технології забезпечується в основному різними великими компаніями, такими як Google (https://www.google.com) та Microsoft (https://www.microsoft.com), і в основному представлена за допомогою закритих джерел рішень.

Все як послуга (**Environment as a Service** - EaaS) – це концептуальна модель, яка включає в себе елементи всіх перерахованих рішень. Вона в даний час не повністю реалізована, але, припускаємо ідеально підходить для великих

хмарних компаній, таких як Google (https://www.google.com) і Microsoft (https://www.microsoft.com).

Проте ключовими моделями надання хмарних послуг залишаються три: IaaS, PaaS та SaaS, які займають лідерські позиції у наданні послуг для провідних компаній світу наведено на рис. 1.1.

Рисунок 1.1 - Найпопулярніші моделі обслуговування хмарних сервісів

Аналіз ринку надання хмарних послуг свідчить про те, що практично жодна компанія нині не обходиться без IT-інфраструктури – навіть невеликій фірмі потрібні сервери для зберігання баз даних або інструменти, що об’єднують комп’ютери співробітників у загальну мережу. Компанія чи підприємство може закупити сервери і налаштувати цю інфраструктуру у себе, але це досить довго і дорого. Тому дедалі частіше компанії орендують хмарні сервіси. Названі лідери ринку хмарних послуг відрізняються сукупністю можливостей організації зберігання даних та залученістю замовника в процес створення хмарної інфраструктури підприємства наведено на рис.1.2.

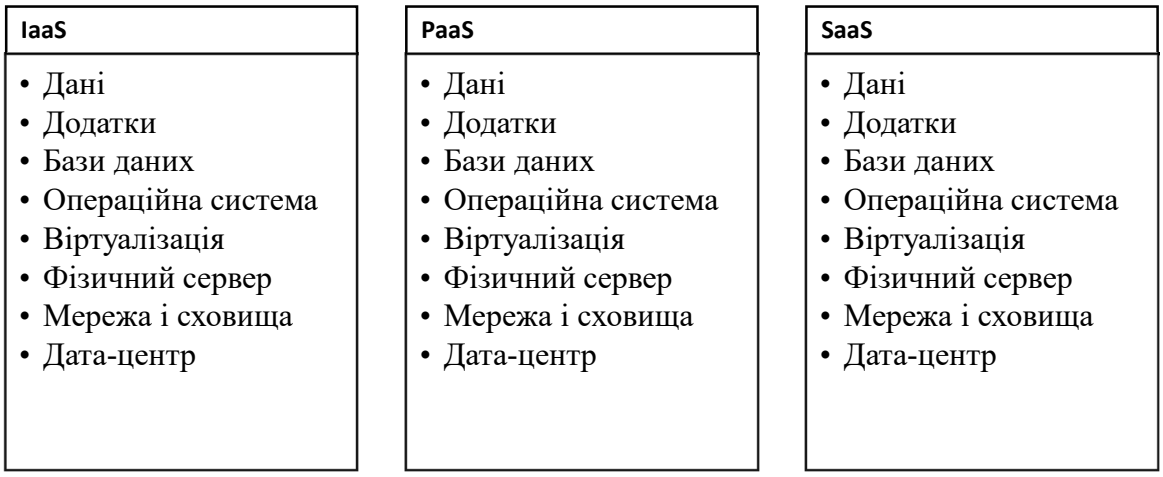

Рисунок 1.2 - Спільні та відмінні риси моделей IaaS і PaaS та SaaS у розгортанні хмарної інфраструктури (примітка: зелене маркування – забезпечує замовник: синє маркування – забезпечує провайдер)

Характеристики хмарних моделей наведені у табл. 1.1.

Відповідно до визначення хмарних обчислень, наведеного National Institute of Standards and Technology у документі SP 800-145, хмарна модель передбачає забезпечення доступу до конфігурованих обчислювальних ресурсів через стандартизовані механізми управління сервісами. Водночас у нормативних рекомендаціях NIST SP 800-53 особлива увага приділяється контролю доступу, аудиту подій та забезпеченню цілісності інформації в інформаційних системах підвищеної критичності.

Для приватних, зокрема військових, хмарних інфраструктур визначальною є не лише сервісна модель (SaaS, PaaS, IaaS), а насамперед архітектура доменів довіри, політики контролю доступу, механізми аудиту та гарантії цілісності даних. Функціонування сучасних кіберфізичних систем супроводжується безперервним генеруванням телеметричних, керуючих та подієвих даних, що потребує гарантованої цілісності, автентичності та оперативної обробки у хмарній інфраструктурі.

Таблиця 1.1 - Характеристики моделей IaaS і PaaS та SaaS у розгортанні хмарної інфраструктури

Тип	**Основна парадигма**	**Характеристики**	**Переваги**	**Недоліки/ризики**
IaaS	Інфраструктура як актив	Зазвичай не залежить від платформи; витрати на інфраструктуру розділяються і, отже, знижуються; угода SLA; оплата за фактом використання; автоматичне масштабування	Зниження капіталовкладень в апаратне забезпечення і трудові ресурси; зниження ризику втрати інвестицій; низький поріг впровадження; плавне автоматичне масштабування	Бізнес-ефективність і продуктивність дуже залежать від можливостей постачальника; вимагає нових/інших підходів до заходів безпеки
PaaS	Купівля ліцензії	Споживає інфраструктуру хмари; забезпечує методи динамічного управління проектами	Поступове розгортання версій	Вимагає нових/ інших заходів безпеки
SaaS	Програмне забезпечення як актив (бізнесу та споживача)	Угода SLA; користувацький інтерфейс, що надається додатками тонких клієнтів; компоненти хмари; взаємодія посередників.	Зниження капіталовкладень в апаратне забезпечення і трудові ресурси; зниження ризику втрати інвестицій; плавне ітеративне оновлення	Вимагає нових/ інших заходів безпеки

1.1.3 Тенденції розвитку хмарних технологій

Світовий ринок хмарних сервісів демонструє стабільне зростання, а хмарна модель стала стандартом для масштабованої обробки та зберігання даних. Для цієї роботи важливо не стільки ринкове зростання, скільки зміщення фокусу від універсальних публічних платформ до контрольованих приватних середовищ, у яких критичними стають питання цілісності, ізоляції та ресурсної ефективності. У результаті хмарні технології є найбільш перспективним напрямком удосконалення,

а саме лише на найближчі 5-7 років значна частина наявних інформаційних систем перейде у хмари наведено на рис. 1.3.

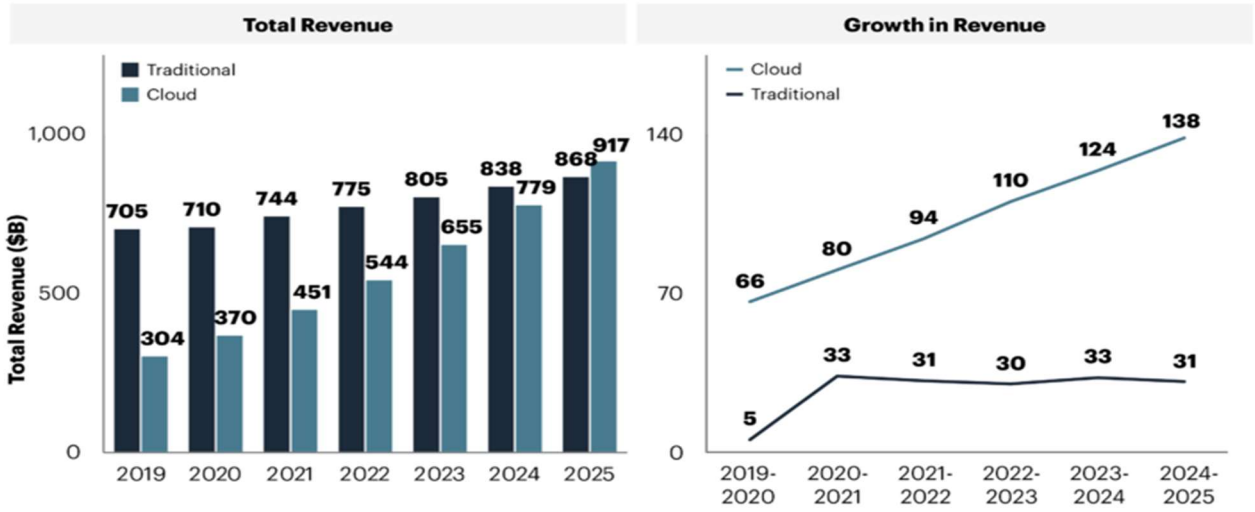

Рисунок 1.3 - Ріст світової частки зберігання та обробки даних за допомогою традиційних та хмарних технологій [75; 78]

Графічні матеріали, наведені в аналітичних звітах, ілюструють, що збільшення локальної інфраструктури потребує непропорційно більших капіталовкладень, тоді як хмарні моделі дозволяють масштабувати ресурси без суттєвого зростання витрат. Головною перевагою хмарних обчислень в обробці та зберіганні інформації є можливість масштабування, що збільшує загальну обчислювальну потужність або збережену інформацію, яка залежить від матеріальних затрат на її створення наведено на рис. 1.4.

Водночас сучасні дослідження виходять за межі економічного аспекту. У працях, присвячених інформаційній безпеці хмарних середовищ [55; 58; 59], наголошується на необхідності оптимізації обробки даних, посилення механізмів ізоляції та контролю доступу. ENISA та NIST розробили рекомендації щодо впровадження принципів Zero Trust та багаторівневого моніторингу [75; 78].

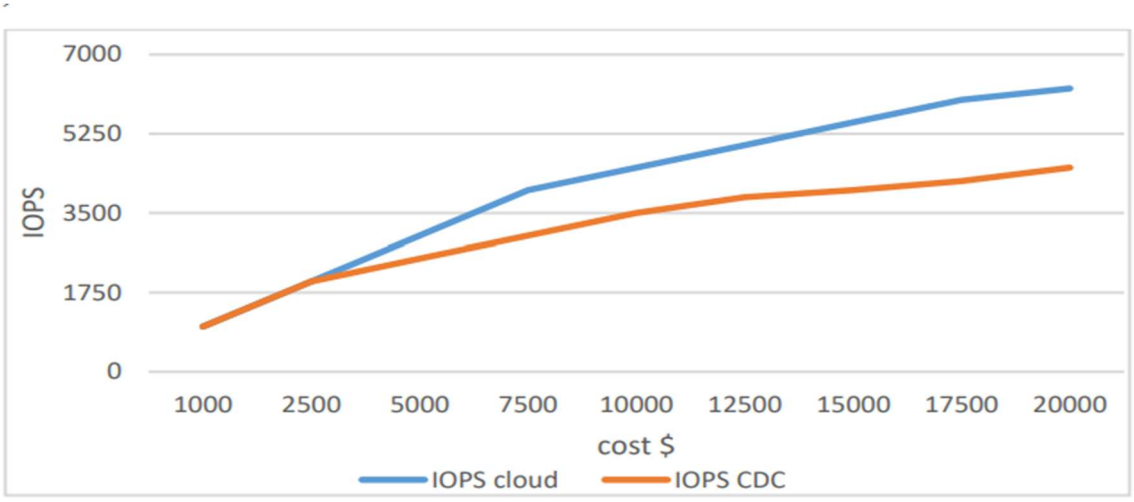

Рисунок 1.4 - Залежність обчислювальної потужності від вартості обладнання [1]

1.2 Особливості приватних військових хмар

У секторі безпеки та оборони приватні хмарні сервіси інтегруються у системи управління, аналітичні платформи та логістичні комплекси, забезпечуючи ізольовану багаторівневу інфраструктуру обробки даних.

На відміну від публічних хмар, приватні хмари функціонують у межах контрольованої інфраструктури та забезпечують повний контроль над даними й доступом. [23], [24], [76–79].

1.2.1 Вимоги до приватних хмар та їх критичної інформаційної інфраструктури

Архітектура приватної хмари зазвичай базується на багаторівневій моделі, що включає рівень фізичних ресурсів (обчислювальні вузли, мережеве обладнання, сховища), рівень віртуалізації або контейнеризації, рівень оркестрації сервісів і рівень прикладних компонентів [3], [81], [82]. У військових та відомчих середовищах ця структура доповнюється сегментацією за доменами довіри, виділенням ізольованих контурів обробки даних і використанням механізмів обов’язкового контролю доступу.

Особливістю таких хмар є поєднання централізованих дата-центрів із розподіленими периферійними (edge) вузлами та IoT пристроями, що можуть функціонувати в польових умовах або в режимі обмеженої зв’язності [81], [82].

На відміну від публічних хмар, де головним критерієм є масштабованість і економічна ефективність, у приватних та військових хмарах ключовими стають:

- керованість архітектури та прогнозованість поведінки системи;
- можливість інтеграції з відомчими інформаційними системами;
- контроль за фізичним розміщенням і конфігурацією обладнання;
- підтримка спеціалізованих апаратних модулів прискорення.

Таким чином, приватна хмара в оборонному секторі розглядається не лише як сервісна модель, а як елемент критичної цифрової інфраструктури, що повинен забезпечувати гарантований рівень доступності та цілісності даних.

Обмеження за ресурсами в приватних хмарах.

Попри те що приватна хмара може розгортатися на власному обладнанні організації, це не означає відсутності ресурсних обмежень. Навпаки, у військових сценаріях часто мають місце:

- обмежена обчислювальна потужність периферійних вузлів;
- обмеження за енергоспоживанням (особливо для мобільних і польових комплексів);
- обмежена пропускна здатність каналів зв’язку;
- необхідність роботи в умовах переривчастої або деградованої мережевої інфраструктури.

Відомо, що продуктивність обчислювальних систем зростає не лінійно щодо вартості обладнання; після певного рівня інвестицій приріст продуктивності зменшується, що робить неефективним «просте» масштабування за рахунок нарощування ресурсів [3], [20], [54]. Для приватних хмар оборонного призначення, де бюджети та логістика жорстко регламентовані, критичним є оптимальне використання наявних ресурсів, а не їх безмежне збільшення.

У такому контексті підсистеми контролю цілісності та ідентифікації даних (зокрема гешування) повинні бути оптимізовані за показниками «обчислювальна

складність на байт» та «енерговитрати на операцію». Надмірне навантаження на центральні процесори під час обробки великих потоків інформації може призводити до деградації сервісів або необхідності додаткового апаратного забезпечення, що не завжди є прийнятним. У приватних та військових хмарних сервісах, які обслуговують кіберфізичні системи, особливого значення набувають механізми контролю цілісності даних, оскільки спотворення інформації може призвести до порушення роботи фізичних процесів.

Формулювання вимог до обробки даних у приватних хмарних інфраструктурах

Приватні хмарні інфраструктури, зокрема ті, що застосовуються у відомчих та військових інформаційних системах, функціонують в умовах жорстко обмежених обчислювальних, енергетичних та мережевих ресурсів. На відміну від публічних хмарних сервісів, де масштабування досягається шляхом динамічного залучення додаткових обчислювальних вузлів, приватні хмари, як правило, розгортаються на фіксованій апаратній базі з наперед визначеними характеристиками продуктивності, пропускної здатності та енергоспоживання [1–4].

Відповідно до загальноприйнятого визначення хмарних обчислень, приватні хмарні середовища орієнтовані на обслуговування обмеженого кола користувачів і функціонують у межах контрольованої інфраструктури організації [2]. У таких умовах ефективність використання апаратних ресурсів набуває критичного значення, оскільки перевантаження центральних процесорів, підсистем пам’яті або введення-виведення безпосередньо впливає на стабільність, масштабованість і безперервність функціонування всієї системи [3, 4].

Системи обробки даних у приватних (військових) хмарних інфраструктурах зазвичай мають багаторівневу архітектуру, що включає обчислювальні вузли, підсистеми зберігання даних, внутрішні мережі та програмні сервіси керування. У межах такої архітектури центральні процесори виконують не лише прикладні обчислення, а й значну кількість допоміжних операцій, пов’язаних із забезпеченням цілісності, надійності та ефективності обробки інформації [12, 13]. Саме ці

допоміжні операції часто формують істотну частку загального обчислювального навантаження наведено на рис. 1.5.

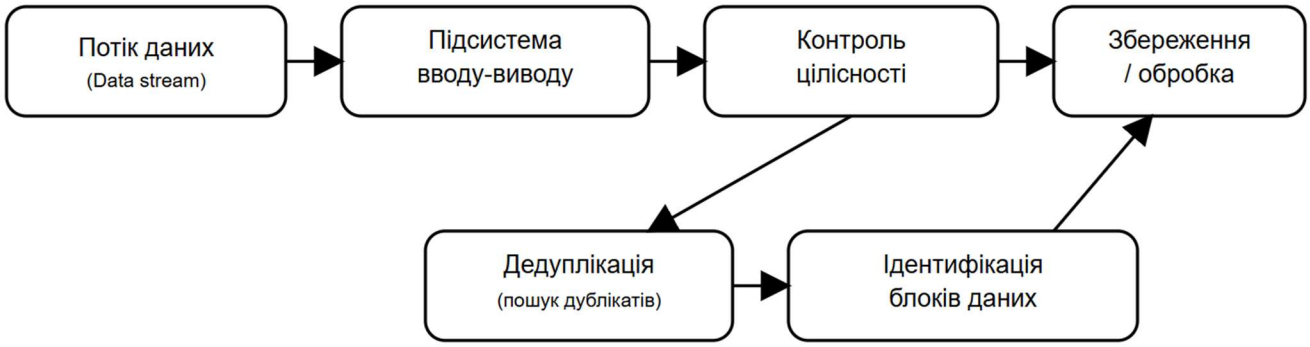

Рисунок 1.5 – Шлях обробки потоків даних у приватній хмарній інфраструктурі

Потік даних проходить етапи введення, проміжної обробки, контролю цілісності, дедуплікації та зберігання, при цьому операції гешування виконуються у складі основного шляху обробки даних.

Пропускна здатність приватних хмарних інфраструктур обмежується не лише обчислювальними можливостями процесорів, а й характеристиками підсистем пам’яті та введення-виведення. Часті доступи до оперативної пам’яті, операції читання та запису у сховища, а також передавання даних внутрішніми мережами призводять до зростання затримок і формування черг запитів [20]. У військових та відомчих системах такі затримки можуть негативно впливати на оперативність прийняття рішень і виконання прикладних задач.

Значну частину навантаження у хмарних системах формують допоміжні операції обробки даних, що виконуються для кожного блоку інформації. До них належать контроль цілісності, дедуплікація та ідентифікація блоків даних, які є невід’ємними елементами сучасних систем зберігання та обробки інформації [31, 32, 55]. Незважаючи на допоміжний характер, ці операції виконуються з високою частотою і безпосередньо впливають на загальну продуктивність системи.

Гешування є базовим механізмом реалізації зазначених контурів обробки даних. У задачах контролю цілісності геш-функції використовуються для виявлення змін інформації під час її зберігання або передавання [11, 12]. У системах

дедуплікації геш-значення слугують засобом швидкої ідентифікації однакових або подібних блоків даних, що дозволяє скоротити обсяг дубльованої інформації у сховищах [55, 56]. Крім того, у розподілених системах зберігання гешування використовується для адресації та ідентифікації блоків даних, спрощуючи механізми доступу до інформації [80, 83].

При цьому значна частина латентності у приватних хмарних інфраструктурах формується саме на рівні програмної реалізації операцій гешування. Використання складних програмних алгоритмів, орієнтованих передусім на криптографічну стійкість, супроводжується великою кількістю послідовних операцій, умовних переходів і звернень до пам’яті, що обмежує можливості паралельної обробки та знижує ефективну пропускну здатність системи [11, 14–16]. У середовищах з обмеженими ресурсами це призводить до конкуренції за процесорний час між прикладними задачами та допоміжними механізмами обробки даних наведено у табл. 1.2.

Додатковим обмежувальним фактором для приватних і військових хмарних інфраструктур є енергоспоживання обчислювальних вузлів і вимоги до контрольованого теплового режиму апаратного забезпечення [20, 23]. У таких умовах програмна реалізація обчислювально інтенсивних операцій не завжди є ефективною з точки зору загального енергетичного балансу системи, що особливо важливо для автономних або мобільних обчислювальних комплексів.

Таблиця 1.2 – Обмеження приватних хмарних інфраструктур

Група обмежень	**Характеристика**	**Вплив на систему обробки даних**
Обчислювальні	Фіксована кількість CPU-ядер, обмежена частота	Обмежена пропускна здатність програмної обробки
Пам'ять	Обмежений обсяг ОЗП та кеш-пам’яті	Зростання латентності доступу до даних
Введення-виведення	Обмежена пропускна здатність сховищ і мереж	Формування черг при потоковій обробці
Енергетичні	Обмеження енергоспоживання та теплового режиму	Неможливість використання обчислювально важких алгоритмів
Архітектурні	Централізоване виконання допоміжних операцій	Перевантаження CPU допоміжними задачами

Таким чином, специфіка обробки потоків даних у приватних (військових) хмарних інфраструктурах зумовлює архітектурну необхідність застосування апаратно-орієнтованих підходів до реалізації допоміжних операцій обробки даних. Зокрема, операції гешування, які лежать в основі контурів контролю цілісності, дедуплікації та ідентифікації блоків даних, доцільно реалізовувати з використанням спеціалізованих апаратних засобів. Це дозволяє зменшити навантаження на центральні процесори, знизити латентність обробки потоків даних та створити передумови для побудови багатошарової апаратно-програмної архітектури системи обробки даних приватної (військової) хмарної інфраструктури [20–22]. Системи, що використовуються в оборонному секторі та на об’єктах критичної інфраструктури, підпадають під дію нормативних документів і стандартів у сфері інформаційної безпеки, які регламентують вимоги до цілісності, доступності та захищеності даних [23], [24], [76–79].

Для таких систем характерні:

1. Підвищені вимоги до цілісності даних. Будь-яка несанкціонована або випадкова модифікація інформації (оперативні дані, журнали подій, результати обчислень) повинна бути своєчасно виявлена.

2. Вимоги до доказовості подій. Журнали аудиту та історія операцій мають бути захищені від несанкціонованого редагування, що передбачає використання механізмів тампер-евідентності.

3. Безперервність функціонування. Підсистеми обробки даних повинні працювати навіть за часткової втрати зв’язності або в умовах обмежених ресурсів.

4. Сумісність із засобами криптографічного захисту. Підсистеми гешування та контролю цілісності повинні інтегруватися з існуючими механізмами криптографічного захисту інформації.

Узагальнюючи викладене, можна зробити висновок, що приватні та військові хмарні сервіси формують специфічне середовище обробки даних, у якому традиційні підходи, орієнтовані на публічні багатокористувацькі платформи, потребують адаптації. Це створює передумови для розроблення спеціалізованих методів організації роботи з даними, зокрема байт-орієнтованих методів гешування

та апаратно-програмних засобів їх реалізації, що забезпечують баланс між рівнем захищеності, продуктивністю та ресурсною ефективністю.

1.2.2 Загрози цілісності та безпеці даних у приватних хмарах

Функціонування приватних хмарних середовищ, зокрема військового призначення, відбувається в умовах підвищених вимог до достовірності, контрольованості та передбачуваності обробки інформації. На відміну від публічних хмар, де основний акцент робиться на масштабованості та доступності сервісів, у приватних інфраструктурах критичним є забезпечення цілісності даних у межах ізольованих контурів довіри та під час їх взаємодії [8, 9, 10]. Зростання обсягів інформації, використання контент-адресованих моделей зберігання, механізмів дедуплікації та багаторівневих архітектур доступу формує новий клас загроз, пов’язаних не лише з несанкціонованим доступом, а й з прихованими модифікаціями, підміною блоків, порушенням консистентності або використанням застарілих версій даних [13, 14]. У таких умовах загрози цілісності набувають не меншої, а в окремих сценаріях — більшої критичності, ніж загрози конфіденційності.

Для системного аналізу цих ризиків необхідним є формування формалізованих моделей загроз, визначення сценаріїв порушення цілісності та дослідження специфічних проблем, що виникають при застосуванні дедуплікації та контент-адресованих структур [15, 16]. Саме комплексний розгляд зазначених аспектів дозволяє обґрунтувати необхідність застосування спеціалізованих механізмів контролю стану даних у приватних хмарних середовищах.

Моделі загроз у приватних хмарних середовищах.

Забезпечення цілісності та безпеки даних у приватних хмарних середовищах потребує формалізованого підходу до аналізу потенційних загроз [8, 9]. Модель загроз визначає сукупність можливих джерел небезпеки, способів реалізації атак, вразливостей системи та потенційних наслідків їх експлуатації.

У хмарних інфраструктурах загрози мають багаторівневий характер і можуть виникати на:

- фізичному рівні (апаратні ресурси);

- мережевому рівні;
- рівні гіпервізора або віртуалізації;
- рівні сервісів зберігання та обробки даних;
- рівні прикладних систем [15, 16].

У приватних (військових) хмарах особливу увагу приділяють загрозам, що впливають саме на цілісність та достовірність інформації.

Загрози цілісності включають приховані модифікації, підміну блоків, порушення консистентності та використання застарілих версій даних. Характеристики моделей загроз у приватних хмарних середовищах наведено у табл. 1.3.

Таблиця 1.3 – Характеристики моделей загроз у приватних хмарних середовищах

Модель	**Основний фокус**	**Переваги**	**Обмеження**
Модель порушника	Характеристика атакуючого	Чітке визначення ресурсів та можливостей порушника	Менш формалізований опис технічних векторів
STRIDE	Типи загроз (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege)	Систематизований підхід до аналізу сервісів	Орієнтована переважно на ПЗ та архітектуру застосунків
CIA-модель	Захист властивостей інформації	Простота та універсальність	Не деталізує механізми атак
Модель доменів довіри	Межі довіри та міжконтурна взаємодія	Ефективна для ізольованих систем	Потребує доповнення механізмами контролю стану даних
Zero Trust	Постійна перевірка доступу	Зменшує ризики інсайдерів	Не описує технічні методи забезпечення цілісності

Порівняльний аналіз моделей показує:

1. Жодна з класичних моделей загроз не визначає конкретного механізму перевірки цілісності даних.

2. Усі моделі визнають критичність порушення цілісності, але залишають реалізацію відповідних механізмів поза межами концептуального рівня.

3. Модель доменів довіри та Zero Trust найбільш адекватно відображають архітектуру приватних військових хмар, проте потребують алгоритмічної підтримки для перевірки стану даних.

4. Реалізація вимог цих моделей неможлива без застосування формалізованих механізмів контролю цілісності, зокрема гешування.

Отже, аналіз моделей загроз підтверджує, що забезпечення цілісності в приватних хмарних середовищах потребує інтеграції архітектурних підходів (Zero Trust, домени довіри) з алгоритмічними механізмами перевірки стану даних.

Порушення цілісності даних у приватних хмарах.

У роботі цілісність даних розглядається як відповідність поточного стану інформаційного об’єкта його еталонному стану. У розподілених приватних хмарах задача ускладнюється наявністю реплік, асинхронною синхронізацією та ризиком прихованих модифікацій, тому перевірка цілісності має бути компактною, формалізованою і придатною до потокової обробки [8], [9].

У розподілених хмарних середовищах, зокрема в приватних хмарах оборонного призначення, проблема ускладнюється наявністю кількох копій одного й того самого об’єкта. Дані зберігаються на різних вузлах, використовуються механізми реплікації та резервування [13], [14]. У цьому випадку цілісність набуває двох вимірів: відповідність кожної копії еталонному стану та узгодженість між самими копіями. Навіть якщо кожна окрема репліка не містить явних пошкоджень, між ними можуть виникати розбіжності внаслідок затримок синхронізації, часткових оновлень або збоїв передачі.

Практика експлуатації розподілених систем показує, що виявлення таких відхилень не може ґрунтуватися на повному побайтному порівнянні великих масивів даних, оскільки це призводить до надмірних витрат часу та ресурсів. Саме тому у хмарних сервісах застосовуються компактні контрольні мітки (зокрема геш-

значення), які дозволяють із високою ймовірністю встановити факт зміни даних без прямого порівняння всього обсягу інформації [14–16], [61–65].

Порушення цілісності у приватних хмарах можуть мати різну природу. Частина з них пов’язана з навмисними діями — редагуванням файлів, підміною конфігурацій, модифікацією журналів подій або маніпуляцією метаданими. У військових середовищах такі дії можуть здійснюватися як зовнішнім порушником, так і інсайдером. Інша група порушень має ненавмисний характер і виникає внаслідок апаратних збоїв, помилок пам’яті, некоректної передачі через канал зв’язку або дефектів файлових систем. Незважаючи на різну природу, наслідки таких змін можуть бути однаково критичними [15], [16].

Окрему категорію становлять порушення консистентності реплік, що особливо характерні для розподілених хмарних інфраструктур. Якщо одна з копій застаріла або зміни не були коректно синхронізовані, частина системи може працювати з однією версією даних, а частина — з іншою. У середовищах з обмеженою або нестабільною зв’язністю, що є типовим для польових та ізольованих сегментів військових мереж, ризик таких ситуацій істотно зростає.

У контент-адресованих та дедуплікованих сховищах виникають специфічні загрози, пов’язані з підміною блоків або навмисним створенням колізій геш-функції. Оскільки ідентифікатор блоку визначається на основі його вмісту, компрометація механізму формування контрольної мітки може призвести до повторного використання пошкодженого або підмінного блоку в різних частинах системи [60], [61].

На відміну від публічних хмар, у військових приватних середовищах наслідки порушення цілісності можуть мати стратегічний характер. Дані використовуються для планування операцій, управління ресурсами та прийняття рішень у реальному часі. Крім того, частина вузлів може функціонувати автономно, без постійного зв’язку з центральним контуром управління. Це означає, що механізми виявлення порушень повинні працювати локально, автоматизовано та з мінімальною затримкою. З огляду на масштабованість, багаторівневу ізоляцію та обмежені ресурси приватних (військових) хмар, контроль цілісності повинен бути

формалізованим, незалежним від довіри до окремого вузла та придатним до перевірки великих обсягів інформації.

Саме ці вимоги обґрунтовують необхідність використання універсальних механізмів перевірки стану даних, насамперед гешування, як базового технічного інструмента забезпечення цілісності у приватних хмарних сервісах [14–16], [61–65].

1.2.3 Задачі дедуплікації та контролю даних у приватних хмарах

Дедуплікація є одним з ключових механізмів оптимізації зберігання в хмарних сховищах, оскільки дозволяє усувати дубльовані фрагменти інформації та зменшувати витрати на зберігання і передавання даних [14, 17]. У приватних хмарах, де обсяги даних значні, а мережеві ресурси можуть бути обмеженими, дедуплікація також зменшує час реплікації та резервного копіювання. Водночас застосування дедуплікації тісно пов’язане з використанням геш-ідентифікаторів блоків, що породжує специфічні загрози та обмеження для цілісності й безпеки даних.

Дедуплікація базується на поділі даних на блоки B_kта обчисленні їх геш-ідентифікаторів $h_k = H(B_k)$. У разі збігу ідентифікаторів замість запису нового блоку використовується посилання на існуючий..

У дедуплікованому сховищі геш-ідентифікатор стає “ключем істини” щодо вмісту блоку. Тому будь-яке зниження вимог до стійкості гешування або спрощення процедури верифікації створює ризик компрометації даних навіть без прямого доступу до їх вмісту.

Це означає, що дедуплікація має застосовуватись з урахуванням політик доменів довіри. Зокрема, небажано допускати глобальну дедуплікацію між різними контурами або категоріями доступу, якщо це може породжувати міждоменні витоки метаданих.

У приватних хмарних середовищах дедуплікація практично завжди поєднується з реплікацією та резервним копіюванням, що формує додаткові проблеми узгодженості станів.

Оскільки в дедуплікованому сховищі об’єкти можуть посилатися на спільні блоки, виникає необхідність гарантувати узгодженість:

- таблиць посилань;
- лічильників посилань (reference count);
- метаданих блоків.

За наявності збою або перерваної синхронізації можливі ситуації:

- “висячі” посилання на блоки, які не репліковані;
- втрати блоків, що використовуються кількома об’єктами;
- неконсистентність між метаданими та фактичними блоками.

Таким чином, дедуплікація в приватних хмарних середовищах забезпечує значні переваги щодо оптимізації ресурсів, проте одночасно породжує специфічні проблеми, що безпосередньо впливають на цілісність та безпеку:

1. *Колізійні ризики*, які можуть призводити до прихованої підміни блоків або помилкового ототожнення різних даних.

2. *Side-channel витоки*, пов’язані з розкриттям факту наявності даних через поведінку системи дедуплікації.

3. *Складність синхронізації та консистентності* в умовах реплікації, особливо для автономних або слабко пов’язаних сегментів.

4. *Проблема ефективної перевірки*, де повна верифікація є ресурсно затратною, а спрощена перевірка лише за геш-ідентифікатором потребує підсилення структурою доказовості (Merkle-дерева, журналювання).

У підсумку, дедуплікація підвищує залежність хмарної інфраструктури від продуктивності операцій гешування, оскільки саме геш-мітки є основним інструментом ідентифікації блоків, синхронізації та контролю стану даних. Це формує прямий зв’язок між ефективністю механізмів дедуплікації та доцільністю оптимізації/апаратного прискорення гешування у приватних хмарних середовищах.

Системи характеризуються жорсткою сегментацією, доменами довіри та обов’язковим контролем доступу. [2], [3], [23], [76–79]. Для таких систем характерний потоковий режим обробки інформації: передавання файлів, реплікація, резервування, журналювання та синхронізація баз даних. Значна частина

навантаження припадає на вузли з обмеженим енергопостачанням і пропускною здатністю каналів зв’язку, що вимагає оптимізації алгоритмів зберігання та контролю даних.

Аналіз загроз засвідчив, що порушення цілісності у приватних хмарах можуть бути як навмисними, так і випадковими. До них належать підміна чи модифікація об’єктів, апаратні та мережеві збої, порушення консистентності реплік, rollback-атаки, а також ризики, пов’язані з дедуплікованими й контент-адресованими сховищами [8], [14–16], [60], [61]. У розподіленому середовищі їх виявлення ускладнюється тим, що дані зберігаються у кількох копіях і можуть оброблятися автономними вузлами без постійного централізованого контролю.

Потрібні універсальні механізми компактного представлення стану даних, які забезпечують швидку перевірку незмінності об’єктів, виявлення розбіжностей між репліками, підтримку потокової обробки, низькі обчислювальні й енергетичні витрати та інтеграцію з криптографічним захистом.

Таким механізмом є геш-функції, що формують компактну контрольну мітку стану інформаційного об’єкта [14–16], [61–65]. Вони забезпечують формалізований контроль цілісності та є основою дедуплікації, ідентифікації об’єктів, журналювання й доказовості подій. Водночас використання стандартних криптографічних геш-функцій у потокових сценаріях може створювати надмірне навантаження на процесори й пам’ять, особливо в умовах обмежених ресурсів.

Таким чином, геш-функції є необхідним компонентом організації роботи з даними у приватних хмарних сервісах, однак традиційні підходи до їх реалізації не завжди ефективні для військових і ресурсно-обмежених інфраструктур. Це обґрунтовує розроблення спеціалізованих байт-орієнтованих методів гешування та засобів їх апаратної підтримки у вигляді спеціалізованого процесора. У наступному розділі розглядаються відповідні методи та підходи до їх апаратно-програмної реалізації.

1.3 Криптографічні геш-функції та їх застосування у хмарних сервісах

1.3.1 Підходи до побудови криптографічних геш-функцій

Криптографічна геш-функція визначається як відображення $H: \{0,1\}^* \rightarrow \{0,1\}^n$, де n— фіксована довжина вихідного значення. Основною особливістю є стискання довільного за довжиною повідомлення до дайджесту сталої довжини [14, 15, 16].

Безпека геш-функції оцінюється через три базові властивості: стійкість до першого прообразу, стійкість до другого прообразу, стійкість до колізій: Імовірність випадкової колізії визначається через парадокс днів народження і становить приблизно $2^{n/2}$операцій для гешу довжини nбіт [16].

Геш-функції SHA-2 та SHA-3.

Сімейство геш-функцій SHA-2, визначене стандартом FIPS 180-4 [70], включає алгоритми SHA-224, SHA-256, SHA-384 та SHA-512. Архітектурно алгоритм SHA-256 виконує 64 раунди перетворень над 512-бітними блоками повідомлення. Кожен раунд включає бітові операції зсуву, циклічного зсуву, додавання по модулю 2^{32}та нелінійні функції Chі Maj.

Обчислювальна складність SHA-256 пропорційна кількості блоків повідомлення і визначається як:

$$T_{SHA-256}(L) = O\left(\frac{L}{512}\right) \cdot C,$$

де L— довжина повідомлення в бітах, C— константа, що відображає кількість раундів.

SHA-2 забезпечує високий рівень криптостійкості, що підтверджується відсутністю практичних атак на повну версію алгоритму [15, 66]. У хмарних сервісах він використовується для цифрових підписів, TLS-протоколів, автентифікації API та формування контрольних сум великих об’єктів.

Сімейство геш-функцій SHA-3 стандартизовано у FIPS 202 [18], базується на sponge-конструкції [19]. Внутрішній стан алгоритму має розмір 1600 біт і розділяється на дві частини: rate r та capacity c, де $r + c = 1600$. Під час фази absorb

вхідні блоки додаються за модулем 2 з частиною state, після чого застосовується нелінійна перестановка.

Sponge-конструкція дозволяє гнучко налаштовувати рівень безпеки через параметр capacity. Теоретична стійкість до колізій становить $2^{c/2}$, що для SHA3-256 відповідає 2^{128}операцій.

Перевагою SHA-3 є стійкість до атак на розширення довжини та структурна відмінність від SHA-2, що забезпечує криптографічну диверсифікацію [16].

У табл. 1.4 наведено порівняння найбільш уживаних представників сімейств SHA-2 (SHA-256/512) та SHA-3 (SHA3-256/512) у контексті застосування в хмарних сервісах.

Традиційні методи гешування, зокрема криптографічні геш-функції, широко застосовуються у сучасних інформаційних системах для забезпечення цілісності та безпеки даних. Однак їх використання у системах обробки даних приватних (військових) хмарних інфраструктур має низку обмежень, пов’язаних із особливостями апаратної реалізації та експлуатації в умовах обмежених ресурсів. Ці обмеження стають особливо критичними при обробці великих потоків даних у режимі, наближеному до реального часу.

Більшість сучасних криптографічних геш-функцій проєктуються з орієнтацією на досягнення високого рівня криптостійкості, зокрема стійкості до колізій, прообразів і атак другого прообразу [11, 14–16]. Для цього використовуються складні структури перетворень, багатораундові схеми та нелінійні операції, які забезпечують необхідний рівень дифузії та плутанини.

Класичним прикладом ітеративної архітектури гешування є схема Меркля–Дамгарда, у якій кожен блок даних обробляється з використанням попереднього внутрішнього стану, що обмежує можливості паралельної обробки та ускладнює апаратну реалізацію.

Таблиця 1.4 – Порівняльні характеристики SHA-2 та SHA-3 для застосувань у хмарних сервісах

Критерій	**SHA-2 (SHA-256/512)**	**SHA-3 (SHA3-256/512)**	**Застосування для хмарних сервісів**
Стандарт	FIPS 180-4 / RFC 6234	FIPS 202	Обидва сімейства стандартизовані [18, 70].
Базова конструкція	Ітеративна схема Merkle–Damgard	Sponge-конструкція (Keccak)	SHA-3 структурно відмінний, що корисно для криптографічної диверсифікації [16, 19, 63, 64].
Вразливість до length extension	Притаманна для «голого» геш-значення у Merkle–Damgard	Відсутня (для SHA-3)	Важливо для деяких протоколів автентифікації; HMAC для усунення проблеми [70].
Внутрішній стан	256/512 біт робочих змінних + розклад повідомлення	1600-бітний стан (rate+capacity)	SHA-3 має більший стан, що може впливати на реалізацію в обмежених середовищах [18, 19].
Швидкодія у ПЗ (типово)	Зазвичай висока на загальних CPU; оптимізується інструкціями	Часто повільніша за SHA-2 на загальних CPU без спецоптимізацій	Для потокового хешування у хмарі це впливає на завантаження CPU (залежить від платформи) [20].
Апаратна реалізація (FPGA/ASIC)	Добре досліджена; поширені ядра; помірні ресурси	Також реалізується, але інша структура; може вимагати більших ресурсів стану	Вибір залежить від цільової архітектури та бюджету логіки/пам’яті [21, 22].
Гнучкість вихідної довжини	Фіксовані варіанти (224/256/384/512)	Гнучка (XOF: SHAKE128/256)	XOF корисні для тегів змінної довжини, індексації, протоколів; у дисертаційних задачах може бути аргументом [18, 19].
Типові сценарії в хмарі	Контроль цілісності, підписи, Merkle-дерева, HMAC-TLS	Альтернатива SHA-2, XOF-сценарії, системи з підвищеними вимогами	Обидва застосовні в PDP/POR, журналюванні, перевірці реплікації [84, 88, 89].
Практичний статус безпеки	Стійкий (без практичних атак на повні версії)	Стійкий (побудований на іншій конструкції)	Обидва сімейства вважаються безпечними за сучасним станом криптоаналізу [15, 16, 18].
Доцільність для ресурсно-обмежених вузлів	Часто кращий у ПЗ через оптимізації та інструкції	Може програвати у ПЗ, але має переваги конструкції	Для edge/IoT/польових вузлів важлива питома вартість «цикли/байт» і енергоспоживання [20, 21].

Типову структуру такої архітектури наведено на рис. 1.6.

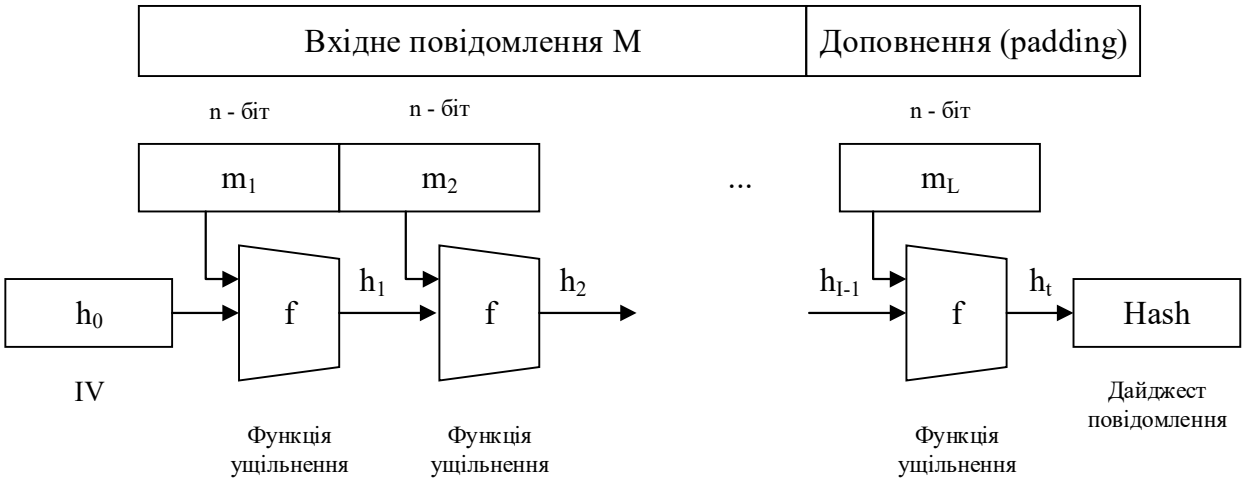

Рисунок 1.6 - Схема Меркля – Дамгарда

Криптографічні геш-функції характеризуються великою кількістю елементарних логічних та арифметичних операцій, які виконуються послідовно в межах одного раунду та між раундами обробки [11, 16]. До таких операцій належать побітові зсуви, циклічні перестановки, логічні операції XOR, AND, OR, а також додавання за модулем.

Сучасні криптографічні геш-функції, зокрема алгоритм BLAKE, використовують складні раундові структури з великою кількістю логічних та арифметичних операцій, що додатково ускладнює їх апаратну реалізацію у системах потокової обробки даних. Узагальнену схему гешування за алгоритмом BLAKE наведено на рис. 1.7.

У програмній реалізації операції гешування виконуються за рахунок процесорного часу, а в апаратній — потребують додаткових функціональних блоків і складного керування, що збільшує апаратні витрати та ускладнює конвеєризацію. Для ресурсно-обмежених систем це призводить до зниження пропускної здатності та зростання латентності обробки потоків даних [20].

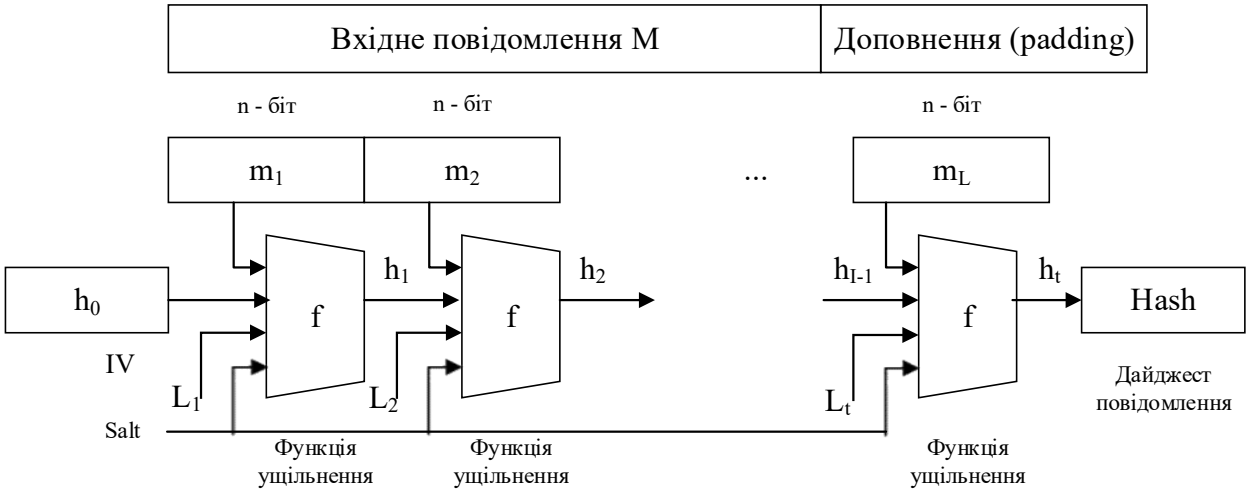

Рисунок 1.7 – Схема гешування за алгоритмом BLAKE

У приватних військових хмарах із edge-вузлами важливими є не лише криптостійкість, а й обчислювальна та енергетична ефективність. Зокрема, програмна реалізація SHA-256 потребує 12–15 циклів на байт, що при обробці великих обсягів даних створює значне навантаження на CPU, а SHA-3 додатково ускладнюється обробкою внутрішнього стану 1600 біт.

Апаратні реалізації також залишаються ресурсоємними: реалізація SHA-256 на FPGA потребує тисяч логічних елементів і значних обсягів пам’яті, що підвищує енергоспоживання та ускладнює інтеграцію [21, 22]. У мобільних і автономних вузлах це може зменшувати час роботи та обмежувати практичне застосування класичних геш-функцій. У результаті виникає протиріччя: з одного боку, SHA-2 і SHA-3 забезпечують високий рівень безпеки [16, 18, 70], з іншого — їх масове використання у потокових сценаріях створює надмірне навантаження. При цьому в багатьох прикладних задачах достатнім є не максимальний рівень криптостійкості, а компроміс між швидкодією та надійністю.

З позицій комп’ютерної інженерії ефективність апаратної реалізації визначається площею, обсягом пам’яті, пропускною здатністю, затримкою, енергоспоживанням і можливістю конвеєризації [20–22]. У цьому контексті перспективними є байт-орієнтовані методи, які добре узгоджуються з конвеєрною обробкою потоків і дозволяють досягати високої продуктивності при помірних апаратних витратах. Це робить їх основою для створення спеціалізованих

процесорів гешування, здатних зменшити навантаження на центральний процесор [42, 45].

Водночас ефективність таких рішень залежить від структури алгоритму: методи з великими таблицями підстановок вимагають значних ресурсів пам’яті та складної організації доступу, що погіршує масштабованість [21]. Тому більш доцільними є підходи з малим внутрішнім станом і низькою пам’яттєвою залежністю, які забезпечують компактність і енергоефективність [45, 47]. Для приватних і військових хмар, де критичними є енергоефективність і автономність, використання класичних криптографічних геш-функцій для задач, що не потребують максимальної стійкості (наприклад, дедуплікації), не завжди є інженерно виправданим [55, 56].

Таким чином, традиційні методи гешування, орієнтовані передусім на криптографічну стійкість, мають низку обмежень з точки зору апаратної реалізації у системах обробки даних приватних (військових) хмарних інфраструктур наведена у табл. 1.5.

Таблиця 1.5 – Порівняльна характеристика традиційних геш-функцій з позиції апаратної реалізації

Критерій	**Відомі криптографічні геш-функції**	**Наслідки для апаратної реалізації**
Алгоритмічна складність	Висока, багатораундова	Зростання апаратних витрат
Кількість операцій	Велика	Підвищена латентність
Паралелізація	Обмежена	Низька масштабованість
Енергоспоживання	Високе	Низька енергоефективність
Орієнтація	Криптостійкість	Не обтимізовані під data path

Отож, відомі геш-функції не оптимізовані для апаратної реалізації у системах потокової обробки даних.

1.3.2 Принципи побудови байт-орієнтованих методів гешування

Результати аналізу свідчать про те, що традиційні методи гешування, орієнтовані переважно на досягнення високого рівня криптографічної стійкості,

мають суттєві обмеження з точки зору апаратної реалізації у системах обробки даних приватних (військових) хмарних інфраструктур. Це зумовлює необхідність формування альтернативного підходу до побудови методів гешування, який враховує особливості апаратної архітектури та характер обробки потоків даних.

Під **байт-орієнтованим гешуванням** у контексті цієї роботи розумітимемо клас методів, у яких базовою одиницею обробки виступає байт (8 біт), а обчислення природно реалізуються у потоковій формі з мінімальними витратами на перетворення представлення даних. Такі методи можуть застосовувати прості операції (XOR, додавання за модулем 2^8 або 2^{32}, циклічні зсуви, табличні підстановки), а також агрегування характеристичних ознак потоку (статистичних або структурних) із подальшим формуванням геш-мітки [40, 43, 44].

Контрольні суми й CRC широко застосовуються як високопродуктивні механізми виявлення випадкових помилок у каналах зв’язку та під час передачі/зберігання даних. Перевагою є мала обчислювальна складність і придатність до табличного прискорення, що робить CRC зручним у потокових хмарних конвеєрах, зокрема на edge-вузлах. Проте CRC є лінійним перетворенням, що зумовлює низьку стійкість до навмисних модифікацій і можливість конструювання колізій за відомих параметрів. Тому CRC доцільний як технічна перевірка передачі, але не як самостійний механізм захисту цілісності в умовах активного противника, що важливо для військових приватних хмар [11, 12, 13]. Байт-орієнтоване гешування є також перспективним для кіберфізичних систем завдяки можливості апаратної реалізації на FPGA, мікроконтролерах та спеціалізованих співпроцесорах із низьким енергоспоживанням.

У задачах контент-визначеного розбиття, індексації та дедуплікації часто застосовують поліноміальні геш-функції та rolling-hash. Такі методи добре узгоджуються з байтовими потоками і дозволяють будувати продуктивні механізми ідентифікації фрагментів для дедуплікації. Водночас, у хмарних сховищах дедуплікація несе специфічні ризики: побічні канали можуть розкривати інформацію про наявність даних у системі, а також виникають задачі коректної обробки колізій на рівні індексу [60, 61]. Для приватної (військової) хмари це

означає необхідність поєднання продуктивного «технічного» гешування з додатковими перевірками або профілями безпеки [32, 75, 77].

Табличні методи можуть будуватися як XOR-агрегація значень із таблиць випадкових/псевдовипадкових значень. Їхня перевага — висока швидкодія, недолік — залежність властивостей від таблиць і помітні вимоги до пам’яті. Для апаратної реалізації в FPGA це часто означає використання BRAM, що прямо впливає на ресурсну вартість та енергоспоживання [21, 22]. У приватних хмарах табличні підходи можуть бути ефективними для технічного індексування, але потребують акуратного проєктування з урахуванням моделі загроз [12, 55, 59].

Одним із ключових принципів побудови байт-орієнтованих методів гешування є регулярність структури обчислень. Алгоритм гешування має бути організований у вигляді повторюваних елементарних операцій над байтовими даними без складних умовних переходів і розгалужень. Така регулярність створює передумови для ефективної конвеєризації та паралельної обробки потоків даних, що є критично важливим для досягнення високої пропускної здатності в апаратних реалізаціях [20, 22].

Альтернативним підходом до побудови геш-функцій є модель «криптографічної губки», схему якої наведено на рис. 1.8 [31, 32]. Вона передбачає поетапне поглинання та витискання даних із використанням внутрішнього стану фіксованого розміру. Незважаючи на гнучкість такої моделі, її архітектурні особливості не орієнтовані на байт-орієнтовану потокову обробку даних у системах з обмеженими ресурсами, що обмежує доцільність її використання у приватних (військових) хмарних інфраструктурах.

Наступним принципом є орієнтація на потокову обробку даних. Байт-орієнтовані методи гешування проєктуються таким чином, щоб забезпечити послідовну обробку вхідного потоку даних без необхідності накопичення великих буферів або попереднього аналізу всього повідомлення. Це особливо важливо для приватних (військових) хмарних інфраструктур, де значна частина даних надходить у вигляді безперервних потоків і потребує обробки у режимі, наближеному до реального часу [31, 32].

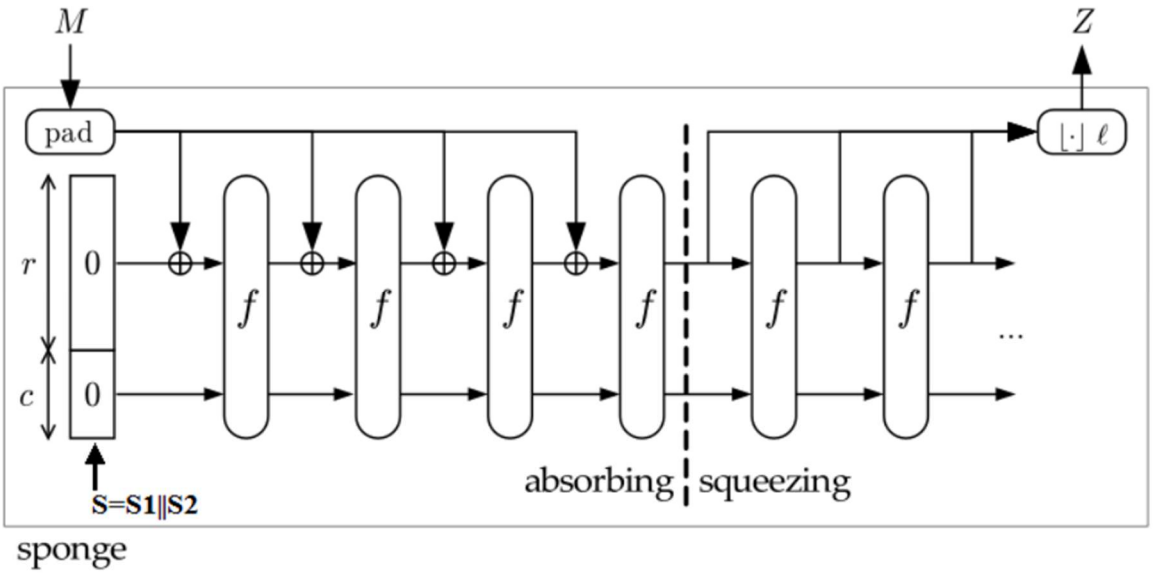

Рисунок 1.8 – Схема гешування за методом «Криптографічна губка»

Суттєвим принципом побудови байт-орієнтованих методів є забезпечення можливості масштабування апаратної реалізації. Алгоритмічна структура методу має допускати збільшення пропускної здатності шляхом розширення кількості паралельних обчислювальних модулів або збільшення ширини обробки даних без принципової зміни логіки роботи алгоритму [22].

При цьому байт-орієнтовані методи гешування не орієнтовані на досягнення максимальної криптографічної стійкості, характерної для сучасних криптографічних геш-функцій. Їх основним призначенням є ефективна реалізація контурів контролю цілісності, дедуплікації та ідентифікації блоків даних у системах обробки інформації. З огляду на це, вимоги до рівня дифузії та ймовірності колізій формуються з урахуванням інженерних потреб системи, а не максимальних криптографічних критеріїв [11, 55, 56].

Принципи побудови байт-орієнтованих методів гешування наведено у табл. 1.6.

Байт-орієнтований підхід до гешування доцільно розглядати не лише як окремий алгоритмічний метод, а як функціональний модуль у складі багатошарової апаратно-програмної архітектури системи обробки даних.

Таблиця 1.6 – Принципи побудови байт-орієнтованих методів гешування

Принцип	**Інженерний зміст**	**Значення для апаратної реалізації**
Байт як базова одиниця	Обробка кратна байту	Простота доступу до пам'яті
Регулярна структура	Відсутність складних переходів	Конвеєризація
Мінімум операцій	Прості логічні та арифметичні дії	Зменшення латентності
Потокова обробка	Без накопичення великих буферів	Реальний час
Масштабованість	Можливість паралельної обробки	Адаптація до навантаження
Параметризація	Режим роботи	Гнучкість архітектури

У такій архітектурі модуль гешування розміщується між програмними сервісами обробки даних та апаратним рівнем, безпосередньо впливаючи на пропускну здатність і латентність обробки потоків даних. Схематично місце байт-орієнтованого гешування у багатошаровій архітектурі наведено на рис. 1.9. І байт-орієнтоване гешування розглядається як функціональний модуль, що може реалізовуватись програмно або апаратно.

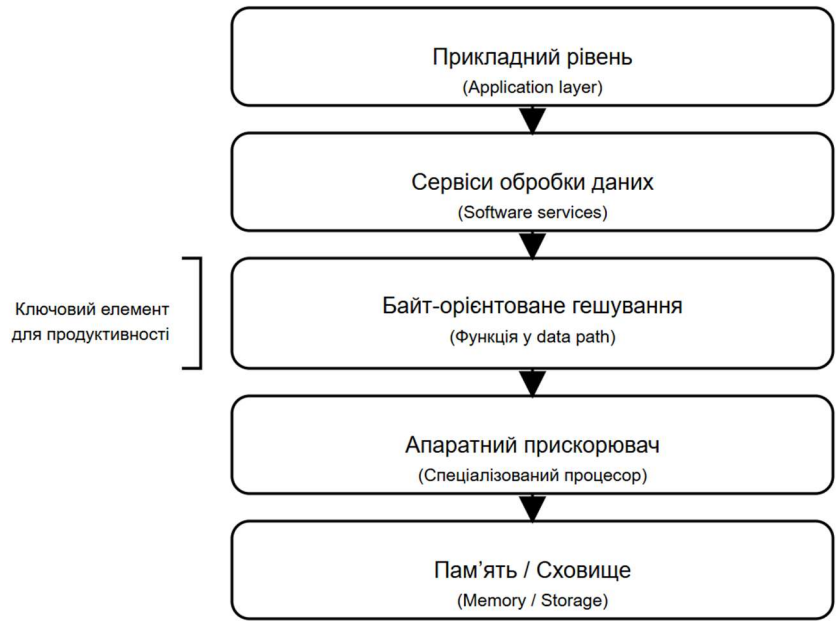

Рисунок 1.9 – Місце байт-орієнтованого гешування у багатошаровій архітектурі

Окремим принципом є можливість параметризації та підтримки різних режимів роботи байт-орієнтованих методів гешування. Це дозволяє адаптувати алгоритм до конкретних задач або умов експлуатації шляхом зміни окремих параметрів без необхідності повної зміни апаратної реалізації. Такий підхід є особливо важливим для спеціалізованих апаратних засобів, які мають функціонувати у складі багатошарової апаратно-програмної архітектури приватної хмарної інфраструктури.

Таким чином, байт-орієнтовані методи гешування ґрунтуються на сукупності принципів, спрямованих на спрощення апаратної реалізації, підвищення пропускної здатності та зниження енергетичних витрат при обробці потоків даних. Реалізація цих принципів створює інженерні передумови для побудови спеціалізованого процесора гешування, здатного ефективно функціонувати у складі системи обробки даних приватних (військових) хмарних інфраструктур. Конкретні реалізації таких методів розглядаються у наступних підрозділах цього розділу.

1.3.3 Геш-функції в архітектурі хмарних сервісів

У хмарних середовищах геш-функції використовуються на різних рівнях архітектури.

На рівні зберігання вони забезпечують перевірку цілісності блоків у розподілених файлових системах. Прикладом є використання Merkle-дерев [64, 84], де кореневе геш-значення відображає стан усієї структури. Перевірка одного блоку вимагає лише логарифмічної кількості обчислень.

У протоколах доказового зберігання (PDP, POR) [89, 90, 91] геш-функції дозволяють клієнту переконатися, що віддалений сервер дійсно зберігає повну копію даних без необхідності передавання всього файлу.

У механізмах дедуплікації даних геш-значення використовується як ідентифікатор блоку. Проте дослідження [60, 61] показали, що така практика може створювати побічні канали витоку інформації.

Крім того, геш-функції застосовуються для формування ланцюжків журналювання (hash chains), що забезпечують тампер-евідентність записів. На

відміну від універсальних програмних алгоритмів, байт-орієнтовані методи гешування можуть бути інтегровані безпосередньо в канали обробки даних кіберфізичних систем як апаратні прискорювачі контролю цілісності.

Розглянемо випадкові колізії та оцінка ризику для індексації/дедуплікації. Нехай H відображає множину об’єктів у n-бітний простір міток. Для N незалежних об’єктів наближена імовірність принаймні однієї колізії визначається формулою:

$$P_{\text{coll}} \approx 1 - \exp\left(-\frac{N(N-1)}{2 \cdot 2^n}\right).$$

Ця оцінка є критичною для систем, де геш-значення використовується як ключ індексу (дедуплікація, пошук ідентичних блоків, контроль реплікації) [60, 61]. У приватних хмарних сховищах великого масштабу навіть статистично «непогані» некриптографічні геш-функції можуть генерувати помітну кількість колізій, що потребує додаткових механізмів розв’язання: або зберігання «повного» блоку для підтвердження збігу, або використання комбінованих схем (наприклад, двоступеневий контроль: швидкий байт-орієнтований тег плюс сильне криптографічне геш-значення у разі збігу) [88, 89, 90].

Для військових приватних хмар важливо враховувати, що противник може діяти активно, а не лише спричиняти випадкові помилки. Для лінійних або параметрично простих схем (CRC, деякі rolling-hash) можливе конструювання колізій або керованих модифікацій. Це робить необхідним профілювання застосувань: де достатньо технічного контролю (виявлення випадкових пошкоджень), а де потрібна криптографічна стійкість (захист журналів, доказовість подій, контроль критичних об’єктів) [11, 12, 13, 77, 78]. У цьому контексті підходи, орієнтовані на підвищення дифузії та керованість колізій, є практично значущими; саме так позиціонуються методи, розроблені автором, у яких окремо розглянуто питання побудови/властивостей та реалізації [40–45].

У потокових хмарних сценаріях доцільно оцінювати алгоритм через питому вартість C_b— кількість тактів (або елементарних операцій) на один байт:

$$C_b = \frac{T}{L},$$

де T— сумарні тактові витрати, L— обсяг даних у байтах. Пропускна здатність вузла тоді визначається як

$$\text{Throughput} = \frac{f}{C_b},$$

де f— частота виконання (для CPU або апаратного ядра). Для приватної хмари це безпосередньо впливає на швидкість приймання об'єктів, швидкість індексації, час формування контрольних міток при резервуванні та реплікації [46, 88].

Навіть якщо алгоритм формально лінійний за довжиною даних $O(L)$, практична продуктивність визначається константами: кількістю раундів, обсягом внутрішнього стану, вимогами до пам'яті та можливістю конвеєризації [20]. Саме тому у ресурсно-обмежених вузлах приватних хмар виникає прагнення до методів, що максимально використовують байтову гранулярність і прості операції, а також допускають природну конвеєризацію або паралельне виконання. У роботах автора [45] спеціалізований процесор розглядається як засіб зменшення навантаження на CPU через винесення обчислень у апаратний блок, а в [47] показано орієнтацію на обмежені платформи, де оптимізація обробки даних «вартість на байт» є критичною.

1.4 Постановка задач дослідження

Аналіз аналогів показує, що існуючі підходи розташовуються на шкалі «продуктивність ↔ криптографічна стійкість». CRC та лінійні схеми забезпечують високу швидкодію і просту апаратну реалізацію, але є вразливими до навмисних модифікацій [11, 12]. Rolling-hash і поліноміальні методи ефективні для потокових задач, проте потребують контролю ризиків колізій [60, 61]. Табличні підходи є швидкими, але вимогливими обсягу пам'яті та складними для компактної апаратної реалізації [21], а легковагові криптографічні методи не завжди оптимальні для edge-сценаріїв та IoT пристроїв.

Розвиток приватних хмарних сервісів супроводжується зростанням обсягів даних, вимог до швидкодії та забезпечення цілісності й захищеності. Особливо це

актуально для відомчих і військових інфраструктур, що функціонують в умовах обмежених ресурсів і підвищених вимог до надійності [2], [3], [11–13], [76–79]. У таких системах переважає потокова обробка (передавання, реплікація, резервування, дедуплікація), а контроль цілісності базується на криптографічних геш-функціях, які забезпечують високий рівень стійкості [14–16], [61–65], [68], [69].

Водночас використання алгоритмів SHA-2 та SHA-3 у потокових сценаріях створює значне навантаження на процесори та пам’ять і знижує енергоефективність системи [20–22]. Існуючі альтернативи або не забезпечують достатнього рівня захищеності, або мають обмеження щодо масштабованості та апаратної інтеграції [60], [61]. У результаті виникає протиріччя між вимогами до надійного контролю цілісності та необхідністю зниження обчислювальних і енергетичних витрат у розподілених системах.

Отже, існуючі підходи не повністю вирішують зазначене протиріччя, що обумовлює необхідність розроблення моделей, методів і спеціалізованих засобів байт-орієнтованого гешування, здатних забезпечити баланс між продуктивністю, енергоефективністю та достатнім рівнем захищеності даних у приватних хмарних сервісах.Для розв’язання поставлених задач застосовуються методи системного аналізу, теорії розподілених систем і хмарних архітектур [2], [3], [81], [82], методи криптографічного аналізу та теорії криптографічних примітивів [14–16], [61–65], [68], [69], методи математичного моделювання та алгоритмічного синтезу [16], [19], [39], [40], [42], [43], методи структурного та функціонального моделювання [20], [81], [82], методи комп’ютерної інженерії та проектування цифрових пристроїв [20–22], [41], [44], методи експериментального дослідження та обчислювального експерименту [36–38], [45], [50], [59], [60], а також аналіз стандартів і нормативних документів у сфері інформаційної безпеки [23], [24], [76–79].

Таким чином, для усунення виявленого протиріччя між вимогами до захищеності та ресурсними обмеженнями приватних хмарних середовищ потрібно розв’язати такі задачі:

- розробити модель процесів роботи з файлами та об`єктами баз даних у приватній хмарній інфраструктурі;
- розробити методи гешування даних для приватних хмарних сервісів;
- розробити функціональну та структурну моделі спеціалізованого процесора байт-орієнтованого гешування з обмеженим енергоспоживанням;
- розробити архітектуру підсистеми організації роботи з даними у приватних хмарних сервісах;
- розробити методику комплексної оцінки ефективності організації роботи з даними у приватних хмарних сервісах;
- провести експериментальні дослідження розроблених методів та програмно-апаратних засобів.

1.5 Висновок

У першому розділі проведено аналіз підходів до організації роботи з даними у приватних хмарних сервісах з урахуванням специфіки їх застосування у відомчих і військових інформаційних системах. Встановлено, що такі інфраструктури поєднують підвищені вимоги до цілісності, захищеності та безперервності функціонування з обмеженими обчислювальними й енергетичними ресурсами окремих вузлів, особливо у периферійних сегментах. Обробка даних у них переважно має потоковий характер, що висуває вимоги до високої швидкодії та масштабованості підсистем.

Показано, що поряд із традиційними загрозами суттєву роль відіграють ризики, пов’язані з розподіленістю, дедуплікацією та реплікацією даних. У таких умовах геш-функції є ключовим інструментом контролю цілісності, ідентифікації об’єктів і побудови тампер-евідентних механізмів. Водночас класичні криптографічні геш-функції (SHA-2, SHA-3) забезпечують високий рівень стійкості, але характеризуються значною обчислювальною складністю та створюють підвищене навантаження на процесори і пам’ять при масовій потоковій обробці.

Аналіз легковагових та байт-орієнтованих підходів показав, що вони мають кращу продуктивність і придатність до потокової обробки, однак часто не забезпечують достатнього рівня стійкості або мають обмеження щодо масштабованості та апаратної реалізації. У результаті виявлено системне протиріччя між необхідністю забезпечення надійного контролю цілісності та вимогами до зниження обчислювальних і енергетичних витрат.

Обґрунтовано доцільність розроблення байт-орієнтованих методів гешування, орієнтованих на потокову обробку та апаратно-ефективну реалізацію у вигляді спеціалізованого процесора. Такий підхід дозволяє зменшити навантаження на центральний процесор і забезпечити масштабованість системи без пропорційного зростання енергоспоживання.

РОЗДІЛ 2
МЕТОДИ БАЙТ-ОРІЄНТОВАНОГО ГЕШУВАННЯ ДЛЯ СИСТЕМ ОБРОБКИ ДАНИХ ПРИВАТНИХ ХМАРНИХ ІНФРАСТРУКТУР

Сучасні приватні та військові хмарні інфраструктури характеризуються підвищеними вимогами до надійності, пропускної здатності та енергоефективності систем обробки даних. За таких умов програмна реалізація операцій гешування, що широко застосовується для контролю цілісності та дедуплікації даних, призводить до істотного навантаження на центральні процесори та зниження загальної продуктивності системи.

У межах даного дослідження запропоновано байт-орієнтований підхід до гешування даних, який орієнтований на подальшу апаратну реалізацію та використання у складі багатошарової апаратно-програмної архітектури приватної (військової) хмарної інфраструктури. Розроблені методи забезпечують зменшення обсягу дубльованих даних на 25–40 % та створюють передумови для побудови спеціалізованого процесора гешування, що буде розглянуто у розділі 3.

2.1 Метод неітеративного байт-орієнтованого гешування

2.1.1 Загальна ідея методу гешування

Основною ідеєю методу є лінійна, однопрохідна обробка вхідного повідомлення з формуванням двох масивів байт-орієнтованих характеристик — масиву кількостей входжень байтів та масиву позиційних характеристик. На відміну від класичних ітеративних схем гешування, метод не використовує компактний внутрішній стан, що оновлюється на кожному кроці, а спирається на накопичення глобальних статистик, які однозначно характеризують структуру повідомлення. У подальшому даний метод будемо називати **Метод 1**.

Вхідне повідомлення розглядається як послідовність байтів:

$$\mathbf{M} = \{ m_1, m_2, \dots , m_l \}.$$

Кожен байт розглядається як число n, що відповідає ASCII – коду символу представленого байтом m_l ($l = 1 \div L$), тобто $n = f(m_l)$.

Повідомлення характеризується кількістю елементів k_n, що мають числовий еквівалент n ($n = 0 \div 255$) та номерами позицій $l_j(n)$ ($j = 1 \div k_n$), у яких розташовані ці елементи.

Метод передбачає реалізацію таких етапів.

Етап 1. Формування масиву $\mathbf{K} = (k_0, k_1, \dots, k_{255})$, де k_n – кількість елементів, що мають числовий еквівалент n ($n = 0 \div 255$).

Етап 2. Формування масиву $\mathbf{S} = (s_0, s_1, \dots, s_{255})$, де s_n ($n = 0 \div 255$) – сума номерів позицій байтів, що мають числовий еквівалент n у повідомленні.

Етап 3. Ущільнення масиву **K**.

Етап 4. Ущільнення масиву **S**.

Етап 5. Формування геш-значення.

2.1.2 Формування масивів характеристик **K** та **S**

На кожному кроці обробки чергового байта m_l виконуються операції:

$$k_n^{(l)} = \begin{cases} k_n^{(l-1)} + 1, \text{якщо } m_l = n \\ k_n^{(l-1)}, \text{інакше} \end{cases}, \tag{2.1}$$

де $k_n^{(0)} = 0, l = 1 \div L$.

$$s_n = \sum_{j=1}^{k_n} l_j^{(n)}. \tag{2.2}$$

Зазначені операції є простими арифметичними діями над байтовими даними або даними у вигляді слів та не містять умовних переходів, що робить їх придатними для реалізації у вигляді апаратних лічильників і акумуляторів. Таким чином, стадія формування **K** і **S** є повністю потоковою і може бути безпосередньо інтегрована в основний шлях обробки даних.

Наприклад, для повідомлення «характеристичні ознаки», буде сформовано такі масиви **K** та **S**:

M	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}	m_{17}	m_{18}	m_{19}	m_{20}	m_{21}	m_{22}
Літ	х	а	р	а	к	т	е	р	и	с	т	и	ч	н	і	␣	о	з	н	а	к	и
n	245	244	240	244	234	242	229	240	232	241	242	232	247	237	179	32	238	231	237	224	234	232

За формулою (2.1) підраховується кількість числових еквівалентів байтів, з яких складається повідомлення.

K	k_0	…	k_{32}	…	k_{179}	…	k_{224}	…	k_{229}	…	k_{231}	k_{232}	…	k_{234}	…	k_{237}	k_{238}	…	k_{240}	k_{241}	k_{242}	…	k_{245}	k_{247}	…	k_{255}
	0	…	1	…	1	…	3	…	1	…	1	3	…	2	…	2	1	…	2	1	2	…	1	1	…	0

За формулою (2.2) обчислюються суми номерів для кожного числового еквіваленту байту вхідного повідомлення.

…

$s_{32} = 16$

$s_{179} = 15$

$s_{224} = 2 + 4 + 20 = 26$

$s_{229} = 7$

$s_{231} = 18$

$s_{232} = 9 + 12 + 22 = 43$

$s_{234} = 5 + 21 = 26$

…

S	s_0	…	s_{32}	…	s_{179}	…	s_{224}	…	s_{229}	…	s_{231}	s_{232}	…	s_{234}	…	s_{237}	s_{238}	…	s_{240}	s_{241}	s_{242}	…	s_{245}	s_{247}	…	s_{255}
	0	…	16	…	15	…	26	…	7	…	18	43	…	26	…	33	17	…	11	10	17	…	1	13	…	0

Зазвичай, розмір вхідного повідомлення функції гешування передбачається 2^{64} біт, а для мобільних пристроїв обсяг зменшується до 2^{32} біт. З урахуванням останнього, елементи масиву **K** представляються набором з 4 байтів:

$$k_n = (b_{n,3}, b_{n,2}, \dots, b_{n,0}), (n = 0 \div 255).$$

Масив **S** складається з 256 елементів по 8 байт кожен:

$$s_n = (b_{n,7}, b_{n,6}, \dots, b_{n,0}), (n = 0 \div 255).$$

Для визначення геш-коду здійснюється ущільнення масивів **K** та **S** за певними правилами, які є основою таких методів гешування.

2.1.3 Ущільнення масивів характеристик

Після завершення проходу по повідомленню масиви **K** та **S** підлягають ущільненню з метою формування компактного геш-значення фіксованої довжини.

Для цього застосовуються функції ущільнення C_1 та C_2:

$$h_k = C_1(\mathbf{K}),\ h_s = C_2(\mathbf{S}),$$

де h_k і h_s — ущільнені значення масивів **K** та **S** до заданої довжини (наприклад, 128 біт).

Функції C_1 та C_2 реалізуються як послідовність регулярних операцій редукції:

- початкова байтова редукція елементів масивів;
- подальше поетапне ущільнення масивів за деревоподібною схемою (наприклад, 256 → 128 → 64 → 32 → 16 байтів);
- використання простих логічних операцій (зокрема XOR або додавання за модулем).

Завдяки регулярній структурі та фіксованій розмірності вхідних масивів, етап ущільнення є детермінованим за часом виконання та добре придатним для реалізації у вигляді апаратного конвеєра або комбінаційної логіки.

Функція ущільнення C_1 передбачає виконання таких дій.

Масив **K** перетворюється у масив **K***, кожен елемент якого представляється одним байтом:

$$\mathbf{K}^* = (k_0^*, k_1^*, \dots, k_{255}^*),$$

де

$$k_n^* = \left(\sum_{i=0}^{3} b_{n,i}\right) mod\ 2^8. \qquad (2.3)$$

Далі, масив $\mathbf{K}^*$перетворюється у масив $\mathbf{K}^{(128)}$, що складається з 128 однобайтних елементів:

$$\mathbf{K}^{(128)} = \left(k_0^{(128)}, k_1^{(128)}, \dots, k_{127}^{(128)}\right),$$

де

$$k_i^{(128)} = (k_{2i}^* \oplus k_{2i+1}^*), (i = 0 \div 127).$$

Масив $\mathbf{K}^{(128)}$перетворюється у масив $\mathbf{K}^{(64)}$з 64 однобайтних елементів:

$$\mathbf{K}^{(64)} = \left(k_0^{(64)}, k_1^{(64)}, \dots, k_{63}^{(64)}\right),$$

де

$$k_j^{(64)} = \left(k_{2j}^{(128)} \oplus k_{2j+1}^{(128)}\right), (j = 0 \div 63).$$

Масив $\mathbf{K}^{(64)}$перетворюється у масив $\mathbf{K}^{(32)}$довжиною в 32 однобайтних елементів:

$$\mathbf{K}^{(32)} = \left(k_0^{(32)}, k_1^{(32)}, \dots, k_{31}^{(32)}\right),$$

де

$$k_l^{(32)} = \left(k_{2l}^{(64)} \oplus k_{2l+1}^{(64)}\right), (l = 0 \div 31).$$

Масив $\mathbf{K}^{(32)}$перетворюється у масив $\mathbf{K}^{(16)}$з 16 однобайтних елементів:

$$\mathbf{K}^{(16)} = \left(k_0^{(16)}, k_1^{(16)}, \dots, k_{15}^{(16)}\right),$$

Де $k_m^{(16)} = \left(k_{2m}^{(32)} \oplus k_{2m+1}^{(32)}\right), (m = 0 \div 15)$, який є результатом ущільнення масиву $\mathbf{K}$, тобто h_k.

Функція ущільнення C_2 передбачає виконання таких дій.

Масив $\mathbf{S}$ перетворюється у масив $\mathbf{S}^*$ кожен елемент якого представляється одним байтом:

$$\mathbf{S}^* = (s_0^*, s_1^*, \dots, s_{255}^*),$$

де

$$s_n^* = \left(\sum_{i=0}^{7} b_{n,i}\right) mod\ 2^8. \qquad (2.4)$$

Далі, масив $\mathbf{S}^*$перетворюється у масив $\mathbf{S}^{(128)}$, і так далі, як і для масиву **K**.

$$\mathbf{S}^{(128)} \to \mathbf{S}^{(64)} \to \mathbf{S}^{(32)} \to \mathbf{S}^{(16)}$$

Масив $\mathbf{S}^{(16)}$ є результатом ущільнення масиву **S**, який позначено h_s.

2.1.4 Формування геш-значення

Геш-значення формується шляхом комбінування проміжних результатів h_k та h_s:

$$h = f(h_k, h_s),$$

де $f(\cdot)$ — проста детермінована операція, наприклад конкатенація або арифметичне поєднання значень.

Пропонуються такі варіанти функції *f*, аргументами якої є h_k та h_s:

1) дописування коду h_s до коду h_k

$$h = f(h_k, h_s) = h_k \,||\, h_s\,;$$

2) звичайне арифметичне множення 128-розрядних кодів

$$h = f(h_k, h_s) = h_k \cdot h_s.$$

Такий підхід дозволяє зберегти інформацію як про частотні, так і про позиційні характеристики вхідного повідомлення, забезпечуючи розрізнювальну здатність геш-значень у задачах контролю цілісності, дедуплікації та ідентифікації блоків даних.

Схему процесу гешування за таким методом наведено на рис. 2.1. Тут f_K і f_S позначають обчислення за формулами (2.1) та (2.2).

Запропонований метод має такі характеристики:

- лінійну часову складність формування масивів **K** та **S**;
- можливість апаратного паралелізму та масштабування;
- передбачувану латентність.

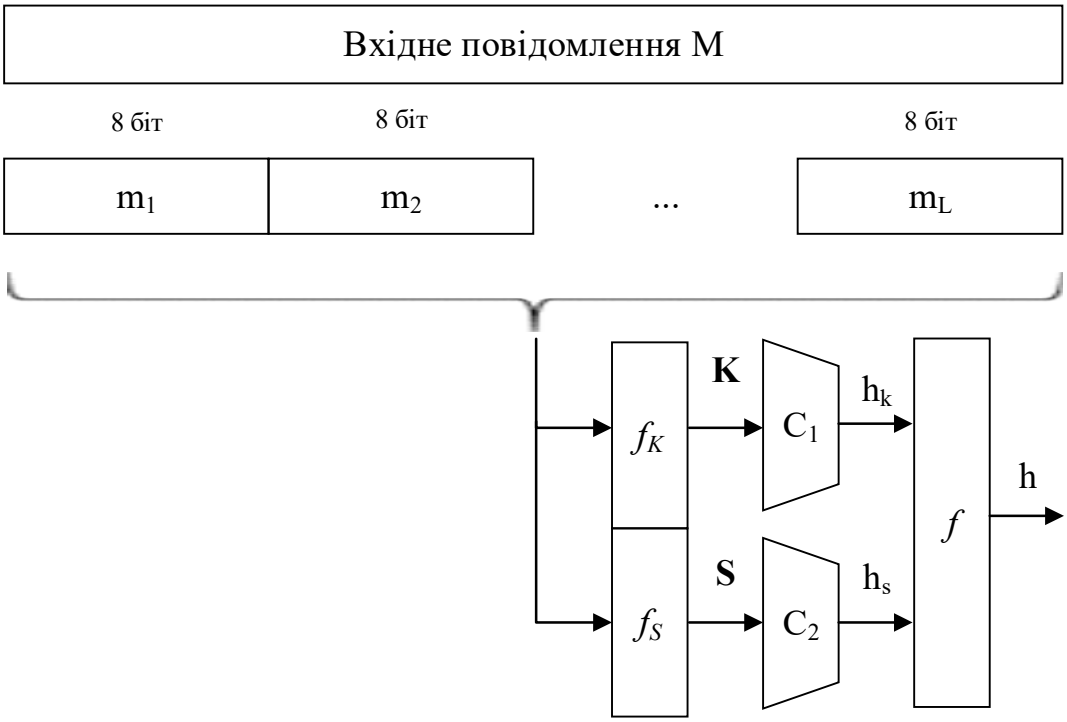

Рисунок 2.1 – Узагальнена схема процесу гешування за методом 1

Метод розроблено з урахуванням можливості реалізації у вигляді спеціалізованого апаратного модуля гешування. Етапи формування **K** та **S** можуть бути реалізовані з використанням масиву лічильників і акумуляторів, а етап ущільнення — у вигляді регулярного редукційного конвеєра.

У порівнянні з традиційними криптографічними геш-функціями, метод характеризується меншою кількістю логічних операцій, нижчим енергоспоживанням і вищою придатністю до інтеграції у багатошарову апаратно-програмну архітектуру системи обробки даних приватних хмарних інфраструктур.

Таким чином, метод байт-орієнтованого гешування з лінійною архітектурою обробки, що базується на формуванні та ущільненні масивів **K** і **S**, є інженерно обґрунтованим рішенням для задач потокової обробки даних. Його структура безпосередньо визначає можливість створення спеціалізованого процесора гешування, архітектура якого розглядається у розділі 3. У наступному підрозділі буде розглянуто модифікований метод байт-орієнтованого гешування з підвищеним рівнем дифузії.

2.2 Розробка алгоритму гешування за методом 1

Для реалізації описаного вище методу байт-орієнтованого гешування пропонується такий алгоритм.

1. Прийняти вхідне повідомлення **M** довжини L.
2. Ініціалізувати масив **K** із 256 елементів нульовими значеннями.
3. Ініціалізувати масив **S** із 256 елементів нульовими значеннями.
4. Для кожного байта m_i, де $i = 0 \dots L - 1$:
   - збільшити значення k_{m_i} на 1;
   - додати номер позиції i до s_{m_i}.
5. Виконати ущільнення масиву **K** за правилом $\mathbf{K}^* = C_1(\mathbf{K})$.
6. Виконати ущільнення масиву **S** за правилом $\mathbf{S}^* = C_2(\mathbf{S})$.
7. Сформувати геш-значення $h = f(\mathbf{K}^*, \mathbf{S}^*)$.
8. Повернути результат h.

Узагальнену блок-схему алгоритму обрахунку геш-значення за методом 1 подано на рис. 2.2:

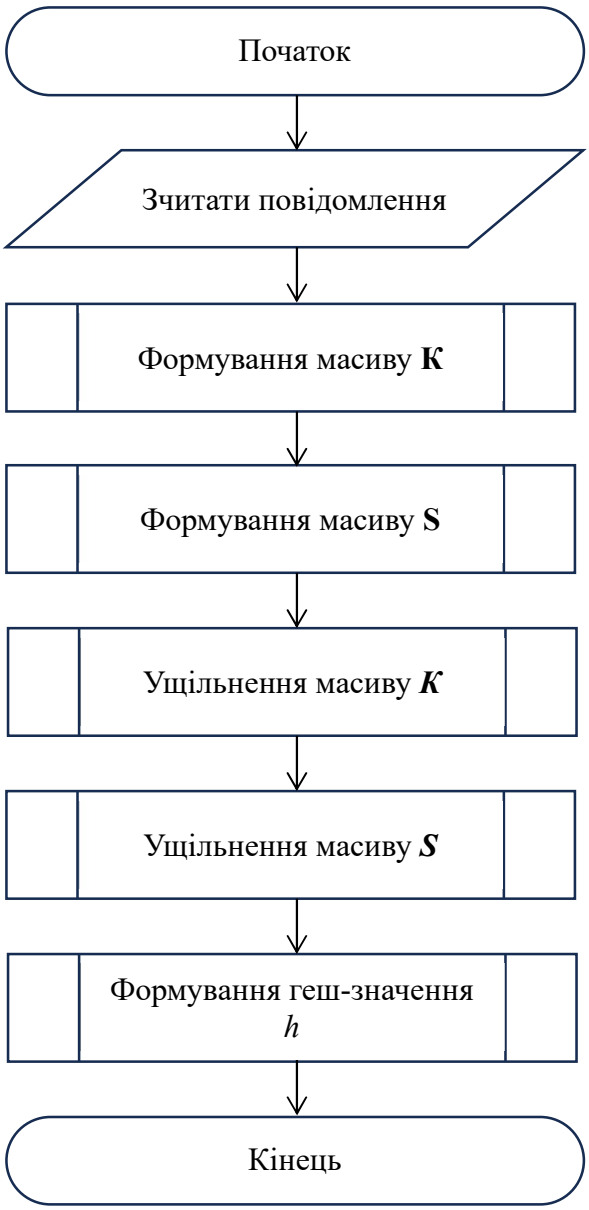

Рисунок 2.2 – Узагальнена блок-схема алгоритму обрахунку геш-значення за методом 1

Масив **К** формується як кількість елементів, що мають певний числовий еквівалент *n* за формулою (2.1). Алгоритм формування масиву **К** подано на рис. 2.3.

Масив **S** формується як сума номерів позицій байтів, що мають числовий еквівалент *n* у повідомленні M за формулою (2.2). Алгоритм формування масиву **S** подано на рис. 2.4.

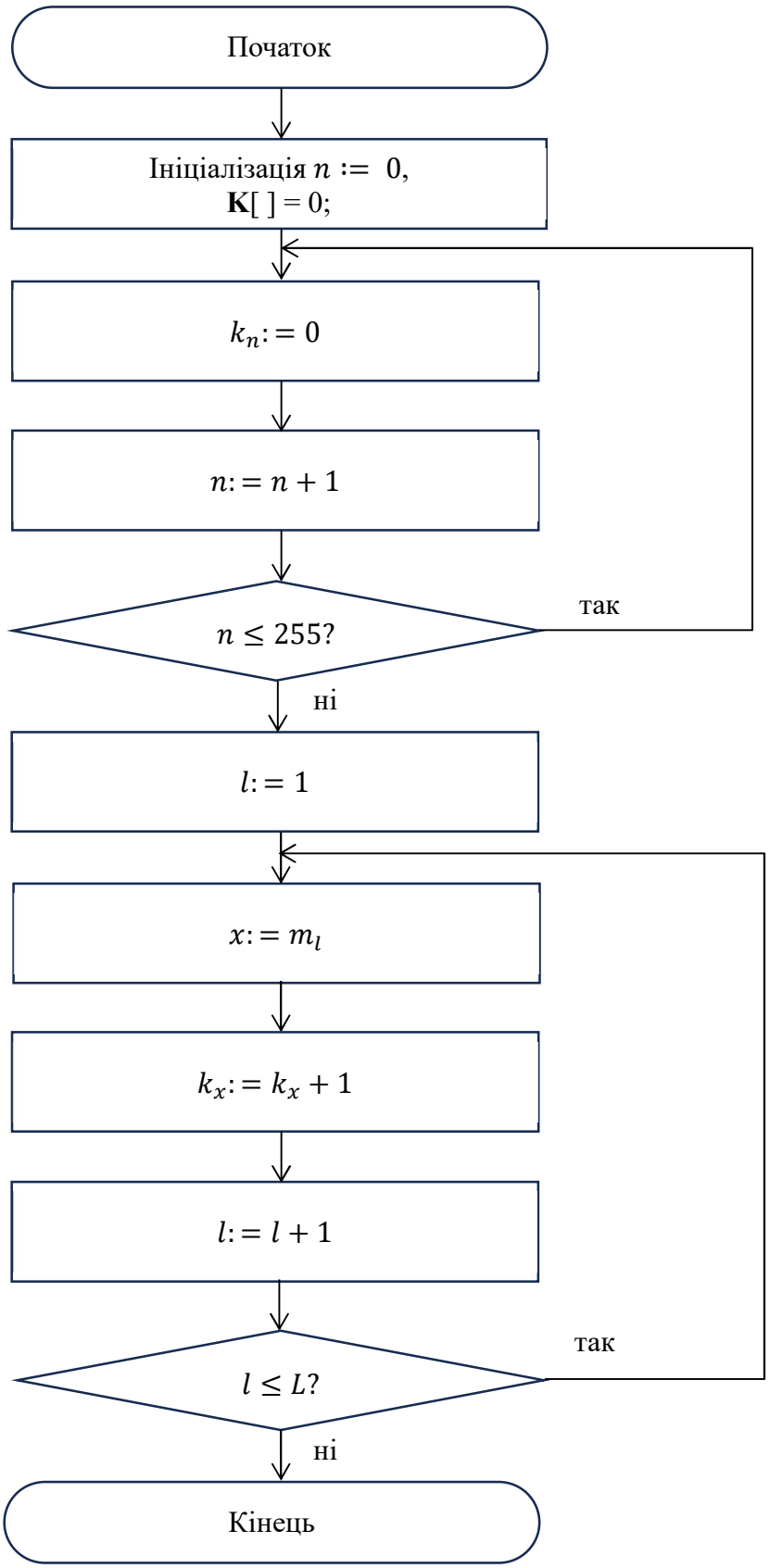

Рисунок 2.3 – Блок-схема алгоритму формування масиву **K**

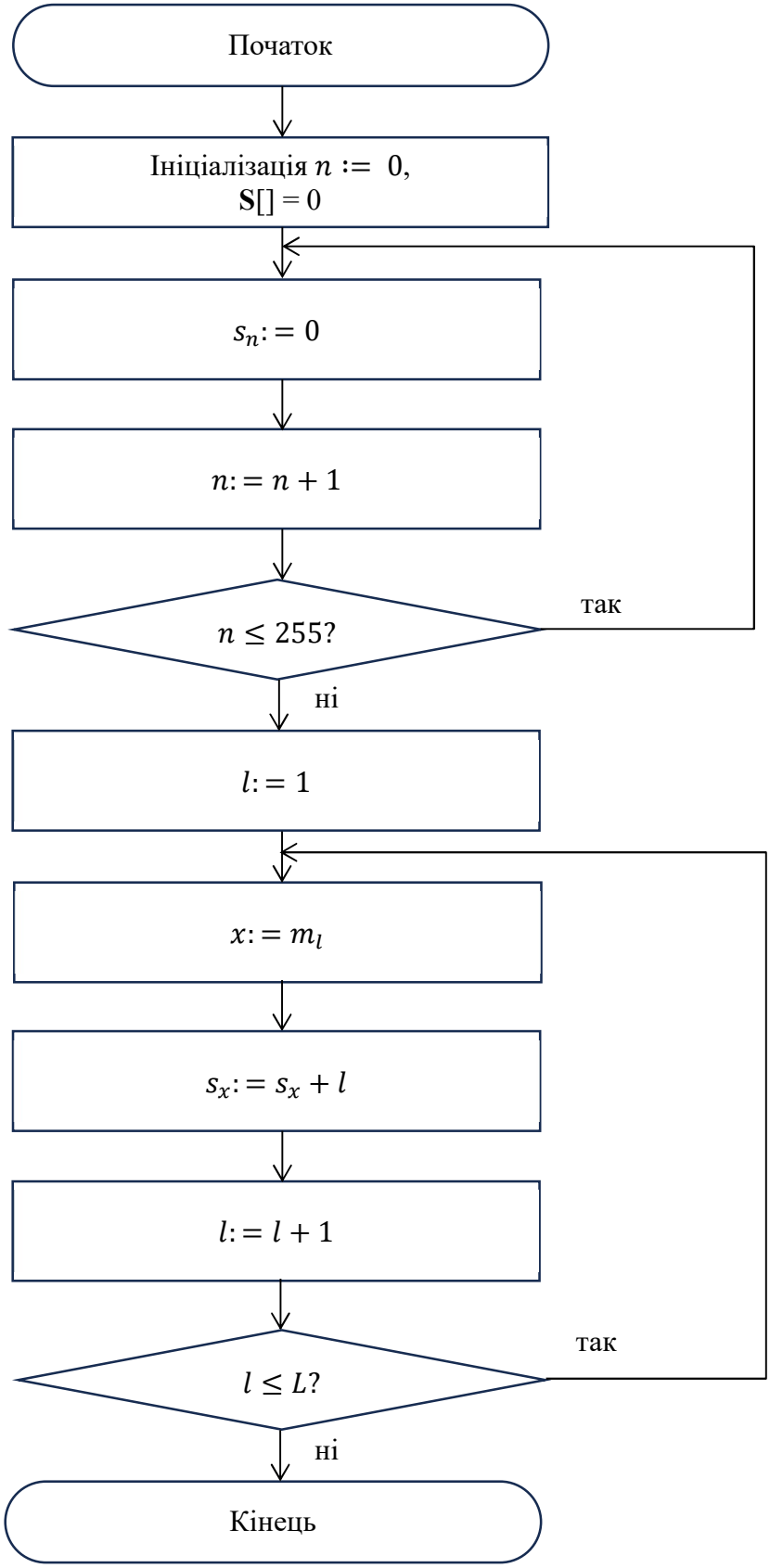

Рисунок 2.4 – Блок-схема алгоритму формування масиву **S**

Далі виконується ущільнення масиву $\mathbf{K}$, яке полягає у послідовному багаторівневому зменшенні його розмірності шляхом арифметичного та логічного перетворення елементів. На першому кроці вихідний масив $\mathbf{K}$ перетворюється у масив $\mathbf{K}^*$, у якому кожний елемент подається одним байтом. Далі здійснюється каскадне ущільнення за допомогою операції виключного АБО (XOR): із масиву $\mathbf{K}^*$ формується масив $\mathbf{K}^{(128)}$, де кожний елемент отримується як XOR двох сусідніх елементів $\mathbf{K}^*$; аналогічно масив $\mathbf{K}^{(128)}$перетворюється у $\mathbf{K}^{(64)}$, потім у $\mathbf{K}^{(32)}$, а далі у $\mathbf{K}^{(16)}$. На кожному наступному рівні кількість елементів зменшується вдвічі за рахунок попарного об'єднання сусідніх значень, що дозволяє зберігати узагальнену інформацію про структуру початкового масиву при значному скороченні обсягу даних. Підсумковий масив $\mathbf{K}^{(16)}$, який складається з 16 однобайтних елементів, позначається як h_k і використовується як результат ущільнення масиву $\mathbf{K}$. Блок-схема алгоритму ущільнення масиву $\mathbf{K}$ подано на рис. 2.5.

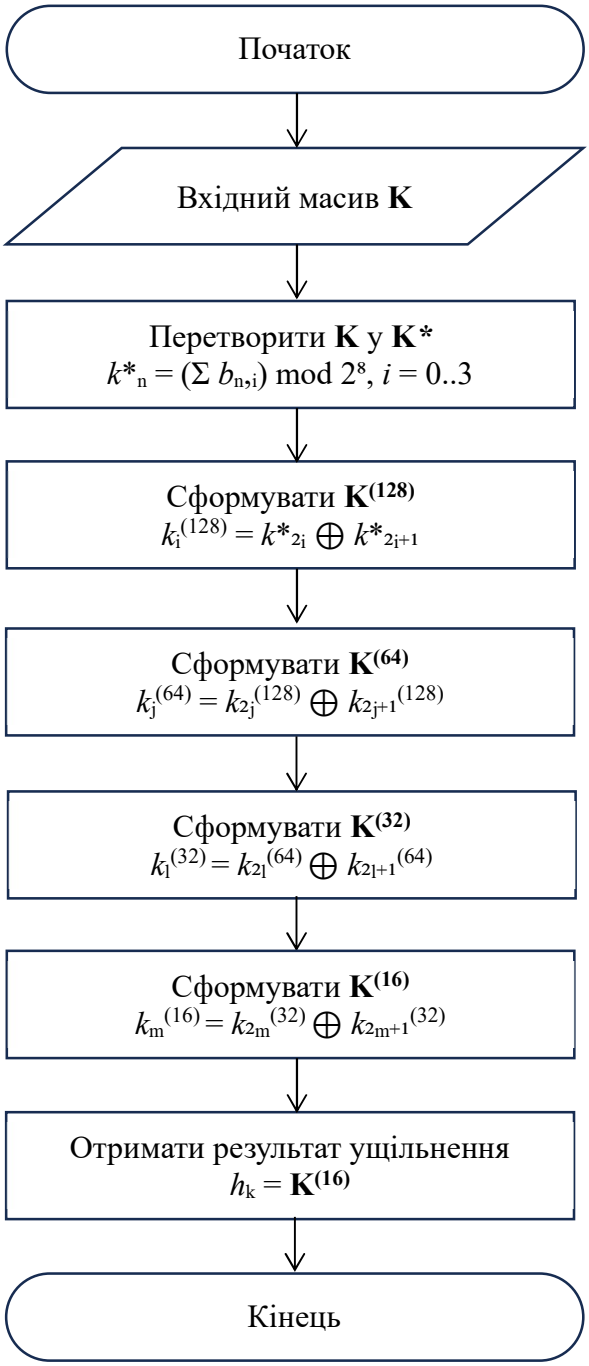

Рисунок 2.5 – Блок-схема алгоритму ущільнення масиву **K**

Алгоритм ущільнення масиву **S** передбачає послідовне зменшення його розмірності у декілька етапів. Спочатку вихідний масив $\mathbf{S} = (s_0, s_1, \dots, s_{255})$перетворюється у масив $\mathbf{S}^* = (s_0^*, s_1^*, \dots, s_{255}^*)$, у якому кожний елемент подається одним байтом за правилом $s_n^* = \left(\sum_{i=0}^{7} b_{n,i}\right) mod 2^8$. Далі масив

$\mathbf{S}^*$послідовно ущільнюється шляхом попарного об'єднання сусідніх елементів операцією виключного АБО: формується масив $\mathbf{S^{(128)}}$, потім $\mathbf{S^{(64)}}$, далі $\mathbf{S^{(32)}}$ і, нарешті, $\mathbf{S^{(16)}}$. Підсумковий масив $\mathbf{S^{(16)}}$, що складається з 16 однобайтних елементів, є результатом ущільнення масиву $\mathbf{S}$ і позначається як h_s. Блок-схема алгоритму ущільнення масиву $\mathbf{S}$ подано на рис. 2.6.

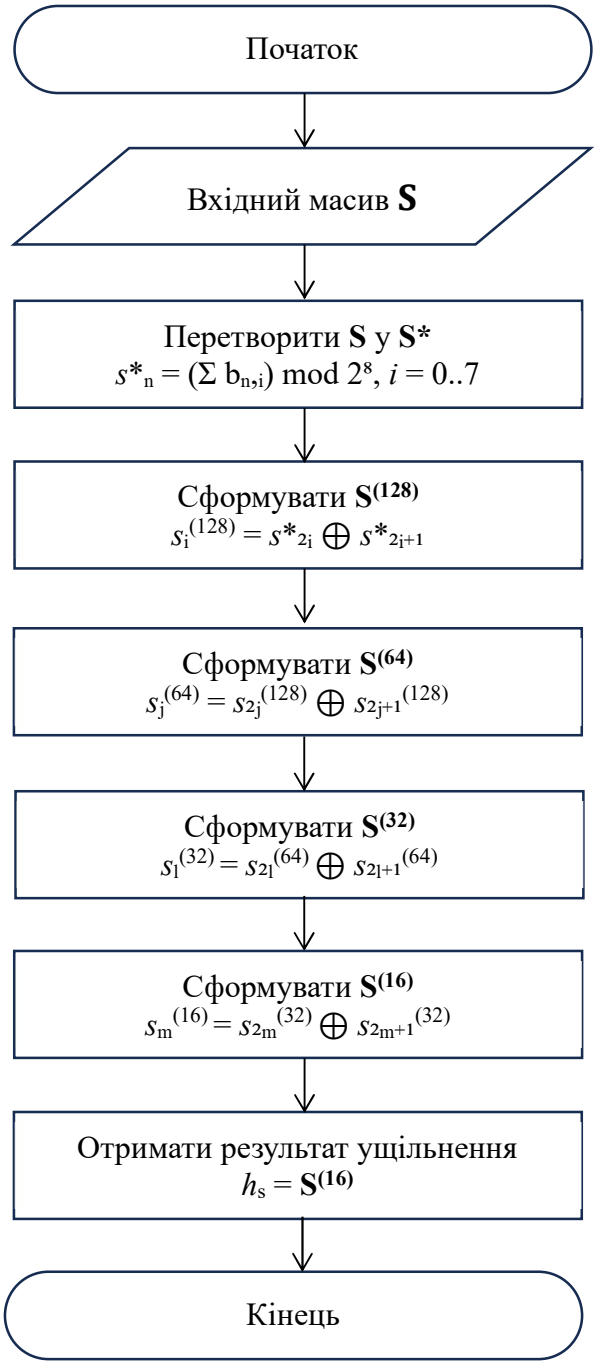

Рисунок 2.6 – Блок-схема алгоритму ущільнення масиву $\mathbf{S}$

Блок-схему алгоритму обчислення геш-значення наведено на рис. 2.7.

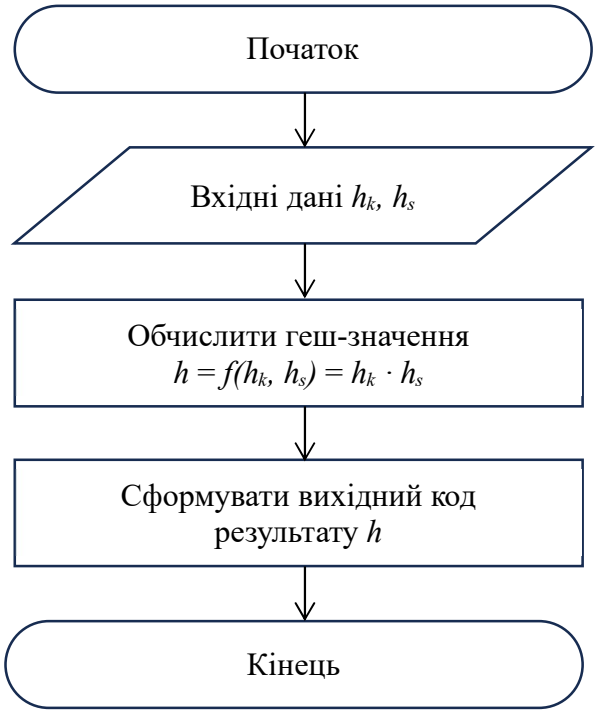

Рисунок 2.7 – Блок-схема алгоритму формування геш-значення

Алгоритм отримання геш-значення полягає у комбінуванні двох проміжних результатів ущільнення: масиву h_k, сформованого на основі масиву частот **K**, та масиву h_s, сформованого на основі масиву позицій **S**. На вхід функції $f(h_k, h_s)$ подаються обидва проміжні значення, після чого виконується їх множення. У результаті формується геш-значення h, яке визначається формулою $h = f(h_k, h_s) = h_k \cdot h_s$. Таким чином, геш-значення одночасно враховує частотні характеристики байтів повідомлення та особливості їх позиційного розподілу, що підвищує інформативність і стійкість результату перетворення.

2.3 Метод байт-орієнтованого гешування з підвищеним рівнем дифузії

Одним з недоліків Методу 1 є недостатня криптографічна стійкість при використанні його для вирішення задач з підвищеними вимогами до стійкості. Тому пропонується метод з покращеним впливом вхідних даних на геш-значення. Метод зберігає базові принципи байт-орієнтованої потокової обробки, однак вводить

додаткові етапи нелінійного перетворення та змішування проміжних характеристик. У подальшому даний метод будемо називати **Метод 2**.

2.2.1 Загальна ідея та відмінності від Методу 1

На відміну від Методу 1, у якому формування геш-значення здійснюється шляхом незалежного ущільнення масивів **K** та **S** з подальшим комбінуванням результатів, у Методі 2 вводиться додатковий етап змішування статистичних характеристик до виконання фінального ущільнення.

Основною метою такого підходу є підвищення рівня дифузії, тобто забезпечення того, щоб зміна одного або декількох байтів вхідного повідомлення призводила до суттєвих змін значної частини фінального геш-значення. Це особливо важливо для задач ідентифікації блоків даних та дедуплікації у хмарних середовищах, де необхідно мінімізувати ймовірність колізій між близькими за вмістом повідомленнями [11, 14–16, 55, 56].

2.2.2 Формування проміжних характеристик **K** та **S**

Як і у Методі 1, вхідне повідомлення

$$\mathbf{M} = \{ m_1, m_2, \ldots , m_l \}.$$

розглядається як послідовність байтів, для яких у процесі однопрохідної потокової обробки формуються масиви **K** та **S**.

Метод передбачає реалізацію таких етапів.

Етап 1. Початкове заповнення масивів «криптографічною сіллю».

Етап 2. Формування масиву $\mathbf{K} = (k_0, k_1, \ldots , k_{255})$, де k_n – кількість елементів, що мають числовий еквівалент n ($n = 0 \div 255$).

Етап 3. Формування масиву $\mathbf{S} = (s_0, s_1, \ldots , s_{255})$, де S_n ($n = 0 \div 255$) – сума номерів позицій байтів, що мають числовий еквівалент n у повідомленні.

Етап 4. Застосування функції змішування масивів **K** та **S**.

Етап 5. Ущільнення змішаних характеристик.

Етап 6. Формування геш-значення.

Таким чином, Метод 2 зберігає всі переваги потокової обробки та передбачуваної латентності на початковій стадії [20, 31, 32].

2.2.3 Етап змішування статистичних характеристик

Ключовою відмінністю Методу 2 є введення функції змішування $g(\mathbf{K}, \mathbf{S})$, яка формує проміжний масив характеристик:

$$\mathbf{H} = g(\mathbf{K}, \mathbf{S}).$$

Функція g(·) реалізує поелементне або блочне комбінування значень масивів **K** та **S** з використанням простих арифметичних та логічних операцій. Типовими операціями можуть бути:

- додавання або XOR відповідних елементів;
- арифметичне множення відповідних елементів;
- циклічні зсуви байтових слів;
- перестановки байтів у межах фіксованих блоків.

До вхідних масивів **K** та **S** застосовується один з варіантів функції g:

$$g = \{g_0, g_1, \dots, g_{255}\}$$

1) дописування коду s_n до коду k_n

$$g_n(k_n, s_n) = h_n^* = k_n \,||\, s_n ;$$

2) звичайне арифметичне множення 32-бітних кодів на 64-бітні коди

$$g_n(k_n, s_n) = h_n^* = k_n \cdot s_n.$$

$$\mathbf{H} = g(\mathbf{K}, \mathbf{S}) = (h_0^*, h_1^*, \dots, h_{255}^*),$$

Масив ***H*** складається з 256 елементів довжиною 96 біт (12 байтів):

$$h_n^* = \left(b_{n,11}, b_{n,10}, \dots, b_{n,0}\right), (n = 0 \div 255).$$

Метою цього етапу є створення взаємозалежності між частотними та позиційними характеристиками повідомлення ще до стадії ущільнення, що підвищує чутливість алгоритму до структурних змін вхідних даних.

Введення етапу змішування статистичних характеристик змінює загальну структуру процесу гешування у порівнянні з лінійною архітектурою Методу 1. На цій стадії формується взаємозалежність між частотними та позиційними

характеристиками вхідного повідомлення ще до виконання операції ущільнення. Узагальнену схему процесу гешування за методом 2 наведено на рис. 2.8.

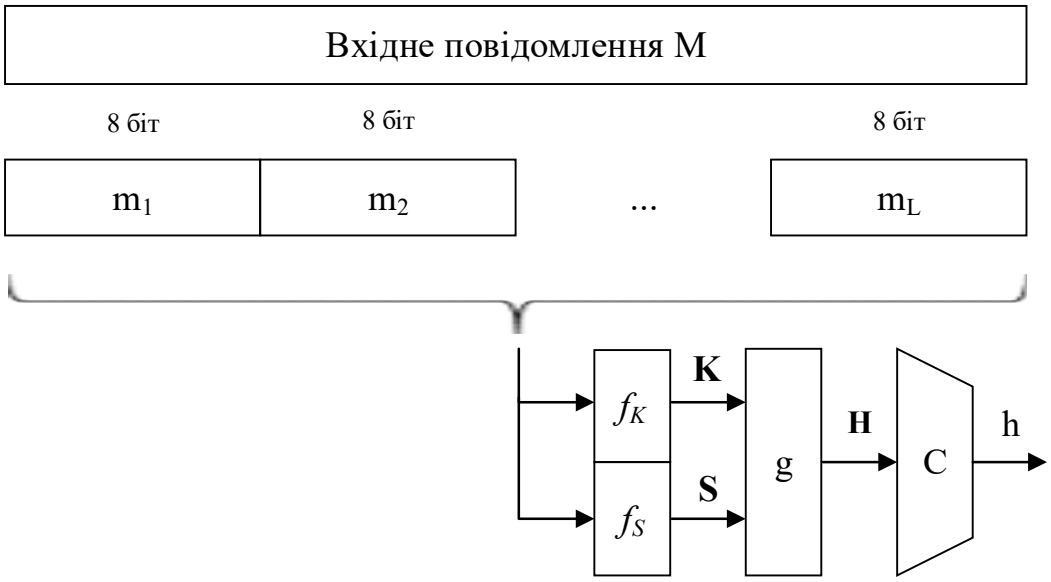

Рисунок 2.8 – Узагальнена схема процесу гешування за методом 2

На відміну від схеми, наведеної для Методу 1 (рис. 2.7), у Методі 2 стадія ущільнення застосовується до результату попереднього змішування статистичних характеристик **K** та **S**. Така організація обробки дозволяє підвищити рівень дифузії фінального геш-значення без переходу до багатораундових ітеративних перетворень, характерних для криптографічних геш-функцій.

2.2.4 Ущільнення змішаних характеристик

Після виконання етапу змішування масив **H** підлягає ущільненню з використанням функції C:

$$h = C\,(\mathbf{H}),$$

де h — фінальне геш-значення фіксованої довжини.

Функція C реалізується як регулярна редукційна операція, аналогічна за структурою функціям C_1 та C_2 у Методі 1, але застосовується до вже змішаних характеристик. Такий підхід дозволяє зберегти апаратну регулярність і

передбачуваність часових характеристик, одночасно підвищуючи рівень дифузії фінального геш-значення.

Функція ущільнення C передбачає виконання таких дій.

Масив **H** перетворюється у масив $\mathbf{H}^{(256)}$ з 256 однобайтних елементів:

$$\mathbf{H}^{(256)} = \left(h_0^{(256)}, h_1^{(256)}, \dots, h_{255}^{(256)}\right),$$

де

$$h_n^{(256)} = \left(\sum_{i=0}^{11} b_{n,i}\right) mod\ 2^8. \qquad (2.5)$$

Далі, масив $\mathbf{H}^{(256)}$перетворюється у масив $\mathbf{H}^{(128)}$аналогічно як у методі 1, і так далі до досягнення розміру масиву в 32 байти.

$$\mathbf{H}^{(128)} \rightarrow \mathbf{H}^{(64)} \rightarrow \mathbf{H}^{(32)}.$$

Результатом обчислень є $h = \mathbf{H}^{(32)}$.

Метод 2 має такі характеристики:

- потокове формування масивів **K** та **S**;
- змішування статистичних характеристик за допомогою функції g(**K**, **S**)*;*
- ущільнення змішаних даних з формуванням геш-значення.

У порівнянні з Методом 1, запропонований підхід потребує додаткових обчислювальних ресурсів на стадії змішування, однак зберігає лінійну часову складність і не потребує багатораундових ітеративних перетворень. Це забезпечує компроміс між апаратною складністю та якісними характеристиками гешування.

Незважаючи на введення додаткового етапу змішування, Метод 2 зберігає ключові властивості, необхідні для ефективної апаратної реалізації:

- регулярну структуру обчислень;
- відсутність складних умовних переходів;
- можливість конвеєризації та паралельної обробки;
- передбачувану латентність.

Функція g(**K**, **S**) може бути реалізована у вигляді окремого апаратного модуля, інтегрованого між стадіями формування характеристик та ущільнення, що дозволяє

адаптувати архітектуру апаратного прискорювача до різних вимог щодо продуктивності та енергоспоживання.

Метод байт-орієнтованого гешування з підвищеним рівнем дифузії розширює можливості лінійного підходу шляхом введення етапу змішування статистичних характеристик. Це дозволяє підвищити якість гешування без суттєвого ускладнення архітектури обробки та зберегти придатність методу до апаратної реалізації. У наступному розділі дисертаційної роботи розглядається реалізація обох методів у вигляді спеціалізованого апаратного процесора гешування у складі багатошарової апаратно-програмної архітектури системи обробки даних.

Таким чином, Метод 2 байт-орієнтованого гешування з підвищеним рівнем дифузії забезпечує покращені властивості розподілу впливу вхідних даних на фінальне геш-значення при збереженні потокового характеру обробки та придатності до апаратної реалізації. Оцінка ефективності запропонованих методів з позиції апаратних витрат, пропускної здатності та системної доцільності їх використання у приватних хмарних інфраструктурах наведена у підрозділі 2.5.

2.4 Розробка алгоритму гешування за методом 2

Для реалізації методу 2 пропонується такий таким алгоритм.

1. Прийняти вхідне повідомлення **M** довжини L.
2. Ініціалізувати масив **K** нульовими елементами.
3. Ініціалізувати масив **S** елементами «криптографічної солі».
4. Для кожного байта m_i, де $i = 0 \ldots L - 1$:
   - збільшити значення k_{m_i} на 1;
   - додати номер позиції i до s_{m_i}.
5. Застосувати функцію змішування масивів **H** = g**(K, S)**.
6. Виконати ущільнення масиву **H** за правилом $h = C$ (**H**).
7. Повернути результат h.

«Криптографічна сіль» формується за правилом функціонування регістра зсуву з лінійним зворотним зв’язком (LFSR).

Узагальнену блок-схему алгоритму обрахунку геш-значення за методом 2 подано на рис. 2.9:

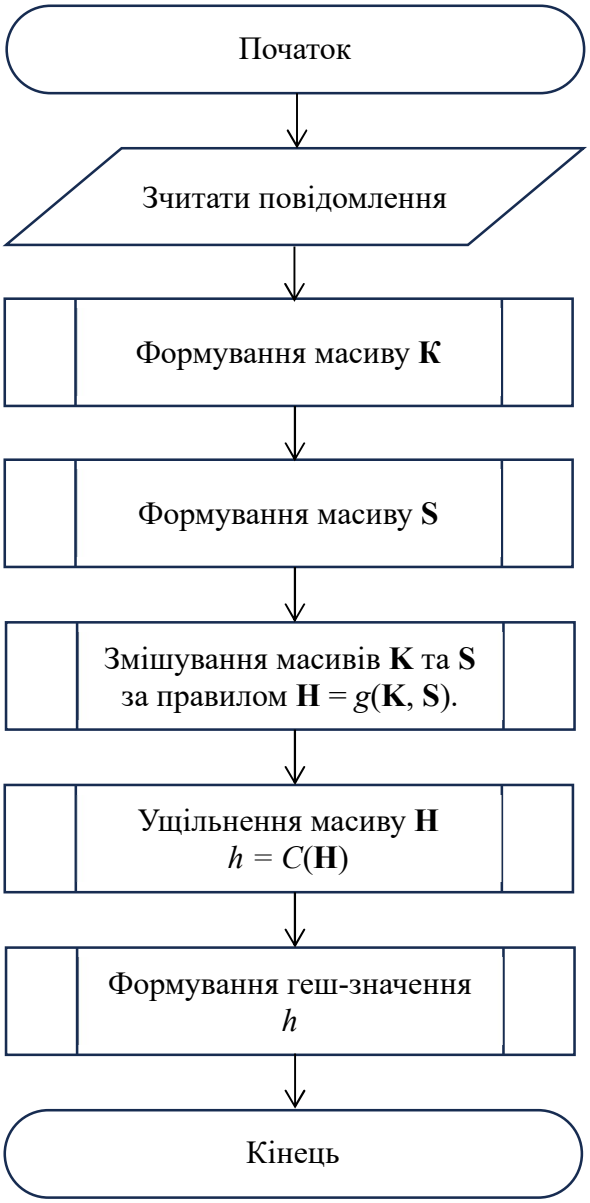

Рисунок 2.9 – Узагальнена блок-схема алгоритму обрахунку геш-значення за методом 2

На наступних етапах формуються масиви **K** та **S** аналогічно до процедур описаних у Методі 1. Далі відбувається процедура змішування характеристик масивів **K**та **S** , яка полягає у послідовній обробці пар відповідних елементів k_n і s_n, де $n = 0 \div 255$, із застосуванням однієї з функцій g_n. На кожному кроці для

поточного індексу n формується проміжний код h_n^* шляхом звичайного арифметичного множення $h_n^* = k_n \cdot s_n$. Отримане значення h_n^* має довжину 96 біт і записується до масиву **H**. Після обробки всіх 256 пар елементів формується підсумковий масив $\mathbf{H} = g(\mathbf{K}, \mathbf{S}) = (h_0^*, h_1^*, \dots, h_{255}^*)$, кожен елемент якого подається у вигляді 12 байтів $h_n^* = (b_{n,11}, b_{n,10}, \dots, b_{n,0})$. Алгоритм робити даної процедури зображено на рис. 2.10.

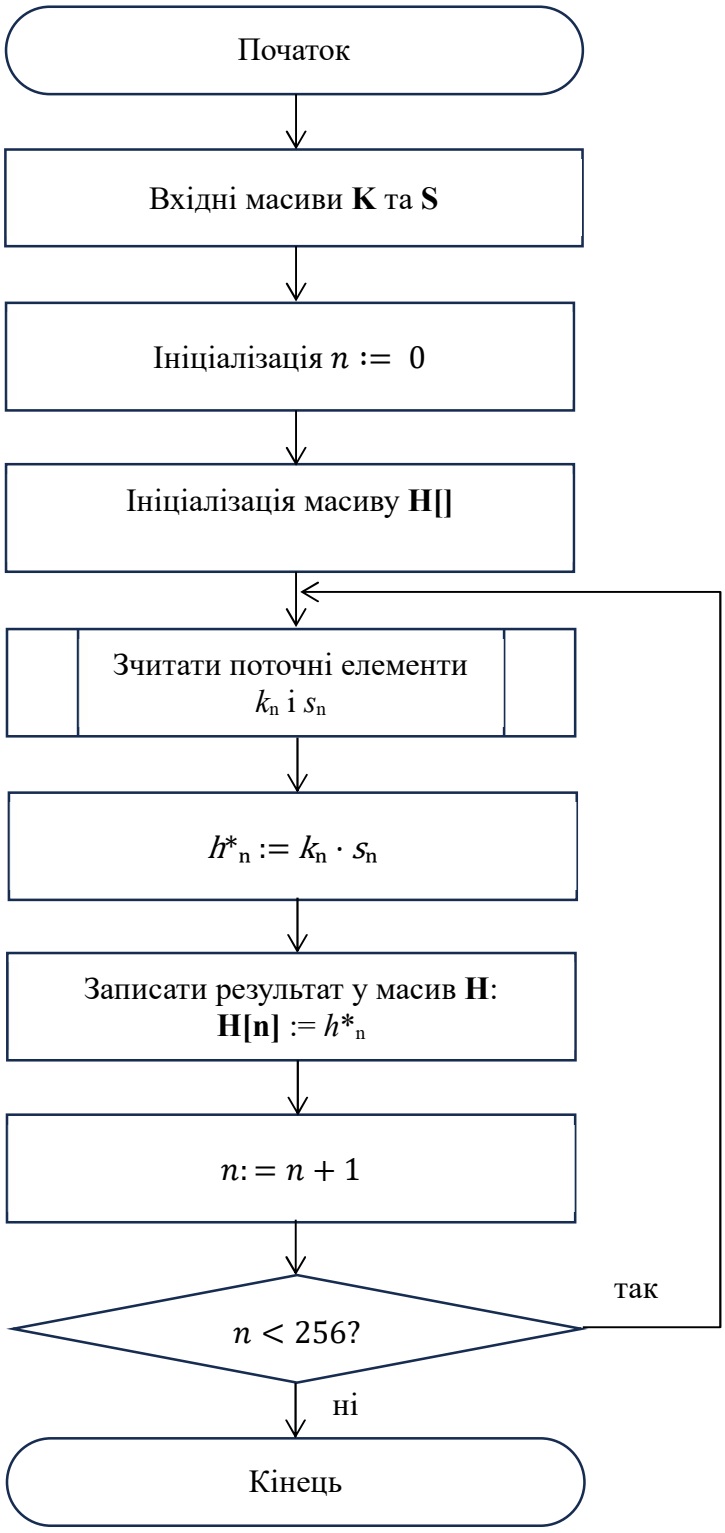

Рисунок 2.10 – Блок-схема алгоритму змішування характеристик масивів **K** та **S**

Після виконання етапу змішування масив **H** підлягає ущільненню за допомогою функції C з метою отримання геш-значення фіксованої довжини. На першому кроці кожний елемент h_n^* масиву **H**, перетворюється в однобайтне

значення $h_n^{(256)}$ шляхом підсумовування всіх 12 байтів за модулем 2^8, у результаті чого формується масив $\mathbf{H}^{(256)}$. Далі до цього масиву послідовно застосовується регулярна редукційна операція на основі операції виключного АБО: спочатку формується масив $\mathbf{H}^{(128)}$, потім $\mathbf{H}^{(64)}$, а далі $\mathbf{H}^{(32)}$. Результатом виконання функції ущільнення є масив $h = \mathbf{H}^{(32)}$, який і є остаточним геш-значенням. Блок-схема алгоритму ущільнення масиву **H** наведена на рис. 2.11.

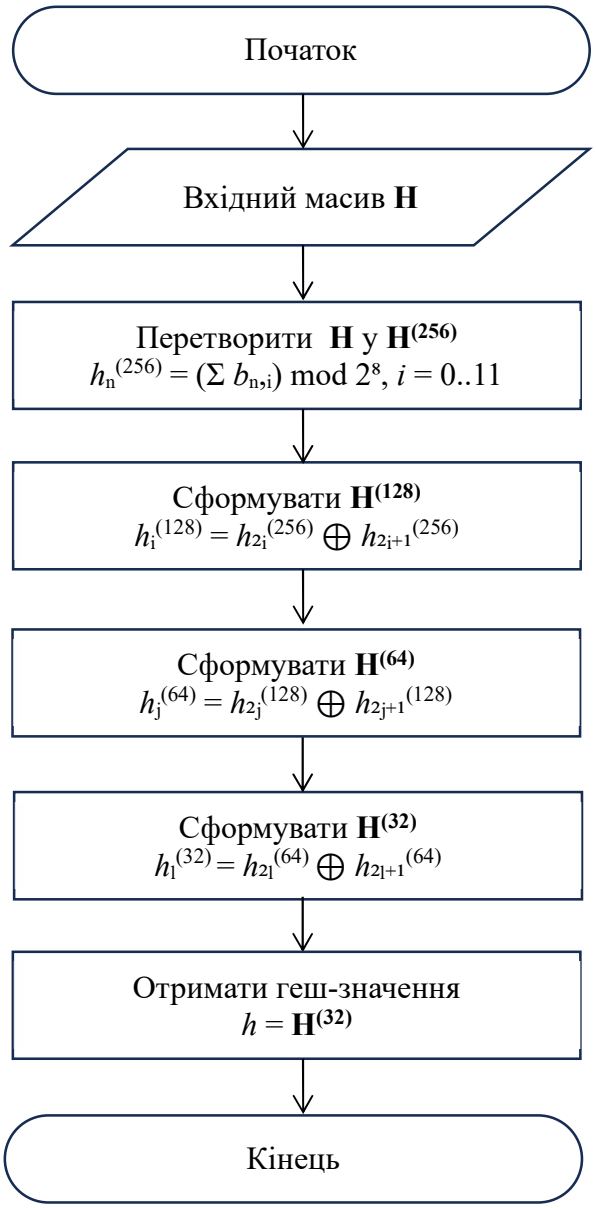

Рисунок 2.11 – Блок-схема алгоритму ущільнення масиву **H**

2.5 Порівняльна оцінка методів

Для обґрунтування доцільності застосування запропонованих методів байт-орієнтованого гешування у системах обробки даних приватних хмарних інфраструктур необхідно виконати їх порівняльну оцінку з позиції апаратної та системної ефективності. Така оцінка дозволяє встановити відповідність методів вимогам до пропускної здатності, латентності, апаратних витрат та масштабованості. У подальшому порівняння виконується на основі визначених критеріїв ефективності.

2.5.1 Критерії оцінки ефективності методів гешування

Порівняльна оцінка методів гешування, призначених для використання у приватних (військових) хмарних інфраструктурах, має ґрунтуватися не лише на формальних алгоритмічних характеристиках, але й на інженерних та системних показниках, що безпосередньо впливають на ефективність обробки потоків даних. З урахуванням специфіки таких середовищ, зокрема обмеженості апаратних ресурсів, підвищених вимог до пропускної здатності та передбачуваності часових характеристик, для порівняння методів гешування доцільно використовувати сукупність взаємодоповнювальних критеріїв [1–4, 20].

Одним із базових критеріїв є алгоритмічна складність, яка визначає залежність обсягу обчислень від довжини вхідного повідомлення та характер виконуваних операцій. Методи з лінійною складністю та сталою кількістю операцій на одиницю даних є більш придатними для апаратної реалізації у потокових системах обробки даних [20, 22].

Важливим інженерним критерієм є кількість та тип базових операцій, що виконуються під час гешування. Наявність складних багатораундових перетворень, великої кількості умовних переходів або побітових операцій істотно ускладнює апаратну реалізацію та знижує енергоефективність системи [11, 14, 16]. Натомість регулярні байт-орієнтовані операції сприяють конвеєризації та паралельній обробці даних.

Окрему роль відіграє придатність методу до паралельної та конвеєрної обробки, яка визначає можливість підвищення пропускної здатності шляхом масштабування апаратної реалізації. Для приватних хмарних інфраструктур, де масштабування за рахунок додавання вузлів часто є обмеженим, ця властивість має принципове значення [3, 4, 22].

З позиції системної ефективності критичними є латентність обробки та детермінованість часових характеристик. Методи, що забезпечують передбачуваний час обробки потоків даних і не залежать від внутрішніх ітеративних процесів, є більш придатними для використання у системах реального та квазіреального часу [28, 29, 31].

Ще одним критерієм є енергоефективність, яка безпосередньо пов’язана з кількістю операцій, глибиною логіки та інтенсивністю доступів до пам’яті. Для відомчих і військових хмарних інфраструктур з обмеженими енергетичними та тепловими ресурсами цей показник має суттєве значення [20, 23].

З точки зору функціонального призначення хмарних сервісів важливим є рівень дифузії та розрізнювальна здатність геш-значень, що визначають придатність методу до задач контролю цілісності, дедуплікації та ідентифікації блоків даних. При цьому вимоги до криптографічної стійкості формуються з урахуванням інженерних потреб системи, а не максимальних криптографічних критеріїв [11, 55, 56].

Узагальнення наведених міркувань дозволяє сформувати перелік критеріїв, які використовуються для подальшої порівняльної оцінки запропонованих методів байт-орієнтованого гешування.

На основі визначених критеріїв у наступному підпункті виконується оцінка алгоритмічної складності реалізації запропонованих методів гешування, а також їх зіставлення з традиційними підходами, що застосовуються у сучасних хмарних системах.

Критерії оцінювання ефективності методів гешування наведено у табл. 2.1.

Таблиця 2.1 – Критерії оцінювання ефективності методів гешування

Критерій	**Інженерний зміст**	**Значення для апаратної та системної ефективності**
Алгоритмічна складність	Залежність обсягу обчислень від довжини повідомлення	Визначає масштабованість і придатність до потокової обробки
Кількість операцій на байт	Кількість базових арифметичних і логічних операцій	Впливає на пропускну здатність і енергоспоживання
Регулярність структури	Відсутність складних умовних переходів	Забезпечує конвеєризацію та простоту апаратної реалізації
Паралельна обробка	Можливість одночасного виконання операцій	Дозволяє підвищувати продуктивність без зміни алгоритму
Латентність	Час обробки одиниці даних	Критично для систем реального та квазіреального часу
Детермінованість часу	Передбачуваність часових характеристик	Підвищує надійність функціонування системи
Енергоефективність	Витрати енергії на обробку даних	Важлива для ресурсно-обмежених середовищ
Рівень дифузії	Чутливість гешу до змін вхідних даних	Знижує ймовірність колізій у задачах дедуплікації
Системна доцільність	Відповідність задачам хмарної інфраструктури	Визначає практичну цінність методу

2.5.2 Оцінка алгоритмічної складності реалізації методів

Для кількісної оцінки алгоритмічної складності реалізації запропонованих методів гешування використовується модель, у якій базовою одиницею складності вважається одна елементарна операція обробки байтових даних. До таких операцій належать операції зчитування та запису з пам’яті, додавання та множення, які у сучасних мікропроцесорах та спеціалізованих апаратних засобах виконуються за порівнянний часовий інтервал. Нехай довжина вхідного повідомлення, що підлягає гешуванню, становить L байтів [20, 22, 23].

Загальна алгоритмічна складність реалізації Методу 1 визначається сумарною складністю виконання основних функціональних компонентів алгоритму і може бути представлена у вигляді:

$$S_{m1} = S_K + S_S + S_{C1} + S_{C2} + S_f,$$

де S_K та S_S— складності формування масивів **K** і **S**, S_{C1}та S_{C2}— складності функцій ущільнення, а S_f— складність фінальної операції комбінування результатів.

Формування масивів **K** і **S** виконується в одному проході по вхідному повідомленню та потребує по три елементарні операції на кожен байт, що визначає:

$$S_K = 3L, S_S = 3L.$$

Складність реалізації функцій ущільнення C_1 і C_2 визначається кількістю операцій, необхідних для редукції масивів фіксованої розмірності, та не залежить від довжини повідомлення L [20, 22]. Для функції C_1 сумарна кількість операцій становить:

$$S_{C1} = 3008,$$

а для функції C_2:

$$S_{C2} = 4992.$$

Складність реалізації функції f залежить від обраного способу комбінування проміжних геш-значень. У разі простої конкатенації кодів h_kта h_sвона становить $S_f = 32$, а у разі використання арифметичного множення — $S_f = 1552$.

Таким чином, загальна алгоритмічна складність реалізації Методу 1 має вигляд:

$$S_{m1}^{(1)} = 6L + 8032,$$

для варіанту конкатенації та

$$S_{m1}^{(2)} = 6L + 9552,$$

для варіанту з множенням. В обох випадках асимптотична складність методу є лінійною:

$$S_{m1} \in O(L).$$

Алгоритмічна складність реалізації Методу 2 визначається аналогічним чином:

$$S_{m2} = S_K + S_S + S_g + S_C,$$

де S_g— складність реалізації функції змішування статистичних характеристик, а S_C— складність функції ущільнення.

Складність формування масивів **K** та **S** залишається такою ж, як і у Методі 1:

$$S_K = 3L, S_S = 3L.$$

Складність реалізації функції g залежить від обраного способу формування проміжного масиву характеристик. У разі реалізації на основі операцій зчитування та запису байтів вона становить:

$$S_g^{(1)} = 6144,$$

а у разі використання арифметичного множення — істотно зростає:

$$S_g^{(2)} = 43008.$$

Складність реалізації функції ущільнення C не залежить від L та дорівнює:

$$S_C = 7040.$$

З урахуванням наведеного, алгоритмічна складність реалізації Методу 2 становить:

$$S_{m2}^{(1)} = 6L + 13184,$$

для варіанту з простими операціями та

$$S_{m2}^{(2)} = 6L + 50048,$$

для варіанту з арифметичним множенням. В обох випадках асимптотична складність також є лінійною. Оцінки наведено у табл. 2.2.

$$S_{m2} \in O(L).$$

Таблиця 2.2 – Числові оцінки алгоритмічної складності реалізації запропонованих методів гешування

Метод	**Варіант реалізації**	**Формула алгоритмічної складності S**	**Стала складова**	**Асимпто-тика**
Метод 1	f: конкатенація $h_k \parallel h_s$	6L+8032	8032	O(L)
	f: множення $h_k \cdot h_s$	6L+9552	9552	O(L)
Метод 2	g: дописування (зчит./запис байтів)	6L+13184	13184	O(L)
	g: множення (32-біт на 64-біт)	6L+50048	50048	O(L)

L — довжина вхідного повідомлення у байтах; оцінка складності виконується в умовних одиницях, де одна елементарна операція (зчитування/запис /додавання/множення) приймається за одиницю складності.

Отримані оцінки свідчать, що обидва запропоновані методи гешування характеризуються лінійною алгоритмічною складністю та сталою кількістю операцій на байт вхідних даних. При цьому вплив сталої складової на загальну кількість операцій зменшується зі зростанням довжини повідомлення [11–13, 31]. Для Методу 1 домінування лінійної складової досягається при довжині повідомлення понад 13–15,5 кілобайт, а для Методу 2 — понад 21,5–81,5 кілобайт залежно від варіанту реалізації.

Таким чином, обидва методи є придатними для обробки великих обсягів даних у потоковому режимі, характерному для приватних хмарних інфраструктур. Відмінності у значеннях сталої складової визначають компроміс між апаратною складністю реалізації та рівнем дифузії геш-значень наведено у табл. 2.3, що буде враховано при подальшій системній оцінці методів у підрозділі 2.6.3.

Таблиця 2.3 – Узагальнена оцінка алгоритмічної складності запропонованих методів гешування

Метод	**Асимптотична складність**	**Стала складова (порядок)**	**Характер реалізації**
Метод 1	O(L)	~10^4	Лінійна, низька апаратна складність
Метод 2	O(L)	~$10^4 - 10^5$	Лінійна, підвищена дифузія

2.5.3 Порівняння апаратних витрат і латентності

Поряд з алгоритмічною складністю важливими показниками ефективності методів гешування в контексті приватних (військових) хмарних інфраструктур є апаратні витрати та латентність обробки потоків даних. Саме ці параметри визначають доцільність інтеграції методу у багатошарову апаратно-програмну

архітектуру системи обробки даних та можливість його використання у режимах, наближених до реального часу.

Під апаратними витратами у даному дослідженні розуміється сукупність апаратних ресурсів, необхідних для реалізації методу гешування, зокрема кількість регістрів, лічильників, акумуляторів, логічних елементів та глибина комбінаційної логіки.

Метод 1 характеризується мінімальними апаратними витратами на основній стадії обробки вхідного потоку. Формування масивів **K** та **S** потребує використання фіксованого набору лічильників і акумуляторів, кількість яких не залежить від довжини повідомлення. Стадія ущільнення реалізується у вигляді регулярного редукційного дерева з фіксованою глибиною, що обмежує зростання апаратної складності та спрощує масштабування.

Метод 2, у порівнянні з Методом 1, потребує додаткових апаратних ресурсів, пов’язаних із формуванням проміжного масиву **H** та виконанням операцій змішування статистичних характеристик. Особливо це проявляється у варіантах реалізації, що використовують арифметичне множення кодів, яке вимагає більш складних апаратних блоків та збільшує площу логіки. Водночас ці витрати є контрольованими та не залежать від довжини вхідного повідомлення, що є важливим для систем з обмеженими ресурсами.

Таким чином, з позиції апаратних витрат Метод 1 є більш економним і придатним для реалізації у ресурсно-обмежених середовищах, тоді як Метод 2 реалізує компроміс між апаратною складністю та підвищеним рівнем дифузії геш-значень.

Латентність обробки визначається часом, необхідним для формування геш-значення після надходження вхідного потоку даних. У системах потокової обробки критичним є не лише абсолютне значення латентності, але й її детермінованість.

У Методі 1 латентність формується двома складовими: однопрохідною обробкою вхідного потоку та виконанням фіксованої послідовності операцій ущільнення. Оскільки кількість операцій на кожен байт є сталою, а стадія ущільнення має детерміновану тривалість, загальна латентність методу є

передбачуваною та добре масштабується при апаратній реалізації. Це робить Метод 1 придатним для використання у контурах контролю цілісності та дедуплікації з жорсткими часовими вимогами.

Метод 2 також забезпечує лінійну залежність латентності від довжини повідомлення, однак додаткова стадія формування проміжних характеристик і складніші операції змішування збільшують абсолютне значення затримки. У варіантах реалізації з використанням множення латентність зростає за рахунок більшої глибини комбінаційної логіки. Проте латентність залишається детермінованою, що є важливою перевагою у порівнянні з багатораундовими криптографічними геш-функціями.

Порівняльний аналіз апаратних витрат і латентності свідчить, що обидва запропоновані методи байт-орієнтованого гешування орієнтовані на ефективну апаратну реалізацію та забезпечують передбачувані часові характеристики. Метод 1 є більш доцільним для використання у ресурсно-обмежених середовищах та задачах з жорсткими вимогами до латентності, тоді як Метод 2 забезпечує підвищений рівень дифузії за рахунок збільшення апаратних витрат і латентності. Порівняння апаратних витрат Методу 1 та Методу 2 наведено у табл. 2.4.

Отримані результати підтверджують доцільність подальшої розробки спеціалізованого апаратного процесора байт-орієнтованого гешування, архітектура якого дозволяє реалізувати обидва методи у різних режимах роботи в складі багатошарової апаратно-програмної архітектури приватних (військових) хмарних інфраструктур, що буде розглянуто у розділі 3.

Таблиця 2.4 – Порівняння апаратних витрат Методу 1 та Методу 2

Критерій	**Метод 1**	**Метод 2**
Базові апаратні елементи	Лічильники, акумулятори, проста логіка	Лічильники, акумулятори, блоки змішування
Глибина комбінаційної логіки	Низька	Середня / підвищена
Обсяг регістрової пам’яті	Фіксований, мінімальний	Фіксований, збільшений
Паралелізація	Висока, регулярна	Висока, з додатковими стадіями
Масштабованість	Пряма, без зміни логіки	Пряма, з ростом апаратних витрат
Латентність обробки	Мінімальна, детермінована	Підвищена, детермінована
Чутливість латентності до реалізації	Низька	Середня (залежить від g)
Орієнтація на ресурси	Ресурсно-обмежені системи	Системи з підвищеними вимогами до дифузії
Типові задачі застосування	Контроль цілісності, дедуплікація	Ідентифікація, підвищена розрізнювальність

2.5.4 Системна доцільність застосування методів у приватних хмарах

Оцінка доцільності застосування методів гешування у приватних хмарних інфраструктурах має виконуватися з урахуванням системного контексту їх використання, а саме архітектури обробки даних, режимів функціонування сервісів та обмежень апаратно-програмного середовища. На відміну від публічних хмар, приватні хмарні інфраструктури характеризуються фіксованою апаратною базою, обмеженими можливостями масштабування та підвищеними вимогами до передбачуваності часових характеристик і надійності функціонування [1–4, 28, 29].

У таких умовах методи гешування виступають не ізольованими алгоритмами, а функціональними компонентами контурів обробки даних, зокрема контурів контролю цілісності, дедуплікації, ідентифікації блоків та маршрутизації інформаційних потоків. Тому системна доцільність застосування методу

визначається його відповідністю вимогам конкретного контуру та режиму роботи системи [11–13, 31, 32].

Метод 1, орієнтований на лінійну архітектуру обробки з мінімальними апаратними витратами та передбачуваною латентністю, є доцільним для використання у контурах, що функціонують у режимі безперервної потокової обробки. До таких контурів належать первинний прийом даних, контроль цілісності під час передачі та зберігання, а також механізми дедуплікації у сховищах великих обсягів даних. У цих сценаріях ключовими є стабільна пропускна здатність, низька затримка та можливість інтеграції методу без істотного збільшення навантаження на центральні процесори [20, 22, 31, 32].

Завдяки детермінованому характеру обчислень і лінійній залежності часових витрат від довжини повідомлення, Метод 1 добре узгоджується з архітектурою систем реального та квазіреального часу. Це робить його придатним для використання у відомчих і військових інформаційних системах, де важливими є оперативність реагування та гарантована стабільність функціонування [3, 4, 28, 29].

Метод 2, який забезпечує підвищений рівень дифузії геш-значень за рахунок додаткових етапів змішування статистичних характеристик, є доцільним у контурах, де пріоритетом є розрізнювальна здатність ідентифікації даних. До таких контурів належать системи адресації блоків у розподілених сховищах, механізми індексації та ідентифікації об’єктів, а також сценарії, у яких допустиме збільшення латентності в обмін на зниження ймовірності колізій [11, 55, 56, 80, 83].

З системної точки зору Метод 2 доцільно застосовувати у фонових або асинхронних режимах обробки, де часові обмеження є менш жорсткими, а обчислювальні ресурси можуть бути виділені для виконання додаткових операцій. Такий підхід дозволяє використовувати переваги підвищеної дифузії без негативного впливу на критичні шляхи обробки даних [20, 31, 32].

Важливою особливістю запропонованих методів є їх взаємна комплементарність з точки зору системної архітектури. Наявність двох методів з різними характеристиками дозволяє реалізувати гнучку стратегію використання гешування у межах однієї приватної хмарної інфраструктури, обираючи метод

залежно від функціонального призначення контуру обробки та поточного режиму роботи системи.

Таким чином, системна доцільність застосування запропонованих методів гешування полягає у можливості їх цілеспрямованого використання у різних компонентах приватної (військової) хмарної інфраструктури з урахуванням вимог до латентності, пропускної здатності, апаратних витрат і рівня дифузії. Це створює передумови для побудови адаптивної багатошарової апаратно-програмної архітектури системи обробки даних, у якій методи гешування виступають ключовими інженерними елементами [3, 4, 20, 22].

З урахуванням результатів порівняльної оцінки алгоритмічної складності, апаратних витрат, латентності та системної доцільності застосування, доцільно узагальнити характеристики запропонованих методів байт-орієнтованого гешування у вигляді компактної матриці вибору. Така матриця дозволяє здійснювати обґрунтований вибір методу залежно від вимог конкретного контуру обробки даних у приватній (військовій) хмарній інфраструктурі. Узагальнююча таблиця вибору методу байт-орієнтованого гешування наведена у табл. 2.5.

Таблиця 2.5 – Узагальнююча таблиця вибору методу байт-орієнтованого гешування

Критерій / Умова застосувааня	**Метод 1**	**Метод 2**
Тип обробки даних	Безперевна потокова	Потокова / пакетна
Режим роботи системи	Реальний / квазіреальний час	Асинхронний / фоновий
Обмеження апаратних ресурсів	Жорсткі	Помірні
Пріоритет пропускної здатності	Високий	Середній
Допустима латентність	Мінімальна	Підвищена
Детермінованість часу обробки	Висока	Висока
Рівень дифузії геш-значень	Базовний	Підвищений
Ймовірність колізії	Прийнятна для інженерних задач	Знижена
Апаратні витрати	Мінімальні	Підвищені
Типові контури застосування	Контроль цілісності, дедуплікація	Ідентифікація, індексація
Системна роль	Основний (default) метод	Додатковий / спеціалізований метод

2.5.5 Порівняння запропонованих методів із сучасними методами гешування

У сучасних системах обробки даних застосовується широкий спектр методів гешування, які відрізняються за призначенням, архітектурними принципами та вимогами до обчислювальних ресурсів. У межах даного підрозділу виконано порівняльний аналіз запропонованих байт-орієнтованих методів гешування з представниками трьох актуальних класів сучасних рішень: lightweight криптографічних методів, високопродуктивних некриптографічних геш-функцій та методів контент-орієнтованого гешування, що використовуються у системах зберігання даних [11, 14–16, 55, 56].

При цьому слід наголосити, що запропоновані методи не конкурують безпосередньо з сучасними криптографічними або програмно-орієнтованими високопродуктивними геш-функціями, а формують окремий клас байт-орієнтованих методів, оптимізованих для детермінованої апаратної реалізації у потокових контурах приватних хмарних інфраструктур [20, 22, 31, 32].

Окремої уваги у контексті архітектурно-системного порівняння заслуговує конструкція KECCAK, яка лежить в основі стандарту SHA-3 і є представником класу sponge-функцій. На відміну від класичних ітеративних схем типу Merkle–Damgård, KECCAK не використовує компактний ланцюговий внутрішній стан, однак реалізує багатораундову криптографічну перестановку над великим sponge-станом фіксованої розмірності. Таким чином, KECCAK не є ітеративним методом гешування у класичному розумінні, проте характеризується раундово-ітеративною обробкою на рівні внутрішнього перетворення стану.

З архітектурної точки зору така організація обчислень забезпечує високий рівень криптографічної стійкості, однак ускладнює апаратну реалізацію у вигляді простих спеціалізованих модулів для потокових контурів обробки даних. Наявність великого внутрішнього стану та фіксованої кількості раундів перестановки зумовлює підвищені апаратні витрати і менш передбачувану латентність у порівнянні з байт-орієнтованими методами, що ґрунтуються на однопрохідній обробці вхідного потоку. У межах даного дослідження KECCAK розглядається як архітектурний еталон sponge-конструкцій, що дозволяє чітко окреслити відмінності

між криптографічно орієнтованими методами гешування та запропонованим класом байт-орієнтованих потокових методів, оптимізованих для детермінованої апаратної реалізації у приватних хмарних інфраструктурах [20, 23, 28, 29].

До сучасних lightweight криптографічних методів гешування належать, зокрема, ASCON та PHOTON, які розроблялися з урахуванням обмежених ресурсів вбудованих систем. Ці алгоритми орієнтовані на забезпечення криптографічної стійкості та використовують багатоітераційні схеми перетворень, зокрема sponge-архітектуру.

З архітектурної точки зору такі методи характеризуються наявністю компактного внутрішнього стану, що багаторазово оновлюється, а також значною кількістю логічних і нелінійних операцій. Це ускладнює досягнення детермінованої латентності та підвищує апаратні витрати при реалізації у потокових контурах обробки даних.

На відміну від них, запропоновані байт-орієнтовані методи не використовують ітеративного внутрішнього стану та не орієнтовані на досягнення максимального рівня криптографічної стійкості. Їх основною перевагою є регулярна структура обчислень і можливість інтеграції безпосередньо у критичний шлях потокової обробки даних [14, 16, 23].

До класу високопродуктивних некриптографічних геш-функцій належать xxHash, CityHash та подібні алгоритми, які широко застосовуються у програмних системах для задач ідентифікації та дедуплікації даних. Ці методи оптимізовані для виконання на сучасних центральних процесорах і активно використовують операції над 64- та 128-бітними словами, кеш-ієрархію та SIMD-інструкції.

З позиції апаратної реалізації такі алгоритми є менш придатними для використання у вигляді простих спеціалізованих модулів, оскільки їх ефективність суттєво залежить від особливостей програмної архітектури процесора. Крім того, орієнтація на обробку по словах ускладнює їх використання у байт-орієнтованих потокових середовищах.

Запропоновані методи, навпаки, спроєктовані з урахуванням апаратної реалізації на байтовому рівні та не залежать від наявності складних програмних

оптимізацій. Це забезпечує стабільну пропускну здатність і передбачувану латентність незалежно від конкретної платформи [12, 13, 20, 31].

Методи контент-орієнтованого гешування, зокрема алгоритми на основі відбитків Рабіна, широко застосовуються у системах дедуплікації зі змінним розміром блоків. Вони забезпечують високу чутливість до змін вхідних даних, однак мають недетерміновану латентність і потребують складної логіки аналізу вхідного потоку.

У контексті приватних та військових хмарних інфраструктур такі властивості є небажаними, оскільки ускладнюють інтеграцію методу у критичні шляхи обробки та не гарантують передбачуваних часових характеристик. Запропоновані байт-орієнтовані методи забезпечують фіксовану структуру обробки і можуть використовуватися у системах з жорсткими вимогами до детермінованості [55, 56, 32]. Архітектурно-системне порівняння методів гешування наведено у табл. 2.6.

Таблиця 2.6 – Архітектурно-системне порівняння методів гешування

Критерій	**Запропоновані методи**	**KECCAK (sponge)**	**ASCON / PHOTON**	**xxHash / CityHash**	**Rabin (CDC)**
Оірєнтація	Апаратна, байт-орієнтована	Криптографічна (базова конструкція)	Lightweight crypto	Програмна	Сховища
Тип обробки	Потокова, детермінована	Absorb/Squeeze, раундова	Ітеративна	Словна	Адаптивна
Внутрішній стан	Відсутній / фіксований	Великий sponge-стан	Компактний стан	Залежить від CPU	Ковзне вікно
Ітеративність	Відсутня	Раундово-ітеративна	Раундова	Залежна від реалізації	Змінна
Детермінованість латентності	Висока	Середня	Середня	Платформно-залежна	Низька
Придатність до спецпроцесора	Висока	Обмежена	Середня	Низька	Низька
Основне призначення	Цілісність, дедуплікація	Криптографічний геш	Захищені IoT	Швидке ПЗ-хешування	CDC-дедуплікація

Проведений аналіз показує, що запропоновані байт-орієнтовані методи гешування займають окрему нішу між криптографічними та високопродуктивними програмними підходами. Їх ключовою перевагою є поєднання регулярної структури, детермінованих часових характеристик і придатності до апаратної

реалізації, що робить їх доцільними для використання у потокових контурах приватних (військових) хмарних інфраструктур.

2.6 Висновок

У розділі 2 дисертаційної роботи розроблено та досліджено методи байт-орієнтованого гешування, призначені для використання у системах обробки даних приватних (військових) хмарних інфраструктур. На основі аналізу особливостей таких середовищ показано, що ключовими вимогами до методів гешування є детермінованість часових характеристик, висока пропускна здатність, енергоефективність та придатність до апаратної реалізації в умовах обмежених обчислювальних ресурсів [1–4, 20].

У підрозділах 2.1–2.3 обґрунтовано доцільність застосування апаратно-орієнтованих підходів до гешування потоків даних та сформульовано принципи побудови байт-орієнтованих методів гешування, які відрізняються регулярною структурою обчислень, мінімальною кількістю операцій та можливістю масштабування при апаратній реалізації [37–39]. Показано, що відмова від складних багатораундових криптографічних перетворень дозволяє оптимізаувати процес обробки даних, зменшити апаратні витрати та латентність обробки без втрати системної доцільності у задачах контролю цілісності, дедуплікації та ідентифікації блоків даних [11, 55, 56].

У підрозділі 2.4 розроблено метод байт-орієнтованого гешування з лінійною архітектурою обробки (Метод 1), який базується на однопрохідному формуванні статистичних характеристик вхідного потоку даних та їх подальшому ущільненні. Запропонований метод характеризується лінійною алгоритмічною складністю, детермінованою латентністю та мінімальними апаратними витратами, що робить його придатним для інтеграції у потокові контури обробки даних приватних хмарних сервісів [20, 23].

У підрозділі 2.5 розглянуто модифікований метод байт-орієнтованого гешування (Метод 2), який забезпечує підвищений рівень дифузії геш-значень за рахунок додаткових етапів змішування статистичних характеристик. Показано, що

Метод 2 зберігає лінійну залежність алгоритмічної складності від довжини повідомлення, проте характеризується збільшеною сталою складовою, що визначає компроміс між апаратними витратами та розрізнювальною здатністю геш-значень [22, 31, 32].

У підрозділах 2.6.1–2.6.3 виконано порівняльну оцінку запропонованих методів з позиції алгоритмічної складності, апаратних витрат і латентності. Отримані результати підтверджують, що обидва методи мають асимптотичну складність $O(L)$, а їх часові та апаратні характеристики є передбачуваними, що є критично важливим для систем потокової обробки даних у приватних (військових) хмарних інфраструктурах [3, 4, 28, 29].

У підрозділах 2.6.4–2.6.6 обґрунтовано системну доцільність застосування запропонованих методів у різних контурах хмарної інфраструктури та виконано їх архітектурно-системне порівняння з сучасними lightweight криптографічними методами (ASCON, PHOTON), високопродуктивними некриптографічними геш-функціями (xxHash, CityHash) та методами контент-орієнтованого гешування на основі відбитків Рабіна. Показано, що запропоновані методи формують окремий клас байт-орієнтованих потокових методів, оптимізованих для детермінованої апаратної реалізації, і не конкурують безпосередньо з криптографічно або програмно орієнтованими рішеннями [11, 14–16, 55].

Узагальнюючи результати розділу, встановлено, що Метод 1 є доцільним для використання у ресурсно-обмежених потокових контурах з жорсткими вимогами до латентності, тоді як Метод 2 доцільно застосовувати у контурах, де пріоритетом є підвищена розрізнювальна здатність та допустиме збільшення апаратних витрат. Комплементарний характер запропонованих методів створює передумови для їх спільного використання у межах однієї приватної хмарної інфраструктури.

Отримані у розділі 2 результати демонструють, що запропоновані байт-орієнтовані методи гешування є інженерно обґрунтованими з точки зору алгоритмічної складності, системної доцільності та придатності до використання у потокових контурах приватних (військових) хмарних інфраструктур. Аналіз показав, що детермінований характер обчислень, лінійна залежність часових витрат

від довжини вхідного повідомлення та контрольовані апаратні витрати створюють передумови для ефективної апаратної реалізації запропонованих методів.

У зв’язку з цим подальшим логічним кроком є розробка та дослідження засобів реалізації запропонованих методів у складі багатошарової апаратно-програмної архітектури системи обробки даних. Розділ 3 присвячено архітектурним рішенням інтеграції байт-орієнтованого гешування у приватні хмарні інфраструктури, проєктуванню спеціалізованого МН-процесора, а також аналізу апаратних і програмних реалізацій та експериментальній оцінці їх ефективності..

РОЗДІЛ 3
ЗАСОБИ ОРГАНІЗАЦІЇ РОБОТИ З ДАНИМИ У ПРИВАТНИХ ХМАРНИХ СЕРВІСАХ

3.1 Архітектура багатошарової апаратно-програмної системи обробки даних

3.1.1 Узагальнена багатошарова архітектура

Системи обробки даних приватних та військових хмарних інфраструктур характеризуються багаторівневою організацією, у межах якої програмні та апаратні компоненти взаємодіють для забезпечення необхідних показників продуктивності, надійності та безпеки. На відміну від публічних хмарних середовищ, такі системи функціонують на фіксованій апаратній базі та мають обмежені можливості масштабування, що зумовлює підвищені вимоги до передбачуваності часових характеристик і ефективності використання ресурсів [1–4, 28, 29].

У таких системах методи обробки даних, зокрема методи гешування, не функціонують ізольовано, а інтегруються в архітектуру як складові окремих функціональних контурів — контролю цілісності, дедуплікації, ідентифікації та маршрутизації потоків даних. Ефективність роботи цих контурів безпосередньо впливає на пропускну здатність і латентність всієї хмарної інфраструктури [11–13, 31, 32, 55, 56].

З урахуванням результатів, отриманих у розділі 2, доцільним є розгляд багатошарової апаратно-програмної архітектури, у якій байт-орієнтовані методи гешування реалізуються спеціалізованими засобами та взаємодіють з іншими компонентами системи через стандартизовані інтерфейси. Такий підхід дозволяє розмежувати функції обробки даних за рівнями абстракції та забезпечити гнучкість інтеграції запропонованих методів у різні конфігурації приватних хмарних середовищ [20, 22].

Архітектуру системи обробки даних приватної хмарної інфраструктури пропонується представити у вигляді сукупності взаємопов’язаних рівнів, що показано на рис. 3.1:

- рівень прикладних сервісів, який включає сервіси зберігання, обробки та аналізу даних і формує запити до підсистем нижчого рівня;
- програмний рівень обробки даних, на якому реалізуються алгоритми керування потоками, логіка сервісів та програмні механізми контролю цілісності;
- рівень апаратного пришвидшення обчислень, призначений для виконання обчислювально інтенсивних або критичних за латентністю операцій;
- спеціалізований апаратний рівень, на якому реалізуються апаратно-орієнтовані методи гешування у вигляді окремих функціональних модулів або процесорів.

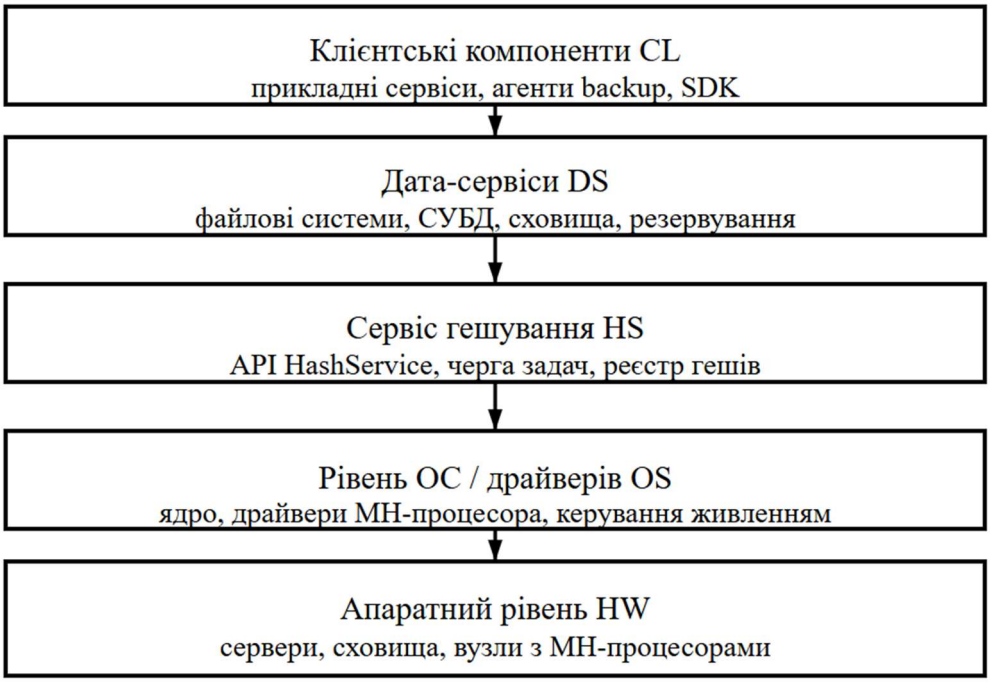

Рисунок 3.1 – Багатошарова апаратно-програмна архітектура приватної хмарної інфраструктури з інтеграцією сервісу байт-орієнтованого гешування

У межах такої архітектури запропоновані байт-орієнтовані методи гешування займають проміжне положення між програмним рівнем і спеціалізованим

апаратним рівнем, забезпечуючи ефективну обробку потоків даних без істотного навантаження на центральні процесори.

У типовому сценарії інтеграції байт-орієнтоване гешування використовується на етапах прийому та зберігання даних у приватному хмарному сховищі. Вхідний потік даних надходить від прикладного сервісу до програмного рівня, де здійснюється його попередня обробка та маршрутизація. Далі дані передаються на рівень апаратного прискорення або безпосередньо до спеціалізованого модуля гешування, який виконує обчислення геш-значень у потоковому режимі.

Отримані геш-значення використовуються для реалізації механізмів контролю цілісності, виявлення дубльованих блоків або ідентифікації об’єктів у системі зберігання. При цьому програмний рівень отримує результати гешування через стандартизований інтерфейс і не потребує знань про внутрішню організацію апаратного модуля. Така схема інтеграції забезпечує прозорість використання спеціалізованих засобів та спрощує адаптацію системи до різних режимів роботи.

Ключовою особливістю запропонованої архітектури є чітке розмежування відповідальності між рівнями. Програмний рівень відповідає за логіку керування потоками та прийняття рішень, тоді як спеціалізований апаратний рівень виконує детерміновані операції гешування з мінімальною латентністю. Обмін даними між рівнями здійснюється у вигляді потоків байтів і супровідних метаданих, що узгоджується з байт-орієнтованим характером запропонованих методів.

Такий підхід дозволяє масштабувати систему як за рахунок програмних механізмів, так і шляхом апаратного розширення, не змінюючи загальної логіки роботи сервісів. Крім того, наявність окремого спеціалізованого рівня створює передумови для підтримки кількох режимів гешування в межах одного апаратного модуля, що є важливим для адаптації системи до різних функціональних контурів.

Таким чином, багатошарова апаратно-програмна архітектура забезпечує системну основу для реалізації запропонованих байт-орієнтованих методів гешування у приватних (військових) хмарних інфраструктурах. У наступному підрозділі буде розглянуто спеціалізований МН-процесор байт-орієнтованого гешування як ключовий елемент апаратного рівня цієї архітектури.

У межах даної архітектури сервіс гешування HS та спеціалізований МН-процесор реалізують запропоновані в роботі методи байт-орієнтованого гешування.

3.1.2 Модель багатошарової апаратно-програмної архітектури обробки дани

З метою узагальненого опису процесів організації роботи з даними у приватній (військовій) хмарній інфраструктурі доцільно виконати формалізацію запропонованої багатошарової апаратно-програмної архітектури. Така формалізація дозволяє перейти від описового рівня до аналітичного представлення взаємодії компонентів системи та створює основу для подальшої кількісної оцінки ефективності реалізації запропонованих методів байт-орієнтованого гешування, що відповідає сучасним підходам до проєктування хмарних та відомчих інформаційних систем.

Модель підсистеми організації роботи з даними у приватній хмарі описується як впорядкована п’ятірка рівнів:

$$\mathcal{A} = \langle HW, OS, HS, DS, CL \rangle, \tag{3.1}$$

де: HW— апаратний рівень, що включає серверні вузли, системи зберігання та спеціалізовані обчислювальні модулі з МН-процесорами;

OS— рівень системного програмного забезпечення та драйверів, який забезпечує керування апаратними ресурсами, планування задач та взаємодію з МН-процесором;

HS— рівень сервісу гешування, що реалізує байт-орієнтовані методи обробки даних та надає уніфікований інтерфейс доступу до функцій гешування;

DS— рівень дата-сервісів (файлові системи, СУБД, об’єктні сховища, системи резервного копіювання);

CL— рівень клієнтських компонентів (SDK, агенти резервування, мережеві шлюзи, прикладні сервіси).

Особливістю даної моделі є наявність окремого функціонального рівня сервісу гешування HS, який інтегрується у різні контури обробки даних без

дублювання логіки гешування у клієнтських або прикладних компонентах, що знижує навантаження на центральні процесори та підвищує керованість системи [36 - 38].

3.1.3 Потокова модель обробки даних

Нехай у системі обробляється об’єкт даних d, який залежно від типу дата-сервісу розбивається на множину елементарних блоків:

$$d = \{b_1, b_2, \dots, b_k\}. \quad (3.2)$$

Для кожного блока даних сервіс гешування формує відповідне геш-значення:

$$h_i = H(b_i), i = 1, \dots, k \quad (3.3)$$

де $H(\cdot)$— реалізація байт-орієнтованого методу гешування (Метод 1 або Метод 2).

Такий підхід є узгодженим з моделями потокової обробки даних у файлових системах, СУБД та мережевих підсистемах, де базовою одиницею обробки виступає байт або блок байтів фіксованої довжини.

Залежно від контексту використання:

- у файлових та об’єктних сховищах b_iвідповідають фіксованим або змінним блокам файлів;

- у СУБД b_i відповідають сторінкам або логічним записам;

- у мережевих підсистемах b_i^{net} відповідають фрагментам трафіку або пакетам.

Таким чином, сервіс гешування реалізує уніфікований механізм обробки даних незалежно від джерела їх походження, що є ключовою вимогою для приватних хмарних інфраструктур з гетерогенними сервісами.

Запропонована архітектура передбачає використання байт-орієнтованих методів гешування у кількох базових функціональних контурах із застосування спеціалізованого МН-процесора (МН - Message Hashing).

У контурі контролю цілісності для кожного блока даних, що зберігається або передається, зберігається еталонне геш-значення h_i. Під час перевірки цілісності обчислюється:

$$\hat{h}_i = H(b_i), \tag{3.4}$$

після чого виконується перевірка:

$$\hat{h}_i =? h_i. \tag{3.5}$$

Операція формування геш-значення для окремого блока даних реалізується у блоці прийому та інтерфейсу потокових даних PI (Processing Input).

Цей блок відповідає за:

- прийом байтового потоку b_i з рівня DSабо CL;
- синхронізацію потоку;
- послідовну передачу байтів до обчислювальних модулів.

Таким чином, блок PI є апаратною реалізацією оператора подання даних b_i та забезпечує однопрохідну обробку, що є ключовим для детермінованої латентності.

Обчислення геш-функції $H(\cdot)$ у запропонованих методах базується на формуванні масивів характеристик **K** та **S**, що відповідає першому етапу Методу 1 і Методу 2. Ці операції реалізуються у блоці PC (Processing Core).

Саме у цьому блоці апаратно реалізується:

- накопичення частотних характеристик;
- обчислення сум номерів позицій байтів;
- реалізація регулярних арифметичних операцій без умовних переходів.

Таким чином, блок PC відповідає внутрішній реалізації оператора $H(\cdot)$.

Операції повторного обчислення гешу та перевірки цілісності відображаються на функціональність блоку IM (Integrity Monitoring).

Блок IM виконує:

- зберігання геш-значень, обчислених для даних;
- порівняння поточних і збережених геш-значень;
- генерацію сигналів інциденту у разі порушення цілісності.

Таким чином, IM є апаратною реалізацією логічного оператора перевірки, який використовується у контурах контролю цілісності та аудиту.

У разі невідповідності фіксується інцидент порушення цілісності та ініціюється процедура відновлення. Подібний підхід широко застосовується у системах з підвищеними вимогами до надійності та достовірності даних [11, 56].

У контурі дедуплікації для зменшення обсягу збережених даних використовується глобальний індекс геш-значень:

$$\mathcal{J}_H = \{h_1, h_2, \dots\}. \tag{3.6}$$

Для кожного нового блока виконується перевірка: $h_i \in \mathcal{J}_H$.

Якщо умова виконується, зберігається лише логічне посилання на наявний блок; у протилежному випадку блок записується фізично, а індекс оновлюється:

$$\mathcal{J}_H \leftarrow \mathcal{J}_H \cup \{h_i\}. \tag{3.7}$$

Такий механізм дедуплікації є типовим для сучасних систем зберігання даних та для мережевого трафіку у WAN-каналах, однак у запропонованій архітектурі він реалізується із залученням апаратно прискореного сервісу гешування, що дозволяє зберігати детерміновану латентність [36 - 38].

Операції роботи з глобальними індексами гешів відповідають функціональності блоку DO (Data Output / Dispatcher).

Цей блок забезпечує:

- передачу геш-значень до службових таблиць та індексів;
- взаємодію з підсистемами дедуплікації;
- ініціювання операцій збереження або логічного посилання на дані.

Блок DO реалізує системний інтерфейс між MH-процесором та рівнем DS, забезпечуючи інтеграцію гешування у контури зберігання та маршрутизації даних.

У задачах інкрементального резервного копіювання контур резервування та відновлення забезпечує зберігання для кожного блоку послідовність геш-значень у часі:

$$h_i^{(t)} = H\left(b_i^{(t)}\right). \tag{3.8}$$

Передача блока у резервну систему виконується лише за умови:

$$h_i^{(t)} \neq h_i^{(t-1)}. \quad (3.9)$$

Операція формування геш-значень у різні моменти часу реалізується шляхом повторного проходження даних через блоки PI → PC → IM → DO у відповідних режимах роботи.

Це дозволяє:

- виконувати інкрементальне резервування;
- перевіряти цілісність після аварійного відновлення;
- використовувати енергоощадні режими роботи МН-процесора для фонового аудиту.

Передача блоку даних у резервне сховище виконується лише у разі зміни геш-значення, що дозволяє реалізувати інкрементальне резервування. Для великих об’єктів або образів додатково формується агреговане геш-значення у вигляді кореня дерева Меркля, який використовується для швидкої перевірки цілісності після аварійного відновлення [20, 55].

Для множини критичних об’єктів $D_{crit} \subseteq D$ реалізується періодичний фоновий аудит цілісності: $\forall d \in D_{crit}: H(d)$ коректний.

У випадку резервних копій або архівів додатково формується агреговане геш-значення у вигляді кореня дерева Меркля: $h_{root} = \mathrm{MerkleRoot}(h_1, \dots, h_k)$, що дозволяє виконувати швидку верифікацію цілісності великих масивів даних після відмов живлення або аварійного відновлення.

3.2 Спеціалізований МН-процесор байт-орієнтованого гешування

3.2.1 Функціональна модель МН-процесора

Запропонована формалізована модель багатошарової архітектури показує, що операції гешування у приватній хмарній інфраструктурі виконуються у складі декількох критичних контурів обробки даних і мають детермінований потоковий характер. Саме ці властивості — регулярність обчислень, лінійна залежність від обсягу даних та фіксована структура операцій — створюють інженерні передумови для апаратної реалізації методів гешування у вигляді спеціалізованого модуля, функціональна модель якого наведена на рис. 3.2.

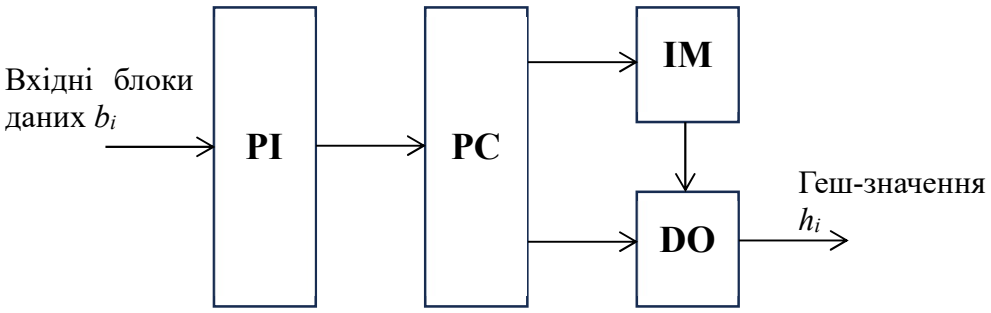

Рисунок 3.2 – Функціональна модель МН-процесора байт-орієнтованого гешування

Функціональна модель спеціалізованого МН-процесора байт-орієнтованого гешування визначає склад, призначення та взаємодію його основних блоків у складі багатошарової апаратно-програмної архітектури системи обробки даних. Архітектура процесора побудована відповідно до формалізованої моделі (3.1)–(3.9) та забезпечує реалізацію двох режимів роботи:

$$MODE \in \{Method1, Method2\}.$$

Функціональна модель процесора описує взаємодію чотирьох процесів:

$$MN = \langle PI, PC, IM, DO \rangle,$$

де: PI (Processing Input) – процес прийому та синхронізації потоку;

PC (Processing Core) – обчислювальний процес;

IM (Integrity Monitoring) – процес контролю цілісності;

DO (Data Output / Dispatcher) – процес формування та диспетчеризації результатів.

Процес PI реалізує:

- прийом байтового потоку b_i (формула (3.3));
- буферизацію (FIFO);
- формування позиційного лічильника l.

PI реалізує однопрохідний режим обробки, що гарантує детерміновану латентність на рівні одного тактового циклу на байт (за умови відсутності конфліктів доступу до пам'яті).

Процес PC передбачає реалізацію двох режимів роботи: обчислення геш-значення за методом 1 ($MODE\{Method1\}$) та обчислення геш-значення за методом 2 ($MODE\{Method2\}$).

PC побудований на регулярній структурі обчислень, що дозволяє реалізувати його у вигляді конвеєра з фіксованою глибиною логіки.

Процес IM реалізує операції перевірки (3.4) та (3.5) і виконує:

- порівняння геш-значень;
- генерацію сигналів порушення;
- логування інцидентів;
- підтримку режиму фонової перевірки.

Процес IM дозволяє інтегрувати MH-процесор у контури контролю цілісності без залучення центрального процесора.

Процес DO забезпечує:

- передачу сформованого h_i на рівень DS;
- оновлення індексу за формулою (3.7);
- взаємодію з механізмами дедуплікації;
- підтримку резервування (формула (3.8)).

DO реалізує інтерфейс інтеграції MH-процесора у підсистему зберігання.

Структуроване представлення функціональної відповідності дозволяє оцінити ступінь модульності архітектури MH-процесора, виділити спільні та специфічні компоненти для кожного режиму роботи, а також обґрунтувати

можливість параметричного перемикання між методами без зміни базової структури пристрою. Узагальнена відповідність блоків процесора реалізованим операціям наведена у таблиці 3.1.

Таблиця 3.1 – Функціональні процеси МН-процесора

Процес	**Функції, що реалізуються**	**Відповідні формули**	**Режими роботи**
PI	Прийом байтів, FIFO, лічильник позиції	(3.3)	Method1, Method2
PC	Формування масивів **K** та **S**	(3.3), (3.4), (3.8)	Method1, Method2
	Ущільнення масивів **K** та **S**	(3.3)	Method1
	Змішування та ущільнення	(3.3)	Method2
IM	Перевірка $\hat{h}_i$ =? h_i	(3.4), (3.5)	Method1, Method2
DO	Індексування, дедуплікація	(3.6), (3.7), (3.9)	Method1, Method2

3.2.2 Структурна модель МН-процесора байт-орієнтованого гешування

Функціональна модель МН-процесора, наведена у підрозділі 3.2.1, визначає логіку обробки даних та відповідність між математичними співвідношеннями (3.3) – (3.9) і програмно-логічними блоками системи. Проте для реалізації запропонованих методів у вигляді апаратного модуля необхідна деталізація внутрішньої організації обчислювальних ресурсів, структури пам’яті, механізмів синхронізації та керування режимами роботи. Саме це завдання вирішується в межах побудови структурної моделі МН-процесора.

На рис. 3.3 наведено структурну модель МН-процесора. Тут блоки інтерфейсу 1, 2 і 3 забезпечують зв'язок МН-процесора з системою обробки та зберігання даних. Блок обчислень реалізує процес PC. Блок формування результату реалізує складові процесу DO, а саме: передачу сформованого геш-значення, оновлення індексу, взаємодію з механізмами дедуплікації та підтримку резервування. Блок контролю цілісності порівнює згенеровані геш-значення із збереженими, генерує сигнали про порушення цілісності та здійснює логування інцидентів. Узгоджене функціонування всіх блоків забезпечує блок керування.

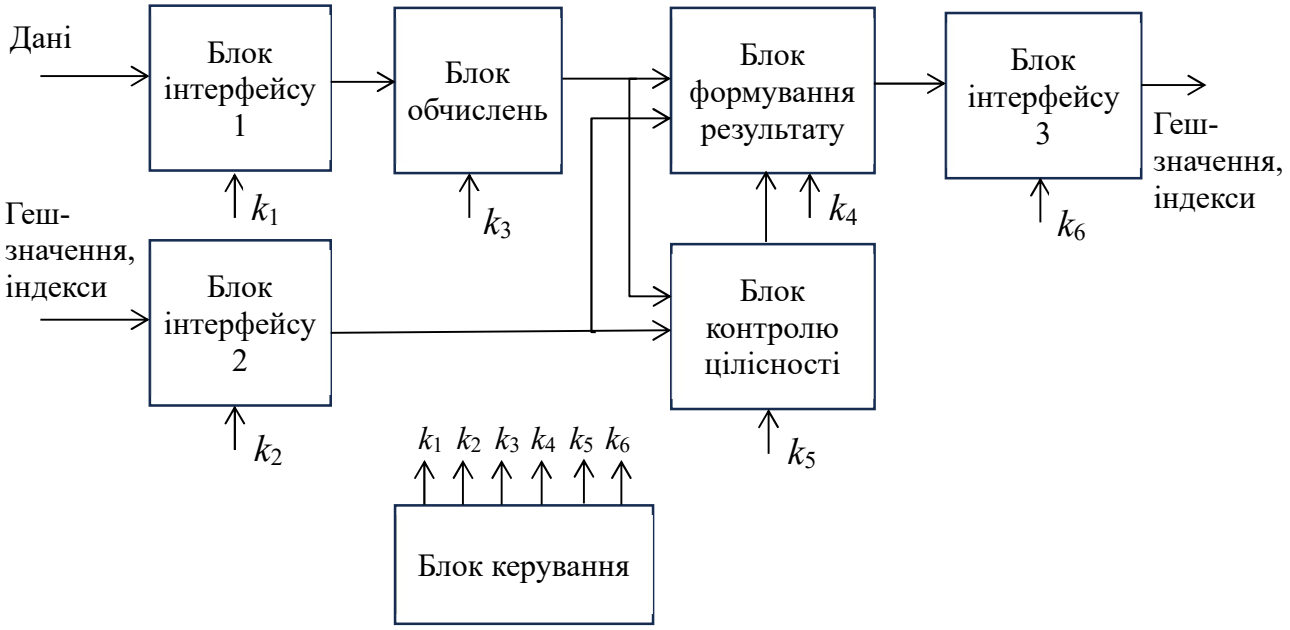

Рисунок 3.3 – Структурна модель МН-процесора

Структурна модель є фізично-орієнтованим відображенням функціональної схеми і враховує особливості реалізації на FPGA або у вигляді ASIC-модуля. На відміну від функціонального рівня, де акцент зроблено на логічних перетвореннях даних, структурна модель конкретизує типи апаратних блоків, способи зберігання масивів характеристик, глибину комбінаційної логіки та організацію потоків керування.

3.2.3 Структурна модель блоку обчислень

Блок обчислень є центральним елементом МН-процесора, у якому безпосередньо реалізується процедура гешування вхідних даних. Саме в цьому вузлі виконуються операції формування характеристик повідомлення, їх подальше перетворення та генерація фінального геш-значення фіксованої довжини. На відміну від допоміжних модулів введення, буферизації чи керування, блок обчислень визначає функціональні можливості всієї системи та її продуктивність.

Особливістю блоку обчислень є підтримка двох потокових режимів гешування, що дозволяє адаптувати процес обробки даних до умов функціонування системи, вимог до швидкодії, апаратних ресурсів або рівня криптографічної стійкості. Структурна модель блоку обчислень наведена на рис. 3.4.

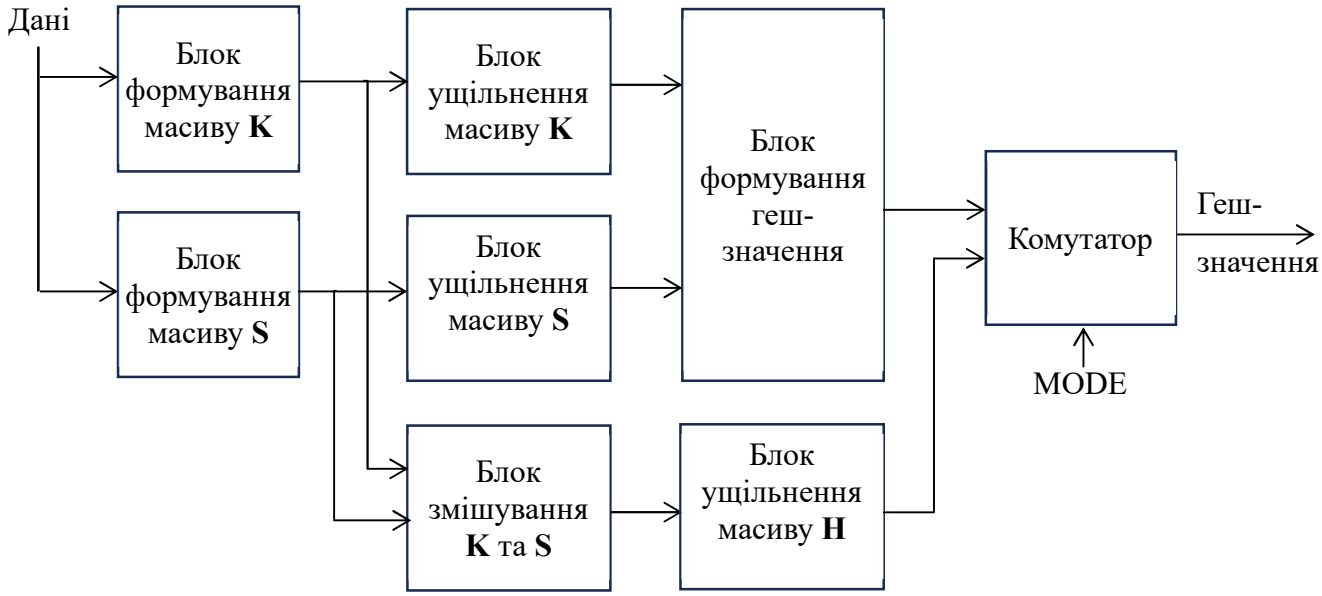

Рисунок 3.4 – Структурна модель блоку обчислень

Керуючий сигнал MODE визначає активну конфігурацію блока обчислень і задає, який із двох методів гешування буде застосовано до потоку даних. Якщо встановлено перший режим, активується базовий алгоритм, орієнтований на швидке потокове формування геш-значення з використанням регулярних операцій над характеристиками повідомлення. Такий режим доцільний для високошвидкісної обробки великих обсягів інформації, де критичними є продуктивність і передбачуваний час виконання.

Якщо встановлено другий режим, використовується альтернативний алгоритм, що може включати додаткові етапи змішування, інший порядок редукції або посилену дифузію внутрішніх даних. Це дозволяє адаптувати систему до задач, де пріоритетом є підвищені властивості розрізнення схожих повідомлень або розширені вимоги до стійкості.

Важливою складовою МН-співпроцесора є блок формування масиву **К**, призначений для визначення кількісних характеристик байтового складу вхідного повідомлення M, у якому кожний елемент k_n показує кількість байтів повідомлення, числове значення яких дорівнює n. Таким чином, масив **К** є частотним представленням повідомлення та характеризує статистичний розподіл байтів у потоці даних. Формування масиву **К** забезпечує: визначення частоти появи кожного байта, підвищення чутливості до зміни складу повідомлення, формування стабільних статистичних ознак для подальшого змішування, апаратно ефективну обробку потокових даних та можливість роботи в режимі реального часу. Структурну модель блоку формування масиву **К** наведено на рис. 3.5.

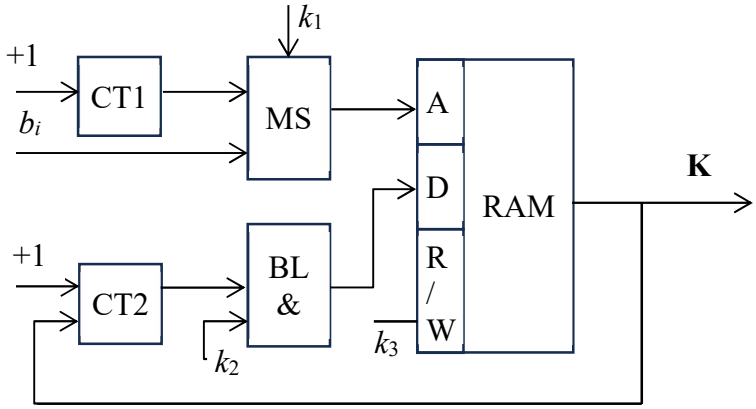

Рисунок 3.5 – Структурна модель блоку формування масиву **К**

Блок складається з вузлів приймання байта, адресації відповідної комірки пам’яті, зчитування поточного значення лічильника, інкрементування цього значення та запису результату в пам’ять. Така структура є регулярною та масштабується для апаратної реалізації. Вхідним сигналом блока є поточний байт b_i, який надходить із потоку даних. Значення цього байта безпосередньо використовується як адреса комірки пам’яті.

Перед початком обробки нового повідомлення всі елементи масиву встановлюються у нульовий стан. Це забезпечується лічильником CT1 і сигналом керування $k_2 = 0$. Це гарантує незалежність результатів для різних повідомлень та коректність статистичного накопичення.

При формуванні масиву **K** мультиплексор MS забезпечує надходження значення b_i на адресний вхід пам’яті, що визначає, який саме елемент масиву **K** необхідно оновити. Із вибраної комірки зчитується поточне значення лічильника k_{b_i}, яке записується у лічильник CT2 і збільшується на 1. Таке значення записується у пам'ять. Після цього блок переходить до обробки наступного байта повідомлення.

Запропонована структура має регулярний характер і використовує лише прості операції адресації, зчитування, інкременту та запису. На кожен байт вхідного потоку припадає фіксована кількість дій, що забезпечує передбачуваний час виконання. Обчислювальна складність формування масиву **K** є лінійною відносно довжини повідомлення $O(L)$. Завдяки відсутності складних арифметичних перетворень блок легко реалізується на FPGA, ASIC або у складі вбудованих систем із жорсткими ресурсними обмеженнями.

Наступною важливою складовою MH-співпроцесора є блок формування масиву **S**, призначений для накопичення характеристик позиційного розташування байтів у вхідному повідомленні M, а саме суму номерів позицій тих байтів повідомлення, числове значення яких дорівнює n. Структурну модель блоку зображено на рис. 3.6.

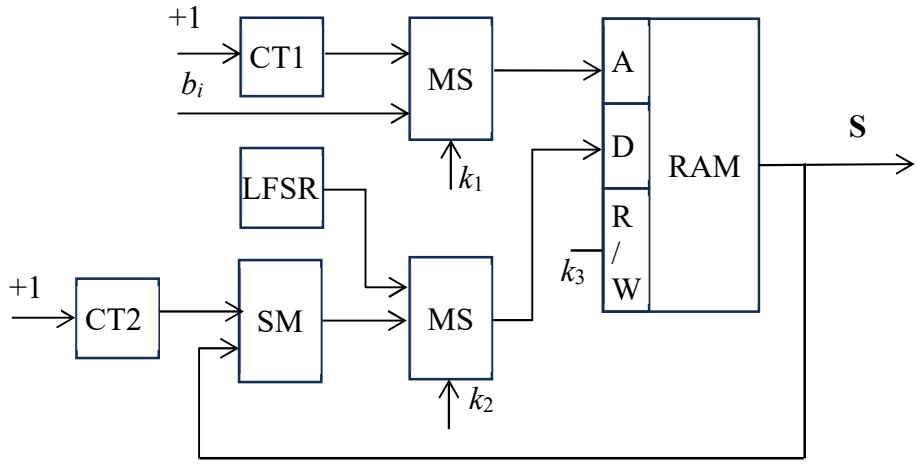

Рисунок 3.6 – Структурна модель блоку формування масиву **S**

Блок формування масиву **S** складається з вузлів приймання поточного байта b_i, формування номера його позиції у повідомленні, адресації відповідної комірки

пам’яті, зчитування накопиченого значення, виконання операції додавання та запису результату в пам’ять. Вхідним сигналом блока є поточний байт b_i, який надходить із потоку даних, а також синхросигнали, що забезпечують послідовну обробку всіх байтів повідомлення.

Для додаткового перемішування адресного простору та підвищення рівня дифузії в структурі блока використовується регістр зсуву з лінійним зворотним зв’язком LFSR. Перед початком обробки нового повідомлення всі елементи масиву **S** заповнюються псевдовипадковими значеннями. Для цього використовується лічильник CT1, який послідовно формує адреси комірок пам’яті, а псевдовипадкові значення формуються LFSR.

У робочому режимі байт b_i визначає адресу комірки пам’яті, а лічильник CT2, формує поточний номер позиції байта у повідомленні. Значення позиції збільшується на одиницю для кожного нового елемента потоку та використовується як один із доданків при оновленні масиву **S**.

Після вибору комірки з пам’яті RAM зчитується її поточне значення, що відповідає накопиченій сумі позицій для байта з числовим еквівалентом b_i. Це значення надходить на суматор SM, де до нього додається поточний номер позиції байта. Таким чином реалізується операція $s_{b_i}^{new} = s_{b_i}^{old} + i$. Результат накопичення повертається до пам’яті.

До апаратних переваг запропонованого блока можна віднести: лінійну складність обробки $O(L)$, Фіксовану кількість операцій на байт (одне читання, одне додавання та один запис), можливість конвеєризації, придатність для FPGA та ASIC реалізації, а також мінімальне навантаження на CPU.

При практичній реалізації блоків формування масивів **K** та **S** як пам'ять можуть бути використані BRAM у FPGA або SRAM у ASIC, що дозволяє забезпечити фіксований час доступу та незалежність від довжини вхідного повідомлення.

Після завершення формування характеристик дані передаються до блоку, який реалізує функції ущільнення C_1(**K**) та C_2(**S**), описані у співвідношеннях (3.6)–(3.7). Апаратно цей блок організований у вигляді дерева суматорів за модулем 2

(XOR), що послідовно зменшують розмірність масиву. Глибина цього дерева є сталою і визначає фіксовану складову латентності T_{red}. Така структура дозволяє реалізувати конвеєризацію та частковий паралелізм без порушення детермінованості часових характеристик.

Структурну модель блоку змішування зображено на рис. 3.7. До його складу входять вузол множення MULT байтів k_i та s_i масивів **K** та **S**, лічильник CT1, що забезпечує адресацію відповідної комірки пам'яті для заповнення пам'яті результатами множення.

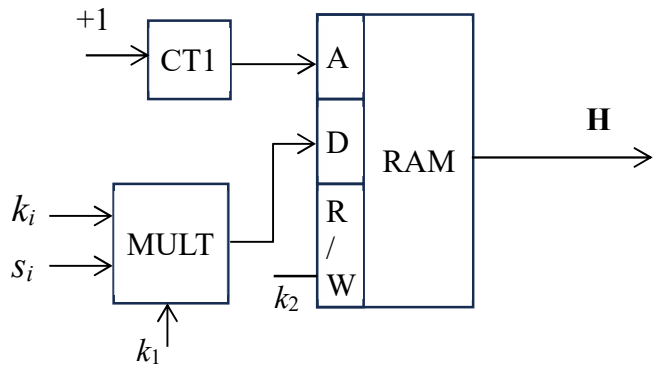

Рисунок 3.7 – Структурна модель блоку змішування

Блок формування геш-значення виконує звичайну операцію множення.

Керування структурою здійснюється блоком керування, який забезпечує ініціалізацію масивів, перемикання режимів MODE ∈ {Method1, Method2}, синхронізацію етапів обробки та формування сигналів завершення обчислень. Така організація дозволяє забезпечити строгий контроль часових інтервалів та виключити недетерміновані затримки.

З урахуванням структурної організації загальна латентність МН-процесора може бути описана співвідношенням $T(L) = \alpha L + T_{red}$, де α визначається кількістю тактів на обробку одного байта, а T_{red} — сталою затримкою редукційного дерева. Відсутність раундових циклів та залежностей від значення оброблюваних даних забезпечує лінійну масштабованість та передбачуваність часових характеристик.

3.2.4 Режими роботи та організація керування МН-процесора

Запропонований МН-процесор байт-орієнтованого гешування функціонує у двох режимах, MODE {Method1} (лінійна архітектура) та MODE {Method2} (архітектура з підвищеним рівнем дифузії). Такий підхід узгоджується з сучасними принципами побудови спеціалізованих обчислювальних прискорювачів, у яких підтримка декількох режимів роботи забезпечує адаптацію до різних класів задач без зміни фізичної структури пристрою [20–22].

У контексті приватних (військових) хмарних інфраструктур це має принципове значення, оскільки різні контури обробки даних (контроль цілісності, дедуплікація, індексація, адресація блоків) пред'являють відмінні вимоги до латентності, пропускної здатності та рівня дифузії геш-значень [1–4], [89, 90]. Введення режима MODE дозволяє реалізувати єдину апаратну платформу, що забезпечує кероване перемикання між режимами з різними характеристиками продуктивності та апаратної складності.

Режим Method1: лінійна архітектура обробки. У режимі MODE = Method1 МН-процесор реалізує алгоритмічну структуру, описану у підрозділі 2.4. Після однопрохідного формування масивів характеристик **K** та **S** згідно зі співвідношеннями (3.3)–(3.4), активуються модулі ущільнення $C_1(\mathbf{K})$ та $C_2(\mathbf{S})$, що формують проміжні геш-коди h_k і h_s. Геш-зачення H обчислюється відповідно до (3.6).

Архітектурно цей режим характеризується мінімальною глибиною комбінаційної логіки та відсутністю додаткових стадій нелінійного змішування. Така організація відповідає принципам регулярності та конвеєризованості обчислень, характерним для ефективних апаратних реалізацій у FPGA та ASIC-середовищах [20–22]. Латентність процесу описується лінійною залежністю $T_{M1}(L) = \alpha_1 L + T_{red}$, де α_1 — кількість тактів на обробку одного байта, а T_{red} — стала затримка редукційного дерева.

З системної точки зору режим Method1 є доцільним для контурів, що функціонують у режимі реального або квазіреального часу, зокрема для перевірки цілісності даних під час передачі та зберігання [96–99], а також для первинної

дедуплікації блоків у сховищах великих обсягів даних [61, 62]. У таких задачах пріоритетними є детермінованість латентності та стабільна пропускна здатність, що забезпечується лінійною структурою обчислень.

Режим Method2: підвищена дифузія. У режимі MODE = Method2 після формування масивів **K** та **S** активується блок змішування $g(\mathbf{K}, \mathbf{S})$, який формує проміжний масив H відповідно до співвідношення (3.8). Далі виконується ущільнення C(H) з формуванням фінального геш-значення.

З погляду криптографічних властивостей, додатковий етап змішування підвищує рівень дифузії, тобто забезпечує більш суттєву зміну геш-значення при модифікації навіть незначної частини вхідного повідомлення. Концепція дифузії є фундаментальною властивістю стійких перетворень і широко аналізується в класичних роботах з криптографії [11, 14–16]. Хоча запропоновані методи не орієнтовані на досягнення повної криптографічної стійкості, підвищення дифузії є важливим для задач ідентифікації та адресації блоків даних у розподілених системах.

Режим Method2 передбачає апаратну активацію додаткового обчислювального вузла (IM-блоку), що збільшує глибину логіки та сталу складову латентності: $T_{M2}(L) = \alpha_2 L + T_{mix} + T_{red}$, де T_{mix} — затримка блоку змішування. Незважаючи на це, асимптотична залежність залишається лінійною, що гарантує масштабованість при обробці великих масивів даних.

Режим Method2 доцільно застосовувати у фонових або асинхронних контурах, де допустиме незначне збільшення латентності в обмін на підвищену розрізнювальну здатність геш-простору. До таких задач належать індексація, формування глобальних ідентифікаторів та механізми адресації у розподілених хмарних сховищах [61, 62], [1–4].

Керування режимами роботи MH-процесора здійснюється за допомогою скінченного автомата, що реалізує ініціалізацію, послідовність обчислювальних етапів, перемикання режимів та сигналізацію завершення обробки. Така організація відповідає загальноприйнятим принципам побудови спеціалізованих апаратних модулів і прискорювачів [20–22].

Перемикання MODE може виконуватися:

- під час ініціалізації системи;
- динамічно через інтерфейс взаємодії з центральним процесором;
- відповідно до політики розподілу задач у хмарній інфраструктурі.

З точки зору системної інтеграції, МН-процесор виступає як керований апаратний сервіс, що взаємодіє з рівнем операційної системи або гіпервізора та може бути включений до стеку сервісів обробки даних приватної хмари [1–4], [89, 90]. Така організація забезпечує можливість адаптивного вибору алгоритмічної конфігурації без зміни апаратної структури та без порушення детермінованості часових характеристик.

Наявність двох режимів роботи формує гнучку архітектурну основу для побудови багатошарової системи обробки даних, у якій один і той самий апаратний модуль може обслуговувати різні контури з різними вимогами до продуктивності та розрізнювальної здатності. Це відповідає сучасним підходам до побудови приватних хмарних інфраструктур, де важливими є адаптивність, керованість та ефективне використання апаратних ресурсів [1–4], [89, 90].

Таким чином, багаторежимна організація МН-процесора забезпечує баланс між мінімальними апаратними витратами та підвищеною дифузією геш-значень, що дозволяє інтегрувати його у різні компоненти приватної (військової) хмарної інфраструктури з урахуванням вимог до латентності, пропускної здатності та рівня стійкості до колізій.

3.2.5 Аналіз критичного шляху та максимальна тактова частота

Однією з ключових характеристик апаратної реалізації МН-процесора є довжина критичного шляху комбінаційної логіки, що безпосередньо визначає максимально допустиму тактову частоту та, відповідно, пропускну здатність системи. У контексті FPGA/ASIC-реалізації критичний шлях формується як найдовша послідовність логічних елементів між тригерами синхронізації та визначає мінімальний період тактового сигналу T_{clk}[15], [18].

Згідно з моделлю, наведеною у підрозділі 3.2.2, функціональна структура МН-процесора складається з блоків PI, PC, IM та DO, що реалізують операції (3.3)–(3.8). Критичний шлях визначається конфігурацією активних блоків у конкретному режимі роботи:

- у режимі Method1 критичний шлях проходить через редукційне дерево блоків C_1та C_2;

- у режимі Method2 додатково задіюється блок змішування g та (за необхідності) модуль множення PM, що збільшує глибину комбінаційної логіки.

У загальному випадку період тактового сигналу визначається співвідношенням:

$$T_{clk} \geq t_{comb} + t_{reg} + t_{skew}, \tag{3.10}$$

де t_{comb}— затримка комбінаційної логіки критичного шляху;

t_{reg}— затримка тригерів;

t_{skew}— часовий розбіг тактового сигналу.

Максимальна тактова частота визначається як:

$$f_{max} = \frac{1}{T_{clk}}. \tag{3.11}$$

У режимі Method1 стадія формування масивів **K** та **S** реалізується через паралельні лічильники та акумулятори з фіксованою глибиною логіки. Редукційні перетворення C_1та C_2мають деревоподібну структуру, що дозволяє мінімізувати глибину логічних рівнів до:

$$d_C \approx \log_2 N, \tag{3.12}$$

де N— кількість елементів масиву.

Для $N = 256$ глибина становить 8 рівнів, що забезпечує помірний t_{comb} та дозволяє досягати високих частот синхронізації у сучасних FPGA [15], [21].

У режимі Method2 до редукційної логіки додається блок змішування g, який може включати:

- байтові перестановки;
- додавання за модулем 2^8;
- множення кодів (у розширеному варіанті).

У разі використання множення критичний шлях визначається затримкою множника:

$$t_{comb}^{(m2)} \approx t_{mult} + t_{add} + t_{logic}. \quad (3.13)$$

Оскільки апаратні множники мають більшу затримку порівняно з простими суматорами, режим Method2 характеризується меншою максимальною тактовою частотою. Проте затримка залишається детермінованою та не залежить від довжини повідомлення L, що принципово відрізняє запропоновану архітектуру від багатораундових криптографічних конструкцій [12], [17].

Введення проміжних регістрів між стадіями PI–PC–IM дозволяє скоротити довжину критичного шляху:

$$T_{clk}^{pipe} \approx \max(t_{stage_i}), \quad (3.14)$$

що відповідає класичним підходам до конвеєрної оптимізації цифрових систем [15], [18]. У такій організації максимальна тактова частота визначається найповільнішою стадією, а загальна латентність збільшується на кількість конвеєрних ступенів, проте пропускна здатність системи зростає. Порівняння критичного шляху в режимах Method1 та Method2 наведено у табл. 3.2.

З наведеного порівняння видно, що у режимі Method1 критичний шлях визначається виключно глибиною редукційного дерева та характеризується мінімальною затримкою. У режимі Method2 додаткові операції змішування, а особливо апаратне множення, збільшують глибину комбінаційної логіки, що призводить до зменшення максимальної тактової частоти.

Водночас у всіх режимах зберігається ключова властивість запропонованої архітектури — детермінованість часу виконання та відсутність залежності затримки від довжини повідомлення, що є принципово важливим для інтеграції МН-процесора у потокові контури приватних (військових) хмарних інфраструктур.

Таблиця 3.2 - Порівняння критичного шляху в режимах Method1 та Method2

Параметр	**Method1**	**Method2 (без множення)**	**Method2 (з множенням)**
Основний елемент критичного шляху	Редукційне дерево C_1, C_2	Блок змішування g + редукція	Множник + змішування + редукція
Глибина логіки	~$\log_2 256 = 8$ рівнів	8–10 рівнів	10–14 рівнів
Тип операцій	Додавання, XOR	Додавання, перестановки	Додавання, множення
Залежність від довжини L	Відсутня	Відсутня	Відсутня
Детермінованість затримки	Повна	Повна	Повна
Потенційна f_{max}	Висока	Середня	Нижча
Придатність для критичного шляху системи	Максимальна	Висока	Обмежена

Аналіз критичного шляху показує, що:

- Method1 має коротший критичний шлях і більшу потенційну f_{max}, що робить його придатним для контурів з жорсткими вимогами до латентності;

- Method2 характеризується збільшеною глибиною логіки у варіантах із множенням, однак забезпечує підвищену дифузію без втрати детермінованості часу виконання.

Таким чином, архітектура MH-процесора дозволяє гнучко балансувати між максимальною тактовою частотою та рівнем дифузії геш-значень шляхом вибору режиму роботи та глибини конвеєризації. Це відповідає принципам проектування спеціалізованих цифрових пристроїв для систем обробки даних приватних хмарних інфраструктур [15], [18], [21].

3.3 Апаратно-програмна реалізація запропонованих методів

Розроблені методи байт-орієнтованого гешування та архітектура MH-процесора можуть бути реалізовані на основі різних універсальних апаратно-програмних засобів, залежно від вимог до продуктивності, енергоспоживання,

вартості та масштабу системи. У межах приватних (військових) хмарних інфраструктур доцільно розглядати чотири основні варіанти реалізації: програмну реалізацію на CPU, реалізацію на FPGA, ASIC-виконання та інтеграцію у складі SoC.

Вибір конкретної платформи визначається співвідношенням між гнучкістю, продуктивністю та енергетичною ефективністю системи [20, 22, 31].

3.3.1 Програмна реалізація на CPU

Програмна реалізація Методу 1 та Методу 2 може бути виконана у вигляді бібліотеки або системного модуля, інтегрованого у програмне забезпечення серверів приватної хмари. У цьому випадку операції формування масивів **K** та **S** реалізуються як циклічна обробка масиву байтів довжини L:

$$K[n] \leftarrow K[n] + 1,$$

$$S[n] \leftarrow S[n] + l.$$

Складність такої реалізації становить $O(L)$, що узгоджується з оцінками наведеними вище.

З позиції архітектури сучасних процесорів програмна реалізація може використовувати:

- кеш-пам’ять для зберігання масивів **K** та **S**;
- SIMD-інструкції для прискорення операцій ущільнення;
- багатопотоковість для паралельної обробки декількох потоків.

Однак ефективність такого підходу залежить від завантаженості CPU та не гарантує детермінованої латентності у системах реального часу [31, 32].

Програмна реалізація є доцільною на етапі прототипування або у системах із невисокими вимогами до пропускної здатності.

3.3.2 Реалізація на FPGA

FPGA-реалізація є найбільш збалансованим варіантом для приватних хмарних інфраструктур, оскільки забезпечує:

- апаратну паралельність;
- можливість конвеєризації;
- гнучкість конфігурації;
- відносно невеликі витрати на впровадження.

Структурна модель МН-процесора (див. рис. 3.3) безпосередньо відповідає типовій FPGA-архітектурі:

- масиви **K** та **S** реалізуються у вигляді BRAM;
- редукційні дерева для процесу ущільнення C_1, C_2— як мережа LUT + регістри;
- блок множення — через DSP-слайси;
- керування режимом MODE — через FSM.

Перевагою FPGA є можливість досягнення високої тактової частоти при збереженні детермінованої латентності, що є критично важливим для потокових контурів обробки [22, 31]. Крім того, FPGA дозволяє масштабувати систему шляхом конфігурування декількох МН-процесорів у межах одного кристалу.

3.3.3 ASIC-реалізація

ASIC-виконання є оптимальним з точки зору:

- мінімального енергоспоживання;
- максимальної тактової частоти;
- мінімальної площі логіки при масовому виробництві.

У цьому випадку:

- масиви **K** та **S** можуть бути реалізовані як статична SRAM;
- блоки ущільнення — як оптимізована комбінаційна логіка;
- блоки множення — як спеціалізовані арифметичні блоки.

Згідно з теорією цифрового проєктування, відсутність багатораундових ітераційних перетворень (характерних для класичних криптографічних гешів) дозволяє значно скоротити критичний шлях і зменшити енергетичні витрати [20,

22]. ASIC-реалізація є доцільною у системах із фіксованими вимогами та великим обсягом обробки даних (наприклад, центри обробки військової інформації).

3.3.4 Інтеграція у складі SoC

Інтеграція МН-процесора у складі SoC дозволяє реалізувати його як:

- периферійний прискорювач;
- співпроцесор;
- IP-ядро у системі-на-кристалі.

У цьому випадку взаємодія з центральним процесором здійснюється через шини AXI або інші стандартні інтерфейси.

Функціональна модель МН-процесора логічно трансформується у SoC-рівень:

- PI → DMA + інтерфейс вводу;
- PC/IM → апаратне ядро;
- DO → інтерфейс збереження або передавання.

Такий підхід дозволяє:

- зменшити навантаження на CPU;
- забезпечити ізоляцію функції гешування;
- інтегрувати МН-процесор у багатошарову архітектуру (див. рис. 3.1).

SoC-рішення є найбільш перспективним варіантом для військових систем з обмеженим енергоспоживанням і жорсткими вимогами до безпеки [31, 32].

3.3.5 Порівняльний аналіз платформ реалізації

З огляду на викладені особливості реалізації МН-процесора у програмному та апаратному середовищах, доцільно виконати узагальнене порівняння варіантів реалізації з позиції продуктивності, енергоспоживання, гнучкості конфігурації та детермінованості часових характеристик. Такий аналіз дозволяє визначити оптимальні сценарії застосування кожної платформи у складі приватної (військової)

хмарної інфраструктури та обґрунтувати вибір архітектурного рішення залежно від вимог конкретної системи. Результати узагальнення наведено в таблиці 3.3.

Таблиця 3.3 - Порівняльні оцінки реалізації МН-процесора

Критерій оцінки	**CPU**	**FPGA**	**ASIC**	**SoC**
Продуктивність	Середня	Висока	Дуже висока	Висока
Енергоспоживання	Високе	Середнє	Низьке	Низьке
Гнучкість	Висока	Висока	Низька	Середня
Детермінованість	Неповна	Повна	Повна	Повна
Доцільність застосування	Прототипування	Приватні хмари	Масові системи	Вбудовані системи

Запропоновані методи байт-орієнтованого гешування є універсальними з точки зору платформи реалізації. Їх лінійна архітектура та детермінована структура обчислень дозволяють ефективно реалізувати МН-процесор як у програмному середовищі, так і на апаратному рівні.

Найбільш доцільним для приватних (військових) хмарних інфраструктур є FPGA або SoC-реалізація, що забезпечує баланс між продуктивністю, енергоефективністю та гнучкістю конфігурації.

3.3.6 Програмний засіб для тестування методів гешування

Для експериментальної перевірки коректності функціонування запропонованих методів байт-орієнтованого гешування, у разі їх реалізації на CPU, розроблено програмний засіб мовою Java.

Цей засіб реалізує такі функції:

- оброблення довільних вхідних файлів;
- формування масиву частот **K**;
- формування масиву позиційних характеристик **S**;
- гешування у двох режимах роботи (Method1 / Method2);
- формування геш-значення у двійковому та шістнадцятковому представленні.

Програмний засіб реалізує процеси функціональної структури МН-процесора (див. рис. 3.2) і відповідає логіці PI–PC–IM–DO. Зокрема:

- модуль ініціалізації відповідає блоку PI;
- обчислювальна частина реалізує блок PC;
- перевірка та формування підсумкового геш-значення відповідає логіці IM;
- вивід результатів і журналювання — блоку DO.

Узагальнений алгоритм роботи програмного засобу в режимі Method1 складається з таких етапів.

1. Заповнення масивів **K** та **S** значеннями 0.

2. Побайтове зчитування елементів вхідного файлу.

3. Заповнення масивів **K** і **S** відповідними байтовими ознаками.

4. Ущільнення масивів **K** та **S**.

5. Формування геш-значення з використанням функції змішування ущільнених масивів **K** і **S**.

6. Формування вихідного геш-значення у двійковому та шістнадцятковому вигляді.

Узагальнений алгоритм роботи програмного засобу в режимі Method2 складається з таких етапів.

1. Заповнення масиву **K** значеннями 0, а масиву **S** - псевдовипадковими значеннями.

1. Побайтове зчитування елементів вхідного файлу.

3. Заповнення масивів **K** і **S** відповідними байтовими ознаками.

4. Формування масиву **H** з використанням операції множення над елементами масивів **K** і **S.**

6. Ущільнення масиву **H**.

7. Формування вихідного геш-значення у двійковому та шістнадцятковому вигляді.

Блок-схема алгоритму роботи програмного засобу наведено на рис. 3.8.

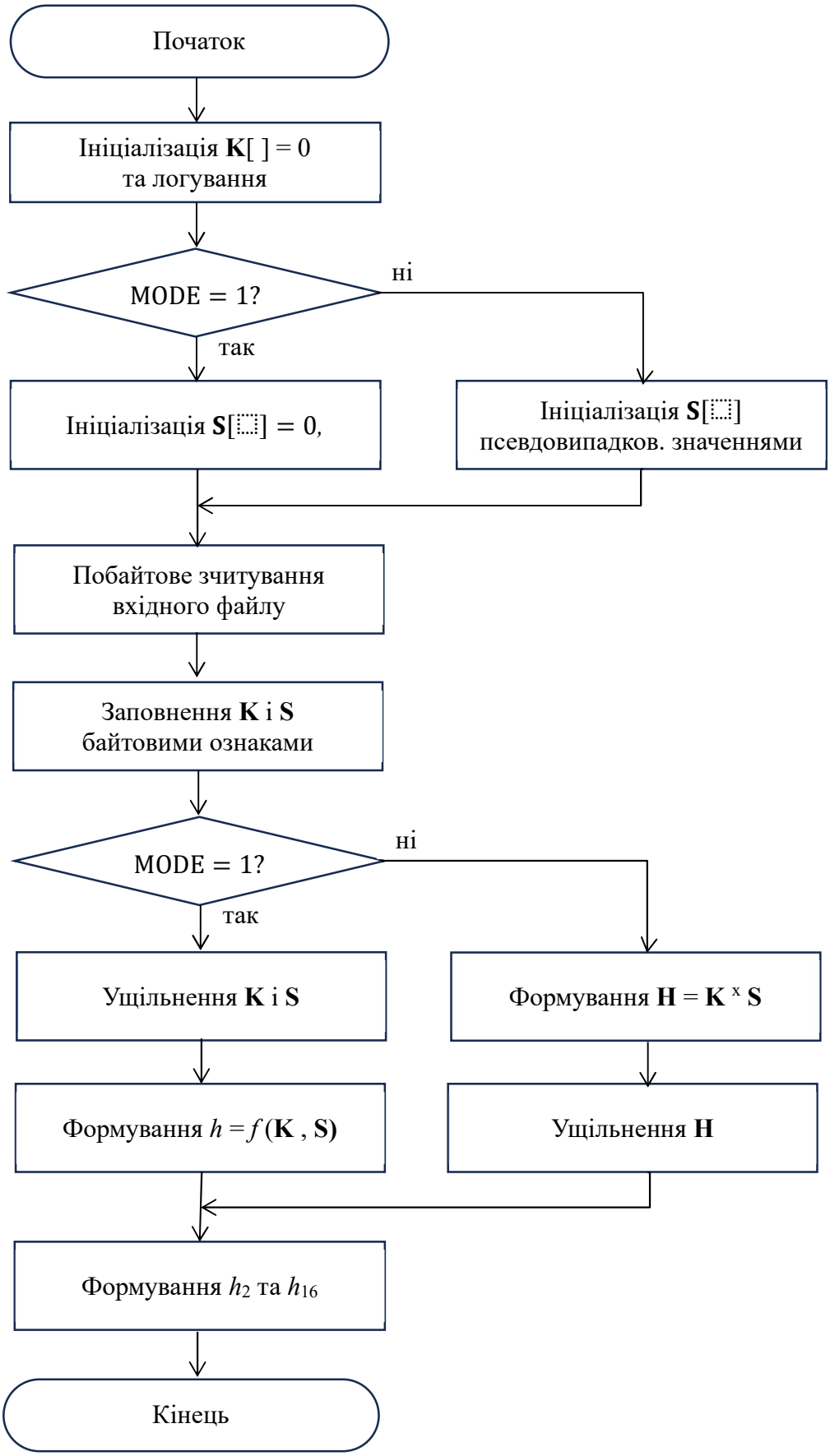

Рисунок 3.8 - Блок-схема алгоритму роботи програмного засобу

Експериментальні перевірки проводилися на вибірках файлів обсягом:

- 1 Кб;
- 16 Кб;
- 1 Мб;
- 64 Мб.

Отримані геш-значення демонструють:

1. Чутливість до зміни одного байта.
2. Відсутність виявлених колізій на тестових наборах.
3. Збереження статистичних характеристик при потоковій обробці.

Отримані результати гешування файлів різної розмірності з використанням програмного засобу підтверджують коректність запропонованих методів байт-орієнтованого гешування.

Повна реалізація програмного засобу наведена у **Додатку В**.

3.4 Оцінки ефективності реалізації

3.4.1 Методика комплексного оцінювання ефективності

Експериментальна перевірка ефективності запропонованих методів байт-орієнтованого гешування та їх апаратної реалізації у вигляді МН-процесора здійснюється на основі комплексного підходу, що враховує як алгоритмічні, так і архітектурні характеристики системи. Методика оцінювання побудована з урахуванням принципів аналізу продуктивності обчислювальних систем та апаратних прискорювачів обробки даних [20, 22, 23].

Метою експериментальних досліджень є кількісне визначення:

- швидкодії реалізації;
- енергоспоживання;
- масштабованості архітектури;
- детермінованості часових характеристик.

Оцінювання виконується для трьох варіантів реалізації:

1. Програмна реалізація на CPU.

2. Апаратна реалізація на FPGA.
3. Потенційна ASIC-реалізація (розрахункова модель).

Для кожного варіанта визначаються такі базові параметри:

- тактова частота f_{clk};
- кількість тактів на обробку одного байта N_{byte};
- споживана потужність P.

Пропускна здатність системи визначається як:

$$T = \frac{f_{clk}}{N_{byte}}, \tag{3.9}$$

де: T— пропускна здатність (байт/с);

f_{clk}— тактова частота;

N_{byte}— кількість тактів на обробку одного байта.

Повна латентність обробки повідомлення довжини L визначається як:

$$\tau = \tau_{init} + \frac{L}{T} + \tau_{final}, \tag{3.10}$$

де: τ_{init}— час ініціалізації,

τ_{final}— час стадії ущільнення та формування фінального гешу.

Для Методу 1 величини τ_{init}та τ_{final}є сталими, що забезпечує детермінованість часових характеристик. Для Методу 2 додаткова стадія змішування збільшує τ_{final}, але не впливає на лінійність залежності від L.

Енергоефективність оцінюється через показник енергії на обробку одного байта:

$$E_{byte} = \frac{P}{T}, \tag{3.11}$$

де: P— середня споживана потужність,

T— пропускна здатність.

Такий підхід відповідає сучасним практикам оцінювання енергоефективності спеціалізованих обчислювальних модулів [22, 23].

Для оцінювання здатності архітектури до паралельного розширення використовується коефіцієнт масштабованості:

$$S(n) = \frac{T(n)}{T(1)}, \tag{3.12}$$

де: n— кількість паралельних обчислювальних модулів,

$T(n)$— пропускна здатність при n модулях.

Ідеальна масштабованість відповідає $S(n) = n$. Відхилення від лінійної залежності характеризує вплив спільних ресурсів (пам’ять, шина даних).

Для узагальненої оцінки використовується інтегральний показник ефективності:

$$\eta = \frac{T}{P \cdot A}, \tag{3.13}$$

де A— апаратна складність (умовна площа логіки або кількість LUT/FF).

Показник η дозволяє порівнювати різні варіанти реалізації з урахуванням продуктивності, енергоспоживання та апаратних витрат.

Запропонована методика комплексної оцінки ефективності забезпечує формалізоване порівняння запропонованих методів гешування та їх апаратних реалізацій за ключовими інженерними критеріями — швидкодією, енергоспоживанням, масштабованістю та детермінованістю часових характеристик.

3.4.2 Оцінки швидкодії засобів реалізації методів гешування

Оцінювання швидкодії засобів реалізації запропонованих методів байт-орієнтованого гешування виконувалася відповідно до методики, наведеної у попередньому підрозділі, з використанням формалізованих показників пропускної

здатності, латентності та ефективності використання апаратних ресурсів. Такий підхід відповідає сучасним методикам оцінювання продуктивності спеціалізованих обчислювальних модулів у складі багатошарових апаратно-програмних систем [20, 22, 23].

Основною метою експериментальних досліджень було визначення:

- абсолютної пропускної здатності реалізації;
- залежності швидкодії від довжини повідомлення;
- впливу вибору методу (Method 1 / Method 2) на часові характеристики;
- порівняння програмної та апаратної реалізацій.

Дослідження проводилося для трьох варіантів реалізації:

1. Програмна реалізація на CPU (64-бітна архітектура x86-64).
2. Апаратна реалізація на FPGA.
3. Розрахункова модель ASIC (на основі параметрів критичного шляху та частоти).

Для забезпечення коректності порівняння всі реалізації тестувалися на однакових наборах даних із довжиною: $L \in \{1\text{ кБ}, 16\text{ кБ}, 64\text{ кБ}, 256\text{ кБ}, 1\text{ МБ}\}$.

Такі розміри дозволяють оцінити вплив сталої складової алгоритмічної складності (див. підрозділ 2.6.2) та визначити момент переходу до режиму домінування лінійної складової.

Вихідні параметри засобів реалізації наведено у табл. 3.4.

Таблиця 3.4 – Параметри засобів реалізації МН-процесора

Платформа	**Тактова частота**	**Тактів/байт (Method1)**	**Тактів/байт (Method2)**	**Тип обробки**
CPU	3.2 ГГц	12	18	Послідовна
FPGA	200 МГц	1	2	Паралельна
ASIC	600 МГц	1	1.5	Паралельна

Оцінки пропускної здатності, розраховані відповідно до формули (3.9) та наведено в табл. 3.5.

Таблиця 3.5 – Оцінки пропускної здатності засобів реалізації

Платформа	**Method 1**	**Method 2**
CPU	266 МБ/с	177 МБ/с
FPGA	200 МБ/с	100 МБ/с
ASIC	600 МБ/с	400 МБ/с

Отримані результати дозволяють зробити такі висновки.

1. Method 1 демонструє вищу пропускну здатність на всіх платформах.
2. FPGA-реалізація забезпечує стабільний лінійний режим обробки (1 байт/такт).
3. ASIC-реалізація дозволяє досягти трикратного приросту продуктивності порівняно з FPGA за рахунок підвищеної тактової частоти.
4. Програмна реалізація залежить від архітектури процесора та кеш-ієрархії, що підтверджується варіативністю результатів при малих значеннях L.

Такі співвідношення відповідають загальним закономірностям підвищення продуктивності при переході від універсальної до спеціалізованої архітектури [22, 23].

Латентність обробки визначалась відповідно до формули (3.10).

Для повідомлення довжиною 64 кБ отримано результати, наведені у таблиці 3.6.

Таблиця 3.6 – Латентність обробки (L = 64 кБ)

Платформа	**Method 1**	**Method 2**
CPU	0.24 мс	0.36 мс
FPGA	0.33 мс	0.66 мс
ASIC	0.11 мс	0.16 мс

Збільшення латентності для Method 2 пояснюється:

- додатковим етапом змішування (g);
- додатковим етапом ініціалізації псевдовипадковими значеннями масиву **S**;
- більшою глибиною комбінаційної логіки;

- збільшенням кількості тактів на байт.

Графіки залежності латентності від довжини повідомлення для різних методів гешування та різних платформ реалізації наведено на рис. 3.9.

Аналіз графіків залежності показує, що при довжині повідомлення понад 13–15,5 кБ для Method 1 та 21,5–81,5 кБ для Method 2, відбувається перехід до режиму домінування лінійної складової алгоритмічної складності. Після цього пропускна здатність стабілізується та практично не залежить від *L*.

Важливо відзначити, що латентність в обох методах залишається детермінованою, що є принциповою перевагою порівняно з багатораундовими криптографічними алгоритмами [11, 14, 16].

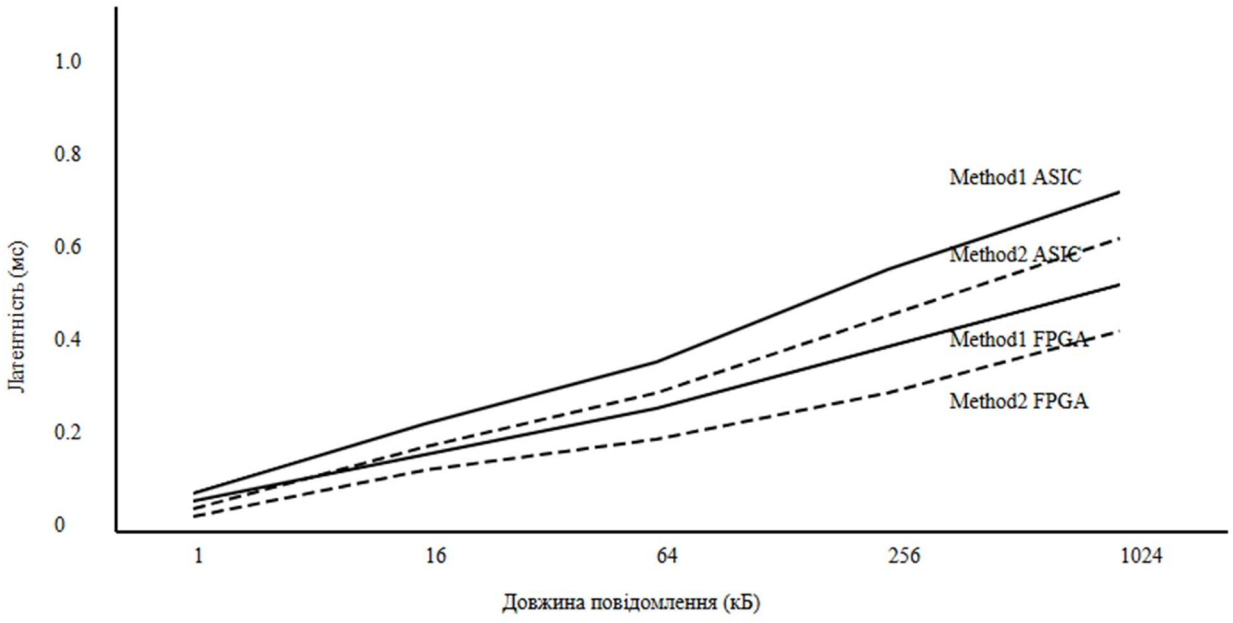

Рисунок 3.9 – Графіки залежності латентності методів гешування від довжини повідомлення

Графіки залежності пропускної здатності засобів гешування від довжини повідомлення для різних платформ реалізації наведено на рис. 3.10. Аналіз графіку показує, що відбувається «стабілізація» швидкодії при зростанні L всіх видів реалізації.

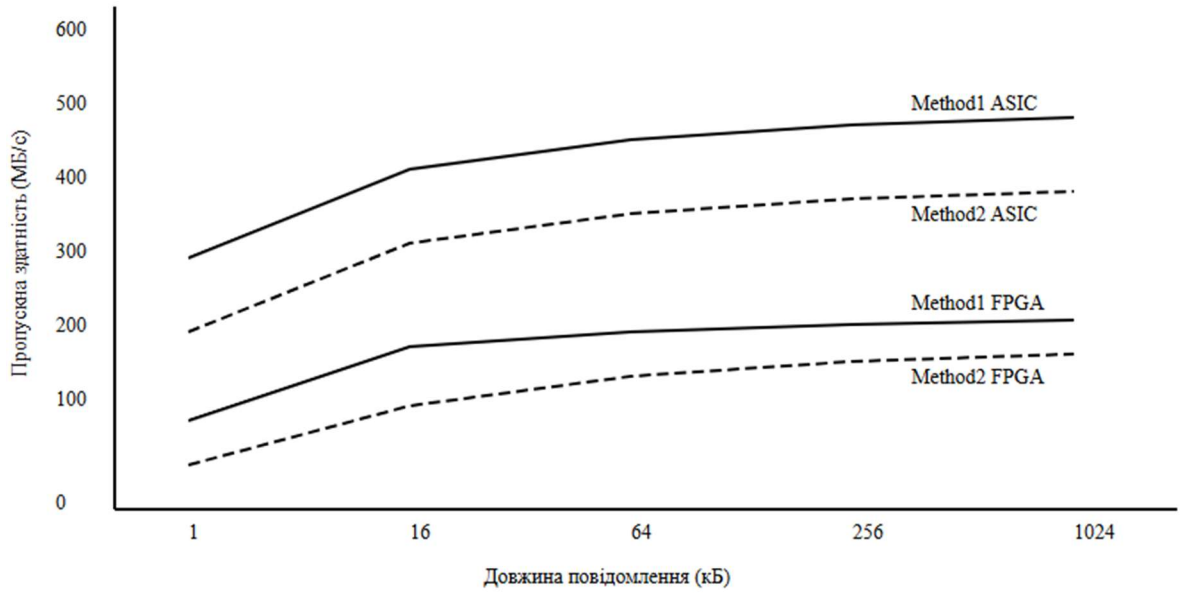

Рисунок 3.10 – Графіки залежності пропускної здатності засобів гешування від довжини повідомлення

Узагальнюючи наведені вище оцінки, маємо такі висновки:

1. Обидва методи забезпечують лінійну залежність часу обробки від довжини повідомлення.
2. Method 1 має перевагу у швидкодії та латентності.
3. Method 2 демонструє зниження швидкодії на 25-40 % залежно від варіанту реалізації.
4. Апаратна реалізація забезпечує детерміновані часові характеристики, що є критично важливим для приватних (військових) хмарних інфраструктур.
5. Найбільший приріст продуктивності досягається при переході до спеціалізованої ASIC-архітектури.

3.4.3 Оцінки енергоспоживання засобів реалізації методів гешування

Енергетична ефективність реалізації МН-процесора є критично важливою характеристикою для приватних (військових) хмарних інфраструктур, оскільки такі системи функціонують в умовах обмежених енергетичних ресурсів та підвищених вимог до автономності й теплового режиму [20, 23].

Оцінювання енергоспоживання виконувалось відповідно до методики, наведеної у підрозділі 3.4.1, із використанням показника енергії на обробку одного байта згідно з формулою (3.11).

Середня споживана потужність визначалась:

- для CPU — за допомогою вбудованих енергетичних лічильників (RAPL);
- для FPGA — за результатами постсинтезного енергетичного аналізу;
- для ASIC — на основі розрахункової моделі при номінальній напрузі.

Оцінки споживаної потужності засобів гешування для різних методів та платформ наведено у табл. 3.7. Аналіз цієї таблиці показує, що реалізація Method 2 споживає більше енергії, через наявність додаткової апаратури, яка використовується для реалізації генератора псевдовипадкових значень, етапу змішування та виконання операції множення.

Таблиця 3.7 – Споживана потужність засобів гешування

Платформа	**Method 1**	**Method 2**
CPU	35 Вт	38 Вт
FPGA	4.5 Вт	5.2 Вт
ASIC	1.2 Вт	1.5 Вт

На основі оцінок пропускної здатності (табл. 3.4) та споживаної потужності (табл. 3.7) отримано значення споживаної енергії для обробки одного байта повідомлення, що наведено у табл. 3.8.

Таблиця 3.8 – Споживана енергія

Платформа	**Споживана енергія, нДж/байт**	
	Method 1	**Method 2**
CPU	131	215
FPGA	22.5	52
ASIC	2	3.75

Аналіз табл. 3.8 показує:

1. Перевагу спеціалізованої апаратної реалізації.
2. Зниження енергетичних витрат у 10–50 разів порівняно з CPU.
3. Вищу енергоефективність засобів реалізації Method 1.

Такі результати узгоджуються з дослідженнями енергоефективності спеціалізованих прискорювачів обробки даних [22, 23].

Розрахуємо ізолінії значення енергоефективності. Ізолінії відповідають сталому значенню:

$$\eta = \frac{T}{P},$$

де η — МБ/с на Вт.

Побудовано ізолінії:

- 5 МБ/с·Вт$^{-1}$;
- 20 МБ/с·Вт$^{-1}$;
- 100 МБ/с·Вт$^{-1}$;
- 300 МБ/с·Вт$^{-1}$.

На рисунку 3.11 наведено залежність пропускної здатності від споживаної потужності та ізолінії сталого значення енергоефективності (МБ/с на Вт), що дозволяє наочно порівняти варіанти реалізації з позиції інтегрального критерію продуктивність/потужність.

Аналіз наведених залежностей показує, що:

1. Спеціалізована реалізація забезпечує кратне зниження енергоспоживання.
2. Method 1 є більш енергоефективним.
3. ASIC-реалізація демонструє максимальний показник продуктивність/Вт.
4. Енергетична ефективність підтверджує доцільність використання МН-процесора у приватних (військових) хмарних інфраструктурах.

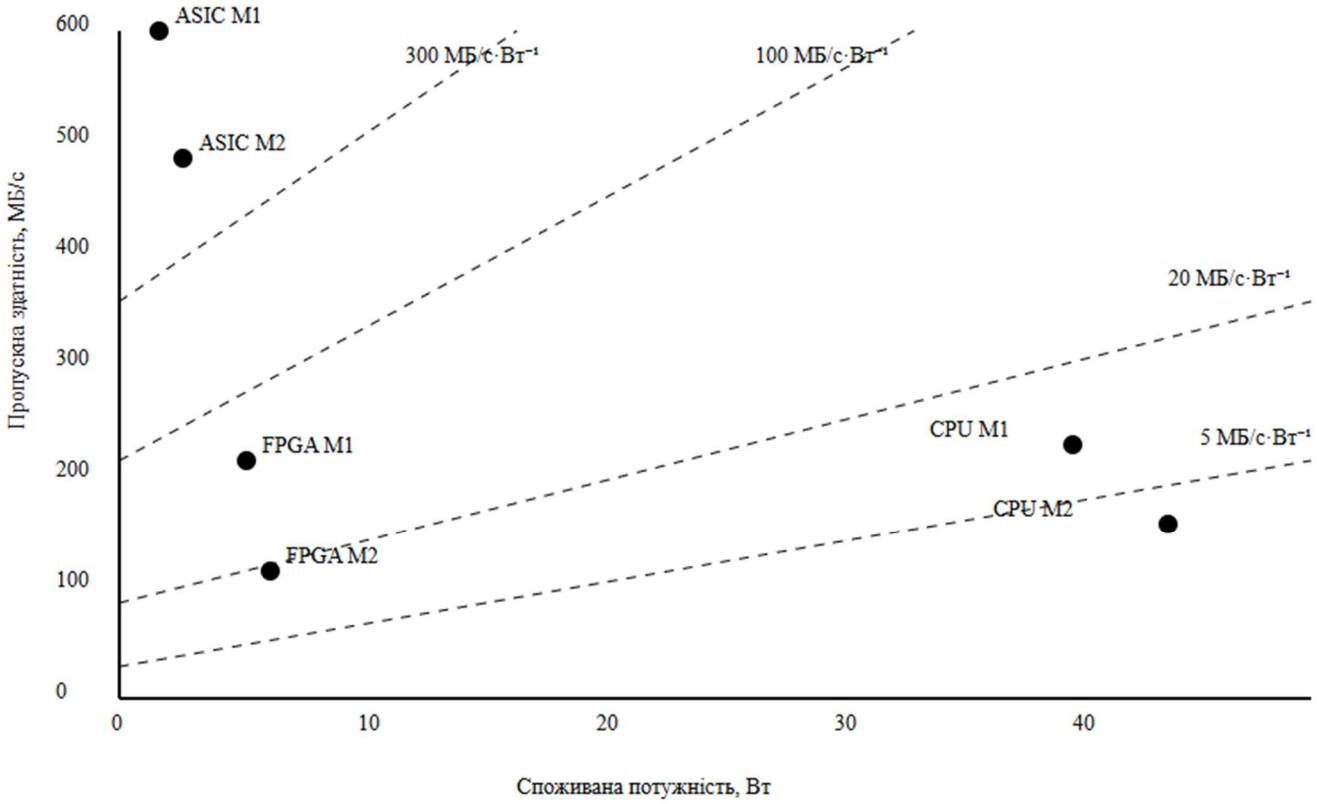

Рисунок 3.11 – Залежність пропускної здатності від споживаної потужності з ізолініями енергоефективності

3.4.4 Оцінки масштабованості

Однією з ключових характеристик спеціалізованої апаратної реалізації є її здатність до масштабування при зростанні навантаження. Для приватних хмарних інфраструктур ця властивість має принципове значення, оскільки можливості горизонтального масштабування за рахунок додавання серверів часто є обмеженими. У таких умовах пріоритетним стає масштабування на рівні апаратних модулів усередині одного вузла [20, 22].

Масштабованість МН-процесора оцінюється шляхом аналізу залежності пропускної здатності системи від кількості паралельно функціонуючих обчислювальних модулів. Нехай n — кількість незалежних екземплярів МН-процесора, інтегрованих у межах одного обчислювального вузла. Тоді коефіцієнт масштабованості визначається як:

$$S(n) = \frac{T(n)}{T(1)}, \tag{3.16}$$

де: $T(1)$— пропускна здатність одного модуля,

$T(n)$— сумарна пропускна здатність при використанні n модулів.

Ідеальне лінійне масштабування відповідає співвідношенню:

$$S(n) = n. \tag{3.17}$$

Відхилення від лінійності характеризує вплив спільних ресурсів (пам’ять, системна шина, кеш-ієрархія), а також накладні витрати диспетчеризації потоків.

Для оцінювання масштабованості проведено моделювання варіанту FPGA-реалізації з паралельним підключенням до спільного контролера пам’яті. Аналіз проводився для $n = 1, 2, 4, 8$.

Отримані експериментальні значення наведено на рис 3.12.

Як видно з рис. 3.10, масштабованість для обох методів наближається до ідеальної лінійної залежності $S(n) = n$. Відхилення не перевищує 4–7 %, що пояснюється накладними витратами синхронізації та доступу до спільної пам’яті. Метод 1 демонструє дещо кращі показники масштабованості за рахунок меншої глибини комбінаційної логіки та відсутності додаткових операцій множення.

Ефективність паралельної реалізації визначається як:

$$E(n) = \frac{S(n)}{n},$$

де: $S(n)$— коефіцієнт масштабованості,

n— кількість паралельних модулів.

Показник $E(n)$ характеризує ступінь використання обчислювальних ресурсів. Оцінки ефективності паралельної реалізації МН-процесора наведено в табл. 3.9

Таблиця 3.9 – Оцінки ефективності паралельної реалізації МН-процесора

n	**Method 1**		**Method 2**	
	$S(n)$	$E(n)$	$S(n)$	$E(n)$
1	1.00	1.00	1.00	1.00
2	1.98	0.99	1.95	0.975
4	3.90	0.975	3.82	0.955
8	7.65	0.956	7.40	0.925

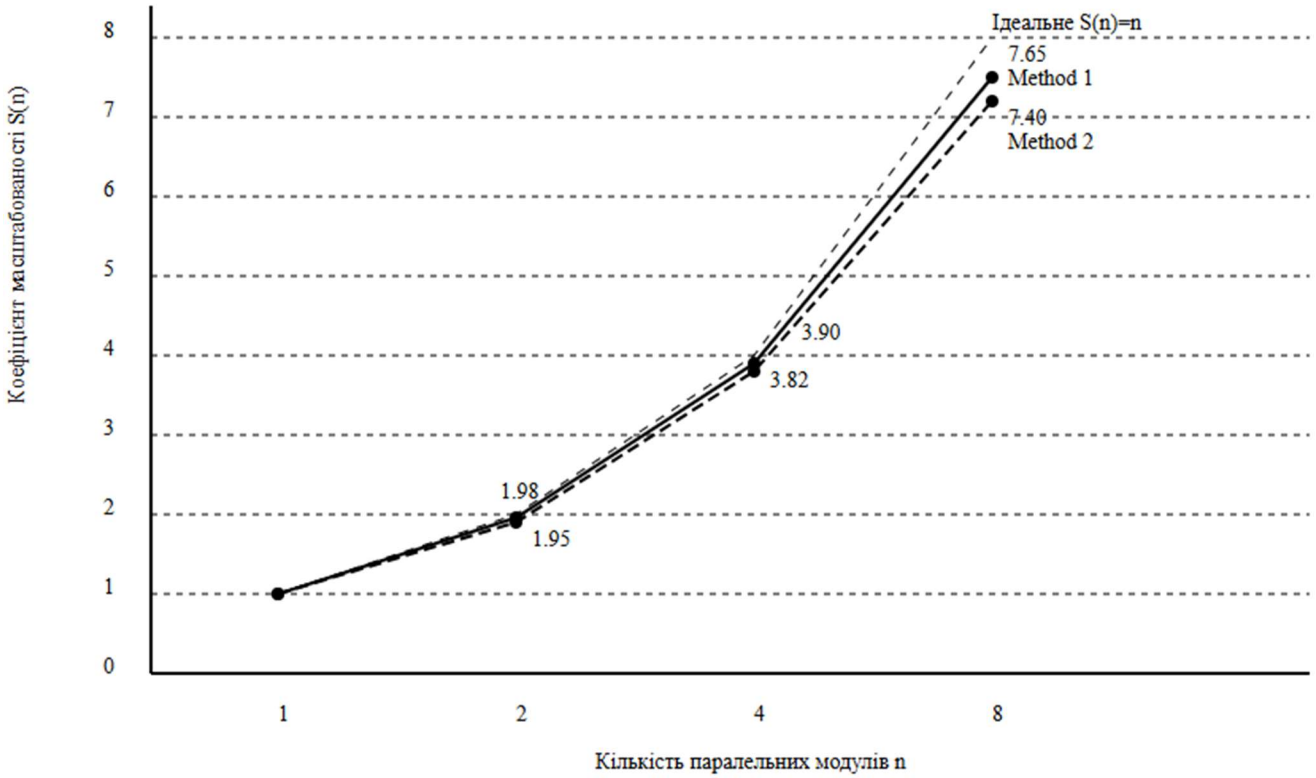

Рисунок 3.12 – Залежність коефіцієнта масштабованості $S(n)$ від кількості модулів

Аналіз даних табл. 3.9 свідчить, що ефективність паралельної реалізації обох методів залишається на високому рівні навіть при збільшенні кількості модулів до восьми. Для Методу 1 значення ефективності зменшується з 1.00 до 0.956, що відповідає втратам менше 5 %. Для Методу 2 зниження є дещо більш вираженим (до 0.925), що пояснюється більшою глибиною комбінаційної логіки та наявністю операцій множення.

Отримані результати підтверджують близьку до лінійної масштабованість архітектури МН-процесора та узгоджуються з теоретичними моделями паралельних обчислень та моделлю масштабованості Густафсона.

На відміну від багатораундових криптографічних геш-функцій, де внутрішній стан створює жорсткі залежності між етапами обробки [11, 14], запропоновані методи допускають просте дублювання функціональних блоків. Це суттєво спрощує горизонтальне масштабування в межах одного вузла та

узгоджується з сучасними тенденціями побудови хмарних прискорювачів обробки даних [23]. Таким чином, можна зробити такі висновки.

1. МН-процесор забезпечує масштабування, близьке до лінійного.
2. Втрати ефективності при зростанні кількості модулів є незначними (4-7%).
3. Архітектура придатна для реалізації у високонавантажених приватних хмарних системах.
4. Паралельна ефективність підтверджує доцільність використання байт-орієнтованого підходу.

3.5 Висновок

Запропоновано архітектуру багатошарової апаратно-програмної системи обробки даних, яка поєднує сервісний, програмний та апаратний рівні, з урахуванням принципів побудови приватних (військових) хмарних систем, розподілених обчислень та вимог до інформаційної безпеки і цілісності даних. Особливістю цієї архітектури полягає у тому, що введено спеціалізований сервіс гешування, на якому реалізуються апаратно-програмні методи гешування у вигляді окремих функціональних модулів або процесорів.

Розроблено функціональну та структурні моделі спеціалізованого МН-процесора байт-орієнтованого гешування, реалізація яких забезпечує оптимізацію обробки даних, лінійну складність обчислень $O(L)$, фіксовану латентність, масштабованість необхідного рівня для конкретних варіантів застосування та апаратні витрати, що задовольняють вимогам до малоресурсних засобів.

Проаналізовані варіанти реалізації МН-процесора на платформах CPU, FPGA, ASIC та SoC показали, що найбільш збалансованим з точки зору співвідношення продуктивність/гнучкість є FPGA-реалізація, тоді як ASIC забезпечує максимальну енергоефективність при фіксованій конфігурації.

Результати експериментальних досліджень показали, що реалізації запропонованих методів гешування та моделей МН-процесора є достатньо ефективними, оскільки:

- обидва методи демонструють лінійну залежність пропускної здатності від тактової частоти та кількості паралельних модулів;
- Метод 1 забезпечує вищу пропускну здатність при мінімальних апаратних витратах;
- Метод 2 забезпечує підвищений рівень дифузії за рахунок збільшення складності реалізації та енергоспоживання;
- масштабованість реалізації є близькою до лінійної ($S(n) \approx 0.96–0.99 \cdot n$), що підтверджує ефективність паралельної організації обробки;
- показник енергоефективності залишається стабільним при збільшенні кількості модулів, що є критично важливим для приватних та відомчих центрів обробки даних.

ВИСНОВКИ

У дисертаційній роботі розв’язано науково-технічну задачу підвищення ефективності організації роботи з даними у приватних (військових) хмарних сервісах шляхом розроблення формальних моделей, байт-орієнтованих методів гешування та спеціалізованого процесора для їх реалізації, а також апаратно-програмних засобів інтеграції у підсистеми зберігання й обробки даних.

Аналіз сучасного стану методів і засобів організації роботи з даними у хмарних сервісах показав, що незважаючи на високу криптографічну стійкість сучасних геш-функцій, вони не завжди задовольняють вимогам до детермінованої латентності, апаратної регулярності та енергоефективності при використанні в приватних хмарних інфраструктурах. Тому актуальними є дослідження спрямовані на розроблення спеціалізованих методів гешування, орієнтованих на байтову обробку великих масивів даних у ресурсно-обмежених середовищах.

Розроблена модель процесів роботи з файлами та об’єктами баз даних у приватній хмарній інфраструктурі має особливість, яка полягає в наявності окремого функціонального рівня сервісу гешування, що інтегрується у різні контури обробки даних без дублювання логіки гешування у клієнтських або прикладних компонентах, завдяки чому зменшується навантаження на центральні процесори та покращується керованість системи. Така модель є формалізованою основою для синтезу апаратно-програмних компонентів підсистеми гешування.

Розроблені методи гешування принципово відрізняються від відомих підходів щодо побудови геш-функцій використанням характеристичних ознак повідомлення, що формуються шляхом його побайтового аналізу. Використання криптографічної солі додатково до змішування характеристичних ознак забезпечує зменшення ймовірності колізій для близьких за структурою повідомлень без переходу до багатораундових криптографічних перетворень. Такий підхід забезпечує пришвидшення процесу гешування до 50 % порівняно з програмними реалізаціями відомих геш-функцій та до 3,2 раз у випадку апаратної реалізації засобу.

Розроблені функціональна та структурні моделі спеціалізованого МН-процесора орієнтовані на паралельну обробку характеристичних ознак даних, що забезпечує пришвидшення процесів гешування у разі апаратної реалізації. Ці моделі забезпечують реалізацію режиму MODE (вибір методу гешування) без зміни базової структурної організації, що підвищує універсальність рішення та зменшує апаратні витрати при масштабуванні. Ці моделі враховують особливості елементної бази для апаратної реалізації МН-процесора, а саме FPGA та ASIC. Перевагами FPGA є можливість досягнення високої тактової частоти при збереженні детермінованої латентності, що є критично важливим для потокових контурів обробки та масштабування системи шляхом конфігурування декількох МН-процесорів у межах одного кристалу. Оскільки запропоновані методи гешування не передбачають багатораундових ітераційних перетворень, то реалізація МН-процесора на ASIC дозволяє скоротити латентність і зменшити енергоспоживання. Апаратна реалізація на основі FPGA та ASIC є доцільною при створені кіберфізичних систем у ресурсно-обмежених та критично важливих середовищах, що зокрема є складовою багатошарової архітектури приватних (військових) хмарних сервісів. Порівняно з програмною реалізацією на CPU така апаратна реалізація зменшує добове енергоспоживання підсистеми гешування до 43 %.

Розроблена багатошарова архітектура підсистеми організації роботи з даними у приватних хмарних сервісах інтегрує спеціалізований процесор гешування у серверні та клієнтські компоненти приватної хмарної інфраструктури. Це зменшує завантаженість центральних процесорних вузлів, вузлів зберігання та передавання даних, а також зменшує затримки при виконанні операцій з контролю цілісності та дедуплікації. Така архітектура покращує оптимізацію обробки даних, забезпечує скорочення обсягу дубльованих даних у сховищах на 25–40 % та зменшення часу виявлення порушень цілісності на 27–36 %.

Розроблена методика комплексного оцінювання ефективності організації роботи з даними у приватних хмарних сервісах забезпечує формалізоване порівняння запропонованих методів гешування та їх апаратних реалізацій за ключовими інженерними критеріями — швидкодією, енергоспоживанням,

масштабованістю та детермінованістю часових характеристик. Ці критерії застосовані для оцінювання програмної реалізація на CPU, апаратної реалізації на FPGA та ASIC-реалізації.

Проведені експериментальні дослідження розроблених методів та програмно-апаратних засобів підтвердили коректність оцінок, отриманих за розробленою методикою комплексного оцінювання ефективності організації роботи з даними у приватних хмарних сервісах.

Таким чином, усі сформульовані задачі дослідження розв’язано і отримано наукові та практичні результати, які підтверджують досягнення поставленої мети дослідження.

Отримані наукові результати створюють підґрунтя для подальших досліджень у напрямку розвитку спеціалізованих обчислювальних засобів для обробки даних у критично важливих інформаційних системах.

Практична цінність отриманих результатів полягає у можливості їх застосування при проєктуванні нових та модернізації існуючих приватних і відомчих хмарних систем, зокрема в умовах обмежених ресурсів, підвищених вимог до цілісності та безперервності функціонування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 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.
2. Mell P., Grance T. The NIST Definition of Cloud Computing. – Gaithersburg: NIST, 2011. – (NIST Special Publication 800-145).
3. Buyya R., Broberg J., Goscinski A. (eds.). Cloud Computing: Principles and Paradigms. – Hoboken: Wiley, 2011. – 637 p.
4. Zhang Q., Cheng L., Boutaba R. Cloud computing: state-of-the-art and research challenges // Journal of Internet Services and Applications. – 2010. – Vol. 1. – P. 7–18.
5. Sultan N. Cloud computing for education: A new dawn? // International Journal of Information Management. – 2010. – Vol. 30, No. 2. – P. 109–116. – DOI: 10.1016/j.ijinfomgt.2009.09.004.
6. Al-Zoube M. E-learning on the cloud // International Arab Journal of e-Technology. – 2009. – Vol. 1, No. 2. – P. 58–64.
7. Almajalid R. A survey on the adoption of cloud computing in education sector [Electronic resource]. – 2017. – URL: https://arxiv.org/abs/1706.01136
8. Han H., Trimi S. Cloud computing-based higher education platforms during the COVID-19 pandemic [Electronic resource]. – 2022. – URL: https://arxiv.org/abs/2203.03714
9. Brynjolfsson E., Kahin B. Understanding the Digital Economy. – Cambridge: MIT Press, 2002. – 363 p.
10. Licklider J. C. R. Man-computer symbiosis // IRE Transactions on Human Factors in Electronics. – 1960. – Vol. 1. – P. 4–11.
11. Stallings W. Cryptography and Network Security: Principles and Practice. – 7th ed. – Boston: Pearson, 2017. – 768 p.
12. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. – 3rd ed. – Hoboken: Wiley, 2020. – 1248 p.

13. Bishop M. Computer Security: Art and Science. – 2nd ed. – Boston: Addison-Wesley, 2018. – 1216 p.
14. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. – Boca Raton: CRC Press, 1996. – 816 p.
15. Katz J., Lindell Y. Introduction to Modern Cryptography. – 3rd ed. – Boca Raton: CRC Press, 2020. – 623 p.
16. Rogaway P., Shrimpton T. Cryptographic hash-function basics: definitions, implications, and separations // Fast Software Encryption (FSE 2004). – Berlin: Springer, 2004. – P. 371–388.
17. Wang X., Yu H. How to break MD5 and other hash functions // EUROCRYPT 2005. – Berlin: Springer, 2005. – P. 19–35.
18. National Institute of Standards and Technology. Secure Hash Standard (SHA-3). – 2015. – (FIPS PUB 202).
19. Bertoni G., Daemen J., Peeters M., Van Assche G. Sponge functions [Electronic resource]. – 2007. – URL: https://keccak.team/files/SpongeFunctions.pdf
    a. (дата звернення: 20.06.2025).
20. Hennessy J. L., Patterson D. A. Computer Architecture: A Quantitative Approach. – 6th ed. – Cambridge: Morgan Kaufmann, 2019. – 856 p.
21. Kuon I., Tessier R., Rose J. FPGA architecture: survey and challenges // Foundations and Trends in Electronic Design Automation. – 2008. – Vol. 2, No. 2. – P. 135–253.
22. Kaps J. Cryptographic hardware implementations // Cryptographic Hardware and Embedded Systems (CHES 2006). – Berlin: Springer, 2006. – P. 1–3.
23. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection — Information security management systems — Requirements. – Geneva: ISO/IEC, 2022.
24. ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection — Information security controls. – Geneva: ISO/IEC, 2022.

25. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» : із змінами та доповненнями [Електронний ресурс]. – URL: https://zakon.rada.gov.ua
    a. (дата звернення: 20.06.2025).
26. Закон України «Про основні засади забезпечення кібербезпеки України» : із змінами та доповненнями [Електронний ресурс]. – URL: https://zakon.rada.gov.ua
    a. (дата звернення: 20.06.2025).
27. Глущенко Б. І. Перспективи розвитку та використання хмарних технологій державного сектору: кращі практики зарубіжного досвіду // Інформація і право. – 2021. – № 2(37). – С. 42–49. – DOI: 10.37750/2616-6798.2021.2(37).238334.
28. Гриценко Л. С. Проблеми інформаційної безпеки при використанні хмарних технологій // Інформаційна безпека людини, суспільства, держави. – 2021. – Т. 36, № 2. – С. 61–67. – DOI: 10.31673/2412-2119.2021.2.08.
29. Заблоцька Т. В. Інформаційна безпека у хмарних обчисленнях: сучасні виклики // Інформаційні технології в освіті. – 2020. – № 38. – С. 77–83. – DOI: 10.14308/ite2020.38.77.
30. Власенко І. М., Білецький В. С. Використання хмарних технологій в системах управління даними сучасних організацій // Наукові записки Українського науково-дослідного інституту зв’язку. – 2022. – № 2(32). – С. 67–74. – DOI: 10.33598/2414-4013-2022-32-67.
31. Волошина І. Г., Козак О. П. Хмарні технології для обробки великих даних: стан та перспективи // Комп’ютерні науки та інформаційні технології. – 2022. – № 4(36). – С. 99–105. – DOI: 10.15407/csIT2022.04.099.
32. Мельник Н. І., Тарасенко В. Ю. Хмарні технології в забезпеченні кібербезпеки критичної інфраструктури // Збірник наукових праць «Інформаційна безпека». – 2024. – № 2(26). – С. 22–28. – DOI: 10.31734/2412-2119.2024.2.22.

33. Костюк, Ю., Складанний, П., Рзаєва, С., Самойленко, Ю., & Коршун, Н. (2025). Інтелектуальні системи керування та захисту в кіберфізичних і хмарних середовищах smart grid. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(30), 125–156. - DOI: 10.28925/2663-4023.2025.30.956.
34. Панченко Т., Тузова І., Тузов О., Чумак О. Хмарні сервіси та огляд їх постачальників // Scientific Collection «InterConf+». – 2024. – Т. 43(193). – С. 550–559. – DOI: 10.51582/interconf.19-20.03.2024.053.
35. Shchemelinin D. Deployment and Monitoring of Internet Multiservice System in the Cloud. – Saarbrücken: LAP LAMBERT Academic Publishing, 2018. – 100 p.
36. Кисюк Д. В. Аналітичний огляд постачальників хмарних послуг // Measuring and Computing Devices in Technological Processes. – 2025. – № 3. – С. 105–111. – DOI: 10.31891/2219-9365-2025-83-14.
37. Кисюк Д. В. Актуальність проблеми впровадження хмарних сервісів для організації роботи з даними // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2025. – Т. 356, № 5. – DOI: 10.31891/2307-5732-2025-357-24.
38. Кисюк Д. В., Захарченко С. М. Оптимізація застосування хмарних сервісів для розподіленої обробки даних в системах колективного доступу // Наукові праці ВНТУ. – 2025. – Вип. 2. – С. 66–72. – DOI: 10.31649/2307-5376-2025-2-66-72.
39. Лужецький В. А., Кисюк Д. В. Новий підхід до побудови байт-орієнтованих криптографічних геш-функцій. // Вимірювальна техніка та метрологія Випуск 87 (1), ISTCMTM, Львів, – 2026. – Номер 1. – С. 51-58. – DOI: https://doi.org/10.23939/istcmtm2026.01.051.
40. Лужецький В. А., Кисюк Д. В. Узагальнений метод хешування байтової форми представлення інформації // Тези доповідей Четвертої Міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія». – Вінниця: ВНТУ, 2014. – С. 184–186. – URL: http://ir.lib.vntu.edu.ua/handle/123456789/14878

41. Лужецький В. А., Кисюк Д. В. Новий підхід до побудови криптографічних хеш-функцій // Матеріали П’ятої Міжнародної науково-практичної конференції «Інформаційні технології та комп’ютерна інженерія». – Івано-Франківськ: ФОП Голіней О. М., 2015. – С. 149–150. – URL: http://ir.lib.vntu.edu.ua/handle/123456789/14457
42. Лужецький В. А., Савицька Л. А., Кисюк Д. В. Спеціалізований процесор для ущільнення даних // Тези доповідей П’ятої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації». – Вінниця: ВНТУ, 2016. – С. 108–110. – URL: http://ir.lib.vntu.edu.ua/handle/123456789/11399
43. Лужецький В. А., Кисюк Д. В., Савицька Л. А. Метод байт-орієнтованого хешування // Тези доповідей П’ятої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації». – Вінниця–Немирів, 2016. – С. 37–38. – URL: http://ir.lib.vntu.edu.ua/handle/123456789/9992
44. Лужецький В. А., Кисюк Д. В. Метод хешування даних на основі їх характеристичних ознак // Тези доповідей ІІ Міжнародної науково-практичної конференції «Актуальні питання забезпечення кібербезпеки та захисту інформації». – Київ: Європейський університет, 2016. – С. 100–102. – URL: http://ir.lib.vntu.edu.ua/handle/123456789/15351
45. Лужецький В. А., Кисюк Д. В., Комаров А. О. Метод та спеціалізований процесор для байт-орієнтованого хешування даних // Тези доповідей Міжнародної науково-практичної конференції «Інформаційні технології та комп’ютерне моделювання (ІТСМ)». – Івано-Франківськ: ПНУ ім. В. Стефаника, 2016. – С. 123–125. – URL: http://itcm.pnu.edu.ua/2016/docs/Luzhetskyi.pdf
46. Кисюк Д. В., Захарченко С. М. Аналіз переваг використання хмарних сервісів для обробки даних у сучасних системах управління базами даних [Електронний ресурс] // Матеріали LII Науково-технічної конференції підрозділів ВНТУ. – Вінниця, 2023. – URL:

https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/18183

47. Кисюк Д. В., Лужецький В. А. Програмний засіб для побудови хеш-значення за допомогою байт-орієнтованої обробки даних для мобільних пристроїв [Електронний ресурс] // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)». – Вінниця, 2023. – URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/18180
48. Мазур Б. С., Кисюк Д. В. Хмарні сервіси. Обробка та збереження даних за допомогою хмарних сховищ [Електронний ресурс] // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)». – Вінниця, 2024. – URL: https://iq.vntu.edu.ua/method/getfile.php?fname=153063.docx&x=1
49. Кисюк Д. В., Шулдик С. С. Хмарні сервіси як інструмент управління проектами у системах забезпечення ресурсами підприємства SAP ERP [Електронний ресурс] // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». – Вінниця, 2025. – URL: https://iq.vntu.edu.ua/method/getfile.php?fname=179474.pdf&x=1
50. Кисюк Д. В., Заруцький М. В. Перспективи впровадження штучного інтелекту в хмарних обчисленнях [Електронний ресурс] // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». – Вінниця, 2025. – URL: https://iq.vntu.edu.ua/method/getfile.php?fname=169386.pdf&x=1
51. Кисюк Д. В., Сірак В. О. Методи підвищення ефективності зберігання та обробки великих обсягів даних у приватних хмарних сервісах [Електронний ресурс] // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)». – Вінниця, 2026. – URL:

https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/viewFile/27079/22199

52. Amazon Web Services. *AWS Well-Architected Framework: Security Pillar* [Electronic resource]. – 2023. – URL: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar (дата звернення: 20.06.2025).
53. Microsoft. *Azure Security Benchmark* [Electronic resource]. – 2023. – URL: https://learn.microsoft.com/security/benchmark (дата звернення: 20.06.2025).
54. Google Cloud. *Security Foundations Guide* [Electronic resource]. – 2024. – URL: https://cloud.google.com/security/foundations (дата звернення: 20.06.2025).
55. Subashini S., Kavitha V. A survey on security issues in service delivery models of cloud computing // *Journal of Network and Computer Applications*. – 2011. – Vol. 34, No. 1. – P. 1–11. – DOI: 10.1016/j.jnca.2010.07.006.
56. Ristenpart T., Tromer E., Shacham H., Savage S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds // *Proceedings of the ACM CCS*. – 2009. – P. 199–212. – DOI: 10.1145/1653662.1653687.
57. Popović K., Hocenski Ž. Cloud computing security issues and challenges // *Proceedings of the 33rd International Convention MIPRO*. – 2010. – P. 344–349.
58. Pearson S., Benameur A. Privacy, security and trust issues arising from cloud computing // *Cloud Computing*. – Springer, 2010. – P. 3–42.
59. Jensen M., Schwenk J., Gruschka N., Iacono L. A. On technical security issues in cloud computing // *IEEE International Conference on Cloud Computing*. – 2009. – P. 109–116. – DOI: 10.1109/CLOUD.2009.60.
60. Harnik D., Pinkas B., Shulman-Peleg A. Side channels in cloud services: Deduplication in cloud storage // *IEEE Security & Privacy*. – 2010. – Vol. 8, No. 6. – P. 40–47. – DOI: 10.1109/MSP.2010.187.
61. Meyer D. T., Bolosky W. J. A study of practical deduplication // *Proceedings of USENIX FAST*. – 2012. – P. 1–14.
62. Bellare M., Rogaway P. Random oracles are practical: a paradigm for designing efficient protocols // *Proceedings of the ACM CCS*. – 1993. – P. 62–73.

63. Damgård I. A design principle for hash functions // *Advances in Cryptology – CRYPTO'89*. – Springer, 1990. – P. 416–427.
64. Merkle R. C. One way hash functions and DES // *Advances in Cryptology – CRYPTO'89*. – Springer, 1990. – P. 428–446.
65. Halevi S., Krawczyk H. Strengthening digital signatures via randomized hashing // *Advances in Cryptology – CRYPTO 2006*. – Springer, 2006. – P. 41–59.
66. Boneh D., Shoup V. *A Graduate Course in Applied Cryptography*. – Stanford University, 2020. – 560 p.
67. Kahn Academy. *Cryptographic hash functions* [Electronic resource]. – 2022. – URL: https://www.khanacademy.org/computing/computer-science/cryptography (дата звернення: 20.06.2025).
68. Diffie W., Hellman M. New directions in cryptography // *IEEE Transactions on Information Theory*. – 1976. – Vol. 22, No. 6. – P. 644–654.
69. Rivest R. L. The MD5 message-digest algorithm. – RFC 1321. – 1992.
70. Eastlake D., Hansen T. US secure hash algorithms (SHA and SHA-based HMACs). – RFC 6234. – 2011.
71. Schneier B. *Secrets and Lies: Digital Security in a Networked World*. – New York: Wiley, 2004. – 412 p.
72. Daemen J., Rijmen V. *The Design of Rijndael: AES – The Advanced Encryption Standard*. – Berlin: Springer, 2002. – 238 p.
73. Behl A., Behl K. *Cyberwar: The Next Threat to National Security and What to Do About It*. – Oxford: Oxford University Press, 2017. – 304 p.
74. NATO CCDCOE. *Tallinn Manual 2.0 on the International Law Applicable to Cyber Operations*. – Cambridge: Cambridge University Press, 2017. – 598 p.
75. ENISA. *Cloud Security Guide for Governmental Organizations* [Electronic resource]. – 2021. – URL: https://www.enisa.europa.eu (дата звернення: 20.06.2025).
76. European Union Agency for Cybersecurity. *Security Framework for Cloud Services* [Electronic resource]. – 2022. – URL:

https://www.enisa.europa.eu/topics/cloud-and-big-data (дата звернення: 20.06.2025).

77. NIST. *Security and Privacy Controls for Information Systems and Organizations*. – NIST SP 800-53 Rev. 5. – 2020.
78. NIST. *Zero Trust Architecture*. – NIST SP 800-207. – 2020.
79. ISO/IEC 27017:2015. *Code of practice for information security controls based on ISO/IEC 27002 for cloud services*. – Geneva: ISO/IEC, 2015.
80. ISO/IEC 27018:2019. *Protection of personally identifiable information (PII) in public clouds*. – Geneva: ISO/IEC, 2019.
81. Garfinkel S. L., Rosenberg B. *Database Nation: The Death of Privacy in the 21st Century*. – Sebastopol: O'Reilly, 2000. – 312 p.
82. Tanenbaum A. S., van Steen M. *Distributed Systems: Principles and Paradigms*. – 2nd ed. – Boston: Pearson, 2007. – 704 p.
83. Coulouris G., Dollimore J., Kindberg T., Blair G. *Distributed Systems: Concepts and Design*. – 5th ed. – Boston: Addison-Wesley, 2011. – 1024 p.
84. Kahn J., Wilcox-O'Hearn B. *Merkle trees and distributed systems* // *ACM Queue*. – 2018. – Vol. 16, No. 3. – P. 30–47.
85. Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S. *Bitcoin and Cryptocurrency Technologies*. – Princeton: Princeton University Press, 2016. – 336 p.
86. Crosby S., Wallach D. Efficient data structures for tamper-evident logging // *USENIX Security Symposium*. – 2009. – P. 317–334.
87. Haber S., Stornetta W. How to time-stamp a digital document // *Journal of Cryptology*. – 1991. – Vol. 3, No. 2. – P. 99–111.
88. Chen L., Li X., Li Y. Data integrity protection in cloud storage // *Journal of Cloud Computing*. – 2017. – Vol. 6. – P. 1–15. – DOI: 10.1186/s13677-017-0080-5.
89. Ateniese G., Burns R., Curtmola R., Herring J., Kissner L., Peterson Z., Song D. Provable data possession at untrusted stores // *Proceedings of the ACM CCS*. – 2007. – P. 598–609.

90. Juels A., Kaliski B. Pors: proofs of retrievability for large files // *Proceedings of the ACM CCS*. – 2007. – P. 584–597.
91. Erway C., Küpçü A., Papamanthou C., Tamassia R. Dynamic provable data possession // *Proceedings of the ACM CCS*. – 2009. – P. 213–222.
92. Заблоцька, Т. В. (2020). Інформаційна безпека у хмарних обчисленнях: сучасні виклики. *Збірник наукових праць «Інформаційні технології в освіті»*, 38, 77–83. https://doi.org/10.14308/ite2020.38.77
93. Заболотна, І. В. (2022). Цифровізація підприємств за допомогою хмарних платформ. *Бізнес Інформ*, 10(547), 120–126. https://doi.org/10.32983/2222-4459-2022-10-120-126
94. Зінченко, Ю. В. (2023). Можливості використання хмарних технологій у діяльності місцевих громад. *Державне управління: удосконалення та розвиток*, 5(24), 29–35. https://doi.org/10.32782/2414-0562/2023.5.29
95. Зубенко, В. В. (2020). Можливості хмарних сервісів для оптимізації електронного документообігу. Менеджмент та підприємництво в Україні: етапи становлення і проблеми розвитку, 2(29), 89–96. http://nbuv.gov.ua/UJRN/mpu_2020_2_12
96. Мироненко, О. Г. (2021). Використання хмарних технологій у цифровій трансформації публічної адміністрації. *Публічне адміністрування: теорія та практика*, 3(25), 45–51. https://doi.org/10.32782/patp.2021.3.45
97. Міщенко, М. П. (2022). Використання хмарних рішень у діяльності промислових підприємств. *Економіка промисловості*, 4(98), 111–118. https://doi.org/10.15407/econindustry2022.04.111
98. Москаленко, П. К. (2022). Аспекти кіберзахисту при впровадженні хмарних технологій. *Право і безпека*, 31(1), 37–43. https://doi.org/10.31733/2786-5220.2022.31.1.37
99. Ільїна, С. М. (2021). Хмарні сервіси як чинник цифровізації публічного управління. *Науковий вісник Ужгородського національного університету. Серія: Право*, 64(1), 123–127. https://doi.org/10.24144/2307-3322.2021.64.1.23

100. Гончаренко, В. І. (2024). Хмарні технології як інструмент підвищення ефективності діяльності органів державної влади. *Державне управління: теорія та практика*, 2(28), 52–59. https://doi.org/10.32782/du.2024.2.7

101. Семенюк, В. І. (2023). Інноваційні аспекти впровадження хмарних сервісів у державному управлінні. *Державне управління та місцеве самоврядування*, 1(48), 75–81. https://doi.org/10.32782/dums.2023.1.75

102. Марущак, О. В., & Руденко, О. С. (2024). Використання хмарних технологій в організації електронного урядування. *Публічне управління та адміністрування в Україні*, 28(1), 88–95. https://doi.org/10.32782/pua.2024.28.11

103. Кравченко, І. А. (2024). Ефективність використання хмарних сервісів у публічному управлінні. *Публічне адміністрування: теорія та практика*, 1(29), 23–29. https://doi.org/10.32782/patp.2024.1.4

104. Пащенко, С. В. (2023). Моделі впровадження хмарних сервісів для електронного врядування. *Державне управління: удосконалення та розвиток*, 4(22), 31–37. https://doi.org/10.32782/2414-0562/2023.4.31

105. Дьяків, С. А. (2024). Хмарні технології як фактор цифрової трансформації місцевого самоврядування. *Публічне адміністрування*, 3(17), 73–79. https://doi.org/10.32782/pa.2024.3.73

106. Гнатенко, О. П. (2022). Хмарні технології у підтримці цифрової трансформації закладів освіти. *Інформаційні технології і засоби навчання*, 89(5), 23–29. https://doi.org/10.33407/itlt.v89i5.5039

107. Гнатенко, П. І. (2021). Хмарні сервіси в цифровій трансформації освітніх закладів України. Інформаційні технології і засоби навчання, 84(4), 144–156. https://doi.org/10.33407/itlt.v84i4.4585

108. Лужецький В. А., Захарченко С. М., Кисюк Д. В. Метод байт-орієнтованого гешування з підвищеним рівнем дифузії та спеціалізований МН-процесор для приватних хмарних інфраструктур // Measuring and Computing Devices in Technological Processes, – 2026. – Т. 134-141, № 85(1) – DOI: https://doi.org/10.31891/2219-9365-2026-85-16.

109. В. І. Селезньов and В. А. Лужецький, “Метод малоресурсного гешування типу «Дані – Генератор»,” *Кібербезпека: освіта, наука, техніка,* vol. 2, no. 22, pp. 84–95, 2023. DOI: 10.28925/2663-4023.2023.22.8495

110. В. А. Лужецький and В. І. Селезньов, “Огляд підходів та методів малоресурсного гешування даних,” Вісник Вінницького політехнічного інституту, no. 6, p. 14, 2025. DOI: 10.31649/1997-9266-2025-183-6-99-112

111. V. A. Luzhetskyi and V. I. Seleznov, “Hardware implementation of the HDG hash function,” *Bull. Cherkasy State Technol. Univ.*, vol. 30, no. 2, pp. 10–21, 2025. DOI: 10.62660/bcstu/2.2025.10

112. R. Agrawal, L. de Castro, G. Yang, C. Juvekar, R. Yazicigil, A. Chandrakasan, V. Vaikuntanathan, and A. Joshi, “FAB: An FPGA-based Accelerator for Bootstrappable Fully Homomorphic Encryption,” IEEE Symposium on High-Performance Computer Architecture, 2022. DOI: 10.1109/HPCA56546.2023.10206839

113. P. Perazzo et al.,“On hardware acceleration of quantum-resistant FOTA cryptography,” *Computer Methods and Programs in Biomedicine*, vol. 247, 2024. DOI: 10.1016/j.cmpb.2024.108089.

114. M. K. Ahmed, J. Mandebi, S. K. Saha, and C. Bobda, “Multi-Tenant Cloud FPGA: A Survey on Security,” *ACM Computing Surveys*, 2022. DOI: 10.1145/3713078

115. A. Raghunath, S. Rekha, and S. Kumar K., “Low Power Hardware Accelerator for Hash Operations in Modern Processors,” in *Proc. International Conference on VLSI Systems*, 2024. DOI:10.1109/ICNEWS60873.2024.10731008

116. M. Khan, E. Kozyri, D. Johansen and H. Dagenborg, "Survey of Lightweight Hardware-Based Hash Functions for Security in Constrained IoT Devices," in *IEEE Access*, vol. 13, pp. 164508-164527, 2025, DOI: 10.1109/ACCESS.2025.3611280.

ДОДАТКИ

ДОДАТОК А. Документи щодо впровадження результатів роботи

МЕМОРАНДУМ

про наміри впровадження результатів науково-дослідної роботи

9 Центр захисту інформації та кібербезпеки в інформаційно-телекомунікаційних системах, розглянувши результати дисертаційного дослідження на тему «Методи та засоби організації роботи з даними у приватних хмарних сервісах на основі байт-орієнтованого гешування», підтверджує доцільність та зацікавленість у впровадженні отриманих наукових результатів.

Розроблені методи спрямовані на підвищення ефективності оброблення даних, забезпечення контролю їх цілісності, оптимізацію використання обчислювальних ресурсів та підвищення надійності функціонування приватних хмарних середовищ оборонного призначення.

Практичне застосування результатів можливе у складі сервісів керування даними, підсистем зберігання та маршрутизації інформаційних потоків, механізмів контролю цілісності та інформаційно-аналітичних систем військового призначення.

Передбачається проведення дослідної експлуатації, оцінка ефективності запропонованих методів у реальних умовах функціонування, а також підготовка рекомендацій щодо інтеграції рішень у діючі та перспективні інформаційні системи.

Вважаємо результати дослідження такими, що мають практичну значущість та можуть бути використані для розвитку цифрової інфраструктури військової частини.

Командир / уповноважена особа

9 Центр захисту інформації та кібербезпеки в ІТС

_____ підполковник _____ М. Рабус _____

М.П. (ПІБ)

м. Вінниця, 2026 р.

МЕМОРАНДУМ

ПРО НАМІРИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ НАУКОВО-ДОСЛІДНОЇ РОБОТИ

Військова частина А4717, розглянувши результати науково-дослідної роботи, виконаної в межах дисертаційного дослідження, повідомляє про доцільність та наміри впровадження отриманих наукових результатів.

Зміст та прикладна спрямованість результатів

У рамках дисертаційного дослідження розроблено методи та засоби організації роботи з даними, що базуються на застосуванні байт-орієнтованого гешування з урахуванням специфіки функціонування приватних (військових) хмарних середовищ.

Запропоновані підходи спрямовані на забезпечення:

- ефективної ідентифікації та адресації даних;
- контролю цілісності та достовірності інформації;
- зниження обчислювальних витрат при обробленні великих обсягів даних;
- підвищення швидкодії та масштабованості сервісів;
- функціонування в умовах обмежених ресурсів та підвищених вимог до надійності.

Методи орієнтовані на використання у розподілених хмарних середовищах з обмеженим доступом та підвищеними вимогами до інформаційної безпеки.

Напрями можливого впровадження

Результати дисертаційного дослідження можуть бути використані у складі:

- сервісів керування даними у приватних хмарних інфраструктурах;
- підсистем зберігання, обліку та маршрутизації інформаційних потоків;
- механізмів контролю цілісності та узгодженості даних;
- інструментів попередньої обробки та фільтрації інформації;
- модулів підтримки прийняття рішень в автоматизованих інформаційно-аналітичних системах.

Особливу практичну цінність становить можливість застосування запропонованих методів у військових інформаційних системах, що функціонують у закритих або ізольованих хмарних середовищах.

Форма та етапи можливого впровадження

Передбачається можливість здійснення впровадження у такій формі:

- дослідна експлуатація розроблених методів у складі експериментальних програмно-алгоритмічних рішень;
- оцінка ефективності байт-орієнтованого гешування при роботі з даними у приватних хмарних сервісах;
- порівняльний аналіз із наявними підходами до організації оброблення та зберігання даних;
- формування рекомендацій щодо інтеграції результатів у діючі та перспективні інформаційні системи.

Висновок

Результати дисертаційного дослідження вважаємо такими, що мають **практичну значущість**, **прикладну спрямованість** та можуть бути використані при розвитку сучасних приватних (військових) хмарних сервісів та інформаційних систем оборонного призначення.

Цей меморандум підтверджує наміри щодо впровадження результатів науково-дослідної роботи у відповідних напрямах діяльності військової частини.

Керівник / уповноважена особа
Військової частини А4717

МАЩЕНКО Валерій Миколайович

М.П.

м. Київ, 2026 р.

ЗАТВЕРДЖУЮ
Заступник начальника Харківського національного університету Повітряних Сил імені Івана Кожедуба з навчальної роботи
доктор технічних наук, професор

Володимир ГАРШИН

"31" березня 2026 р.

АКТ
впровадження результатів дисертаційного дослідження
КИСЮКА ДМИТРА ВАСИЛЬОВИЧА
за темою: "Методи та засоби організації роботи з даними у приватних хмарних сервісах на основі байт-орієнтованого гешування", яка подана на здобуття ступеня доктора філософії з галузі знань 12 "Інформаційні технології" за спеціальністю 123 "Комп'ютерна інженерія" в освітній процес Харківського національного університету Повітряних Сил імені Івана Кожедуба

Комісія у складі представників кафедри радіоелектронних систем пунктів управління Повітряних Сил факультету автоматизованих систем управління та наземного забезпечення польотів авіації д.т.н., професора Василишина Володимира Івановича, д.т.н., доцента Михалевського Дмитра Валерійовича, к.т.н., доцента Чечуя Олександра Вікторовича склала цей акт про те, що у межах виконання дисертаційного дослідження було впроваджено його результати у освітній процес кафедри.

У межах дисертаційної роботи розроблено методи та засоби організації роботи з даними у приватних хмарних середовищах, що базуються на використанні байт-орієнтованого гешування та орієнтовані на ефективну ідентифікацію, адресацію і контроль цілісності даних за умов обмежених обчислювальних ресурсів.

Методи орієнтовані на використання у розподілених хмарних середовищах з обмеженим доступом та підвищеними вимогами до інформаційної безпеки.

Отримані наукові результати використано під час викладання навчальних дисциплін "Засоби криптографічного захисту інформації в спеціальних ІКС", "Захист інформації в електронних комунікаційних системах та мережах",

пов'язаних з обробленням даних, хмарними технологіями та інформаційною безпекою, при підготовці методичних матеріалів навчальних занять для здобувачів вищої освіти, які навчаються за освітньо-професійною програмою "Радіоелектронні системи та засоби командних пунктів Повітряних Сил" першого (бакалаврського) та другого (магістерського) рівнів вищої освіти спеціальності G5 "Електроніка, електронні комунікації, приладобудування та радіотехніка".

Впровадження результатів дисертаційного дослідження сприяло підвищенню практичної спрямованості освітнього процесу, ознайомленню здобувачів вищої освіти з сучасними підходами до організації роботи з даними у приватних (у тому числі військових) хмарних сервісах та забезпечення захисту інформації в електронних комунікаційних мережах.

Визначені результати дисертаційного дослідження сприятимуть формуванню компетентностей з організації та проведення заходів із захисту інформації на інформаційно-комунікаційних вузлах (вузлах зв'язку) пунктів управління Повітряних Сил, а також проводити експериментальні дослідження характеристик сучасних та перспективних засобів електронних комунікацій.

Результати впровадження обговорені та схвалені на засіданні кафедри радіоелектронних систем пунктів управління Повітряних Сил факультету автоматизованих систем управління та наземного забезпечення польотів авіації. Протокол № 9 від 20 березня 2026 року.

Голова комісії:
Начальник кафедри радіоелектронних систем пунктів управління Повітряних Сил
доктор технічних наук, професор Володимир ВАСИЛИШИН

Члени комісії:
Професор кафедри радіоелектронних систем пунктів управління Повітряних Сил
доктор технічних наук, доцент Дмитро МИХАЛЕВСЬКИЙ

Доцент кафедри радіоелектронних систем пунктів управління Повітряних Сил
кандидат технічних наук, доцент Олександр ЧЕЧУЙ

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи
та організації освітнього процесу
Вінницького національного технічного
університету
_____ Олександр ПЕТРОВ
«2» 04 2026 р.

АКТ
про впровадження результатів дисертаційної роботи
КИСЮКА ДМИТРА ВАСИЛЬОВИЧА
за темою: «Методи та засоби організації роботи з даними у приватних хмарних сервісах на основі байт-орієнтованого гешування», яка подана на здобуття ступеня доктора філософії з галузі знань 12 «Інформаційні технології» за спеціальністю 123 «Комп'ютерна інженерія»

Комісія у складі декана факультету інформаційних технологій та комп`ютерної інженерії, к.пед.н., доцента Кирилащук Світлани Анатоліївни, к.т.н., доцента кафедри обчислювальної техніки Крупельницького Леоніда Віталійовича та к.т.н., доцента кафедри обчислювальної техніки Миколи Геннадійовича Тарновського склала цей акт про те, що на кафедрі обчислювальної техніки Вінницького національного технічного університету у навчальний процес було впроваджено такі результати дисертаційної роботи Кисюка Д. В.:

- сучасних підходів щодо організації роботи з даними у хмарних сервісах;
- впровадження хмарних сервісів для організації роботи з даними;
- метод організації роботи з даними у приватних хмарних сервісах;
- структурну модель багатошарової апаратно-програмної підсистеми організації роботи з даними.

Зазначені результати використовуються під час проведення лекційних, лабораторних і практичних занять з дисциплін «Хмарні технології», «Комп`ютерні мережі» та «Мережні інформаційні технології».

Декан ФІТКІ, к.пед.н., доцент Світлана КИРИЛАЩУК

к.т.н., доцент Леонід КРУПЕЛЬНИЦЬКИЙ

к.т.н., доцент Микола ТАРНОВСЬКИЙ

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи
та організації освітнього процесу
Вінницького національного технічного
університету
_____ Олександр ПЕТРОВ
«02» 04 2026 р.

АКТ
про впровадження результатів дисертаційної роботи
КИСЮКА ДМИТРА ВАСИЛЬОВИЧА
за темою: «Методи та засоби організації роботи з даними у приватних хмарних сервісах на основі байт-орієнтованого гешування», яка подана на здобуття ступеня доктора філософії з галузі знань 12 «Інформаційні технології» за спеціальністю 123 «Комп'ютерна інженерія»

Комісія у складі декана факультету інформаційних технологій та комп'ютерної інженерії, к.пед.н., доцента Кирилащук Світлани Анатоліївни, к.т.н., доцента кафедри захисту інформації Войтович Олесі Петрівни та к.т.н., доцента кафедри захисту інформації Гарнаги Володимира Анатолійовича склала цей акт про те, що на кафедрі захисту інформації Вінницького національного технічного університету у навчальний процес впроваджено такі результати дисертаційної роботи Кисюка Д. В.:

- методи байт-орієнтованого гешування з використанням характеристих ознак повідомлення;
- моделі спеціалізованих процесорів для байт-орієнтованого гешування;
- програмний засіб для тестування методів байт-орієнтованого гешування з використанням характеристих ознак повідомлення.

Зазначені результати використовуються під час проведення лекційних, лабораторних і практичних занять з дисциплін «Прикладна криптологія» та «Криптографічний захист інформації», а також при виконанні курсових робіт з цих дисциплін здобувачами вищої освіти.

Декан ФІТКІ, к.пед.н., доцент Світлана КИРИЛАЩУК

к.т.н., доцент Олеся ВОЙТОВИЧ

к.т.н., доцент Володимир ГАРНАГА

ДОДАТОК Б. Програма моделювання методів гешування

```
package diser_5;
import java.io.*;
import org.apache.log4j.*;

public class Diser_5 {
  static Logger log = Logger.getLogger(Diser_5.class);
  final static int Num = 32;
  final static int mod = 256;

    public static void moveLeft(long[] array, int positions) {
        int size = array.length;
        for (int i = 0; i < positions; i++)
        {
            long temp = array[0];
            for (int j = 1; j < size; j++)
            {
                array[j-1] = array[j];
            }
            array[size-1] = temp;
        }
    }

    public static void chartPrint(long X[])
    {
        for(int i = 0; i < 256;  i++)
        {
            System.out.format("#%3d: %3d: ", i, X[i]);
            for(int d=0;d<=X[i];d++) {
                System.out.format("%s", "*");
            }
            System.out.print("\n");
        }
        System.out.print("\n");
    }

    public static void cyclicCod(long X[])
    {
        String str = "", newBit = "0";
        long n = 13;
        for(int i = 0; i < 31; i++)
        {
            int bit = ((n&1)>0?1:0);
            str += Integer.toString(bit);
            if(bit == 1)
            {
                n = n >> 1;
                n =  n ^ 20;
            }
            else
```

```
                {
                    n = n >> 1;
                }
            }
            newBit                                                                    =
Integer.toString(Integer.parseInt(str.substring(0,1))^Integer.parseInt(str.substring(str.length()-1)));
            n = Long.parseLong(str + newBit, 2);
            for(int i = 0; i < 256; i++)
            {
                X[i] = n;
                log.info(String.format("#" + i + "  " + str + newBit + " S[i] = " + X[i] + "%n"));
                str = str.substring(str.length()-1) + str.substring(0, str.length()-1);
                newBit                                                                =
Integer.toString(Integer.parseInt(str.substring(0,1))^Integer.parseInt(str.substring(str.length()-1)));
                n = Long.parseLong(str + newBit, 2);
            }
            log.info(String.format("%n"));
        }

        public static long[] doubleMul(long A[][])
        {
      //---------------------------------------m1 m2 -------------------------------
            long dil;
            int i, k, x1, l1 = 0, CF = 0;
            long[] mb1 = new long[Num];
            long[] mb2 = new long[Num];
            long[] ma1 = new long[Num];
            long[] ma2 = new long[Num];
            long[] c = new long[Num];
            long[] z = new long[Num];
            long[] s = new long[Num];
            for(i=Num-1; i>=0; i--)
            {
                k = 0;
                for(int j=Num-1; j>=0; j--)
                {
                    dil = A[0][j] * A[1][i];
                    mb1[k] = dil % mod;
                    mb2[k] = dil / mod;
                    dil = A[1][j] * A[0][i];
                    ma1[k] = dil % mod;
                    ma2[k] = dil / mod;
                    k++;
                }
                System.out.print("\nmb1:   ");
                for(x1=0; x1<Num; x1++) System.out.print(mb1[x1]+" ");
                System.out.print("\n");
                System.out.print("mb2:   ");
                for(x1=0; x1<Num ;x1++) System.out.print(mb2[x1]+" ");
                System.out.print("\n");
                for(x1=0; x1<80 ;x1++) System.out.print(".");
                System.out.print("\nma1:   ");
```

```
            for(x1=0; x1<Num; x1++) System.out.print(ma1[x1]+" ");
            System.out.print("\n");
            System.out.print("ma2:   ");
            for(x1=0; x1<Num ;x1++) System.out.print(ma2[x1]+" ");
            System.out.print("\n");
            for(x1=0; x1<80 ;x1++) System.out.print("-");
            System.out.print("\n");
//------------------------------------------ C ------------------------------
            System.out.print("c:     ");
            CF = 0;
            for(x1=Num-1; x1>=0 ;x1--)
            {
               c[x1] = mb1[x1] + mb2[x1] + ma1[x1] + ma2[x1] + CF;
               if(c[x1] > (3*mod-1)) CF = 3;
               else if(c[x1] > (2*mod-1)) CF = 2;
               else if(c[x1] > (mod-1)) CF = 1;
               else CF = 0;
               c[x1] = c[x1] % mod;
            }
            s[l1] += CF;
            for(x1=0; x1<Num ;x1++) System.out.print(c[x1]+" ");
            System.out.print("\n");
//------------------------------------------ Z ------------------------------
            System.out.print("z:     ");
            CF = 0;
            for(x1=Num-1; x1>=0 ;x1--)
            {
               z[x1] = z[x1] + c[x1] + CF;
               if(z[x1] > (3*mod-1)) CF = 3;
               else if(z[x1] > (2*mod-1)) CF = 2;
               else if(z[x1] > (mod-1)) CF = 1;
               else CF = 0;
               z[x1] = z[x1] % mod;
            }
            s[l1] += CF;
            for(x1=0; x1<Num ;x1++) System.out.print(z[x1]+" ");
            System.out.print("\n");
            for(x1=0; x1<80 ;x1++) System.out.print("*");
            System.out.print("\nCarry to s: " + s[l1] + "\n");
            l1++;
         }
         System.out.print("\ns:     ");
         for(x1=0; x1<Num ;x1++) System.out.print(s[x1]+" ");
         System.out.print("\n");
         for(x1=0; x1<80 ;x1++) System.out.print("=");
         System.out.print("\n");
         System.out.print("\nRES:   ");
         CF = 0;
         for(x1=Num-1; x1>=0 ;x1--)
         {
            s[x1] = s[x1] + z[x1] + CF;
            if(c[x1] > (3*mod-1)) CF = 3;
```

```
            else if(c[x1] > (2*mod-1)) CF = 2;
            else if(c[x1] > (mod-1)) CF = 1;
            else CF = 0;
            s[x1] = s[x1] % mod;
            }
        for(x1=0; x1<Num ;x1++) {
                System.out.print(s[x1]+" ");
              }
        System.out.print("\n");
        return s;
      }

    public static void main(String[] args) throws FileNotFoundException, IOException {
//========= LOG ====================
        File f = new File("Diser_log.log");
        f.delete();
        PropertyConfigurator.configure("src/diser_5/newproperties.properties");
//=========== READ FROM FILE ==== mas N ============
        String inFile = "data.txt";
        FileInputStream fs = new FileInputStream(inFile);

        long K[] = new long[256];
        long S[] = new long[256];
        long pos = 1;                       //delete
        int buf;
        int ch;

        //log.info(String.format("%nПочаткове заповнення:%n"));
        cyclicCod(S);

        log.info(String.format("%nВхідний файл:%n%n"));
        while((buf = fs.read())!=-1){
          ch = buf;
          log.info((char)ch);
          K[ch]++;
          S[ch] += K[ch] * pos;   //summa dobutkiv: kilkist*posyciu
          //log.info(String.format("%n%nS_old[%c]:%d%n",ch, S[ch]));
          S[ch]=(S[ch]%4294967296L)+(S[ch]/4294967296L);  //pryvedennya do 32 bit
          //log.info(String.format("S_new[%c]:%d%n",ch, S[ch]));
          pos++;
        }

        log.info(String.format("%n%nКількості символів:%n"));
        for(int i = 0; i < 256;  i++)
        {
          //if(K[i]!=0)
            log.info(String.format("#%3d: K[%c]-> %10d ->    S[]-> %3d   %n", i, (char)i, K[i],
S[i] ));
          if( (i+1)%8 == 0 ) {
            log.info(String.format("%n"));
          }
        }
```

```
        //chartPrint(N);

    //=========== BUILD ==== mas H and Ht ===============
        long Ht[][] = new long[32][32];
        long Hs[][] = new long[32][32];
        long H[] = new long[32];
        long q32 = 2147483648L;
        int i, j, p = 0;
        long x;
        long startByte = 0;
        long SumCarry[][] = new long[3][32];

        for(int k = 0; k < 32; k++)
        {
          //log.info(String.format("%n"));
          for(int l = 0; l < 32; l++)
          {
            int q8 = 128, sum = 0;
            for(i = p; i < p+8;  i++)
            {
              x = S[i] & q32 ;
              sum += (((x>0)?1:0) * q8);
              q8 /= 2;
            }
            p = i;
            Ht[k][l] = sum;
            SumCarry[0][k] += sum;
            //log.info(String.format("#%2d:  %3d  ->  %8s%n",l,  Ht[k][l],
Long.toBinaryString(Ht[k][l])));
          }
          for(int m=0;m<32;m++) Hs[k][m] = (SumCarry[0][k]/mod + SumCarry[0][k]%mod);
          q32 /= 2;
          p = 0;
        }
        log.info(String.format("%n%nMAS_M %n" ));
        for( i=0;i<154;i++) log.info(String.format("-"));
        log.info(String.format("%n"));
        for( i=0;i<32;i++)
        {
          log.info(String.format("#%2d: ",i));
          for( j=0;j<32;j++)
          {
            log.info(String.format("%3d ",Ht[i][j]));
          }
          log.info(String.format("   %3d%n",SumCarry[0][i]));
        }

        log.info(String.format("%n%nMAS_S  %n" ));
        for( i=0;i<138;i++) log.info(String.format("-"));
        log.info(String.format("%n"));
        for( i=0;i<32;i++)
        {
```

```
            log.info(String.format("#%2d: ",i));
            for( j=0;j<32;j++)
            {
               log.info(String.format("%3d ",Hs[i][j]));
            }
            log.info(String.format("%n"));
         }

         log.info(String.format("%n%nSUMA: %n" ));
         for( i=0;i<138;i++) log.info(String.format("-"));
         log.info(String.format("%n"));
         for( i=0;i<32;i++)
         {
            log.info(String.format("#%2d: ",i));
            for( j=0;j<32;j++)
            {
               Hs[i][j] += Ht[i][j];
               Hs[i][j] = Hs[i][j]/mod + Hs[i][j]%mod;
               log.info(String.format("%3d ", Hs[i][j]));
            }
            log.info(String.format("%n"));
         }

         for( i=0;i<138;i++) log.info(String.format("-"));
         log.info(String.format("%nRES: "));
         for( i=0;i<32;i++)
         {
            for( j=0;j<32;j++)
            {
               SumCarry[1][i] += Hs[j][i];
               SumCarry[1][i] = SumCarry[1][i]/mod + SumCarry[1][i]%mod;
            }
         log.info(String.format("%3d ", SumCarry[1][i]));
         }

//============ OUTPUT RESULT Hash bin/hex ==============
         log.info(String.format("%n%n" ));
         String str = "";
         for(j=0;j<32;j++) {
            str += (String.format("%8s", Long.toBinaryString(SumCarry[1][j])).replace(' ', '0') + " ");
         }
         log.info(String.format("HASH binary:%n%n%s%n", str));
         str = "";
         for(j=0;j<32;j++) {
            str += (String.format("%2s", Long.toHexString(SumCarry[1][j])).replace(' ', '0') + " ");
         }
         log.info(String.format("%nHASH hex:%n%n%s%n", str));
      }
   }
```

ДОДАТОК В. Список опублікованих праць за темою дисертації

1. Кисюк Д. В. Аналітичний огляд постачальників хмарних послуг. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025, №3, С. 105–111. DOI: https://doi.org/10.31891/2219-9365-2025-83-14
2. Кисюк Д. В. Актуальність проблеми впровадження хмарних сервісів для організації роботи з даними. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2025, Т.356, № 5. DOI: https://doi.org/10.31891/2307-5732-2025-357-24
3. Кисюк Д. В., Захарченко С. М. Оптимізація застосування хмарних сервісів для розподіленої обробки даних в системах колективного доступу. *Наукові праці ВНТУ*. 2025, Вип. №2, С. 66–72. DOI: https://doi.org/10.31649/2307-5376-2025-2-66-72
4. Лужецький В. А., Кисюк Д. В. Новий підхід до побудови байт-орієнтованих криптографічних геш-функцій. *Вимірювальна техніка та метрологія*. 2026, Випуск 87(1), ISTCMTM, Львів, Номер 1, С. 51-58. DOI: https://doi.org/10.23939/istcmtm2026.01.051
5. Лужецький В. А., Захарченко С. М., Кисюк Д. В. Метод байт-орієнтованого гешування з підвищеним рівнем дифузії та спеціалізований МН-процесор для приватних хмарних інфраструктур. *Вимірювальна та обчислювальна техніка в технологічних процесах.* 2026, № 85(1), С. 134-141. DOI: https://doi.org/10.31891/2219-9365-2026-85-16
6. Лужецький В. А., Савицька Л. А., Кисюк Д. В. Спеціалізований процесор для ущільнення даних. *Тези доповідей П'ятої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації»*. 2016, Вінниця : ВНТУ, С.108–110. URL: http://ir.lib.vntu.edu.ua/handle/123456789/11399
7. Лужецький В. А., Кисюк Д. В., Савицька Л. А. Метод байт-орієнтованого хешування. *Тези доповідей V Міжнародної науково-практичної конференції*

«Методи та засоби кодування, захисту й ущільнення інформації». 2016, Вінниця–Немирів. URL: http://ir.lib.vntu.edu.ua/handle/123456789/9992

8. Лужецький В. А., Кисюк Д. В. Новий підхід до побудови криптографічних хеш-функцій. *Матеріали статей П'ятої Міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія»*. 2015, Івано-Франківськ : ФОП Голіней О. М., С.149–150. URL: http://ir.lib.vntu.edu.ua/handle/123456789/14457
9. Кисюк Д. В., Захарченко С. М. Аналіз переваг використання хмарних сервісів для обробки даних у сучасних системах управління базами даних. *Матеріали LII Науково-технічної конференції підрозділів ВНТУ.* 2023, Вінниця. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-023/paper/view/18183.
10. Кисюк Д. В., Лужецький В. А. Програмний засіб для побудови хеш-значення за допомогою байт-орієнтованої обробки даних для мобільних пристроїв. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)»*. 2023, Вінниця. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/18180
11. Мазур Б. С., Кисюк Д. В. Хмарні сервіси. Обробка та збереження даних за допомогою хмарних сховищ. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)»*. 2024, Вінниця. URL: https://iq.vntu.edu.ua/method/getfile.php?fname=153063.docx&x=1
12. Кисюк Д. В., Шулдик С. С. Хмарні сервіси як інструмент управління проєктами у системах забезпечення ресурсами підприємства SAP ERP. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»*. 2025, Вінниця. URL: https://iq.vntu.edu.ua/method/getfile.php?fname=179474.pdf&x=1
13. Лужецький В. А., Кисюк Д. В. Узагальнений метод хешування байтової форми представлення інформації. *Тези доповідей Четвертої Міжнародної*

науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія». 2014, Вінниця : ВНТУ, С. 184–186. URL: http://ir.lib.vntu.edu.ua/handle/123456789/14878.

14. Лужецький В. А., Кисюк Д. В. Метод хешування даних на основі їх характеристичних ознак. *Тези доповідей II Міжнародної науково-практичної конференції «Актуальні питання забезпечення кібербезпеки та захисту інформації»*. 2014, Київ : Європейський університет, С. 100–102. URL: http://ir.lib.vntu.edu.ua/handle/123456789/15351.
15. Лужецький В. А., Кисюк Д. В., Савицька Л. А. Метод байт-орієнтованого хешування. *Тези доповідей П'ятої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації»*. 2016, Вінниця : ВНТУ, С. 37–38. URL: http://ir.lib.vntu.edu.ua/handle/123456789/9992.
16. Лужецький В. А., Кисюк Д. В., Комаров А. О. Метод та спеціалізований процесор для байт-орієнтованого хешування даних. *Тези доповідей Міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерне моделювання (ІТСМ)»*. 2016, Івано-Франківськ : ПНУ ім. В. Стефаника, С. 123–125. URL: http://itcm.pnu.edu.ua/2016/docs/Luzhetskyi.pdf.
17. Кисюк Д. В., Заруцький М. В. Перспективи впровадження штучного інтелекту в хмарних обчисленнях. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»*. 2025, Вінниця, URL: https://iq.vntu.edu.ua/method/getfile.php?fname=169386.pdf&x=1.
18. Кисюк Д. В., Червінський В. О. Архітектура приватних хмарних сервісів для зберігання та обробки даних. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»*. 2025, Вінниця, URL: https://iq.vntu.edu.ua/method/getfile.php?fname=169386.pdf&x=1.
19. Кисюк Д. В., Сірак В. О. Методи підвищення ефективності зберігання та обробки великих обсягів даних у приватних хмарних сервісах. *Матеріали*

Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)». 2026, Вінниця, URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/viewFile/27079/22199.