Розділ 5

ФУНКЦІЇ ГЕШУВАННЯ. СУТНІСТЬ ПЕРЕТВОРЕНЬ, ЗАСТОСУВАННЯ ТА НАПРЯМИ РОЗВИТКУ

Як уже зазначалося в п. 2.1 розділу 2 цієї книги, функції гешування є примітивами, що використовуються в різних криптографічних і некриптографічних додатках. Але практика сьогодення підтвердила, що найбільш важливі застосування функції гешування пов'язані з електронним цифровим підписом і криптографічними протоколами, перш за все щодо автентифікації та встановлення ключів. Цим питанням міжнародні організації стандартизації ISO та IEC приділяють дуже велику увагу, розроблено та застосовується стандарт гешування ISO/IEC 10118-1, 2, 3, 4 [190–193]. Опис складових цього стандарту наведений у 2.1. У цьому розділі розглядаються основні проблемні питання застосування, аналізуються вимоги, наводиться короткий огляд функцій гешування в контексті їх історичної появи та застосування, а також розглядаються й аналізуються нові вимоги та шляхи їх виконання в перспективі.

Зважаючи на особливу актуальність і роль функції гешування в криптографічних додатках, необхідність покращення їх властивостей, у 2007 році Національним інститутом стандартизації США (NIST) розпочато відкритий конкурс на проект нового федерального стандарту гешування SHA-3, що отримав назву «NIST SHA-3 Competition» [196, 189]. Така значна необхідність розробки проекту нового стандарту гешування перш за все обумовлена появою повідомлень про успішну, але все-таки теоретичну, побудову атак на такі функції гешування, як SHA-1 [199, 195] та ГОСТ Р 34.11-94 (ГОСТ 34.311-95) [195], що широко застосовуються на практиці. Функції гешування є примітивами, що використовуються здебільшого в різних криптографічних додатках. Найбільш важливі застосування відносяться до автентифікації та в схемах електронного цифрового підпису. Можна виділити три основних підходи до побудови функцій гешування:

1) функції гешування, що побудовані з використанням блокових шифрів [191];

2) функції гешування, що засновані на арифметиці обчислень за модулем [193];

3) замовлені функції гешування [192].

Міжнародна організація із стандартизації ISO/IEC розробила стандарт для опису різних класів функцій гешування ISO/IEC 10118 [190], який нині переглядається з метою включення до нього функцій гешування, таких як SHA-2 федерального стандарту США FIPS 180-2.

У частині ISO/IEC 10118-1 подано загальні визначення, вимоги та схеми функцій гешування.

У частині ISO/IEC 10118-2 визначені функції гешування, що засновані на блокових шифрах у конструкції Matyas-Meyer-Oseas [191, 201]. У цій конструкції використовується незалежний блоковий шифр у алгоритмі MDC-2 із двома й більше функціями, який дозволяє обчислити геш-значення подвійної та потрійної довжини відповідно.

Частина ISO/IEC 10118-3 визначає три замовлених алгоритми: RIPEMD-128, RIPEMD-160 і SHA-1. Ця частина стандарту на цей час уже переглянута. До неї уже включені такі нові функції гешування, як SHA-2/256, SHA-2/384, SHA-2/512, що пройшли широкі дослідження в NIST США та включені до федерального стандарту США FIPS 180-2 [198].

Частина ISO/IEC 10118-4 описує MASH-1 і MASH-2 функцій гешування, що використовують арифметику модульних перетворень.

Необхідно відзначити, що нині широко застосовуються однонаправлені й безколізійні функції гешування та функції вироблення КАП (кодів автентифікації повідомлень). До них необхідно віднести такі, як ГОСТ 34.311-95, HAVAL, SHA-1, RIPEMD-160, MD4, MD5, ГОСТ 28147-89 режим 4, Rijndael CBC–MAC, Whirlpool, SHA-2, та UMAC [189–204]. По суті, їм на зміну приходять такі функції гешування, як SHA-2/256, SHA-2/384, SHA-2/512 [198, 201]. Також необхідно відзначити успішність виконання проекту «NIST SHA-3 Competition» [201–204]. Усе вищезазначене враховується в цьому розділі.

5.1. ВИЗНАЧЕННЯ ТА ВИМОГИ ДО ФУНКЦІЙ ГЕШУВАННЯ

Конструктивними елементами електронних цифрових підписів і кодів автентифікації повідомлень є функції гешування, функції стиснення та ітераційні функції гешування. Визначення названих функцій гешування та їх криптографічних властивостей розглянемо у трактуванні Блека та Рогавея [190–191, 201].

Визначення 1. ***Функцією гешування*** називається функція відображення $h: D \to R$, де область значень $D = \{0,1\}^*$, а $R = \{0,1\}n$ для деякого $n \geq 1$.

Визначення 2. ***Функцією стиснення*** називається функція відображення $f: D \to R$, де $D = \{0,1\}a \times \{0,1\}b$ та $R = \{0,1\}n$ для деяких $a, b, n \geq 1$ та $a + b \geq n$.

Визначення 3. ***Ітераційною функцією гешування*** від функції стиснення $f: (\{0,1\}n \times \{0,1\}b) \to \{0,1\}n$ є функція гешування $h: (\{0,1\}b)^* \to \{0,1\}n$, що визначена як $h(X_1 \ldots X_t) = H_t$, де $H_i = f(H_{i-1}, X_i)$ при $1 \leq i \leq t$ $(H_0 = \text{IV})$.

Більшість функцій гешування – це ітеративні конструкції, що використовують функцію стиснення f, роблять попередню розбивку даних X на

підблоки X_i та виконують зв'язування по зворотному входу проміжних результатів обчислення геш-значень.

Стандартна модель ітеративної функції гешування визначена в ISO/IEC 10118-1. На рис. 5.1 наведена загальна модель ітеративної функції гешування.

Узагальнена модель ітеративної функції гешування для t підблоків визначається таким алгоритмом ітеративних обчислень:

$$H_0 = \text{IV},$$
$$H_i = f(H_{i-1}, X_i),\ 1 \le i \le t, \tag{5.1}$$
$$h(K, X) = g(H_t),$$

де цифра IV позначає початкове значення, що може бути визначене як деяка константа в описі функції гешування, а g є вихідною результуючою функцією перетворення.

Стандарт ISO/IEC 10118-1 не визначає методи доповнення рядка даних, але передбачається, що можуть бути використані методи, визначені в ISO/IEC 9797.

Визначальними вимогами до функцій гешування є їхня стійкість до обчислення прообразу, другого прообразу, а також стійкість до колізій [201, 190, 7, 9, 10].

Стійкість функції гешування до обчислення прообразу визначає її як однобічну в тому сенсі, що за даного значення функції гешування h практично неможливо визначити повідомлення, для якого воно було обчислене.

*Визначення 4. **Стійкість до обчислення прообразу*** [7, 9, 10, 190, 201]. Функція гешування h:{0,1}* → R є стійкою до обчислення прообразу (t, ε, I_h), якщо не існує ймовірнісного алгоритму I_h, із вхідними значеннями $X \in R$ і значеннями на виході $Y \in$ {0,1}*, часом виконання менше (не більше) ніж t, де $h(X) = Y$, та ймовірністю не менше ніж оцінена при випадковому виборі Y та I_h.

Ця властивість функції гешування має важливе значення для систем автентифікації, що ґрунтуються на використанні геш-значення паролів і таємних ключів.

*Визначення 5. **Стійкість до обчислення другого прообразу*** [7, 9, 10, 190, 201]. Нехай S – кінцева підмножина {0,1}*. Функція гешування h:{0,1}*→ R є стійкою до обчислення другого прообразу із захищеністю (t, ε, S), якщо не існує ймовірнісного алгоритму S_h, з $X \in R$ та $X' \in$ {0,1}*, з часом виконання менше ніж t, де $X' \neq X$ та $h(X') = h(X)$ і ймовірністю, не меншою ніж оцінена при випадковому виборі X і S_h.

Стійкість функцій гешування до обчислення другого прообразу визначає безпеку систем автентифікації.

*Визначення 6. **Стійкість до колізій*** [7, 9, 51, 190, 201]. Функція гешування h: {0,1}* → R є стійкою до колізій із захищеністю (t, ε, R), якщо не існує ймовірнісного алгоритму C_h з відомими вхідними значеннями X, $X' \in$ {0,1}*, часом виконання менше ніж t, де $X' \neq X$ і $h(X) = h(X')$, і ймовірністю, не меншою ніж оцінена при випадковому виборі C_h.

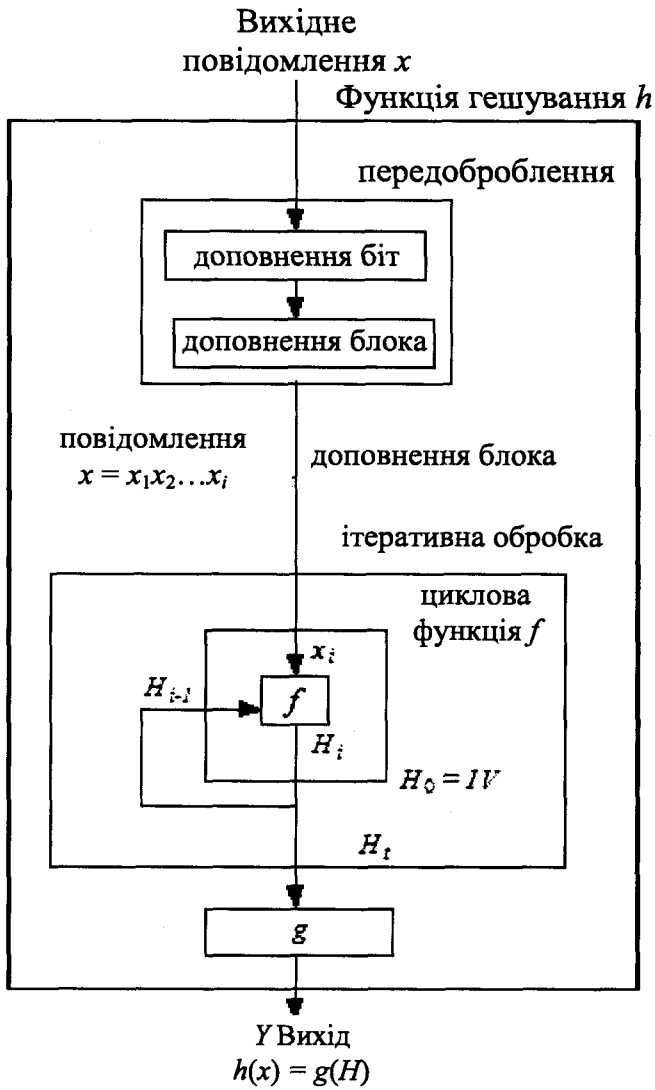


Рис. 5.1. Ітеративна модель функції гешування

Як уже зазначалося в розділі 3, стійкість функцій гешування до колізій визначає безпеку систем автентифікації з цифровим підписом.

Визначення 7. ***Функція гешування*** називається ***простою,*** або ***слабкою (однонаправленою)*** [7, 10, 51, 190, 201], якщо вона є стійкою тільки до обчислення прообразу та до обчислення другого прообразу.

Визначення 8. ***Функція гешування*** називається ***сильною,*** або ***стійкою до колізій*** [174], якщо вона є стійкою до обчислення прообразу, стійкою до обчислення другого прообразу та стійкою до колізій.

Визначення сильної функції гешування показує, що для неї обчислювано неможливо знайти яку-небудь колізію; вона також забезпечує захист атак, що базуються на парадоксі про «день народження» [7, 51, 201].

Усі атаки на функції гешування можна розділити на дві групи:

1) атаки, що засновані на уразливості алгоритму перетворень (у подальшому аналітичні);

2) атаки, що не залежать від алгоритму.

В однобічних і колізійно стійких функціях гешування при обчисленнях не використовують секретну інформацію. Тому немає необхідності класифікувати атаки в залежності від таких характеристик інформації та їх доступності порушнику. Порушник має за мету знайти перший або другий прообраз для геш-значення або за допомогою колізії нав'язати помилкові дані. Забезпечення стійкості до колізії необхідне в схемах цифрового підпису, де використовуються однобічні функції гешування. Слід зазначити, що деякі атаки засновані на визначенні прообразу або колізії для випадкових повідомлень, інші надають порушникові достатньо свободи у виборі необхідних текстів даних.

Відомо [7, 10, 51, 190, 201], що рівень безпеки функції гешування можна досліджувати через аналіз її функції стиснення. Колізійно стійка функція стиснення може бути розширена на колізійно-стійку функцію гешування для довільної довжини вхідних даних. Проте, атаки на функцію стиснення не обов'язково призводять до атак на функцію гешування. Так, атака знаходження прообразів або колізії для f з обраним H_{i-1} призводить до атаки на h функцію гешування, але така сама атака з випадковим значенням H_{i-1} не призводить до атаки на h (якщо початкове значення IV не може бути змінене). Іншим прикладом є атака, що дозволяє створити колізії для функції стиснення, де початкове ланцюгове значення не є однаковим для обох повідомлень $H_{i-1} \neq H'_{i-1}$. Атаки на функцію стиснення, які не можна розширити щодо стійкості, визначають як сертифікаційні слабкості функцій гешування.

5.2. АТАКИ НА ФУНКЦІЇ ГЕШУВАННЯ

На основі аналізу робіт [7, 9, 10, 51, 190, 201–204] визначені основні атаки на функції гешування. Розглянемо їх детальніше.

1. ***Атака методом «Груба сила»***, або атака на знаходження випадкового (другого) прообразу, може бути виконана для знаходження прообразу за заданим геш-значенням або для знаходження прообразу, що дає задане геш-значення. Сутність атаки полягає в послідовному або випадковому перебиранні вхідних повідомлень і порівнянні результату виконання функції гешування із заданим. Успіх атаки при рівноймовірних появленнях геш-значень буде дорівнювати 2^{-n}, де n – довжина геш-значення в бітах. Складність такої атаки оцінюється $2^n - 1$ операцій обчислення геш-значень.

2. ***Атака методом парадоксу «Дня народження»*** виконується для знаходження двох різних повідомлень з однаковими геш-значеннями. Ця атака заснована на парадоксі «дня народження» і полягає в тому, що у двох генерованих

множинах геш-значень, що містять n_1 і n_2 елементів відповідно, імовірність знаходження елементів, що збігаються, між цими множинами оцінюється виразами: $P << 1 - \exp(-n_1 n_2 / 2^n)$. При $n_1 = n_2 = 2^{n/2}$ складність атаки оцінюється як $2^{n/2} + 1$ операцій обчислення геш-значень, а ймовірність успіху дорівнює $P << 1 - 1/e << 0{,}63$. Більш точні оцінки наведено в [7, 51].

3. ***Атака «Зустріч посередині»*** [7, 10, 51, 190, 201] є модифікацією атаки методом парадоксу «дня народження» і використовується для геш-функцій із циклічною структурою, якщо циклова функція $f()$ є такою, що інвертується стосовно проміжного значення X або блоку повідомлення M_i. Ця атака за складністю порівнюється з атакою методом парадоксу «дня народження».

4. ***Атака з корекцією блоку*** [7, 10, 51, 190, 201] використовується у випадку, якщо порушник має повідомлення і хоче змінити в ньому один або більше блоків без зміни геш-значення. Наприклад, функція гешування MD5 є вразливою до цієї атаки. Так, порушник, беручи блок повідомлення M_i (16 слів по 32 біти), залишає 11 слів, модифікує одне слово й обчислює 4, що залишилися. У результаті виходить блок M'_i, що відображається в те саме геш-значення, що обчислене для дійсного M_i. Повна версія MD5 не вразлива до цієї атаки.

5. ***Атака з фіксованою точкою*** [7, 10, 51, 190, 201] може застосовуватися за умови, що циклова функція f має одну чи декілька фіксованих точок. Фіксованою точкою називається блок повідомлення X_i, для якого виконується умова:

$$f(H_{i-1}, X_i) = H_{i-1},$$

тобто існує блок повідомлення X_i, що не змінює проміжний результат H_{i-1}. Таким чином, у повідомлення X можна додавати або видаляти блоки X_i без зміни геш-значення. Захистом від таких атак служить обчислення довжини повідомлення й додавання її наприкінці повідомлення.

6. ***Атака на базовий алгоритм шифрування*** [7, 10, 51, 190, 201] використовується для атаки на функції гешування, що базуються на блокових симетричних шифрах. Оскільки алгоритми шифрування розроблялися як зворотні (підтримують зворотне перетворення), то це може збільшити вразливість унаслідок застосування функції стиснення.

7. ***Диференційні атаки.*** Диференційний криптоаналіз [7, 10, 51, 190, 201] є потужним інструментом для аналізу не тільки блокових шифрів, але також і функцій гешування. За такої атаки досліджується вплив різниці вхідних даних функцій стиснення на відповідні вихідні різниці. Колізія виникає, якщо вихідна різниця дорівнює нулю.

8. ***Аналітичні слабкості.*** Велике число атак може бути засновано при блокуванні дифузії вхідних даних функції гешування, за яких зміни не впливають або можуть бути легко скасовані на наступних кроках. Серед найкращих атак у цьому класі – атаки, розвинені Доббертином (Dobbertin) на MDx-сімейство [201, 10, 51, 190]. Вони поєднують звичайні методи оптимізації та криптоаналітичні методи аналізу. Інша особливість цих атак полягає в тому, що зміни контролюються тільки в деяких точках алгоритмів; на відміну від диференційного криптоаналізу, де аналізу піддаються всі спільні диференціали деякого ступеня.

Теоретичні значення обчислювальної стійкості функцій гешування наведені в таблиці 5.1.

Таблиця 5.1. **Обчислювальна стійкість функцій гешування**

Тип функції гешування[a]	Ціль атаки	Ідеальна стійкість
Однонаправлена функція гешування	Знаходження першого прообразу	$2^n - 1$
	Знаходження іншого прообразу	$2^n - 1$
Колізійно стійка функція гешування	Знаходження будь-якої колізії	$2^{n/2}$

Опис і аналіз функцій гешування, що на цей час широко використовуються на міжнародному рівні, наведено нижче в таблиці 5.2. Їхню стійкість пропонується оцінювати кількістю бітів захисту.

Стійкість функції гешування ГОСТ Р 34.11 (ГОСТ 34.311-95) необхідно оцінювати, використовуючи оцінки, що наведені в таблиці 5.1, при значенні $n = 256$. У [7, 10, 51, 190] наведено дані щодо складності знаходження прообразу та складності виникнення колізії.

Таблиця 5.2. **Стійкість функцій гешування в криптографічних застосуваннях відносно симетричного шифру**

Біти захисту (симетричне перетворення)	Цифрові підписи та геш-функції, що застосовуються	HMAC	Функції вироблення ключів	Генерація випадкових чисел	Інше (має бути зазначене)
80	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	Має бути зазначене	Має бути зазначене	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	Має бути зазначене
112	SHA-224 SHA-256 SHA-384 SHA-512			SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	
128	SHA-256 SHA-384 SHA-512			SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	
192	SHA-384 SHA-512			SHA-224 SHA-256 SHA-384 SHA-512	
256	SHA-512			SHA-256 SHA-384 SHA-512	

5.3. АНАЛІЗ ІСНУЮЧИХ СТАНДАРТІВ ФУНКЦІЙ ГЕШУВАННЯ

На цей час можна виділити щонайменше два базових підходи до оцінки якості функцій гешування. Перший із них ґрунтується на вимогах, що були прийняті щодо функцій гешування в проекті NESSIE [111], другий – що були прийняті в міжнародному проекті «NIST SHA-3 Competition» [189, 196, 201–204].

Основними характеристиками алгоритмів гешування, за якими виконується їх порівняльна оцінка відповідно до рекомендацій проекту NESSIE є такі [196, 189, 201]:

– рівень захищеності функцій гешування від загальних атак;

– швидкодія алгоритмів гешування;

– статистичні властивості розподілів геш-значень.

Функції гешування є ітераційними конструкціями, що використовують функції стиснення з установленим розміром вхідних даних. Нижче наведені функції стиснення, що засновані на застосуванні блокових шифрів, модульній арифметиці та вимогах, що спеціально розроблені для цілей гешування.

5.3.1. Функції гешування, що засновані на блокових шифрах

З практичної точки зору, побудова функцій гешування на основі використання n-розрядного блокового шифру є ефективна програмна чи апаратна реалізації блокового шифру, що застосовується в системі захисту. Це забезпечує велику функціональність криптоалгоритмів і зниження витрат на їх розробку. В ISO/IEC 10118-2 визначаються дві функції гешування:

1) функції гешування однократної довжини;

2) функції гешування подвійної довжини.

Циклова функція є блоковим шифром і характеризується такими основними параметрами:

n – довжина вхідного блоку даних;

k – довжина ключа;

$m \le n$ – довжина вихідного блоку.

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

Поряд з параметрами n, k, m враховується характеристика швидкості, тобто кількість операцій при блоковому шифруванні, яку необхідно виконати для формування геш-значення довжиною, що дорівнює довжині блоку шифру.

Функції гешування однократної довжини

В основу методу побудови функцій гешування однократної довжини покладено схеми Матіаса-Мейєра-Озіса (Matyas-Mayer-Oseas), Девіса-Мейєра (Davies-Meyer) і Міагуччі-Пренеля (Miyaguchi-Preneel) [190, 201]. Загальним для цих схем обчислень є те, що рядок даних X, що гешується, розбивається на n-розрядні блоки $X_1, X_2, \ldots X_t$. Останній блок, у разі потреби, доповнюється необхідною кількістю символів. Вектор ініціалізації IV визначається як деяка n-розрядна константа. Основними визначеннями для цього є такі.

Визначення 9. ***Циклічна функція Матіаса-Мейєра-Озіса*** визначається аналітичним співвідношенням:

$$H_i = Eg\, H_{i-1}\, X_i \oplus X_i \quad \text{(рис. 5.2}a\text{)}.$$

Довжина блоку даних дорівнює довжині блоку шифру. Довжина гешзначення також приймається такою, що дорівнює довжині блоку блокового шифру, тобто $m = n$. Функція усікання забезпечує взяття найменших значущих k бітів рядка H_t.

Визначення 10. ***Циклічна функція Девіса-Мейєра*** визначається аналітичним співвідношенням

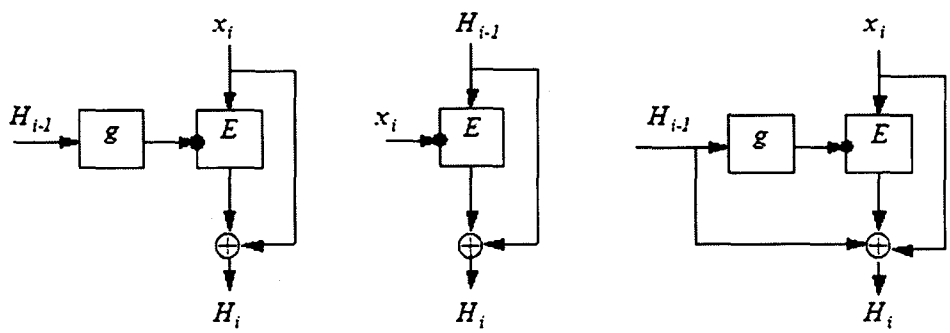
$$H_i = EX_i\, H_{i-1} \oplus H_{i-1} \quad \text{(рис. 5.2}б\text{)}.$$

Ця схема є дуальною попередній у тому сенсі, що X_i та H_{i-1} міняються місцями. Це призводить до того, що в схемі Девіса-Мейєра швидкість може бути й менше одиниці. Так, якщо за основу взяти алгоритм DES з 56-розрядним ключем, то швидкість функції гешування буде дорівнювати $56/64 < 1$.

Визначення 11. ***Циклічна функція Міагуччі-Пренеля*** визначається аналітичним співвідношенням:

$$H_i = Eg\, H_{i-1}\, X_i \otimes X_i \otimes H_{i-1} \quad \text{(рис. 5.2}в\text{)}.$$

Алгоритми, що реалізують вищенаведені схеми, мають такий вигляд (рис. 5.2).



а) *б)* *в)*

Рис. 5.2. Одношвидкісні функції гешування однократної довжини

Стійкість даних функцій гешування базується на стійкості блокового шифру E, що використовується в їх основі. Крім того, для даних алгоритмів не повинні реалізовуватися атаки, що засновані на використанні яких-небудь особливих властивостей і особливостей алгоритмів шифрування. За вказаних умов пошук прообразу або колізії вимагає виконання порядку 2_n і $2_{n/2}$ операцій шифрування відповідно.

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

Функції гешування з подвійною довжиною. Стандарт 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 1/2. Стандарт орієнтований на використання DES алгоритму [191]. Однак загальна конструкція може бути використана й для побудови функції гешування з іншими блоковими шифрами.

Визначення 12. Функція гешування подвійної довжини MDC-2 визначається аналітичним співвідношенням:

$$h(x) = H_t \| \tilde{H}_t,$$

де значення H_i та $\tilde{H}$ і для $1 \le i \le t$ обчислюються за формулами:

$$\begin{aligned} &H_0 = IV; \quad k_i = g(H_{i-1}); \quad C_i = E_{k_i}(x_i) \oplus x_i; \quad H_i = C_i^L \| C_i^R; \\ &\tilde{H}_0 = IV'; \quad k_i = \tilde{g}(\tilde{H}_{i-1}); \quad \tilde{C}_i = E_{\tilde{k}_i}(x_i) \oplus x_i; \quad \tilde{H}_i = \tilde{C}_i^L \| C_i^R. \end{aligned} \qquad 5.3$$

Значення C_i^L, C_i^R визначають ліву і праву 32-розрядні частини результату шифрування C_i відповідно. Функції g і $\tilde{g}$ відображають 64-бітові рядки у 56-бітові ключі для DES шифру за допомогою перетворень:

g(U) = u1 10 u4u5u6u7u9u10...u63, $\tilde{g}$(U) = u1 01 u4u5u6u7u9u10...u63.

Вектори ініціалізації IV і IV′ за замовчуванням визначаються значеннями:

IV = 5252525252525252x, IV′ = 2525252525252525x.

Результатом гешування є 128-бітове геш-значення. Алгоритм роботи MDC-2 наведений на рис. 5.3.

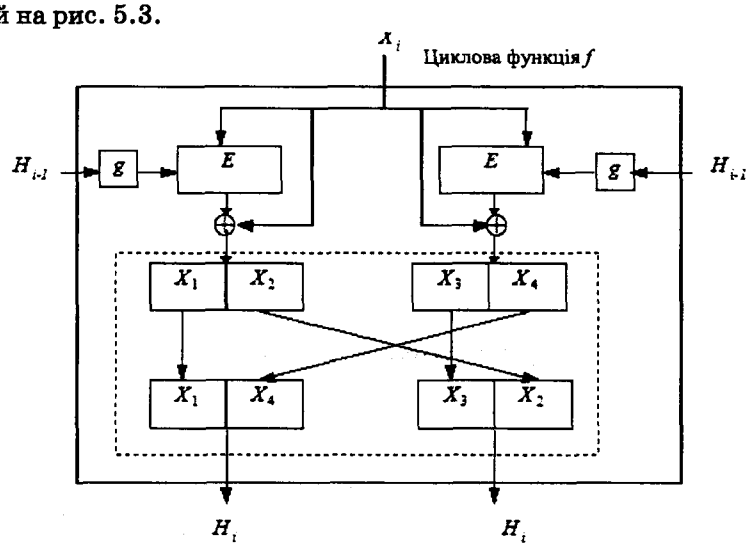


Рис. 5.3. Алгоритм функції гешування MDC-2

На основі MDC-2 побудована ***функція гешування MDC-4***. Одна ітерація циклової функції MDC-4 включає послідовне виконання двох циклових функцій MDC-2. Швидкість MDC-4 дорівнює ¼.

Оцінки стійкості функцій гешування, заснованих на блокових шифрах (для DES алгоритму), та інших функцій гешування наведені в таблиці 5.3 [194, 199–204].

Таблиця 5.3. **Верхні границі стійкості функцій гешування**

Функція гешування	n	m	Стійкість до прообразу	Стійкість до колізії	Примітки
Матіс-Мейєр-Озіса	N	n	2^n	$2^{n/2}$	
MDC-2 (DES)b	64	128	2^{282}	2^{254}	Довжина ключа = n
MDC-4 (DES)	64	128	2^{109}	4^{254}	швидкість 0,5
Мерхнули (DES)	106	128	2^{112}	2^{56}	швидкість 0,25
MD4	512	128	2^{128}	2^{20}	швидкість 0,276
MD5	512	128	2^{128}	2^{64}	
RIPEMD-128	512	128	2^{128}	2^{64}	
SHA-1, RIPEMD-160	512	160	2^{160}	2^{80}	

Така ж сама стійкість передбачається для функцій гешування Девіса-Мейєра та Міягучі-Принеля.

Стійкість може бути збільшена шляхом використання шифру з довжиною ключа, що дорівнює довжині блоку шифру.

Схеми Матіаса-Мейєра-Озіса і MDC-2 включені в ISO/IEC 10118-2 [191], і стандарт для функції гешування не конкретизує блоковий шифр, що застосовується. Стійкість таких схем може бути збільшена шляхом використання шифру з більшою довжиною ключа та довжиною блоку шифру. Швидкість гешування буде визначатися складністю блокового симетричного шифрування.

5.3.2. Функції гешування, засновані на асиметричних перетвореннях

У функціях гешування, що засновані на асиметричних перетвореннях, для побудови ітераційних функцій стиснення використовуються модульні обчислення аналогічно, як і в асиметричних криптографічних системах, тобто відкритими ключами. Їх найбільший недолік полягає у великій складності, і, як наслідок, низькій швидкості гешування (наприклад, у порівнянні із замовленими функціями гешування). Алгоритми гешування MASH-1 та MASH-2, подані в цій частині стандарту, розроблені з урахуванням аналізу аналітичних атак відносно таких асиметричних функцій гешування.

Функція стиснення в MASH-1 побудована на операції модульного піднесення до ступеня, де бітовий розмір модуля обчислень M вибирається достатнім для криптографічних застосувань. Попередньо вхідний блок даних X розбивається на $n/2$ бітові блоки Y_i. Функція гешування MASH-1 має таке визначення представлення [193].

Визначення 13. ***Функція гешування MASH-1*** визначається функцією стиснення H_i, $1 \le i \le t$ вигляду:

$$H_i = (((H_{i-1} \otimes Y_i \vee A)\ 2 \bmod M)\ n) \otimes H_{i-1} \qquad (5.4)$$

Геш-значення MASH-1 є n-бітовим блоком $h(x) = H_t$, де $n = 16\ n' < m$, а m визначає бітову довжину модуля $M = pq$, p і q є таємні великі прості числа. Вектор ініціалізації IV визначається значенням $H_0 = 0$, а постійна $A = 0\ xf_0 \ldots 0 \ldots 0$. Операції $\otimes$ і $\vee$ є побітовими операціями, що виключають

АБО (за модулем 2), і звичайною операцією АБО відповідно. Операція n визначає збереження найменших n-бітів з m-бітів результату.

Безпека MASH-1 залежить частково від складності модульних обчислень (для складеного числа невідомо його розкладання), тобто від факторизації модуля перетворення. Надлишкові біти також роблять свій внесок, що в цілому призводить до такого рівня захисту: для підробки геш-значення необхідно виконати $2^{n/2}$ операцій; для створення колізії необхідно виконати $n \times 2^{n/4}$ операцій.

MASH-2 є одним із варіантів MASH-1, єдина відмінність полягає в тому, що використовується модульне піднесення до ступеня з $e = 28 + 1$ (замість модулярного піднесення до ступеня з $e = 2$). Функції гешування MASH-1 і MASH-2 включені в стандарт ISO/IEC 10118-4 [193]. Цей стандарт також визначає додаткове перетворення виходу, що використовується для приведення довжини геш-значення до $n/2$ бітів і менше.

5.3.3. Функції гешування, що розроблені за замовленням

Функції гешування за замовленням є функціями, спеціально розробленими для задач гешування. Вони оптимізують тимчасові витрати на обчислення, що пов'язані, насамперед, при виконанні алгоритму зі звертанням до пам'яті. Функції цього класу використовують алгоритми обчислень, що вперше реалізовані в проекті алгоритму MD4.

У MD4-подібних алгоритмах поділяють повідомлення на блоки X_i та ланцюгову змінну H_{i-1} на 32-бітові слова або на 64-бітові слова для алгоритмів, запропонованих останнім часом. Функція стиснення модифікує ланцюгову змінну до нового значення H_i за допомогою поточного блоку повідомлення. Обчислення виконується за кроками. На кожному кроці змінюється попереднє значення, яке є словом ланцюгової перемінної в залежності від слова повідомлення або інших реєстрових значень. На кожному кроці обчислення функції стиснення виконується кілька циклів, і в кожному циклі слова блоку повідомлення використовуються один раз.

Основний крок алгоритму MD4 описується виразом вигляду:

$$A = (A + f(B, C, D) + X[j] + y) << s[j]. \qquad (5.5)$$

Тут (A, B, C, D) – регістри, що містять чотири 32-бітових слова ланцюгової змінної. Операція «+» означає складання (доповнення) за модулем 232, та $(...) << s$ – побітовий циклічний зсув слова на s позицій вліво. Нелінійна функція f визначає побітові обчислення, $X[j]$ – слово повідомлення на j кроці, y – константа та $s[j]$ – постійна зсуву. Функція f і константа y змінюються на кожному циклі обчислень. MD4 включає три цикли, що використовують мультиплексування, мажоритарну вибірку і функції виключно АБО (сума за модулем 2). Слова блоку повідомлення використовуються в обчисленнях на всіх кроках циклу.

Кроки кожної операції є зворотними (можна виконати обчислення у зворотному порядку), але на останньому етапі обчислення функції стиснення здійснюється додаткова операція, що додає при обчисленні ланцюгової змінної H_i початкове реєстрове значення до результуючого значення. Це робить функцію стиснення незворотною. Ланцюгова змінна H_i, що отримана на останньому кроці при обчисленні блоку повідомлення, який містить дані про довжину, є результатом гешування.

На основі алгоритму MD4 зроблено декілька вдосконалень алгоритмів, що використовують для збільшення стійкості до прообразу та стійкості до колізії додаткові механізми. Ці ідеї включають збільшення числа циклів, деякі побітові операції на кроці обчислень і збільшення довжини ланцюгових змінних, що призводить до більш довгого значення функції гешування. Так, у сімействі SHA-1 використовується процедура розширення блоку повідомлення, що обчислює різницю слів на кожному кроці. У RIPEMD використовуються дві рівнобіжні схеми обчислення, які є модифікованими версіями MD4. У таблиці 5.4 наведено порівняльні дані MD4 подібних алгоритмів.

Таблиця 5.4. **Параметри MD4 подібних функцій гешування**

Алгоритм	Розмір геш-значення (бітів)	Розмір вихідного блоку (бітів)	Розмір слова (бітів)	Число циклів на число кроків у циклі
MD4	128	512	32	3×16
MD5	128	512	32	4×16
RIPEMD-128	128	512	32	4×16×2
RIPEMD-160	160	512	32	5×16×2
SHA-1	160	512	32	4×20
SHA-2/256	256	512	32	1×64
SHA-2/384	384	1024	64	1×80
SHA-2/512	512	1024	64	1×80

Функція гешування SHA-1 визначена у федеральному стандарті США FIPS 180-1 [197], вона рекомендована NIST разом з DSA стандартом для цифрових підписів. NIST обновив цей стандарт, затвердивши стандарт FIPS 180-2 [198], що включає, крім SHA-1, ще три нових функції гешування SHA-2/256, SHA-2/384 і SHA-2/512 з функціями гешування відповідно довжинами 256, 384 та 512 бітів. Додатково ANSI США прийняло також удосконалені банківські стандарти з відкритими ключами: стандарт X 9.30, у якому разом з DSA використовується алгоритм SHA-1 і стандарт X 9.31, у якому разом зі схемою RSA використовується алгоритм MDC-2 [201].

Також в ISO/IEC 10118 додатково введено функції гешування, що включені у FIPS 180-2. У частині 10118-2 визначені функції гешування, що засновані на блокових шифрах у конструкції Matyas-Meyer-Oseas, де незалежний блоковий шифр в алгоритмі MDC-2 із двома й більше функціями використовується для обчислення значення геш-функції подвійної та потрійної довжини відповідно. Частина 10118-3 [192] визначає три замовлених алгоритми: RIPEMD-128, RIPEMD-160 і SHA-1. Ця частина стандарту на цей час переглянута. Окрім зазначених трьох алгоритмів, прийняті функції гешування: SHA-2/256, SHA-2/384, SHA-2/512 і Whirlpool. Частина 10118-4 [192] описує MASH-1 та MASH-2 функції гешування, що використовують модульну арифметику.

Розглянемо перспективні функції гешування: Whirlpool та SHA-2/256, SHA-2/384, SHA-2/512.

У таблицях 5.5 і 5.6 подано характеристики, властивості й аналітичні результати оцінки швидкодії алгоритму Whirlpool у порівнянні з іншими відомими алгоритмами.

З аналізу цих таблиць випливає, що Whirlpool уступає багатьом алгоритмам за швидкодією, але однією з головних переваг алгоритму Whirlpool залишається розмір геш-значення, що дорівнює 512 бітам.

У таблиці 5.6 подано дані, що показують витрати часу на обчислення геш-значення/ ініціалізацію програми/ ініціалізацію програми та її завершення.

При цьому одиницями виміру є:

– час обчислення геш-значення, вимірюється в числі циклів процесора на байт;

– час ініціалізації програми та її завершення, у числі циклів процесора.

Аналіз безпеки алгоритму Whirlpool

Доказ безпеки алгоритму Whirlpool ґрунтується на недоведеності вразливості схеми Міягуччі-Пренеля при побудові функції гешування засобом використання блокового симетричного шифру. Ця схема є доказово стійкою, якщо вважати, що використовуваний алгоритм шифрування є ідеальним. Блоковий шифр W, що використовується у Whirlpool, дуже схожий на алгоритм AES Rijndael.

У таблиці 5.7 наведені порівняльні характеристики для блокових шифрів Rijendael і Whirlpool. Його основна відмінність полягає в тому, що Rijndael може застосовуватись з блоками довжиною 128, 192, 256 бітів, а Whirlpool тільки 512 бітів. Зважаючи на подібність алгоритмів, частина аналізу безпеки алгоритму Rijndael є застосовною й до алгоритму Whirlpool.

Стійкість алгоритму Whirlpool ґрунтується на виборі таблиці підстановок S і наявності квадратичних залежностей між вхідними і вихідними блоками таблиці, а також впливу числа циклів вбудованого блокового шифру на безпеку алгоритму.

Вибір *S*-блоку

В алгоритмі Whirlpool використовується блок підстановки для відображення 8 вхідних бітів у 8 вихідних бітів. В оригінальній версії алгоритму використовувалася випадково генерована таблиця підстановки, що обрана за умови забезпечення неможливості успішного проведення лінійного й диференційного криптоаналізу. Однак, при цьому була припущена істотна помилка, що призводить до появи лінійного параметра. У представленій до розгляду в проекті NESSIE [111] версії алгоритму використовувалася інша таблиця підстановки, що складається з 4-бітових таблиць, яка усувала цей недолік.

Зменшення числа циклів

Версія алгоритму зі зменшеною кількістю циклів має гірші властивості випадковості. Були проведені дослідження впливу числа циклів на стійкість функції стиснення. Сутність їх полягає в такому.

Нехай є 256 текстів довжиною 64 байти, що відрізняються в одному байті, а в інших співпадають. Із цього випливає, що після 2-х циклів шифрування тексти приймуть усі 256 значень у кожному з 64-х байтів, і після 3-х циклів шифрування сума всіх 256 байтів на кожній позиції всіх текстів буде дорівнювати 0. Така структура має назву «інтеграл». Відзначимо, що інші 63 структури подібні, поки позиція байта, що змінюється, може приймати кожне з 64-х значень.

Таблиця 5.5. **Характеристики функцій гешування**

Назва примітиву	Розмір слова (bits)	Розмір підключа (bits)	Розмір допоміжної таблиці (bits)	Число зрушень/ перестановок + множень	XOR, ADD, MULT (bit size)	AND, OR, NOT	Загальне число логічних операцій (8-bit)	Розмір коду (k)
Whirlpool	8	5120	1280	1188	1175 (64-bit)	1268 (64-bit)	19544	65
SHA-1	32			0/224 (32-bit)	312 (32-bit), 325 (32-bit)	100 (32-bit)	2948	
SHA-2/256	32			96 (32-bit)/ 576 (32-bit)	640 (32-bit), 600 (32-bit)	384 (32-bit)	6496	
SHA-2/384 SHA-2/512	64			128 (64-bit)/ 736 (64-bit)	816 (64-bit), 760 (64-bit)	480 (64-bit)	16448	

Таблиця 5.6. **Продуктивність функцій гешування**

Найменування примітиву	Розм. геш	Тип ЕОМ							
		PIII/Linux	PIII/MS	PI/MMX	Pentium 4	Pentium 2	486	Mac	AMD
Whirlpool 2nd impl (MMX)	512 512	79/59/5534 46/63/3199	82/47/5657 73/50/5026	190/260/13K 177/244/11K	108/130/8559 60/132/4495	97/51/6804 83/76/5992	460/122/30K N/A	133/82/8818 N/A	104/42/7012 99/51/6573
MD4	128	4.7/15/440	4.5/12/411	6.9/44/639	6.4/34/600	4.7/13/444	4.2/36/601	7.0/14/518	4.7/8.3/419
MD5	128	7.2/15/602	6.8/12/558	8.9/44/768	9.4/34/799	7.2/12/611	5.8/36/704	10/14/752	7.5/10/599
RIPEMD	160	18/16/1339	16/14/1207	33/54/2363	26/36/1929	23/15/1651	27/38/3741	20/17/1384	21/12/1493
SHA-1	160	15/16/1024	13/14/929	27/54/1830	25/20/1497	16/15/1086	17/39/1500	9.7/16/872	12/12/825
SHA-2	256 384 512	39/44/2736 83/157/11K 83/156/11K	39/20/2643 74/101/9907 74/101/9940	71/82/4804 187/432/25K 187/433/25K	40/118/3159 122/292/16K 122/292/16K	40/34/2860 84/244/11K 84/244/11K	60/50/4271 176/207/24K 176/207/24K	29/32/2052 93/206/12K 93/206/12K	34/39/2369 71/105/9672 71/106/9752
BC HASH-Rijndael	128 256	49/15/1712 112/44/3775	45/14/1588 116/20/3913	90/42/3078 241/93/8082	77/19/2579 152/118/5254	45/13/1605 123/35/4118	214/32/6963	59/22/2043 –/–/–	47/11/1650 149/40/4889

Таблиця 5.7. **Порівняльний аналіз блокових шифрів**

Властивості	Rijndael	Whirlpool
Розмір блоку (у бітах)	128, 192, 256	тільки 512
Число циклів	10, 12, 14	тільки 10
Розгортання ключа	апріорний алгоритм	безпосередньо циклічна функція
GF (2^8) поліном	$x^8 + x^4 + x^3 + x + 1$ (0×11B)	$x^8 + x^4 + x^3 + x^2 + 1$ (0×11D)
Базис для *S*-box	відображення $u -> u - 1$ афінне перетворення в полі GF (2^8)	рекурсивна структура
Базис для циклів	поліном x^i над GF (2^8)	наступний вміст *S*-box
Розсіювання	множення циркулярної 4×4 матриці MDS *cir* (2, 3, 1, 1)	множення циркулярної 8×8 матриці MDS *cir* (1, 1, 4, 1, 8, 5, 2, 9)

Поданий вище трьохцикловий «інтеграл» може бути перетворений на чотирьохцикловий, якщо розглядати структуру з 2^{64} текстів. Головна відмінність полягає в тому, що після першого циклу він приймає всі 2^{64} значення у верхньому ряді, а 3 цикли, що залишилися, можуть бути представлені як набір 2^{56} варіантів трьохциклових інтегралів. Оскільки тексти в кожній інтегральній сумі дають 0 у будь-якому байті після 4-х циклів, то це ж можна сказати і про суму всіх 2^{64} текстів.

Аналогічно можна визначити трьохцикловий зворотний «інтеграл» як 3 цикли інверсного шифру Whirlpool. У версії алгоритму Whirlpool з 6 циклами можна комбінувати перші 3 цикли 4-циклового прямого інтеграла та 3-х циклового зворотного інтеграла для перекриття всіх циклів шифру з імовірністю 1.

Таким чином, з тимчасовою складністю 2^{120} можна знайти набір 2^{120} вхідних значень Whirlpool зі зменшеною кількістю циклів до 6, оскільки кожен байт вхідних і вихідних значень приймає всі значення багато разів. Передбачається, що для того, щоб знайти структуру з 2^{120} текстів із властивостями, схожими з Whirlpool, зі зменшеною кількістю циклів до 6, буде потрібна генерація 2^{128} вихідних значень і значна кількість пам'яті.

У версії Whirlpool зі зменшеною кількістю циклів до 7 можна комбінувати повний 4-цикловий прямий інтеграл і 3-цикловий зворотний інтеграл для покриття семи циклів з імовірністю 1. На цей момент немає результатів щодо того, яким чином такий інтеграл з числом циклів 7 можна застосувати для ефективного розрізнення Whirlpool від випадкової 512-бітової перестановки.

Квадратичні залежності

Були проведені дослідження існування квадратичної залежності у вхідних і вихідних значеннях блоку підстановки алгоритму Whirlpool. Було показано, що для блоків підстановки Rijndael і Serpent існують квадратичні рівняння для вхідних і вихідних бітів з імовірністю 1. Такі рівняння завжди існують для n бітів n-бітового блоку підстановки, якщо $n \leq 6$, але для $n > 6$ не завжди.

Також проведені дослідження з виявлення квадратичних залежностей у блоці перестановки Whirlpool. Блок перестановки – це 8-бітова перестановка. Існує максимум 137 можливих ступенів свободи в багатомірному виразі для 8 вхідних і 8 вихідних бітів. Найпростішим методом перевірки наявності таких залежностей є обчислення 2^{56} раз визначника 137-мірних двійкових матриць. Для повної таблиці підстановки Whirlpool не було знайдено квадратичних залежностей. Проте, оскільки блок підстановки складається з декількох 4-бітових підстановок, задача побудови маленької системи багатомірних квадратичних рівнянь істотно спрощується.

5.4. ОПИС І АНАЛІЗ АЛГОРИТМУ SHA-1

Функція гешування SHA-1 [191, 199] є федеральним стандартом США FIPS 180-2 та міжнародним стандартом ISO/IEC 10118. Незважаючи на те, що довжина результату гешування 160 бітів є недостатньою для функції, які мають бути стійкими до колізій, існує достатньо практичних застосувань, що використовують функції гешування із 160 бітами. Алгоритм SHA-1 оперує з 512 бітами блоку даних, розділеними на 32-бітові слова й обчислює геш-значення довжиною 160 бітів (рис 5.4).

SHA-1 є посиленою версією алгоритму MD4 і визначається як ітераційна функція стиснення. Як початкові використовуються такі дані:

IV = 67452301x efcdab89x 98badcfex 10325476x c3d2e1f0x.

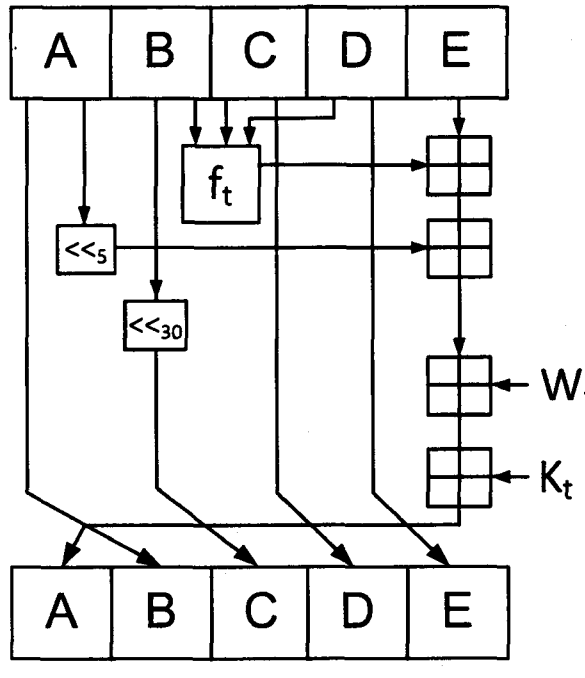


Рис. 5.4. Функція гешування SHA-1

На кожному кроці обчислення функції стиснення використовується 5 слів IV і 16 слів повідомлення, у результаті утворюються п'ять вихідних слів a_i, b_i, c_i, d_i, e_i, які використовуються в наступному циклі замість початкових значень IV. Останнє обчислення функції стиснення визначає результуюче геш-значення. Довжина оброблюваних даних повинна бути кратною 512 бітам і включати дані своєї довжини.

Функція стиснення SHA-1

Нехай $M = [W_0||W_1||:::||W_{15}]$ є блок повідомлення в 512 бітів, і W_i – 32-бітові слова. SHA-1 використовує процедуру розширення, що визначена як

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) << 1; \quad 16 \le i \le 79.$$

Використовуються такі константи:

K_i = 5a827999x; $0 \le i \le 19$,
K_i = 6ed9eba1x; $20 \le i \le 39$,
K_i = 8f1bbcdcx; $40 \le i \le 59$,
K_i = ca62c1d6x; $60 \le i \le 79$,

і булеві функції

$F_i(X,Y,Z)$ = fif $(X,Y,Z) = (X \& Y) \mid (\overline{X} \& Z)$, $0 \le i \le 19$,
$F_i(X,Y,Z)$ = fxor $(X,Y,Z) = X \oplus Y \oplus Z$, $20 \le i \le 39$, $60 \le i \le 79$,
$F_i(X,Z)$ = fmaj $(X,Y,Z) = (X \& Y) \mid (X \& Z) \mid (Y \& Z)$, $40 \le i \le 59$.

Припустимо, що початкові значення A_0, B_0, C_0, D_0, E_0 задані. Функція стиснення виконує обчислення за $0 \le i \le 79$ кроків (з додатком за mod 232):

$A_{i+1} = A_i << 5 + f_i(B_i, C_i, D_i) + E_i + W_i + K_i$,
$B_{i+1} = A_i$,
$C_{i+1} = B_i << 30$,
$D_{i+1} = C_i$,
$E_{i+1} = D_i$.

Нарешті вихід функції стиснення визначається як

$$A = A_0 + A_{80}, \; B = B_0 + B_{80}, \quad C = C_0 + C_{80}, \; D = D_0 + D_{80}, \quad E = E_0 + E_{80}.$$

Аналіз безпеки алгоритму SHA-1

Вимоги до алгоритму SHA-1 відносно колізійної стійкості є такими: при довжині геш-значення 160 бітів складність колізії має складати 2_{80} спроб, а знаходження другого прообразу – порядок 2^{160}. На цей момент невідомі атаки, що спростовували б ці припущення для повного числа циклів гешування. У [189] наведено дані про атаку, яка при числі циклів 61 має складність 2^{69}.

Є істотна особливість функції стиснення, пов'язана з розширенням 16 слів повідомлення до 80 слів, що призводить до того, що для двох різних блоків з 16 слів результуючі 80 значень слів відрізняються у великому числі бітових позицій. Це виключає можливість реалізації атаки, що використовувалася на інших алгоритмах MD-x сімейства.

Слайд-атака SHA-1

Складність слайд-атаки на алгоритм SHA-1 складає 232 [189, 199]. Хоча важко уявити, яким чином ця атака може бути практично реалізована. Аналіз функції стиснення показує несподівану особливість.

Розглянемо два повідомлення: $M = [W_0||W_1||:::||W_{15}]$ і $M' = [W'_0||W'_1||:::||W'_{15}]$. Основне полягає в тому, що процедуру для розширення повідомлення можна модифікувати, ніби рухати. Наприклад, просто вибираємо $W'_i = W_{i+1}$ для $0 \le i \le 14$ та $W'_{15} = (W_0 \oplus W_2 \oplus W_8 \oplus W_{13}) << 1$. Після розширення повідомлення буде справедливим вираз $W'_i = W_{i+1}$ для $0 \le i \le 78$. Друге спостереження полягає в тому, що для 20 кроків у кожному циклі функції стиснення булева функція f_i й адитивні константи K_i є незмінними. Отже, будь-які два послідовних кроки (крок i та крок i+1) у функції стиснення є подібними, якби не три перетворення між різними циклами (це відбувається, коли $i = 19; 39; 59$). Припустимо тепер, що обчислення геш-значення для M и M' робиться від початкових значень A, B, C, D, E та A', B', C', D', E' відповідно, які пов'язані таким чином:

$B' = A, C' = B >> 30, D' = C, E' = D.$

Ціль атаки полягає в тому, щоб знайти повідомлення та початкові значення, для яких те саме відношення (між ланцюговими перемінними) збережеться наприкінці функції стиснення. Така пара з повідомлення і зв'язаного з ним початкового значення називається *слайд-парою*. Метод виявлення слайд-пари технічно непростий, але загальна стратегія полягає в тому, щоб вибрати підходящі значення для ланцюгових перемінних на кроках $i = 19$ та $i = 39$, і виконати атаку «зустріч посередині». Ця процедура повторюється до перетворення на кроці $i = 59$ з імовірністю успіху 2^{-32}. Витрати часу та складність атаки має порядок 2^{32}.

Диференційний аналіз

Диференційний криптоаналіз є доречним для функції SHA-1. Хоча поширення різниці в початкових значеннях через функцію стиснення не дозволяє знайти прообрази або колізії для функції гешування. Це може вказувати на небажані властивості функції стиснення.

Для SHA-1 існує диференційна характеристика з 5-ма кроками за будь-якими 5-ма кроками у функції стиснення з імовірністю 1. Handschuh і Naccache [199] показали, що за більш ніж 80 кроків у функції стиснення краща диференційна характеристика має ймовірність близько 2^{-116}. Підкреслимо, що ця оцінка є «сприятливою» для криптоаналізу, оскільки має бути неможливо пов'язати всі встановлені характеристики так, щоб досягти цих зсувів. Bogaert та Rijmen [199] знайшли оптимальні диференційні характеристики, які задовольняли вимозі, що вага Хеммінга кожного вхідного слова з 32-ма бітами була обмежена зверху значенням 2. Було виявлено, що є дві 10-крокові характеристики для fif з імовірністю 2^{-12} (це у 2 рази краще, ніж у 10-крокової характеристики) для fxor у кращому випадку з імовірністю 2^{-12}, і 20-крокові характеристиці для fif і fmaj з імовірностями 2^{-42} та 2^{-41} відповідно. Ці значення наведені в [199]. Також знайшли диференційну 28-крокову характеристику з імовірністю 2^{-107} і диференційну 30-крокову характеристику з імовірністю 2^{-138}. Це показує, що, коли число кроків збільшується, імовірність диференційних характеристик зменшується швидко (оскільки вага Хемінга в різних словах збільшується). Границя безпеки SHA-1 проти диференційного аналізу вважається дуже великою.

5.5. ОПИС І АНАЛІЗ АЛГОРИТМІВ SHA-2

SHA-2 [197, 201] включений у новий стандарт функції гешування NIST (FIPS 180-2), він забезпечує обчислення геш-значень з довжиною 256, 384 або 512 бітів. Головна причина появи нового стандарту полягає в тому, що необхідно забезпечити такий самий рівень стійкості, як і при використанні федерального стандарту США – блокового симетричного шифру AES (FIPS-197) із довжинами ключів 128, 192 та 256 бітів відповідно).

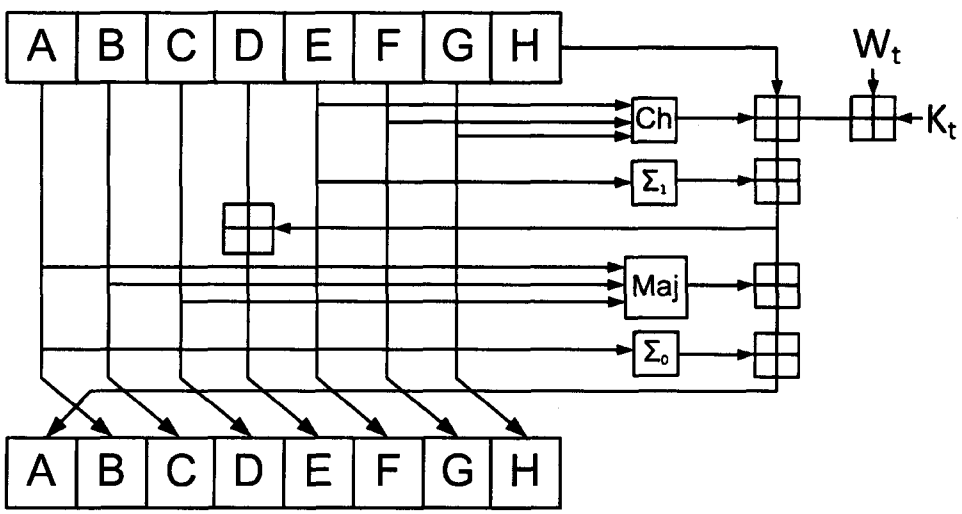


Рис. 5.5. Функція гешування SHA-2

Алгоритм SHA-2/256

У SHA-2/256 функція гешування обробляє 512-бітові вихідні дані, розділені на 32-бітові слова та обчислює геш-значення з довжиною 256 бітів. Алгоритм визначений як ітераційна функція стиснення.

Обчислення починається з 8 початкових значень:

$$IV = 6a09e667x\ bb67ae85x\ 3c6ef372x\ a54ff53ax\ 510e527fx\ 9b05688cx\ 1f83d9abx\ 5be0cd19x.$$

При кожному застосуванні функції стиснення використовуються 8 слів як початкові значення і 16 слів вхідного повідомлення, що дає 8 вихідних слів, які використовуються як початкові значення для наступного кроку обчислень.

На заключному кроці формується геш-значення.

Функція стиснення SHA-2/256 [198]

Нехай $M = [W_0||W_1||:::||W_{15}]$ є блок повідомлення довжиною 512 бітів, і W_i – 32-бітові слова. SHA-2/256 використовує процедуру розширення, що визначена як

$$W_i = \sigma_1 W_{i-2} \oplus W_{i-7} \oplus \sigma_0 W_{i-15} \oplus W_{i-16}, \quad 16 \le i \le 63,$$

де σ_0 і σ_1 визначаються в такий спосіб:

$\sigma_0(X) = X >> 7 \oplus X >> 18 \oplus X > 3,$

$\sigma_1(X) = X >> 17 \oplus X >> 19 \oplus X > 10.$

Операції (...) > i (...) >> позначають зрушення i циклічне зрушення слова вправо.

Використовуються такі функції:

$\mathrm{Ch}(X, Y, Z) = (X \& Y) \oplus (X \& Z),$

$\mathrm{Maj}(X, Y, Z) = (X \& Y) \oplus (X \& Z) \oplus (Y \& Z),$

$\Sigma_0(X) = X >> 2 \oplus X >> 13 \oplus X >> 22,$

$\Sigma_1(X) = X >> 6 \oplus X >> 11 \oplus X >> 25.$

Припустимо, що початкові значення $A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0$ задані.

Функція стиснення обчислюється на кроках $0 \le i \le 63$ (додавання за mod 2^{32}):

$A_{i+1} = \Sigma_0(A_i) + \mathrm{Maj}(A_i, B_i, C_i) + \Sigma_1(E_i) + \mathrm{Ch}(E_i, F_i, G_i) + H_i + W_i + K_i,$

$B_{i+1} = A_i,$

$C_{i+1} = B_i,$

$D_{i+1} = C_i,$

$E_{i+1} = D_i + \Sigma_1(E_i) + \mathrm{Ch}(E_i, F_i, G_i) + H_i + W_i + K_i,$

$F_{i+1} = E_i,$

$G_{i+1} = F_i,$

$H_{i+1} = G_i.$

32-бітові константи K_i є різними на кожному із 64-х кроків функції стиснення. Обчислення функції стиснення завершується як

$A = A_0 + A_{64}, \quad B = B_0 + B_{64}, \quad C = C_0 + C_{64}, \quad D = D_0 + D_{64},$

$E = E_0 + E_{64}, \quad F = F_0 + F_{64}, \quad G = G_0 + G_{64}, \quad H = H_0 + H_{64}.$

Алгоритм SHA-2/512

Головна відмінність між SHA-2/256 і SHA-2/512 полягає в тому, що останній алгоритм використовує довжину слова в 64 бітів (замість 32 бітів). Це дозволяє обчислити значення геш-функції повідомлення, що у два рази довше в порівнянні з SHA-2/256, без зміни самої структури алгоритму. Інша відмінність полягає в числі кроків у функції стиснення, що збільшене з 64 до 80. Таким чином, функція SHA-2/512 працює на блоках довжиною 1024 бітів, розділених на 64-бітові слова, й обчислює геш-значення довжиною в 512 бітів. Алгоритм визначений як ітеративна функція стиснення.

Обчислення робляться з початкових значень:

IV = 6a09e667f3bcc908x bb67ae8584caa73bx 3c6ef372fe94f82bx
a54ff53a5f1d36f1x
510e527fade682d1x 9b05688c2b3e6c1fx 1f83d9abfb41bd6bx
5be0cd19137e2179x.

При кожному використанні функції стиснення робиться 8 слів як початкові значення і 16 вхідних слів повідомлення, що дає 8 вихідних слів, які потім використовуються як початкові значення для наступного застосування функції стиснення. Останнє обчислення визначає значення геш-функції.

Ці перетворення зображені на рис. 5.6.

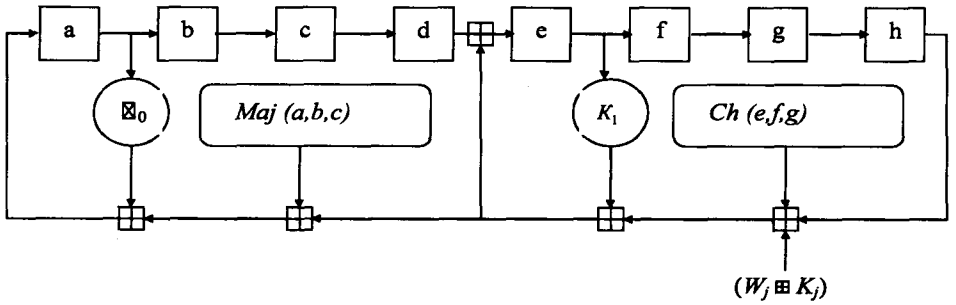


Рис. 5.6. Перетворення стиснення

На рис 5.6 прийняті такі позначення: «+» – додавання за модулем 2^{32}, K_j – циклові константи j-го циклу, W_j – 32-бітовий елемент розширеного повідомлення j-го циклу. Розширене повідомлення W формується з повідомлення M з використанням перетворення, зображеного на рис. 5.7. Початковим заповненням регістра є повідомлення M.

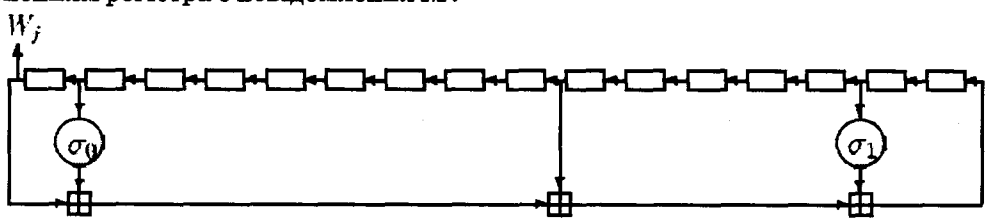


Рис. 5.7. Перетворення розширення

Нехай $M = [W_0 || W_1 || ::: || W_{15}]$ є блок повідомлення в 1024 бітів, і W_i – 64-бітові слова. SHA-2/512 використовує процедуру розширення, що визначена як

$$W_i = \sigma_1 W_{i-2} \oplus W_{i-7} \oplus \sigma_0 (W_{i-15}) \oplus W_{i-16};\ 16 \le i \le 79,$$

де σ_0 і σ_1 визначюється в такий спосіб:

$\sigma_0 (X) = X >> 1 \oplus X >> 8 \oplus X > 7,$
$\sigma_1 (X) = X >> 19 \oplus X >> 61 \oplus X > 7.$

Операції (...) > i (...) >> позначають зрушення i циклічне зрушення слова вправо.

Використовуються такі функції:

$\text{Ch}\ (X, Y, Z) = (X \&Y) \oplus (X \& Z),$
$\text{Maj}\ (X, Y, Z) = (X \&Y) \oplus (X \& Z) \oplus (Y \& Z),$
$\Sigma_0 (X) = X >> 28 \oplus X >> 34 \oplus X >> 39,$
$\Sigma_1 (X) = X >> 14 \oplus X >> 18 \oplus X >> 41.$

Припустимо, що початкові значення A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0 задані.

Функція стиснення обчислюється на кроках $0 \le i \le 79$ (додавання за модулем 264):

$A_{i+1} = \Sigma_0 (A_i) + \text{Maj}\ (A_i, B_i, C_i) + \Sigma_1 (E_i) + \text{Ch}\ (E_i, F_i, G_i) + H_i + W_i + K_i,$
$B_{i+1} = A_i,$

$$C_{i+1} = B_i,$$
$$D_{i+1} = C_i,$$
$$E_{i+1} = D_i + \Sigma_1(E_i) + \mathrm{Ch}(E_i, F_i, G_i) + H_i + W_i + K_i,$$
$$F_{i+1} = E_i,$$
$$G_{i+1} = F_i,$$
$$H_{i+1} = G_i.$$

64-бітові константи K_i є різними на кожному з 80-ти кроків функції стиснення. Обчислення функції стиснення завершується як

$$A = A_0 + A_{80}, \quad B = B_0 + B_{80}, \quad C = C_0 + C_{80}, \quad D = D_0 + D_{80},$$
$$E = E_0 + E_{80}, \quad F = F_0 + F_{80}, \quad G = G_0 + G_{80}, \quad H = H_0 + H_{80}.$$

Алгоритм SHA-2/384

Функція SHA-2/384 визначена так само, як SHA-2/512, з наступними двома змінами.

1. Обчислення робляться з іншими початковими значеннями:

IV = cbbb9d5dc1059ed8x 629a292a367cd507x 9159015a3070dd17x
152fecd8f70e5939x 67332667ffc00b31x 8eb44a8768581511x
db0c2e0d64f98fa7x 47b5481dbefa4fa4x.

2. Результуюче геш-значення з 384-ма бітами отримують шляхом усікання результату до крайніх лівих 384 бітів.

Аналіз безпеки SHA-2

Вимоги колізійної стійкості до алгоритмів SHA-2 є такими: за довжини геш-значення n бітів порядок утворення колізії має складати $2^{n/2}$ спроб, а знаходження іншого прообразу повинне мати порядок 2^n, де n = 256, 384, 512 для SHA-2/256, SHA-2/384 і SHA-2/512 відповідно. На цей момент невідомі атаки, що спростовували б ці припущення.

SHA-2 – новий проект, у якому по суті розвиваються ідеї SHA-1, але є й важливі розходження в структурі. Для 256-бітової версії число кроків у функції стиснення менше, ніж для SHA-1 (64 кроки в порівнянні з 80). З іншого боку, на кожному кроці функції стиснення оновлюються дві змінні, у той час як в SHA-1 тільки одна змінна. Диференційний криптоаналіз показав, що існує 4-крокова характеристика за будь-якими 4-ма кроками у функції стиснення з імовірністю 1. Імовірність диференційних характеристик зменшується в SHA-2 швидше, ніж у SHA-1. Це визначається багаторазовими обертаннями у функціях Σ0 (·) і Σ1 (·). Слайд-атака, що може бути застосована на SHA-1, на SHA-2 не поширюється, тому що в SHA-2 на кожному кроці функції стиснення використовується унікальна адитивна постійна. Потрібна обережність в оцінці криптостійкості SHA-2, оскільки проведений на сьогодні аналіз і дослідження безпечності є недостатніми.

Попередні висновки щодо безпеки застосування функцій гешування

Проведений аналіз і результати, подані в [189, 194, 196, 201], дозволили дійти висновку, що криптографічне застосування алгоритмів гешування здебільшого визначається вимогами захищеності функцій гешування від відомих атак та їх складністю (швидкодією).

Найбільшої уваги заслуговують алгоритми Whirlpool і SHA-2. На сьогоднішній день для цих примітивів гешування не були знайдені ніякі слабкості з точки зору безпечності. Але у зв'язку з тим, що це нові примітиви, вони були піддані тільки обмеженій оцінці з боку фахівців.

Алгоритм Whirlpool має високу стійкість, як і функція гешування з SHA-2/512, але невисоку швидкодію, і, відповідно, його можна рекомендувати до застосування в системах, де необхідно забезпечити стійкість протягом тривалого періоду часу, де критерій стійкості є визначальним і набагато важливішим за швидкодію. Математична простота алгоритму, досягнута в процесі розробки, робить прозорим процес аналізу його стійкості. Довжина геш-значення в 512 бітів, забезпечує ефективний захист від атак, заснованих на парадоксі «день народження», а також поліпшує показники стійкості до колізій. Оптимальна довжина геш-значення, що відповідає сучасним вимогам, дозволила цьому алгоритму стати переможцем у категорії «Алгоритм, стійкий до колізій» у рамках проекту NESSIE, і на сьогодні він має найбільшу перевагу.

Проект SHA-2 ввійшов у новий NIST стандарт функцій гешування (FIPS 180-2) з геш-значеннями 256, 384 і 512 бітів. Головна причина появи нового стандарту полягає в тому, щоб забезпечити рівень захисту, що відповідає рівню захисту нового NIST стандарту блокового шифру AES (із ключовою довжиною 128, 192 або 256 бітів відповідно).

При використанні алгоритму SHA-2/512 забезпечуються такі ж вимоги колізійної стійкості, як і в алгоритмі Whirlpool, з деякою перевагою відносно швидкості обчислень. Аналіз продуктивності алгоритмів показує, що SHA-2/256 виконується вдвічі швидше алгоритмів SHA-2/384 і SHA-2/512 на більшості операційних платформ. Ці примітиви є новими, дещо подібними до SHA-1, але при цьому є важливі розходження в структурі. Приблизно, SHA-2 мають великий рівень безпеки проти відомих нападів. На цей момент не відомі атаки, які спростовували б припущення про колізійну стійкість цього алгоритму.

Стандарти примітивів SHA-1 і RIPEMD-160 не задовольняють вимогам безпеки NESSIE [111], що потрібні для симетричних криптопримітивів. Ці стандарти рекомендуються для застосувань, де цей рівень безпеки в 160 бітів достатній. За довжини геш-значень 160 бітів порядок утворення колізії має складати 2^{80} спроб, а порядок знаходження другого прообразу 2^{160}. На цей час на SHA-1 і RIPEMD-160 не відомі атаки, які спростовували б ці припущення для повного числа циклів. За продуктивністю ці алгоритми практично такі, як і RIPEMD-160, й у 2–3 рази швидші в порівнянні з SHA-2/256.

5.6. СУТНІСТЬ І ВЛАСТИВОСТІ ФУНКЦІЇ ГЕШУВАННЯ ГОСТ 34.311-95 (ГОСТ Р 34.11-94)

У ГОСТ 34.311-95 [195] вироблення геш-значення базується на використанні симетричного блокового шифру ГОСТ 28147-89. Необхідно виділити три основні етапи обчислення геш-значення:

1) обчислення крокової функції гешування;

2) процедура обчислення геш-значення;

3) прикінцеві обчислення.

Розглянемо кожний етап обчислення геш-значень.

5.6.1. Крокова функція гешування

Алгоритм обчислення крокової функції гешування містить у собі трискладові частини, які послідовно реалізують:

– генерацію ключів довжиною 256 бітів;

– перетворення типу зашифрування 64-бітових підслів слова H на ключах K_i ($i = 1, 2, 3, 4$) із використанням алгоритму простої заміни ГОСТ 28147-89;

– перетворення проміжних значень перемішуванням результату зашифрування.

Генерація ключів

Розглянемо бітову послідовність $X = (b_{256}, b_{255}, ..., b_1) \in V_{256}(2)$.

Нехай $X = x_4 \| x_3 \| x_2 \| x_1 = \eta_{16} \| \eta_{15} \| ... \| \eta_1 = \xi_{32} \| \xi_{31} \| ... \| \xi_1$,

де $x_i = (b_{i*64}, ..., b_{(i-1)*64+1}) \in V_{64}(2)$, $i = \overline{1,4}$;

$\eta_j = (b_{j*16}, ..., b_{(j-1)*16+1}) \in V_{16}(2)$, $j = \overline{1,16}$;

$\xi_k = (b_{k*8}, ..., b_{(k-1)*8+1}) \in V_8(2)$, $k = \overline{1,32}$ ...

Позначимо $A(X) = (x_1 \oplus x_2) \| x_4 \| x_3 \| x_2$.

Будемо використовувати перетворення

$P: V_{256}(2) \to V_{256}(2)$

слова $\xi_{32} \| ... \| \xi_1$ у слово $\xi_{\varphi(32)} \| ... \| \xi_{\varphi(1)}$,

де $\varphi(i + 1 + 4(k - 1)) = 8i + k$; $i = 0 \div 3$, $k = 1 \div 8$.

Для генерації ключів потрібно використовувати такі вихідні дані:

– слова $H, M \in V_{256}(2)$;

– параметри слів C_i ($i = 2, 3, 4$), що приймають такі значення:

– $C_2 = C_4 = 0^{256}$ та $C_3 = 1^8 0^8 1^{16} 0^{24} 1^{16} 0^8 (0^8 1^8)^2 1^8 0^8 (0^8 1^8)^4 (1^8 0^8)^4$;

FF00FFFF000000FF0000FF00FFFF0000FF00FF00FF00FFFF00FF00FF00FF0.

Для генерування ключів реалізується такий алгоритм.

1. Присвоїти значення $i := 1$; $U := H$; $V := M$.

2. Виконати обчислення $W = U \oplus V$, $K_1 = P(W)$.

3. Присвоїти $i := i + 1$.

4. Перевірити умову $i = 5$. При позитивному виході перейти до кроку 7. При негативному – перейти до кроку 5.

5. Виконати обчислення: $U := A(U) \oplus C_i$, $V := A(A(V))$, $W := U \oplus V$, $K_i = P(W)$.

6. Перейти до кроку 3.

7. Кінець роботи алгоритму.

Як результат роботи наведеного вище алгоритму є 4 генеровані ключі K_i.

Перетворення, що здійснює шифрування

На даному етапі здійснюється зашифрування 64-бітових підслів слова H на ключах K_i ($i = 1, 2, 3, 4$).

Для перетворення, що здійснює зашифрування, необхідно використовувати такі вихідні дані:

$H = h_4 \| h_3 \| h_2 \| h_1$, $h_i \in V_{64}(2)$, $i = \overline{1,4}$ і набір ключів K_1, K_2, K_3, K_4.

У результаті реалізації алгоритму зашифрування отримують слова

$$s_i = E_{k_i}(h_i), \text{ де } i = 1, 2, 3, 4,$$

які подаються у вигляді:

$$S = s_4 \| s_3 \| s_2 \| s_1.$$

Перетворення, що здійснює перемішування

На цьому етапі здійснюється перемішування отриманої послідовності із застосуванням регістру зсуву.

Вихідними даними є: слова H, M ∈ $V_{256}(2)$ і слово S ∈ $V_{256}(2)$ (регістру зсуву 2).

Нехай відображення $\psi: V_{256}(2) \to V_{256}(2)$

перетворить слово $\eta_{16}\|\eta_{15}\|\ldots\|\eta_1$, $\eta_i \in V_{256}(2)$, $i = \overline{1,16}$

на слово $\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus \eta_4 \oplus \eta_{13} \oplus \eta_{16}\|\eta_{16}\|\eta_{15}\|\ldots\|\eta_2$.

Тоді в якості значення крокової функції гешування приймається слово

$\chi(M, H) = \psi^{61}(H \oplus \psi(M \oplus \psi^{12}(S)))$,

де ψ^i – *i*-й ступінь перетворення ψ.

5.6.2. Процедура обчислення геш-значення

Вихідними даними для процедури обчислення значення функції *h* є послідовність M ∈ B^*, що підлягає гешуванню. Параметром є стартовий вектор гешування H – фіксоване слово з $V_{256}(2)$. Процедура обчислення функції h на кожній ітерації використовує такі розміри:

M ∈ B^* – частина послідовності M, відносно якої не зроблено гешування на попередніх ітераціях;

H ∈ $V_{256}(2)$ – поточне значення геш-функції;

Σ ∈ $V_{256}(2)$ – поточне значення контрольної суми;

L ∈ $V_{256}(2)$ – поточне значення довжини, обробленої на попередніх ітераціях частини M.

Алгоритм обчислення значення функції h містить такі етапи:

1 етап. Присвоїти початкові значення поточних розмірів:

1) M: = M;

2) H: = H;

3) Σ: = 0^{256};

4) L: = 0^{256}.

2 етап. Перевірити умову $|M| > 256$. При позитивному виході перейти до етапу 3. В іншому випадку виконати послідовність обчислень:

1) L: =< $L + |M| > 256$.

2) M′: = $0^{256-|M|}\|M$.

3) Σ: = $\Sigma \oplus' M'$.

4) H: = $\chi(M', H)$.

5) H: = $\chi(L, H)$.

6) H: = $\chi(\Sigma, H)$.

Кінець роботи алгоритму.

3 етап. Обчислити підслово $M_s \in V_{256}(2)$ слова M ($M = M_p\|M_s$). Далі виконати послідовність обчислень:

1) H: = $\chi(M_s, H)$.

2) L: =< $L + 256 > 256$.

3) Σ: = $\Sigma \overline{\oplus'} M_s$.

4) M: = M_p.

5) Перейти до етапу 2.

Значення величини H, отримане на кроці 6 етапу 2, є значенням функції гешування $h(M)$.

Аналіз та оцінка властивостей і характеристик ГОСТ 34.311-95 [195] дозволяє зробити висновки, що він не задовольняє сучасним вимогам з точки зору великої складності обчислень. Також, як випливає із [189, 194], відносно ГОСТ 34.311-95 (ГОСТ 34.11-94) розроблено ряд ефективних атак. Окрім того, цей стандарт дозволяє генерувати геш-значення тільки з довжиною 256 бітів. У той же час уже необхідно генерувати геш-значення і з більшими довжинами, наприклад, як для SHA-2 довжинами 384 та 512 бітів.

Таким чином, як стандарт гешування Російської федерації ГОСТ Р 34.11-94, так і його, по суті, копія – міждержавний стандарт ГОСТ 34.311-95 уже не задовольняють вимогам як щодо стійкості, так і швидкодії.

5.7. ЗАВДАННЯ ТА ПОПЕРЕДНІ ПІДСУМКИ ПРОЕКТУ СТВОРЕННЯ SHA-3

Національним інститутом стандартизації США (NIST) у 2007 році розпочато відкритий конкурс на проект нового федерального стандарту гешування, який отримав назву «NIST SHA-3 Competition» [189, 194, 196, 199–204]. Потреба розробки нового стандарту гешування, що отримав назву SHA-3, обумовлена появою повідомлень про існування атак на функції гешування SHA-1 та MD5, які широко використовуються у світі. Так, атака створення колізій відносно SHA-1 має складність 2^{69} [189, 201]. Дійсно, оскільки довжина вихідного блоку функції гешування SHA-1 складає 160 бітів, то складність знаходження колізії через парадокс «днів народження» оцінюється як 2^{80}, але значення 2^{69} значно менше. Необхідно відзначити, що значення складності 2^{69} на даному рівні розвитку обчислювальної техніки є практично недосяжним, але його наявність довела можливість зменшення складності такого класу атак. Хоча атаку зі складністю 2^{69} поки ще необхідно вважати скоріше теоретичною, а не практичною.

Щодо MD5 необхідно відзначити, що точної оцінки складності атаки на MD5 невідомо, як і алгоритм атаки. Але автори [200–201] стверджують, що час, необхідний для пошуку колізії для MD5 на персональному комп'ютері, приблизно дорівнює 8 годинам, і, отже, така атака практично може здійснюватись. Вказується, як, користуючись цією атакою, можливо побудувати фальшивий центр сертифікації ключів і здійснити таким чином атаку man-in-the-middle на криптографічні протоколи, які використовують сертифікати, підписані з використанням функції гешування MD5. Найбільш застосовуваним з таких протоколів є протокол SSL.

Але все ж таки більш тривожним є той факт, що MD5 і SHA-1 побудовані на спільних принципах і мають схожу архітектуру. Не виключено, що атака на MD5 після певних модифікацій потенційно може бути застосована і проти SHA-1, SHA-256 та SHA-512, які мають схожу внутрішню структуру. Тому щоб запобігти ситуації, коли всі стандартизовані функції гешування можуть бути компрометованими, NIST США у 2007 році оголосив, по суті, міжнародний конкурс на перспективні функції гешування.

Окрім того, з точки зору практики, з'явилося ряд побажань, а скоріше вимог щодо підвищення швидкодії гешування, тобто зменшення складності обчислення функцій гешування. Зазначена вимога особливо актуальна з боку таких додатків, як електронний цифровий підпис, вироблення псевдовипадкових послідовностей,криптографічні протоколи тощо.

5.7.1. Основні умови організації та проведення конкурсу

Особливості організації та проведення конкурсу, на наш погляд, є такими, що повинні використовуватись і в перспективі. У зв'язку з цим розглянемо та проаналізуємо їх дещо детальніше. Аналіз умов проведення конкурсу та практика його проведення показали, що до участі в конкурсі були допущені всі бажаючі [196, 201], які надали у встановлені терміни необхідні дані щодо свого кандидата. По закінченню строку прийому було проведено конференцію, головною метою якої було обґрунтування розробниками обраної методики побудови й аналізу, а також обговорення алгоритмів кандидатів. Оцінювання проводиться у два етапи, на кожному з яких алгоритми кандидатів ретельно досліджуються групою експертів NIST і бажаючими. Обидва етапи є відкритими, і кожен бажаючий може взяти участь у процесі дослідження та обговорення того чи іншого алгоритму. Також усім зацікавленим користувачам надається можливість вільного доступу до специфікацій алгоритмів кандидатів та до звітів досліджень, що формують групи експертів NIST. Організаторами також підтримуються ініціативи щодо вдосконалення методики досліджень та критеріїв оцінки.

Перший етап проводився у вигляді відкритого обговорення, у якому беруть участь усі бажаючі та експерти NIST. До нього були допущені всі кандидати, які були прийняті до участі в конкурсі. На цьому етапі проведено оцінку криптографічних вразливостей та додаткових переваг і недоліків запропонованих функцій гешування. Право визначати тривалість першого етапу залишив за собою NIST, але попередньо відведений час становив 12 місяців. У процесі відкритого обговорення з метою ефективної організації конкурсу заборонялося вносити зміни до алгоритму гешування. По закінченні першого етапу для аналізу результатів виконаної роботи була проведена конференція. Метою конференції було визначення перших п'яти кандидатів, що найбільше відповідають висунутим вимогам.

Попередньо організаторами планувалось, що в другому етапі візьмуть участь п'ять алгоритмів, що будуть обрані за результатами відкритого обговорення. Проте, після додаткового обговорення в подовженому першому етапі NIST, було дозволено брати участь чотирнадцяти алгоритмам [201]. Хоча на кінець 2009 р. залишилося 12 кандидатів. На другому етапі планується провести більш детальний аналіз складових частин і алгоритму кандидата в цілому. Форма його проведення майже не відрізняється від попереднього, термін проведення становить 12–15 місяців і може бути змінений організаторами. Але до його початку було дозволено внести незначні зміни в алгоритм для усунення можливих недоліків, після чого модернізовані версії алгоритмів стали доступними для всіх зацікавлених на офіційній сторінці сайту конкурсу. За результатами другого етапу конкурсу заплановано провести конференцію та оприлюднити

алгоритм, що буде включений до нового стандарту FIPS. Також організатори не відкидають можливості того, що висунутим вимогам задовольнятимуть одразу декілька кандидатів. У такому випадку всі алгоритми, можливо, будуть включені до нового стандарту.

5.7.2. Вимоги до перспективних кандидатів на стандарт гешування

Таким чином, метою проведення конкурсу SHA-3, як уже відзначалося вище, є розробка алгоритму, який зміг би замінити діючий стандарт функцій гешування. В інформаційному листі NIST наведено ряд вимог до кандидатів на новий стандарт [201, 196]. Основною вимогою є підтримка алгоритмом-кандидатом можливості вироблення геш-значень довжиною 224, 256, 384 та 512 бітів. Така вимога необхідна для того, щоб не порушити спадкову послідовність відносно SHA-2 і таким чином домогтися сумісності нового стандарту гешування з уже існуючими, у яких його планується застосовувати. Можливість функції гешування виробляти значення іншої довжини є перевагою, але за умови, що її визнають учасники відкритого обговорення. NIST планує використовувати новий алгоритм гешування в таких застосуваннях [201–204]:

– вироблення та перевіряння електронного цифрового підпису за FIPS 186-2 та FIPS 186-3, Digital Signature Standard;

– обчислення кодів автентифікації повідомлень за FIPS 198, The Keyed-Hash Message Authentication Code (HMAC);

– встановлення ключів згідно SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography;

– генерування псевдовипадкових чисел згідно з NIST SP 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators (DRBGs).

Зважаючи на перелік сервісів, що пропонуються NIST, вимоги до її безпеки мають бути жорсткими. Відповідно до цього узгоджені вимоги щодо безпеки функції гешування з довжиною геш-значення n бітів такі:

– складність знаходження колізії, не менше $2^{n/2}$;

– складність відновлення прообразу, не менше 2^n;

– складність знаходження другого прообразу, не менше 2^n;

– стійкість до атак length extension;

– стійкість до усічених колізій;

– відсутність атак розпізнавання для генераторів псевдовипадкових послідовностей, що використовують HMAC, побудованих на базі функції гешування зі складністю, меншою ніж знаходження другого прообразу, і кількістю запитів до генератора не менше $2^{n/2}$;

– також, як випливає з аналізу, відкритість алгоритмів гешування та можливість мінімізації складності гешування як за апаратної, так і за програмної реалізації.

Для геш-функцій з ітеративною структурою застосовується додатковий критерій – розмір внутрішнього стану має бути щонайменше вдвічі більший за розмір вихідного значення. Причина його введення – це протидія атаці на ітеративну структуру, що наведена у [201–204].

Оцінка кандидатів зроблена способом відкритого обговорення, у якому беруть участь усі бажаючі та експерти NIST. Критерії, за якими буде оцінюватися функція гешування, наведено в інформаційному листі, проте організатори підтримують внесення пропозицій щодо вдосконалення запропонованих критеріїв та впровадження нових. Усі критерії оцінювання можна умовно поділити на критерії стійкості, складності обчислення та характеристик застосування.

Критерії стійкості визначають мінімальний рівень захищеності функції гешування від криптоаналітичних атак. Так, алгоритм-кандидат не повинен мати вразливостей до відомих атак, а також, бажано, і до атак, що можуть бути розроблені в майбутньому. Критерії формуються згідно з вимогами безпеки, які має задовольняти функція гешування.

На відміну від критеріїв стійкості, *критерії складності* обчислення не мають чітких значень. Організаторами запропоновано два з них: ефективність обчислень, яка характеризує швидкодію алгоритму, та об'єм пам'яті, необхідний для обчислення геш-значення, тобто просторова складність. Оцінки за цими критеріями дозволяють визначити вимоги до систем, у яких планується застосовувати алгоритм гешування.

Всього для участі в конкурсі було подано 56 заявок [201]. На першому етапі обговорення вибули 10 кандидатів як такі, що визнані «зламаними». Ще 5 відкликані авторами.

5.7.3. Аналіз підходів щодо побудування перспективних функцій гешування

Аналізуючи специфікації геш-функцій, подані на конкурс, можна виділити такі методи їх побудови:

- на основі ітеративної структури Меркле-Дамгарда (МД) [201–204];
- на основі потокової архітектури;
- метод обчислення за деревом;
- на основі архітектури Беларе.

Ітеративна структура Меркле-Дамгарда (МД) – це класична архітектура, запропонована в роботах.

Потокова архітектура може бути представлена як потоковий шифр, який працює у зворотному напрямку, тобто не генерує гаму, а приймає повідомлення як такої гами на вхід. Геш-значенням у цьому випадку стає внутрішній стан лінійного рекурентного регістра. Цей метод характеризується високою швидкодією. На конкурсі представлено всього три кандидати Streamhash, Boole та Waterfall [201], які вже визнані зламаними. Тому новим і актуальним завданням є детальне вивчення умов і вимог, яким повинні задовольняти функції перетворення.

Обчислення за деревом – це інший, порівняно новий метод щодо побудування функцій гешування, який запропоновано в роботі [203]. Особливістю цього методу є можливість виконання паралельних обчислень, що дозволяє значно підвищити швидкодію алгоритму шляхом ефективного використання сучасних багатоядерних апаратних платформ. За визначенням NIST, така властивість є перевагою і враховується при виборі геш-функції.

Метод, що ґрунтується на використанні архітектури Беларе, запропонований у [202]. Сутність цієї архітектури полягає в розділенні операції стиснення блоку повідомлення й операції комбінування. Причому, всі операції стиснення можуть бути виконані одночасно й паралельно. Операція комбінування являє собою деяку бінарну операцію, як правило, у групі. Складність операції комбінування, у порівнянні з операцією стиснення, значно менше, тому функції гешування з такою архітектурою мають найкращі можливості з використання багатоядерних обчислювальних засобів.

У ході конкурсу були визначені нові цікаві особливості до побудування функцій гешування. Так, більшість нововведень спрямовані на зменшення складності гешування, тобто підвищення швидкодії. Деякі кандидати мають навіть меншу за значенням числа тактів/ байтів складність, ніж навіть у AES.

У таблиці 5.8 наведено типи архітектури алгоритмів, які до кінця першого етапу були визнані незламними [194, 201].

Таблиця 5.8. **Функції гешування та типи їх архітектури**

Назва алгоритму	Автор	Тип архітектури
ARIRANG	Jongin Lim	МД
BLAKE	Jean-Philippe Aumasson	МД
Blue Midnight Wish	Svein Johan Knapskog	МД
CHI	Phillip Hawkes	МД
ECHO	Henri Gilbert	МД
FSB	Matthieu Finiasz	МД
Fugue	Charanjit S. Jutla	МД
Grøstl	Lars R. Knudsen	МД
Hamsi	Özgül Küçük	МД
Keccak	The Keccak Team	МД
LANE	Sebastiaan Indesteege	МД
Lesamnta	Hirotaka Yoshida	МД
Luffa	Dai Watanabe	МД
MD6	Ronald L. Rivest	Дерево
SANDstorm	Rich Schroeppel	Дерево
Shabal	Jean-François Misarsky	МД
SHAvite-3	Orr Dunkelman	МД
SIMD	Gaëtan Leurent	МД
Skein	Bruce Schneier	Дерево
SWIFFTX	Daniele Micciancio	МД

Як видно із даних таблиці 5.8, абсолютну першість за популярністю у розробників посідає класична ітеративна архітектура Меркле-Дамгарда. Причиною такого широкого використання є бажання розробників скористатися перевагами консервативного підходу до побудови, а саме – високим рівнем розвитку механізмів захисту від витончених атак. Проте, суттєвим недоліком такої архітектури є її непристосованість до виконання паралельних обчислень, що не дозволяє підвищувати швидкодію алгоритму без збитків для стійкості.

Також необхідно відзначити, що загалом на конкурс було представлено 6 алгоритмів, які підтримують паралельні обчислення. У таблиці 5.9 наведено перелік кандидатів, що забезпечували паралельні обчислення при гешуванні та типи їхньої архітектури [194, 201].

Таблиця 5.9. **Паралельні алгоритми, що було представлено на конкурс**

Назва алгоритму	Автор	Тип архітектури
ECOH *	Daniel R.L. Brown	Белларе
EnRUPT *	Sean O'Neil	Дерево
ESSENCE *	Jason Worth Martin	Дерево
MD6	Ronald L. Rivest	Дерево
SANDstorm	Rich Schroeppel	Дерево
Skein	Bruce Schneier	Дерево

Примітка

* Зламані алгоритми (на момент закінчення 1-го етапу).

Аналіз показав, що незалежно від принципів побудови швидкість функцій гешування суттєво відрізняється. У таблиці 5.10 подано дані, що відібрані із специфікацій алгоритмів, які ілюструють показники складності (швидкодії) кандидатів, що претендували на участь у другому етапі [194, 201].

Таблиця 5.10. **Заявлена авторами швидкодія функцій гешування**

Назва алгоритму	Автор	Тип архітектури	Швидкодія (тактів / байтів)*	
			32 біти	64 біти
1	2	3	4	5
Arirang	Jongin Lim	МД		
224			21,6	16,1
256			21,5	16,1
384			65,1	12,8
512			65,2	12,9

Продовження табл 5.10

1	2	3	4	5
Blake	Jean-Philippe Aumasson	МД		
224			27,4	18,4
256			27,4	18,4
384			61,3	13,8
512			61,3	13,8
Blue Midnight Wish	Svein Johan Knapskog	МД		
224			29,28	26,28
256			9,01	8,10
384			13,06	4,29
512			13,14	4,27
CHI	Phillip Hawkes	МД		
224			51	26
256			51	26
384			80	18
FSB	Matthieu Finiasz	МД		
224			257	–
256			297	–
384			324	–
512			423	–
Fugue	Charanjit S. Jutla	МД		
224				
256				
384				
512				
Grøstl	Lars R. Knudsen	МД		
224			77,9	27,2
256			77,9	27,2
384			123,4	45,5
512			123,4	45,5
Hamsi	Özgül Küçük	МД		

Продовження табл 5.10

1	2	3	4	5
224			–	–
256			–	–
384			–	–
512			–	–
Keccak	The Keccak Team	МД		
224			–	–
256			–	–
384			–	–
512			–	–
LANE	Sebastiaan Indesteege	МД		
224			40,46	152,24
256			40,46	152,24
384			26,17	145,31
512			26,17	145,31
Lesamnta	Hirotaka Yoshida	МД		
224			68,9	78,4
256			68,9	78,4
384			97,7	65,4
512			97,7	65,4
Luffa	Dai Watanabe	МД		
224			–	–
256			–	–
384			–	–
512			–	–
MD6	Ronald L. Rivest	Дерево		
224			63	26
256			68	28
384			87	36
512			106	44
SANDstorm	Rich Schroeppel	Усічене дерево		
224			–	
256			–	

Закінчення табл 5.10

1	2	3	4	5
384			–	
512			–	
Shabal	Jean-François Misarsky	МД		
224			–	
256			–	
384			–	
512			–	
SHAvite-3	Orr Dunkelman	МД		
224			35,3	
256			35,3	
384			58,4	
512			58,4	
SIMD	Gaëtan Leurent	МД		
224			90	
256			90	
384			118	
512			118	
Skein	Bruce Schneier	Дерево		
224			38,1	
256			38,1	
384			44,3	
512			44,3	
Swifftx	Daniele Micciancio	МД		
224			–	
256			–	
384			–	
512			–	

Примітка

* На платформі, що заявлена NIST: Intel Core 2 Duo Processor, 2.4GHz clock speed, 2GB RAM, running Windows Vista Ultimate 32-bit (x86) and 64-bit (x64) Edition, наведені дані, взяті із специфікацій алгоритмів, наданих авторами.

Стосовно алгоритмів Echo, Fugue, Hamsi, Кессак, Luffa, SANDstorm, Shabal та Swifftx атак отримати не вдалося.

Наприкінці першого етапу серед кандидатів залишилась 21 функція гешування, відносно яких ще не було знайдено жодних атак [205]. Детально з аналізом кандидатів можна ознайомитися в роботі [204].

Однак експерти NIST визначили 14 кандидатів, що можуть бути представлені у другому етапі змагання за умови внесення певних змін, які посилюють стійкість, у специфікації. Всупереч очікуванням серед претендентів на місце нового стандарту не виявилось алгоритму MD6, який показав необхідний рівень стійкості, високу гнучкість і здатність до паралельного обчислення геш-значень. Вочевидь, це зумовлено тим, що його функція стискання побудована за тими самими принципами, що й MD-x сімейство, на яке було знайдено ряд атак.

У таблиці 5.11 подані кандидати, що пройшли до другого етапу та їх характеристики щодо складності гешування (швидкодії) [201].

Таблиця 5.11. **Геш-функції, що пройшли до другого етапу**

Алгоритм	Швидкодія (тактів/байтів)*
Blake	30.8633
BlueMindightWish	18.082
CubeHash	3657.95
Echo	72.1367
Fugue	122.855
Grostl	106.777
Hamsi	246.703
JH	89.1289
Keccak	73.4688
Luffa	38.6367
Shabal	19.7969
Shavite	59.5078
SIMD	149.445
Skein	46.2891

Примітка

* Тестова платформа: Core 2 Duo 2.66 GHz, RAM 2.0 Gb, OS Win XP Prof. (32 bit). Compiler-MVS 2005.

5.8. СУТНІСТЬ І РЕЗУЛЬТАТИ 2 ЕТАПУ КОНКУРСУ SHA-3

Попередній аналіз [201–205] стану вирішення вказаної проблеми дозволив зробити висновок, що відповідного методичного та математичного апарату оцінювання функцій гешування немає. Тому розробка науково-методичного й математичного забезпечень порівняння функцій гешування з метою вибору кращої із запропонованих кандидатів є на цей момент актуальним завданням. Процедура порівняння ускладнюється тим, що розробники пропонують різні підходи до створення функцій гешування [205–208]. Так, наприклад, одним із

сучасних підходів до побудування функцій гешування є створення архітектурних моделей, у яких з метою збільшення швидкодії використовуються паралельні обчислення. Далі такі функції гешування називатимемо паралельними. Надалі, зважаючи на сучасні тенденції розвитку, необхідно вдосконалити вже існуючі системи критеріїв і показників і розробити методики порівняння функцій гешування з урахуванням нових пропозицій щодо особливостей їх архітектури. Особливу увагу необхідно приділити такому показнику, як складність (швидкодія). Наприклад, для паралельної функції гешування значення швидкодії на процесорах з декількома обчислювальними ядрами відрізняється від значення швидкодії на процесорах з одним обчислювальним ядром. Окрім того, є суттєва різниця у виконанні паралельних обчислень відносно апаратних, програмних та апаратно-програмних реалізацій. Так, у програмних реалізаціях потоки виконання по своїй суті є конкуруючими, тому необхідні додаткові витрати на синхронізацію тощо.

Нижче викладено результати щодо основних вимог, критеріїв і показників оцінки, обґрунтування та розроблення методики оцінки перспективних функцій гешування і, як наслідок, можливості її застосування для оцінювання в тому числі перспективних функцій гешування міжнародного проекту SHA-3 [201–207].

5.8.1. Аналіз вимог до функцій гешування

У подальшому будемо вважати, що функція гешування дозволяє обчислювати вихідне значення з довжиною *hlen* при довільній довжині l_M вхідного повідомлення M.

Аналіз існуючих вимог [206–208] показав, що сучасні й перспективні функції гешування повинні неодмінно відповідати таким вимогам:

1) складність знаходження колізії

$$C_{col} \geq 2^{hlen/2}; \tag{5.6}$$

2) складність відновлення прообразу M

$$C_{preim} \geq 2^{hlen}; \tag{5.7}$$

3)складність знаходження іншого прообразу

$$C_{sec\_preim} \geq 2^{hlen}; \tag{5.8}$$

4)складність знаходження колізії, усіченої на ht символів

$$C_{tr\_col} \geq 2^{(hlen-ht)/2}. \tag{5.9}$$

Наведені вище вимоги, по суті, є безумовними критеріями оцінки функцій гешування.

По суті [196, 201], розглянуті критерії безпеки функції гешування є булевими змінними, які приймають значення «1» (істина), якщо відповідна умова виконується, та «0» в іншому випадку. При цьому стійкість j-ї функції гешування можливо описати вектором булевих змінних $(w_1^{(j)}, w_2^{(j)}, w_3^{(j)}, w_4^{(j)})$, які формально можуть бути визначені як:

$$
\begin{aligned}
w_1^{(j)} &= \begin{cases} 1, C_{col} \geq 2^{hlen/2} \\ 0 \end{cases} \\
w_2^{(j)} &= \begin{cases} 1, C_{preim} \geq 2^{hlen} \\ 0 \end{cases} \\
w_3^{(j)} &= \begin{cases} 1, C_{sec\_preim} \geq 2^{hlen} \\ 0 \end{cases} \\
w_4^{(j)} &= \begin{cases} 1, C_{tr\_col} \geq 2^{(hlen-ht)/2} \\ 0 \end{cases}
\end{aligned}
\qquad (5.10)
$$

Вектор $(w_1^{(j)}, w_2^{(j)}, w_3^{(j)}, w_4^{(j)})$ будемо з деяким уточненням називати інтегральним безумовним критерієм.

Щодо такого показника, як складність (швидкодія) гешування, то він вже не є однозначним. Тому необхідно визначити й обґрунтувати нові показники та відповідні умовні критерії.

Зважаючи на те [196, 201], що обчислення функції гешування може здійснюватись паралельно, розглянемо загальні питання паралельних обчислень і вимоги до них.

Так, згідно із законом Амдала [189, 205], коефіцієнт прискорення K_N при здійсненні N паралельних обчислень визначається за формулою:

$$K_N = \frac{1}{(1-P)+\frac{P}{N}}, \qquad (5.11)$$

де P – частка паралельних обчислень, $(1-P)$ – відповідно частка послідовних обчислень.

Фізичний зміст цього коефіцієнта прискорення K_N можна трактувати як відношення нормованої швидкодії алгоритму з використанням паралельних обчислень при його виконанні на N обчислювальних ядрах до швидкодії цього ж алгоритму при його виконанні на одному ядрі.

У той же час визначити частку паралельних обчислень в алгоритмі, базуючись на його специфікації, практично неможливо. Але розрахувати коефіцієнт прискорення K_N при використанні відповідної кількості обчислювальних ядер можливо, використовуючи результати вимірювань швидкодії функції гешування на одному та кількох обчислювальних ядрах

$$K_N = \frac{S_N}{S_1}, \qquad (5.12)$$

де S_1 – швидкодія алгоритму гешування на одному обчислювальному ядрі, а S_N – швидкодія алгоритму гешування на N обчислювальних ядрах.

Використовуючи (5.12), можна отримати таку формулу (5.13), що дозволяє обчислити частку паралельних обчислень, знаючи коефіцієнт прискорення на N обчислювальних ядрах і кількість цих ядер. Формула має вигляд:

$$P = \frac{(K_N - 1)N}{(N-1)K_N}, \tag{5.13}$$

де, як і раніше, K_N – коефіцієнт прискорення на N обчислювальних ядрах, P – частка паралельних обчислень.

Безпосередній аналіз згідно (5.11) показав, що нарощування кількості обчислювальних ядер має сенс до певної величини. Причому кожне нове додане обчислювальне ядро дає менший приріст швидкодії, ніж попереднє. Коефіцієнт прискорення K_{max} при цьому обмежений зверху значенням $\frac{1}{1-P}$.

Дійсно, якщо $N \to \infty$, то

$$K_{max} = \lim_{N\to\infty}\left(\frac{1}{1-P+P/N}\right) = \frac{1}{1-P}. \tag{5.14}$$

Але на практиці вже доведено, що максимальне значення коефіцієнта прискорення є недосяжним, тому з (5.14) у загальному випадку неможливо обчислити максимальну кількість ядер, що можуть ефективно використовуватися алгоритмом. Задача може бути вирішена таким чином.

Обчислення максимально можливого прискорення можна зробити, якщо замість K_{max} використати деяке $K'_{max} < K_{max}$. При цьому наскільки K'_{max} близьке до K_{max} можна визначити, використовуючи значення коефіцієнта наближення r, що близький до одиниці, але менше за неї:

$$K'_{max} = rK_{max}. \tag{5.15}$$

Для практичних розрахунків рекомендується використати значення $r = 0{,}95$. Вибір такого значення r можна пояснити тим фактом, що збільшення кількості обчислювальних ядер після досягнення K'_{max} практично не впливає на швидкодію. Тому саме це значення доцільно використовувати для обчислення інших показників.

5.8.2. Критерії та показники оцінки функцій гешування

По аналогії з [99] усі критерії, які можуть бути застосовані до геш-функцій, поділимо на два класи: безумовні та умовні. Функції гешування, що оцінюються, повинні обов'язково відповідати всім безумовним. Використовуючи умовні критерії, можна оцінити якість гешування.

Як правило, до безумовних критеріїв належать усі критерії, пов'язані зі стійкістю та вимогами, що повинні бути виконані безумовно. Це, по суті, часткові безумовні критерії (5.6)–(5.9). Згідно з (5.10), інтегральний критерій можна застосовувати таким чином: функція гешування вважається прийнятною, якщо відповідні часткові показники приймають значення «істина», тобто функція гешування є стійкою проти атак типу (5.6)–(5.9).

Як умовні показники, що достатньою мірою характеризують якість функції гешування, можна використати максимально досяжну швидкодію і максимальну кількість обчислювальних ядер, які можуть бути ефективно задіяні в паралельних обчисленнях. При цьому під максимально досяжною швидкодією розуміємо швидкодію алгоритму обчислення функції гешування, яка може бути

принципово досягнута при оптимізованій реалізації у випадку використання паралельних обчислень і відповідно використання максимальної кількості обчислювальних ядер.

Відповідний критерій формулюємо таким чином. У парі функцій гешування перевага надається тій функції гешування, яка має найбільш максимально досяжну швидкодію. Під максимальною кількістю обчислювальних ядер, які можуть бути ефективно використані алгоритмом, розуміємо таку кількість обчислювальних ядер, перевищення якої не принесе значного приросту швидкодії при використанні паралельних обчислень в алгоритмі.

Умовний критерій формулюємо таким чином. У парі функцій гешування, що використовують паралельні обчислення і мають однакову максимально досяжну швидкодію, перевага надається тій функції гешування, що має найменше значення максимальної кількості обчислювальних ядер. Обґрунтування цього критерію робиться на основі того, що паралельні обчислення використовуються для підвищення швидкодії, а отже, за однакової швидкодії функція гешування, що використовує меншу кількість обчислювальних ядер, економічно більш вигідна і може мати простішу реалізацію.

Основні співвідношення для обчислення максимально досяжної швидкодії вже було наведено вище. Підставивши формулу (5.14) у (5.15), отримаємо уточнену формулу для обчислення максимально досяжного коефіцієнта прискорення:

$$K'_{\max} = r\frac{1}{1-P}, \tag{5.16}$$

де, як і раніше, r – коефіцієнт наближення, P – частка паралельних обчислень.

З (5.16) та (5.14) випливає, що максимально досяжна швидкодія може бути обчислена як

$$S_{\max} = S_1 r\frac{1}{1-P}, \tag{5.17}$$

де S_1 – швидкодія алгоритму гешування на одному обчислювальному ядрі функції гешування.

Із формули (5.13) отримуємо вираз для визначення максимального значення коефіцієнта прискорення:

$$N_{\max} = \frac{P}{\frac{1}{K'_{\max}} - 1 + P}, \tag{5.18}$$

де $K'_{\max}$ – значення максимального коефіцієнта прискорення, розраховане за формулою (5.16).

Слід зауважити, що внаслідок присутності залежностей між даними в алгоритмах максимальна кількість обчислювальних ядер і максимальна досяжна швидкодія можуть в деяких випадках відрізнятися від обчисленої.

5.8.3. Методики порівняння функцій гешування за критерієм складності

Огляд доступних літературних джерел проекту SHA-3 дозволив виділити щонайменше три підходи до реалізації паралельних обчислень функцій гешування. Вони ґрунтуються на методах, в основу яких покладено:

1) використання класичної архітектури Меркле-Дамгарда (Merkle-Damgard) [207] з паралельним виконанням частини операцій всередині ітерації;

2) використання архітектури Міхіра Белларе [208] з незалежним перетворенням вхідних блоків повідомлення і комбінуванням результатів цього перетворення;

3) гешування за деревом [209].

Ітеративна архітектура Меркле-Дамгарда є класичною архітектурою реалізації функції гешування. У ній вхідне повідомлення M розбивається на блоки певної довжини $M = M_0 \| \ldots \| M_i \| \ldots \| M_n$, після чого виконується ітеративне обчислення за такими формулами [3]:

$$\begin{aligned} h_0 &= f(M_0, IV); \\ h_i &= f(M_i, V_{i-1}), \end{aligned} \tag{5.19}$$

де IV – початковий вектор (ключ) гешування.

Обчислене h_n значення для останнього блока повідомлення M_n і є геш-значенням повідомлення.

Із (5.19) видно, що наведена вище архітектура є послідовною, тому програмні реалізації функцій гешування з такою архітектурою не здатні ефективно використовувати обчислювальні потужності сучасних багатоядерних процесорів.

Проведемо більш детальний аналіз класичної архітектури Меркле-Дамгарда. Пропонується використовувати паралельні обчислення всередині ітерації [207–209]. Проте, в такому разі обчислення потребують або спеціальних інструкцій процесора (таких як MMX, SSE та інші), або чіткої синхронізації між потоками в програмних реалізаціях. Указане вимагає деяких додаткових потужностей, унаслідок чого реальний виграш при здійсненні паралельних обчислень геш-значень завжди менший за очікуваний. Окрім того, для даного алгоритму, як правило, наявні залежності даних, які необхідно враховувати при обчисленнях. Залежність даних полягає в тому, що незважаючи на велику частку паралельних обчислень в алгоритмі, кількість обчислювальних ядер, що можуть бути ефективно використані для паралельних обчислень, менша, ніж обчислена за законом Амдала, і визначається особливостями внутрішньої структури ітерації алгоритму.

За вказаних умов для методу, що ґрунтується на архітектурі Меркле-Дамгарда, доречно застосувати нижченаведену методику обчислення пари показників $\{S_{\max}, N_{\max}\}$.

Методика 1. **Обчислення пари показників** $\{S_{\max}, N_{\max}\}$

1. Виміряти швидкодію S_1, S_2 для алгоритмів з архітектурою Меркле-Дамгарда, використовуючи два обчислювальні ядра. Значення кількості обчислювальних ядер для вимірювань визначаються тим фактом, що до початку

вимірювання значення максимальної кількості обчислювальних ядер невідоме, а перевищення цього значення значно погіршить точність оцінювання. При цьому, для функцій гешування, у яких не використовуються паралельні обчислення, значення S_1, S_2 практично співпадатимуть, а значення K'_{max} та N_{max} дорівнюватимуть одиниці.

2. Обчислити коефіцієнт прискорення K_2 за формулою (5.12).

3. За формулою (5.13) обчислити частку паралельних обчислень в алгоритмі.

4. Шляхом експертного аналізу специфікації алгоритму визначити наявність залежностей даних і максимальну кількість обчислювальних ядер, що можуть бути ефективно використані алгоритмом.

5. За формулами (5.16) та (5.18) обчислити максимальну кількість обчислювальних ядер, які можуть ефективно використовуватись для обчислення значення функції гешування.

6. Порівняти значення, отримані на кроках 4 та 5, і для подальших розрахунків використовувати найменше з них.

7. На основі формул (5.11) та (5.12) обчислити максимально досяжну швидкодію S_{max}:

$$S_{max} = \frac{S_1}{(1-P) + \frac{P}{N_{max}}}. \qquad (5.20)$$

Таким чином, наведена методика дозволяє отримати пару значень показників $\{S_{max}, N_{max}\}$ для функції гешування з архітектурою Меркле-Дамгарда з паралельними обчисленнями всередині ітерації.

Метод гешування, що ґрунтується на архітектурі Міхіра Белларе повністю орієнтований на використання паралельних обчислень. Схема цієї архітектури показана на рисунку 5.7.

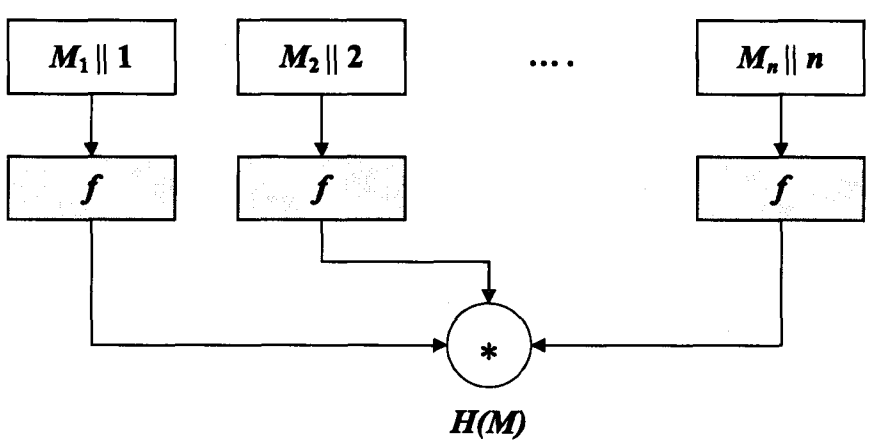


Рис. 5.7. Архітектура функції гешування Міхіра Белларе

Геш-функція, що реалізується з використанням архітектури Міхіра Белларе має дві складові – функцію стискання f та операцію комбінування *. Як видно на рис. 5.5, кожний блок даних обробляється незалежно, тому існує можливість здійснення їх паралельної обробки. Якщо комбінуюча операція є комутативною, то результати обробки блоків даних функцією f можуть бути подані на її вхід у довільному порядку. З урахуванням того, що в такій архітектурі відсутні залежності даних, обчислення значень показників функцій гешування з такою архітектурою можна зробити згідно такої методики.

Методика 2. **Обчислення пари показників $\{S_{\max}, N_{\max}\}$ для архітектури Міхіра Белларе**

1. Виміряти швидкодію алгоритму S_1 та S_2 для архітектури Міхіра Белларе, використовуючи два ядра.
2. Обчислити коефіцієнт прискорення K_2 за формулою (5.12).
3. За формулою (5.13) обчислити частку паралельних обчислень в алгоритмі.
4. За формулою (5.20) обчислити максимально досяжну швидкодію $S_{\max}$.
5. За формулами (5.16) та (5.18) обчислити максимальну кількість обчислювальних ядер, що можуть ефективно використовуватись для обчислення геш-значення.

У результаті виконання обчислень для функцій гешування згідно з методикою 2 отримуємо пару значень показників $\{S_{\max}, N_{\max}\}$.

Як уже зазначалося, паралельні обчислення геш-значення можна також реалізувати на основі метода, що ґрунтується на архітектурі типу «дерево». Цей підхід було запропоновано в роботі [209]. Основою реалізації паралельних обчислень при застосуванні цього методу є приведення структури алгоритму гешування до структури двійкового дерева порядку t (рис. 5.8).

Кількість обчислювальних ядер у дереві порядку t визначається виразом 2^t. Між собою обчислювальні ядра з'єднані таким чином, що дані з виходу обчислювальних ядер P_{2i} та P_{2i+1} подаються на вхід обчислювального ядра P_i.

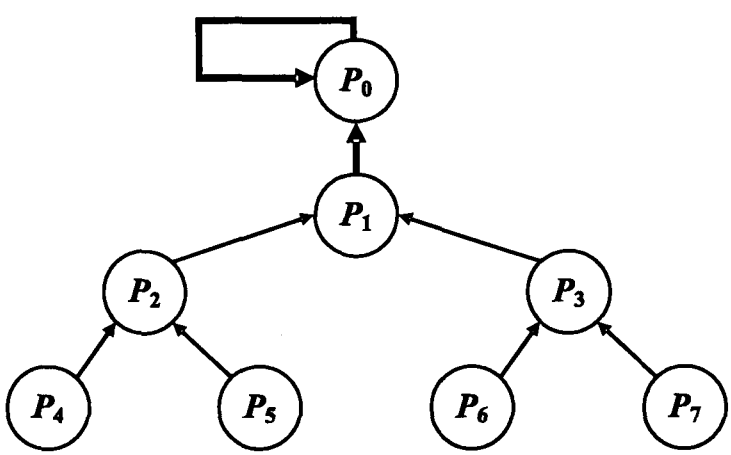


Рис. 5.8. Двійкове дерево порядку $t = 3$

Кожне обчислювальне ядро реалізує стійку проти колізій функцію стискання f, яка приймає на вхід бітовий рядок довжиною n і повертає бітовий рядок довжиною m, причому $n \geq 2m$. У роботі [209] представлено інкрементний алгоритм, що реалізує двійкове дерево порядку t на $2^{t'}$ обчислювальних ядрах, де $0 \leq t' \leq t-1$ з відповідним зростанням часу роботи алгоритму. Тобто якщо реалізувати цей алгоритм для дерева порядку t на 2^{t-1} обчислювальних ядрах, то час виконання збільшиться у два рази порівняно з реалізацією на 2^t обчислювальних ядрах. Для $t=0$ ця архітектура перетворюється на послідовну архітектуру Меркле-Дамгарда. Тому відносно алгоритму обчислення геш-значення із застосуванням двійкового дерева максимальне число обчислювальних ядер $N_{\max}$, що можуть бути ефективно використані для обчислення, дорівнює 2^t.

Значення показника максимально досяжної швидкодії для такого алгоритму можна виміряти безпосередньо на $N_{\max}$ обчислювальних ядрах, якщо доступна така конфігурація ЕОМ, або обчислити з результатів вимірювання швидкодії реалізації інкрементного алгоритму на меншому числі обчислювальних ядер з відповідним перерахунком до $N_{\max}$.

Коефіцієнт прискорення K для дерева визначається як

$$K = \frac{\Psi}{R}, \tag{5.21}$$

де Ψ – кількість звернень до функції стискання, R – загальна кількість операцій, що виконуються паралельно. Число звертань для дерева можна визначити, використовуючи [209]:

$$\Psi = (q+2)2^t + 2\left\lceil \frac{r}{2n-2m} \right\rceil - 1, \tag{5.22}$$

де n – довжина блоку в бітах на вході функції стискання, q – кількість повних блоків повідомлення, які подаються на вхід дерева, t – порядок дерева, r – довжина в бітах останнього неповного блоку, m – довжина виходу функції стискання f у бітах. Значення R обчислюється за формулою:

$$R = q + t + 2, \tag{5.23}$$

де q – кількість повних блоків повідомлення, які подаються на вхід дерева, t – порядок дерева, а величина K обчислюється за формулою:

$$K = \frac{(q+2)2^t + 2\left\lceil \frac{r}{2n-2m} \right\rceil - 1}{q+t+2}, \tag{5.24}$$

Як видно з (5.22), зі зростанням q коефіцієнт прискорення наближається до 2^t. Дійсно, значення для межі має вигляд:

$$\lim_{q \to \infty} \frac{(q+2)2^t + 2\left\lceil \frac{r}{2n-2m} \right\rceil - 1}{q+t+2} = 2^t. \tag{5.25}$$

Для практичних обчислень у [5] пропонується використовувати наближену формулу:

$$K \approx 2^t \left(\frac{1}{1+t/(q+2)} \right), \tag{5.26}$$

де q – кількість блоків повідомлення, які подаються на вхід дерева, t – порядок двійкового дерева.

Аналіз виразу (5.26) показує, що високий порядок двійкового дерева забезпечить збільшення швидкості обчислень тільки для «довгих» повідомлень. Причому, як і у випадку з архітектурою Міхіра Белларе, максимальне значення коефіцієнта прискорення 2^t є недосяжним. Тому для дерева при визначенні коефіцієнта максимального прискорення K_{max} необхідно вважати, що повідомлення є достатньо довгим, але в разі коли коефіцієнт прискорення досягає $2^t r$.

Число повних блоків q повідомлення, що подаються на вхід дерева, необхідне для досягнення максимального прискорення, визначається з наведеного наближення для визначення коефіцієнта швидкодії (5.26).

Необхідно зважати на те, що теоретичної межі прискорення досягнути неможливо, тому при практичному визначенні коефіцієнта максимальної швидкодії пропонується використовувати коефіцієнт наближення $r = 0{,}95$ і відповідно використовувати формулу:

$$K'_{max} = 0{,}95 \cdot 2^t, \tag{5.27}$$

де t – порядок двійкового дерева, а K'_{max} – максимальний коефіцієнт прискорення.

Підставляючи замість K у (5.26) значення K'_{max} з (5.27) визначимо кількість блоків повідомлення q, необхідних для досягнення максимальної швидкодії:

$$q = \frac{t}{\left(\frac{1}{0{,}95} - 1 \right)} - 2, \tag{5.28}$$

де, як і раніше, q – кількість повних блоків повідомлення, t – порядок двійкового дерева.

Практично для розрахунків показників функцій гешування з подібною архітектурою можливо використати співвідношення:

$$\begin{aligned} N_{max} &= 2^t, \\ S_{max} &= K'_{max} S_1. \end{aligned} \tag{5.29}$$

5.8.4. Прийняття рішень при порівнянні функцій гешування

Раніше зазначалося, що кожна функція гешування повинна задовольняти всім безумовним критеріям. Тому, як запропоновано в [99], узагальнений безумовний критерій W треба записувати за допомогою булевої операції кон'юнкції:

$$W = w_1 \wedge w_2 \wedge w_3 \wedge w_4. \tag{5.30}$$

У той же час наявність двох умовних критеріїв вигляду (5.29) обумовлює складність процедури неформального порівняння й вибору. Їх неможливо об'єднати між собою і з узагальненим безумовним критерієм за допомогою булевих

операцій. Для цього випадку пропонується використати деяку узагальнюючу цільову функцію $D(W, S_{\max}, N_{\max})$, максимальне значення якої відповідатиме «найкращій» функції гешування. Наведемо основні властивості цієї функції.

Для всіх можливих значень $S_{\max}, N_{\max}$ виконується умова

$$D(0, u_1, u_2) = 0. \tag{5.31}$$

Для всіх можливих значень $N_{\max}$ та $\Delta S > 0$ виконується нерівність

$$D(1, S_{\max} + \Delta S, N_{\max}) > D(1, S_{\max}, N_{\max}). \tag{5.32}$$

Для всіх можливих значень $S_{\max}$ та $\Delta N > 0$ виконується нерівність

$$D(1, S_{\max}, N_{\max} + \Delta N) < D(1, S_{\max}, N_{\max}). \tag{5.33}$$

Для побудування такої цільової функції можна скористатися методами підтримки прийняття рішень, наприклад методом аналізу ієрархій. Співвідношення (5.31), (5.32) та (5.33) можуть бути використані для перевірки адекватності побудованої цільової функції D.

Таким чином, запропоновані критерії оцінки швидкодії алгоритму гешування дозволяють дати більш повну й адекватну оцінку переваг однієї функції гешування над іншою, та здатні враховувати зміни в підходах до побудування функцій гешування, перш за все зважаючи на сучасні тенденції розвитку комп'ютерної техніки. Методики порівняння функцій гешування враховують архітектурні особливості розглянутих методів гешування. Результати порівняння функцій гешування за сукупністю критеріїв можуть бути легко формалізовані, що дозволить застосувати методи оптимізації для підтримки прийняття рішень. Такий підхід максимально спрощує та дає можливість автоматизувати процес оцінки й порівняння функцій гешування.

Ми розуміємо, що наведені вище результати є попередніми. У наступні 2–3 роки безумовно будуть розроблені та прийняті скоріше декілька стандартів гешування. У цілому, успішне виконання міжнародного проекту «NIST SHA-3 Competition» суттєво вплине на процеси розвитку та прийняття національних і регіональних стандартів щодо функцій гешування. Безумовним є також той факт, що успішне виконання проекту «NIST SHA-3 Competition» значно вплине на наші погляди та застосування різноманітних криптографічних перетворень, криптографічних механізмів і протоколів.