1 АНАЛІЗ ЗАГАЛЬНОВІДОМИХ ПІДХОДІВ ПОБУДОВИ ГЕШ-ФУНКЦІЙ

1.1 Відомі підходи до побудови геш-функцій

Геш-функція – це функція, яка приймає довільний вхідний рядок будь-якої довжини і перетворює його в обмежений за обсягом вихідний рядок фіксованої довжини, який зазвичай називають хеш-значенням. Геш-функції використовуються в багатьох галузях, включаючи шифрування, пошук інформації та бази даних [1].

Значення геш-функції генерується функцією *H* виду:

,

де: *M* – повідомлення фіксованої довжини,

*Н(M)* – значення геш-функції будь-якої довжини.

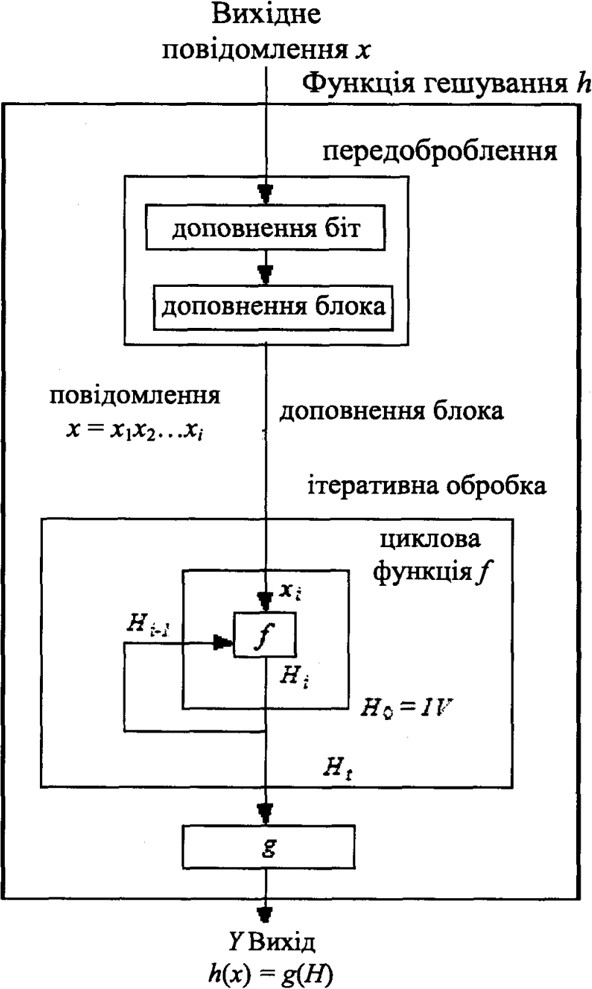
Геш-функція *H* повинна мати наступні властивості:

* На виході функції повинно бути значення фіксованої довжини;
* Застосовується до блоків даних будь-якої довжини;
* Значення функції *H(x)* обчислюється відносно легко і застосовуваний алгоритм розрахунку повинен бути практичним;
* Геш-функція повинна бути односторонньою. У цьому випадку не можна обчислити *x*, де *H(x) = h*;

Загальний принцип побудови геш-функції, що на рис. 1.1 полягає в тому, щоб приймати вхідні дані будь-якої довжини та повертати геш-значення фіксованої довжини. Геш-функція призначена для забезпечення надійного зберігання інформації, перевірки цілісності даних і шифрування [2].

Вхідні дані (повідомлення, файл) розглядаються як послідовні *m* бітові блоки типу 𝑀0, 𝑀1, …. , 𝑀𝐿−1. Усі значення отриманні на виході геш-функції перед цим обробляються послідовно блок за блоком. Проміжні та остаточне значення геш-функції мають бітове представлення.

Однією з основних вимог до геш-функції є те, що вихідні дані не можуть бути відновлені за геш-значенням. Це забезпечує криптографічну стійкість та випадковість геш-функції. Крім того, геш-функція повинна мати високу швидкість обчислення, щоб бути ефективною у різних галузях застосування. На рис. 1.2 наведено ітераційну конструкцію Меркля-Дамгарда [2].

Рисунок 1.1 – Загальний принцип побудови геш-функції

При створенні якісних геш-функцій дуже важливо враховувати розподіл ключів цієї функції. Вибір геш-функції має здійснюватися таким чином, щоб вона не корелювала з закономірностями, властивими існуючим даним. Іноді до геш-функцій можуть висуватися більш суворі вимоги, ніж просто забезпечення рівномірного гешування.

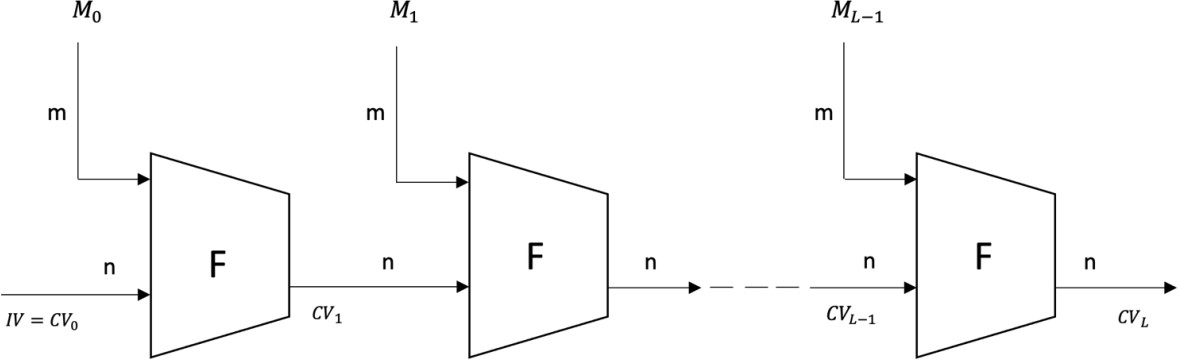


Рисунок 1.2 – Ітераційна конструкція Меркля-Дамгарда

Ітеративність перетворень, що реалізуються конструкцією Меркля-Дамгарда (1.1), стала основною причиною появи атак з передобчисленнями [3].

Існують такі загальновідомі підходи до побудови геш-функцій:

* На основі будь-якої складної математичної задачі;
* на основі алгоритмів блокового шифрування;
* розроблені «з нуля».

Кожен з перелічених методів має свої переваги та недоліки, проте найбільш популярними сьогодні є останні два. Це пов'язано з тим, що при розробці геш-функцій з нуля виникає можливість врахувати їхню властивість ефективної програмної реалізації. Широке використання геш-функцій, побудованих на основі алгоритмів блокового шифрування, є результатом ретельного дослідження стійкості багатьох з існуючих алгоритмів [2].

Геш-функції, засновані на складних математичних задачах, можуть базуватися на швидкому перетворенні Фур’є,.

1.2 Геш функцїі на основі складних математичних задач

Серед типових представників методів гешування теоретично доведеної стійкості є методи, що базуються на швидкому перетворенні Фур’є [4, 5]. Одним з таких методів гешування є метод, запропонований у роботі [4]. Цей метод передбачає представлення вхідного повідомлення у вигляді матриці розмірністю (*k* – секретний параметр), а кожний елемент матриці є невід’ємним цілим числом, що не перевищує 4 або 8 [4].

Вхідні дані оброблюються шляхом множення кожного стовпця на для відповідно обраної константи w, а також застосовування швидкого перетворення Фур’є до кожного рядка [4, 5]:

*,*

де – швидке перетворення Фур’є.

Обчислюється випадкове лінійне об’єднання кожного стовпця:

.

Вихідними даними є вектор (). Друга операція ущільнює вхідні дані та досягає «заплутування» [4, 5].

Всі арифметичні операції у формулі виконуються за модулем малого простого числа *p* (або добутку простих чисел) типу . Недоліком цього методу є його низька швидкість (щонайменше в 15 раз повільніше за сучасні геш-функції) та стійкість менша за стійкість ідеальної геш-функції [4, 5]. Крім того, така геш-функція незручна з практичної точки зору, оскільки різні режими її роботи дають вихідні геш-значення довжиною, що не парна 8 бітам.

Найпоширенішими задачами теоретично доведеної складності є задачі дискретного логарифмування в полі простого числа. Прикладом геш-функцій такого типу є спосіб [6]: та пропозиції Діффі-Хеллмана щодо гешування. Ці задачі формалізуються таким порівнянням [7]:

*,*

де – просте число;

– деякі натуральні числа.

Для того, щоб задача зламу такого криптографічного перетворення зводилась до задачі пошуку дискретного логарифму в полі простого числа, необхідно, щоб a було примітивним елементом за модулем числа . В такому випадку при зміні *x* від 1 до (де – значення функції Ейлера для числа ), число також прийматиме значення від 1 до і ці значення не повторюватимуться.

Складність пошуку дискретного логарифму становить викликів функції ущільнення. Якщо використовувати цю задачу для побудови функцій ущільнення в гешуванні, то оцінка зламу такої геш-функції за умови, що до , становитиме або ще менше, якщо використовувати деякі евристичні алгоритми, зокрема алгоритми Адлемана, Широкауера та інших. Оцінки реалізації цих алгоритмів значно менші за , однак вони теоретично гарантовані, а отже, для низки задач, де дані періодично оновлюються, таких оцінок достатньо для забезпечення цілісності та автентичності даних в комп’ютерній системі.

Метод гешування, запропонований у роботі [6], передбачає розбиття повідомлення на блоки рівної довжини та піднесення їх до степеня за модулем простого числа:

,

де– частина ключа;

– просте число.

Недоліком цього методу є те, що не всі блоки даних є примітивними елементами за модулем , а тому стійкість гешування до колізій може істотно варіюватися залежно від конкретного значення блоку даних .

Окрім того, всі методи, що зводяться до задач теоретично доведеної складності, вимагають значного часу порівняно з іншими підходами до проектування методів гешування через використання операцій, що можуть бути неефективними для сучасних мікропроцесорів, наприклад, піднесення до степеня, не забезпечуючи при цьому високу якість отриманих геш-функцій. [8] Через це на практиці такий підхід до побудови геш-функцій широко не використовується.

Проте ці методи викликають інтерес через можливість гарантування певного рівня стійкості та їх потенційне використання для розв'язання задач гешування, де відсутні обмеження на час гешування або обробляються невеликі повідомлення, зокрема у завданнях обчислення MAC для архівних даних.

1.3 Геш функції на основі алгоритмів блокового шифрування

Побудова геш-функцій на основі блокових шифрів - це метод, який комбінує блокові шифри та геш-функції для забезпечення цілісності та захисту від підробки даних. У цьому методі блокові шифри перетворюють вхідні блоки даних в блоки фіксованої довжини, які називаються геш-значеннями.

Зазвичай цей метод включає фазу гешування та фазу побудови коду аутентифікації повідомлення. Під час фази гешування вхідне повідомлення зберігається у буфері та піддається гешуванню за допомогою обраної геш-функції. Після цього, після додавання ключа та можливої додаткової інформації, геш розбивається на блоки та проходить через фазу, в якій застосовується блоковий шифр для створення коду аутентифікації повідомлення.

Ще один підхід - це побудова геш-функції на основі режиму шифрування відкритого тексту ECB. Повідомлення розбивається на блоки, які шифруються за допомогою блокового шифру, а потім об'єднуються та піддаються гешуванню за допомогою обраної геш-функції [9].

Загалом, побудова геш-функцій на основі блокових шифрів є досить складним процесом, оскільки потрібно враховувати різні атаки та забезпечити відповідний рівень безпеки. Тому перед використанням таких геш-функцій варто детально вивчити метод та провести відповідне тестування на стійкість до атак.

Загальні схеми ітерації геш-функцій на основі блокового шифрування наведено на рис. 1.3-1.4.

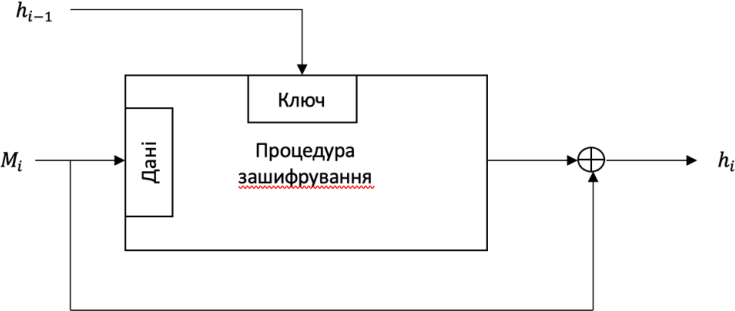


Рисунок 1.3 – Схема 1 ітерації безпечної геш-функції

ℎ𝑖 = 𝐸ℎ𝑖−1 (𝑀𝑖)⨁𝑀𝑖

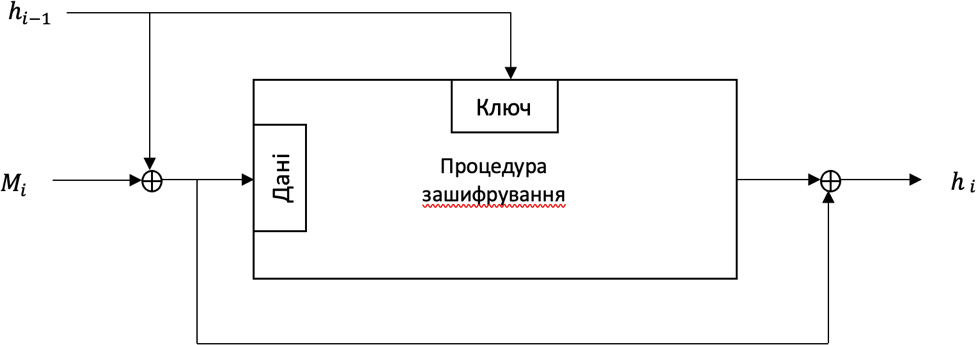


Рисунок 1.4 – Схема 2 ітерації безпечної геш-функції

ℎ𝑖 = 𝐸ℎ𝑖−1 (𝑀𝑖⨁ℎ𝑖−1)⨁𝑀𝑖⨁ℎ𝑖−1

З практичної точки зору, побудова геш-функції на основі використання n-розрядного блокового шифру може бути ефективною як програмна, так і апаратна реалізація блокового шифру, що використовується в системі захисту. Це може забезпечити велику функціональність криптоалгоритмів і зниження витрат на їх розробку.

ISO/IEC 10118-2 - це міжнародний стандарт, що визначає методи гешування повідомлень з використанням криптографічних хеш-функцій. Він встановлює три рівні безпеки, що відповідають різним розмірам вихідних значень геш-функцій.

ISO/IEC 10118-2 включає в себе такі геш-функції, як MD2, MD4, MD5, SHA-1, SHA-224, SHA-256, SHA-384 та SHA-512. Кожна з цих функцій має свої особливості, наприклад, розмір вихідного значення та спосіб обчислення [10].

Стандарт ISO/IEC 10118-2 включає в себе вимоги до безпеки геш-функцій, такі як опірність до колізій та здатність захищатись від атак. Крім того, він надає вказівки щодо використання геш-функцій у різних криптографічних протоколах, таких як SSL, TLS, IPSec та інші.

Один з найважливіших аспектів стандарту полягає в рекомендації використовувати ключі з великою довжиною в бітовому виразі. Це сприяє високому рівню безпеки при використанні геш-функцій для захисту від колізій та атак.

Узагальнюючи, стандарт ISO/IEC 10118-2 є важливим інструментом для забезпечення безпеки в криптографії та захисту від різних видів атак [10].

Циклова функція є блоковим шифром і характеризується такими основними параметрами:

* *n* - довжина вхідного блоку даних;
* *k* - довжина ключа;
* *𝑚* ≤ *𝑛* - довжина вихідного блоку даних.

У більшості випадків ключ (*k*) має таку саму довжину, що і блок (*n*), який обробляється, тобто *n* розрядів. Але в деяких випадках функції гешування можуть використовувати ключі більшої довжини.

Також з параметрами *n*, *k*, *m* обчислюється характеристика швидкості. Кількість операцій при блоковому шифруванні, яку необхідно виконати для формування геш-значення довжиною, що дорівнює довжині блоку шифрування.

Функції гешування однократної довжини базуються на схемах Матіаса Майєра Озіса та Девіса Майєра. Суть обчислень у цих схемах полягає в розбитті рядка даних *Х*, який гешується, на n-розрядні блоки 𝑋1, 𝑋2, … , 𝑋𝑡. При необхідності останній блок може бути доповнений до потрібної кількості символів [11].

Циклічна функція Матіаса Майєра Озіса визначається аналітичним співвідношенням:

Довжина блоку баних дорівнює довжині блоку шифру. Також, довжина геш-значення дорівнює довжині блоку, що використовується у блоковому шифруванні, тобто . Функція усікування вибирає найменші *k* бітів з хеш-значення рядка *Ht*.

Стійкість цих функцій базується на стійкості блокового шифру *E*, який використовується в їхній основі. Також важливо, щоб для цих алгоритмів не було можливості використати атаки, що базуються на специфічних властивостях або особливостях алгоритмів шифрування.

Функції гешування однократної довжини використовуються для побудови однобічних функцій гешування на основі блокових шифрів з довжиною блоку *n* = 64 біта або побудови вільних від виникнення колізій функцій гешування на основі блокових шифрів з довжиною блоку n = 128 бітів.

Функції гешування з подвійною довжиною стандарту ISO/IEC 10118-2 визначає алгоритм MDC-2 як функцію гешування подвійної довжини. У MDC-2 на кожному кроці ітерації паралельно обчислюється відразу два блокових шифрування і таким чином, швидкість гешування дорівнює ½ [11].

Функція гешування подвійної довжини визначається аналітичним співвідношенням:

* 1. **Геш функції розроблені «з нуля»**

Методи цього типу значно відрізняються у механізмах перетворень, які вони використовують, порівняно з методами інших типів. Це пояснюється тим, що вони базуються на ідеї незалежних розв'язків для вибору послідовностей перетворень, які використовуються для вирішення задачі. Ця особливість призводить до широкого спектру можливих підходів.

Прикладом такого гешування є стандарт SHA-2 [12], що передбачає для кожного блоку повідомлення ітеративні перетворення, які полягають у тому, що блок розбивається на 16 32-розрядних слів та розширюється до 64 32-розрядних слів за таким правилом. Потім ініціалізуються вісім 32-розрядних робочих змінних ,,,,,,,.

Виконуються раундові перетворення для всіх [8, 12]:

;

;

де та – проміжні змінні;

та – результат перетворення вхідного аргументу для гешування [12].

Наступні проміжні геш-значення отримуються шляхом додавання до попереднього проміжного геш-значення отриманих значень робочих змінних *A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*.

Основними недоліками методів гешування SHA-2 є їх вразливість до загальних атак на основі мультиколізій таких як атака Нострадамуса та атака Жу. Це обумовлюється тим, що вони побудовані на основі конструкції Меркля-Дамгаарда [13].

Сімейство геш-функцій SHA-2 є найбільш стійким представником серед геш-функцій, що використовують підхід до проектування Рівеста. Представниками таких геш-функцій є MD5, RIPE-MD, SHA-1 [8]. Однак функції ущільнення в останніх виявились вразливими до спеціальних атак [14].

Один з підходів до побудови «з нуля» функцій гешування полягає у використанні керованих криптографічних примітивів, тобто примітивів, параметри перетворень яких залежать від вектора керування [15, 16, 17, 18]. Використовуючи SP-мережі як основні перетворення, пропонується використовувати вектор керування для вибору типу підстановки та перестановки, що виконуватиметься в кожному випадку. Приклад криптографічних примітивів для класу F2/1 (2 вхідних/вихідних біти, 1 керуючий) в алгебраїчній нормальній формі наведено у табл. 1.1.

Таблиця 1.1 – Приклад керованих криптографічних примітивів з класу F2/1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Примітив | № | Примітив | № | Примітив |
| 1 |  | 5 |  | 9 |  |
| 2 |  | 6 |  | 10 |  |
| 3 |  | 7 |  | 11 |  |
| 4 |  | 8 |  | 12 |  |

Кількість нелінійних примітивів можна подвоїти, додавши до арифметичної нормальної форми примітивів, наведених у табл. 1.1, одиницю (тобто інвертувавши дані примітиви). Попарно комбінуючи такі примітиви можна визначити найкращі комбінації відповідно критеріїв збалансованості та нелінійності вихідних значень, отримуючи SP-мережу класу F2/1. Аналогічно отримують SP-мережі класів F2/2 та F3/1.

У роботах [15, 16], крім керованих перестановок, керування пропонується виконувати за допомогою операції інвертування, а також виконанням прямого чи оберненого перетворень, використанням складних чи простих функцій. Саме тому, особливістю роботи [15, 16] є те, що вектор керування може приймати більше значень, ніж кількість комбінацій вихідних бітів. Для задач безключового гешування така «надмірність» вектора керування може стати причиною колізій, оскільки декілька різних наборів векторів керування призводять до однакових вихідних значень при однакових вхідних даних.

Недоліком підходу керованої SP-мережі є висока складність реалізації програмного забезпечення і, отже, низька швидкість гешування. Це пов'язано з тим, що мережа SP в першу чергу орієнтована виключно на апаратну реалізацію [15, 16]. Реалізація перестановки 2 або 3 суміжних бітів в програмному забезпеченні з використанням сучасної обчислювальної платформи комп'ютерної системи створює проблеми, пов'язані з резервним копіюванням частини блоку даних, застосуванням маски і прив'язкою результату до 1 операнду. Спроба перетасувати біти по всьому блоку даних викликає подібну проблему, але в апаратних додатках це робиться лише шляхом розгалуження провідників.

У роботі [17] розглянуто підходи до розробки методів гешування на основі логічних функцій, що використовуються у геш-функції SHA-2. Але на відміну від [12] у роботі [17] пропонувалось використовувати додатковий криптографічний примітив – зсув, що керується даними. Недоліком такого методу є те, що він не враховує властивість простого зсуву «втрачати» дані, які зсуваються та надання можливості зловмиснику впливати на керований параметр шляхом нав’язування певних блоків даних, що було використано у роботі [18] для зламу цих геш-функцій.

Наразі відомі різні методи ущільнення для гешування, засновані на задачах з теоретично доведеною складністю, а також перетворення, що базуються на відомих шифрах. Хоча ці методи забезпечують високий рівень стійкості, вони не дозволяють досягти високої швидкості гешування на сучасних універсальних мікропроцесорах через їхню неприродність.

З іншого боку, перетворення, розроблені "з нуля", можуть бути швидшими у виконанні, але не гарантують такої ж високої стійкості. Тому найбільш перспективним підходом для гешування є комбінація методів з теоретично доведеною складністю, що забезпечують гарантований рівень стійкості, та перетворень, розроблених "з нуля", які дозволяють досягти високої швидкості й гнучкості при адаптації до різних конструкцій.

**ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Joux A. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. Advances in Cryptology *–* CRYPTO 2004. Berlin, Heidelberg, 2004. С. 306–316. URL: https://doi.org/10.1007/978-3-540-28628-8\_19 (дата звернення: 07.05.2024).
2. В. А. Лужецький, Ю. В. Баришев. Конструкції гешування стійкі до мультиколізій. Наукові праці Вісник Вінницького політехнічного інституту. 2010. № 1. 8 с. URL: https://praci.vntu.edu.ua/index.php/praci/article/view/191. (дата звернення: 09.05.2024)
3. Хешування і захист інформації. URL: http://isearch.kiev.ua/index.php/uk/searchpractice/internetsecurity/837-hashing-message-digest (дата звернення: 13.05.2024).
4. Lyubashevsky V., Micciancio D., Peikert C., Rosen A. Provably Secure FFT Hashing. 2006. 12 с. URL: http://www.eecs.harvard.edu/~alon/PAPERS/lattices/description.pdf. (дата звернення: 14.05.2024)
5. Lyubashevsky V. Towards Practical Lattice-Based Cryptography: A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy. 2008. 75 с. URL: http://www.eecs.harvard.edu/~alon/PAPERS/lattices/description.pdf. (дата звернення: 14.05.2024)
6. Патент України на корисну модель № 55211 МПК H 04 L 9/06. Конвеєрний криптографічний обчислювач. / Корченко О. Г., Паціра Є. В., Панасюк А. Л., Гнатюк С. О., Кінзерявий В. М.; заявник та патентовласник Національний авіаційний університет. – № u201006041; заявл. 19.05.10; опубл. 10.12.10, Бюл. №23.
7. Кузнецов О. О., Євсєєв С. П., Король О. Г. Захист інформації в інформаційних системах. Методи традиційної криптографії : навчальний посібник. Харків : Вид. ХНЕУ, 2010. 316 с.
8. Schneier B. Applied Cryptography: Protocols, Algorithms and Source Code in C. Wiley & Sons, Incorporated, John, 2015. 784 с.
9. Горбенко Ю. І., Горбенко І. Д. Інфраструктури відкритих ключів. Харків : УДК, 2010. 608 с.
10. Лужецький В. А. Метод паралельного гешування. Вінниця : Вінницький національний технічний університет, 2023. 8 с.
11. Основи компʼютерних алгоритмів : [збірник матеріалів] (м. Київ, 7 червня 2021 р.) / Київський національний університет імені Тараса Шевченка, факультет компʼютерних наук та кібернетики. Київ : Компʼютерні науки, 2021.
12. U.S. Department of Commerce, National Institute of Standards and Technology. Secure Hash Standard: Federal Information Processing Standards Publication 180-4. CreateSpace Independent Publishing Platform, 2012. 36 с.
13. Klima V., Gligoroski D. Generic collision attacks on narrow-pipe hash functions faster than birthday paradox, applicable to MDx, SHA-1, SHA-2, and SHA-3 narrow-pipe candidates // Cryptology ePrint Archive. 2010. 4 с. URL: http://eprint.iacr.org/2010/430.pdf. (дата звернення: 16.05.2024)
14. McDonald C., Hawkes P., Pieprzyk J. Differential Path for SHA-1 with complexity O(252) // Cryptology ePrint Archive. 2009. 14 с. URL: http://eprint.iacr.org/2009/259.pdf. (дата звернення: 17.05.2024)
15. Рудницький В. М. Модель уніфікованого пристрою криптографічного перетворення інформації / В. М. Рудницький, В. Г. Бабенко // Системи обробки інформації. – №3. – 2009. – С. 91-95..
16. Рудницький В. М., Бабенко В. Г. Синтез математичних моделей пристроїв декодування інформації для криптографічних систем // Системи обробки інформації.. – № 2. – 2010 – С. 124–128.
17. Xu Z. Dynamic SHA // Cryptology ePrint Archive. 2007. 34 с. URL: http://eprint.iacr.org/2007/476.pdf. (дата звернення: 19.05.2024)
18. Cryptanalysis of Dynamic SHA(2) / J.-P. Aumasson та ін. *Selected Areas in Cryptography*. Berlin, Heidelberg, 2009. С. 415–432. URL: https://doi.org/10.1007/978-3-642-05445-7\_26 (дата звернення: 21.05.2024).