результате оказывается более эффективным на коротких сообщениях, и это дает дополнительную гибкость для верификации. Можно выбирать, сколько из параллельных вычислений использовать в MAC коде, учитывая временные затраты и уровень гарантий.

IV Анализ безопасности UMAC алгоритма

UMAC код аутентификации сообщения основан на семействах универсальных хэш-функций и предлагает доказуемую безопасность в том смысле, что имеются теоретические границы коллизии для хешируемой части при вычислении MAC кода, так что безопасность в конечном счете зависит от криптографического примитива, используемого для зашифрования показателя новизны.

Примитив, заявленный в спецификации UMAC кода - AES (Rijndael) блочный шифр, является достаточно надежным. Имеется дополнительное преимущество, которое состоит в том, что шифрование выполняется на коротком показателе новизны. Не были найдены недостатки в доказательстве безопасности UMAC [2]. Для уровня, использующего NH универсальное семейство хэш-функции, первое положение доказательства безопасности состоит в том, что NH является 2^{-W}– почти универсальным. Это означает, что вероятность коллизии не больше, чем 2^{-W} для строк равной длины и слов длиной в w бит. Это соответствует использованию NH на одном блоке установленной длины сообщения. Второе положение в доказательстве безопасности позволяет расширить этот результат на NH алгоритм, работающий на любом числе строк, в соответствии с параметрами в спецификации UMAC. Вероятность коллизии после NH уровня в UHASH16 схеме равна 2^{-15} , что больше чем 2^{-16} из-за знаковой арифметики.

UHASH16 и UHASH32 имеют два дополнительных уровня хеширования, использующие RP и IP универсальные семейства хэш-функций, и они повторяют схему с тремя слоями четыре раза и соответственно два раза. Доказано, что семейство UHASH16 является 4-связанным ($2^{-15}+2^{-18}+2^{-28}$) почти универсальным, а семейство UHASH32 является 2-связанным ($2^{-31}+2^{-33}$) почти универсальным, что гарантирует вероятность коллизии 2^{-60}.

V Выводы

UMAC (1999, 2000) реализации имеют самую высокую скорость вычислений для MAC примитивов, представленных проекту NESSIE. Алгоритм UMAC основан на применения композиционной схемы с многократным универсальным хешированием и криптографическим вычислением тега аутентификации, гарантирует вероятность коллизии 2^{-60} и по итоговым показателям является одним из лучших.

*Литература: **1**. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P. UMAC: Fast and secure message authentication. In Advances in Cryptology 'CRYPTO '99 (1999), vol. 1666 of Lecture Notes in Computer Science, Springer-Verlag, pp. 216-233. **2.** T. Krovetz, J. Black, S. Halevi, A. Hevia, H. Krawczyk, and P. Rogaway, "UMAC." Primitive submitted to NESSIE, Sept. 2000. [p. 5, 157, 160] **3.** B. Preneel, "Cryptographic primitives for information authentication state of the art." in State of the Art in Applied Cryptography (B. Preneel and V. Rijmen, eds.), no. 1528 in Lecture Notes in Computer Science, pp. 50-105, Springer-Verlag, 1998. **4.** Halevi, S., and Krawczyk, H. MMH: Software message authentication in the Gbit/second rates. In Proceedings of the 4th Workshop on Fast Software Encryption (1997), vol. 1267, Springer-Verlag, pp. 172-189. **5.** Carter, L., and Wegman, M. Universal classes of hash functions. J. of Computerand System Sciences 18 (1979), 143-154. **6.** Krovetz, T., and Rogaway, P. Variationally universal hashing. In preparation, 2000. **7.** Rogaway, P. Bucket hashing and its application to fast message authentication. In Advances in Cryptology 'CRYPTO '95 (1995), vol. 963 of Lecture Notes in Computer Science, Springer-Verlag, pp. 313-328. **8.** Bosselaers, A., Govaerts, R., and Vandewalle, J. Fast hashing on the Pentium. In Advances in Cryptology 'CRYPTO '96 (1996), vol. 1109 of Lecture Notes in Computer Science, Springer-Verlag, pp. 298-312. Updated timing at http://www.esat.kuleuven.ac.be/ bosselae/fast.html.*

УДК 004.056 : 004.424.47

ПІДВИЩЕННЯ СТІЙКОСТІ ФУНКЦІЙ ХЕШУВАННЯ В СХЕМАХ АВТЕНТИФІКАЦІЇ

Олександр Іщенко, Юрій Яремчук

Вінницький національний технічний університет

Анотація: **Розглянуто проблеми, пов'язані з використанням функцій хешування в схемах автентифікації. Запропоновано метод усунення атаки на подовження/скорочення повідомлення.**

Summary: **Problems of using hash-function at authentification were considered. Elimination method of message lengthening/abridge attack was offered.**
Ключові слова: **Інформаційна безпека, функції хешування, загрози.**

I Вступ

Шифрування інформації забезпечує захист від атак пасивної форми (розкриття змісту повідомлення). Іншою проблемою є захист від активних атак (фальсифікація даних і транзакцій). Захист від таких атак забезпечується автентифікацією повідомлень.

Одним із методів автентифікації є генерування за допомогою деякого секретного ключа невеликого блоку даних та приєднання його до повідомлення. Такий блок даних зазвичай називають криптографічною контрольною сумою або кодом автентичності повідомлення (Message Authentification Code – MAC). При цьому розуміється, що дві сторони, які приймають участь в обміні даними, наприклад, А і Б, використовують загальний секретний ключ $К_{АБ}$.

Для генерування кодів автентичності існує низка алгоритмів. Одним з варіантів використання кодів автентичності повідомлення, що в останній час привертає до себе велику увагу, є однобічна функція хешування. Вимоги, що висуваються до таких функцій, наведені в [1].

В одних випадках метою зловмисника є пошук прообразу хеш-значення (для заданого x знайти таке m, що $h(m) = x$) або виявлення деяких структурних закономірностей у вихідних даних функції хешування. Таку атаку визначають як атаку на прообраз. Дана атака потребує виконання приблизно 2^n кроків.

В більшості випадків універсальною атакою на функцію хешування є атака, в основу якої покладено парадокс задачі про дні народження. Така атака спрямована на знаходження колізії. Для функції хешування з n-бітовим вихідним значенням появу колізії слід очікувати приблизно через $2^{n/2}$ кроків [2]. Саме такий підхід найчастіше застосовують для здійснення атак на системи автентифікації.

II Постановка задачі

В загальному вигляді схема автентифікації може мати вигляд, представлений на рис 1. Користувач А відсилає повідомлення користувачеві Б. Після цього він здійснює автентифікацію повідомлення, надсилаючи значення $h(К_{АБ} \parallel m)$, де $К_{АБ}$ – секретний ключ, відомий тільки користувачам А і Б, а m – повідомлення.

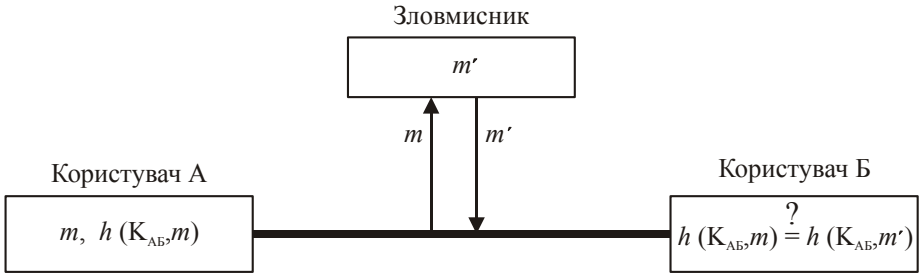

Рисунок 1 – Загальна схема автентифікації

Аналізуючи наведену схему автентифікації слід відзначити суттєвий недолік, що полягає в можливості подовження/скорочення повідомлення. Суть проблеми – деяке повідомлення m розбивається на блоки m_1, ... , m_n та хешується, в результаті чого отримують значення H. Деяке повідомлення m' розбивається на блоки m_1, ... , m_n, m_{n+1}. Оскільки перші n блоків повідомлення m' ідентичні першим n блокам повідомлення m, то значення $h(m)$ відповідає проміжному результату, що з'являється при обчисленні $h(m')$ після обробки перших n блоків повідомлення m'. Отримуємо, що $h(m') = h'(h(m), m_{n+1})$, де h' – функція ущільнення хеш-функції. Такий недолік можна використати в MD5 або в будь-якій функції сімейства SHA. Необхідно лише конструювати повідомлення m' таким чином, щоб в його склад включались біти доповнення й поле довжини повідомлення. Але це не є проблемою, оскільки метод побудови таких полів відомий, наприклад, в [3].

Доповнення повідомлення можливе й не тільки одним блоком. Зловмисник може доповнити повідомлення довільною кількістю блоків m_{n+1}, ... , m_{n+k}, а останній m_{n+k} блок підібрати таким чином, щоб $h'(h'(m_{n+k-1}), m_{n+k}) = h(m)$, так як це показано на рис. 2. Атака потребує від зловмисника побудови кількох необхідних пар (m, m') і перевірки їх на предмет колізії. Перевірка полягає в обчисленні значень функції хешування, що в свою чергу не потребує великих затрат часу, а отже, така атака є достатньо швидкою.'

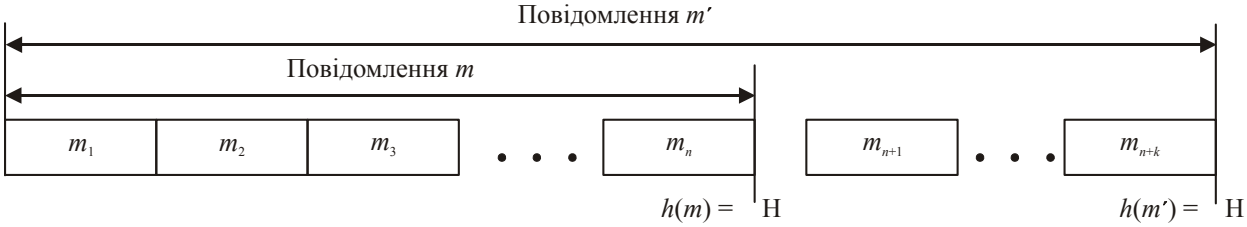

Рисунок 2 – Атака на подовження повідомлення

Також, з використанням цього ж принципу стає можливим і скорочення початкового повідомлення. Зловмисник може видалити будь-яку кількість блоків, скажімо, m_i, ... , m_j, замінивши їх своїм блоком m_e таким, що при його обробці функцією $h'(m_e)$ отримане значення буде дорівнювати результату обробки функцією ущільнення останнього видаленого блоку m_j, тобто $h'(m_e) = h'(m_j)$. Замінити блоки m_i, ... , m_j можна й на довільну кількість блоків m_{ei}, ... , m_{ej}, підібравши певним чином останній m_{ej} блок.

Таким чином зловмисник може подовжити початкове повідомлення своєю інформацією як з кінця повідомлення так і з середини, а також скоротити повідомлення, за умови, що довжина останнього більше довжини блоку m_i.

Ще один недолік функцій хешування пов'язаний з наявністю в більшості з них ітеративної структури. Пояснюється це на конкретному прикладі. Припустимо, що є система, що проводить автентифікацію повідомлення m за допомогою значення $h(m \parallel K_a)$, де K_a – ключ автентифікації. Ітеративна структура дає можливість зловмисникові знаходити рядки m та m', які при хешуванні функцією h призводять до колізії. Використовуючи атаку, в основі якої лежить парадокс задачі про дні народження, це можна виконати лише приблизно за $2^{n/2}$ кроків. Потім зловмисник автентифікує в системі повідомлення m і замінює його повідомленням m'. Оскільки значення h обчислюється ітеративно, то, якщо при хешуванні частини другого повідомлення виникне колізія, а всі вхідні значення, що залишилися, будуть такими ж, як і для першого повідомлення, значення хеш-коду теж не зміниться. Оскільки хешування повідомлень m і m' дає те саме значення, то $h(m \| X) = h(m' \| X)$ для всіх X [4].

В цьому випадку, при знаходженні повідомлення m', зловмисник також може використати атаку на подовження/скорочення повідомлення, що робить її найбільш серйозною, а проблему усунення можливості здійснення цієї атаки – актуальною задачею.

III Аналіз можливості усунення проблеми подовження/скорочення повідомлення

Є декілька способів, що дозволяють усунути проблему подовження/скорочення.

Перший спосіб [4] полягає в тому, що замість функції хешування $h(m)$ використовується функція

$$h_{DBL}(m) := h(h(m) \| m). \quad (1)$$

Як видно, перед хешуванням повідомлення m до його початку було дописано значення $h(m)$. Завдяки цьому ітеративне обчислення хеш-коду буде одразу залежати від усіх бітів повідомлення, що робить неможливими атаки, які відбуваються шляхом часткового хешування або подовження повідомлення. Якщо h є хеш-функцією, що забезпечує рівень безпеки в $n/2$ біт, то запропонована конструкція буде мати рівень безпеки в n біт, де n – розмір результату хешування.

Недоліком цього підходу є низька швидкість роботи. Повідомлення доводиться хешувати двічі, що займає в двічі більше часу, порівняно зі звичайним хешуванням. Ще одним недоліком цього підходу є необхідність буферизації всього повідомлення m. В результаті цього більше не буде можливості підраховувати хеш-код потоку даних, що поступово надходять до системи. Деякі програми потребують таку можливість і тому функція h_{DBL} для них не підходить.

Для збереження повної швидкості роботи існує другий спосіб усунення вище наведених недоліків. Замість використання $h(m)$ використовують таку функцію [4]:

$$h_d := h(h(m)). \quad (2)$$

Дана функція має рівень безпеки, що дорівнює $\min(k, n/2)$, де k – рівень безпеки функції h, а n –розмір результату функції хешування.

Використання таких підходів обумовлює застосування спеціальних програм або спеціально написаного коду, що буде використовувати функцію хешування для обчислення повідомлення, а потім знову використовувати функцію хешування для обчислення попередньо отриманого результату.

Запропонований нижче метод вдосконалює саму функцію хешування.

IV Вдосконалення функції хешування

Якщо повідомлення можна буде подовжувати до нескінченості, то і кількість колізій буде нескінченною. Вирішивши проблему подовження повідомлення, ми підвищимо стійкість функцій хешування до колізій. Для усунення цього недоліку потрібно, щоб в процесі роботи функції хешування приймали участь біти, що відповідають за довжину повідомлення. Такими бітами можуть бути біти довжини повідомлення. Застосовувати ці біти слід в функції ущільнення самої хеш-функції, адже саме при роботі функції ущільнення перемішуються, ущільнюються біти повідомлення та отримується хеш-код.

В результаті, при спробі зловмисника змінити повідомлення, дописавши до нього своє чи видаливши існуючи блоки, зміниться значення довжини повідомлення. Тоді при послідовному обчислені 512-бітових блоків повідомлення зміниться проміжне значення результату функції ущільнення. Обчисливши останній блок отримуємо хеш-значення, що відрізняється від хеш-значення, отриманного від первинного повідомлення. Ця відмінність буде свідчити про те, що повідомлення було змінено. Загальна схема запропонованого методу наведена на рис. 3.

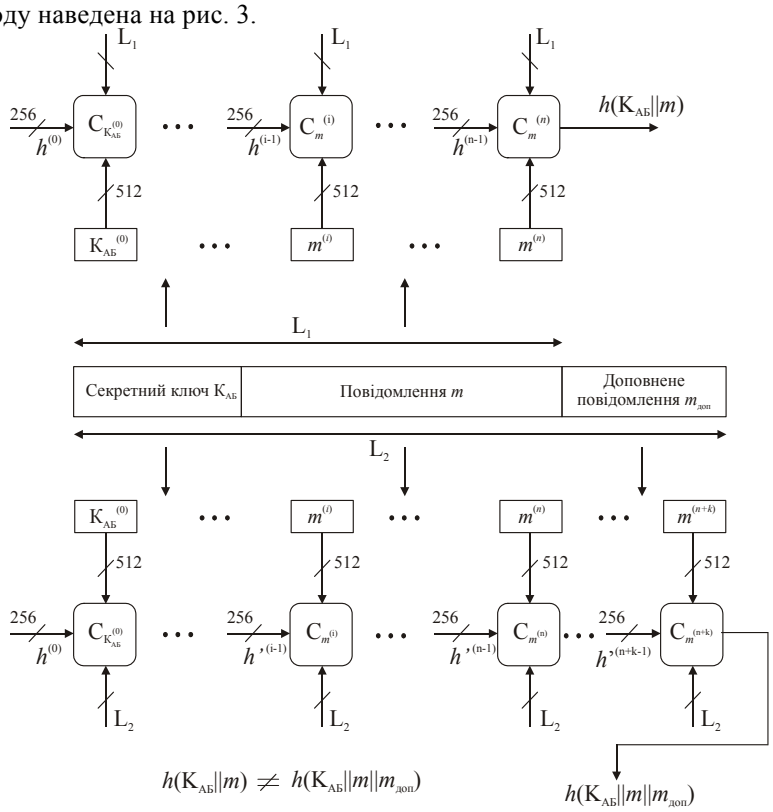

Рисунок 3 – Загальна схема роботи запропонованого методу

Як видно з рис. 3, при обчисленні хеш-коду первинного повідомлення m в роботі функції ущільнення приймає участь значення довжини повідомлення L_1. Якщо зловмисник, отримавши з каналу передавання даних повідомлення m та код автентичності $h(K_{АБ}||m)$ (див. рис. 1), все ж таки доповнить повідомлення m своїм повідомленням $m_{доп}$ та підбере певним чином останній блок так, щоб результат обчислення функції ущільнення $h'(m_{n+k})$ дорівнював коду автентичності $h(K_{АБ}||m)$ (див. рис. 2), то при автентифікації повідомлення користувачем Б (див. рис. 1) в блоках, які складають секретний ключ $K_{АБ}$, вже буде приймати учать значення довжини L_2 і результат роботи функції ущільнення буде зовсім інший. Це в свою чергу спричинить обчислення зовсім іншого результату всього повідомлення. В результаті $h(K_{АБ} \| m) \neq h(K_{АБ} \| m \| m_{доп})$ і система автентифікації видасть негативну відповідь.

Розглянемо роботу запропонованого методу на конкретному прикладі. За функцію хешування візьмемо SHA-256 [3]. На сьогодні вона забезпечує мінімальний рівень безпеки в 128 біт, а швидкість роботи алгоритму достатня для сучасних обчислювальних систем.

Слід певним чином застосовувати значення довжини повідомлення в функції ущільнення, адже використання одного й того ж значення на кожному кроці роботи функції ущільнення хеш-функції може призвести до небажаних закономірностей, які можуть виявитися при диференційному аналізі функції

хешування [5, 6]. Тому значення довжини повідомлення необхідно рандомізувати для кожного кроку роботи функції ущільнення.

Як відомо, всі операції в SHA-256 виконуються за модулем 2^{32}, а довжина значення повідомлення при цьому дорівнює 64 біти [3]. Отже, постає необхідність виконання операції розрядністю в 32 біти.

Процес перетворення 64-х бітового значення довжини повідомлення в 32-х бітове зображено на рис. 4.

Для виконання цього перетворення складемо за модулем 2 перші 32 біти довжини повідомлення з іншими 32 бітами довжини повідомлення. Нехай L – довжина повідомлення; l_1 – перші 32 біти; l_2 – другі 32 біти.

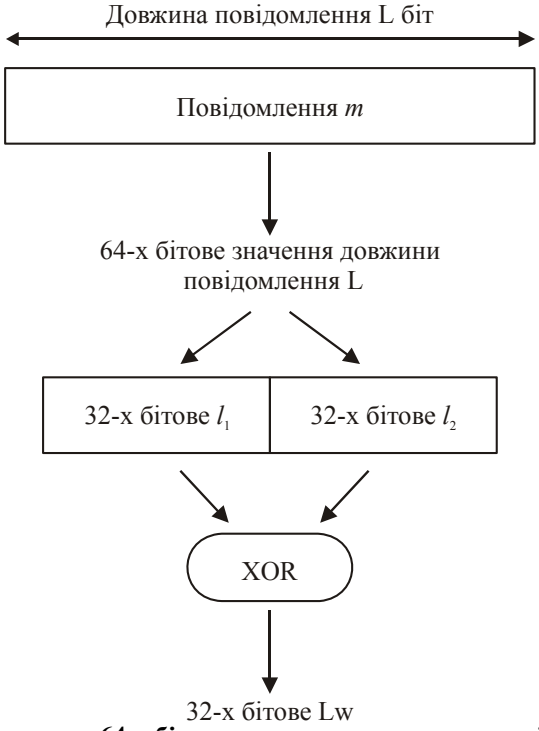

Рисунок 4 – Процес перетворення 64-х бітового значення довжини повідомлення в 32-х бітове

Нехай Lw буде зберігати результат виконання операції суми доданків l_1 та l_2 за модулем 2. Таким чином довжина Lw буде дорівнювати 32 бітам. Це дозволить виконувати всі необхідні операції за модулем 2^{32}, як визначено у хеш-функції SHA-256:

$$\mathrm{Lw} = l_1 \oplus l_2 . \tag{3}$$

Для того, щоб рандомізувати значення довжини повідомлення для кожного кроку роботи функції ущільнення, виконаємо операцію додавання вище отриманого результату з 32-бітовим елементом розширеного повідомлення j-го кроку W_j за модулем 2^{32}. Нехай результат виконання операції додавання за модулем 2^{32} для кожного j-го кроку одного блоку буде зберігатися в масиві $\mathrm{ArrayLw}_j$.

$$\mathrm{ArrayLw}_j = \mathrm{Lw} + \mathrm{W}_j \tag{4}$$

Оскільки повідомлення обробляється 512-бітовими блоками, а значення елементів масиву $\mathrm{ArrayLw}_j$ залежить від W_j, то вміст масиву $\mathrm{ArrayLw}_j$ буде оновлюватись для кожного 512-бітового блоку повідомлення. Кожний елемент масиву $\mathrm{ArrayLw}_j$ буде тільки один раз використовуватися на відповідному кроці роботи функції ущільнення. Використання полягає в здійсненні додавання кожного j-ого елементу масиву $\mathrm{ArrayLw}_j$ з регістром c. Результат додавання заноситься в 32-бітовий регістр d. Далі виконуються елементарні логічні операції згідно з алгоритмом SHA-256.

Схематично кожний крок раунду функції ущільнення модифікованої хеш-функції матиме вигляд, представлений на рис. 5.

Основні дії зловмисника, що пов'язані з функціями хешування зводяться до таких операцій:

1 знаходження прообразу m за заданим $y = h(m)$; така атака особливо небезпечна для систем автентифікації, що використовують хеш-значення паролів та секретних ключів;

2 знаходження прообразу m' для заданого m, для яких би виконувалася умова $h(m) = h(m')$; ця атака може бути використана для фальсифікації повідомлення, що підписане схемою цифрового підпису.

Тому для запобігання вище перерахованим атакам зловмисника висуваються основні вимоги до захищених функцій хешування з точки зору їх крипостійкості [1]:

1 *стійкість до обчислення прообразу* (*one-way property*) – неможливість побудови невідомого прообразу для будь-яких попередньо заданих хеш-значень, тобто для заданої хеш-функції h обчислювально неможливо знайти невідомий прообраз m при попередньо заданому хеш-значенні $y = h(m)$ для будь-якого значення y;

2 *стійкість до колізій* (*collision resistance*) – неможливість побудови двох прообразів, для яких вироблялося б однакове значення, тобто для заданої хеш-функції h обчислювально неможливо знайти два прообрази m і m', $m \neq m'$, для яких виконувалася б умова $h(m) = h(m')$.

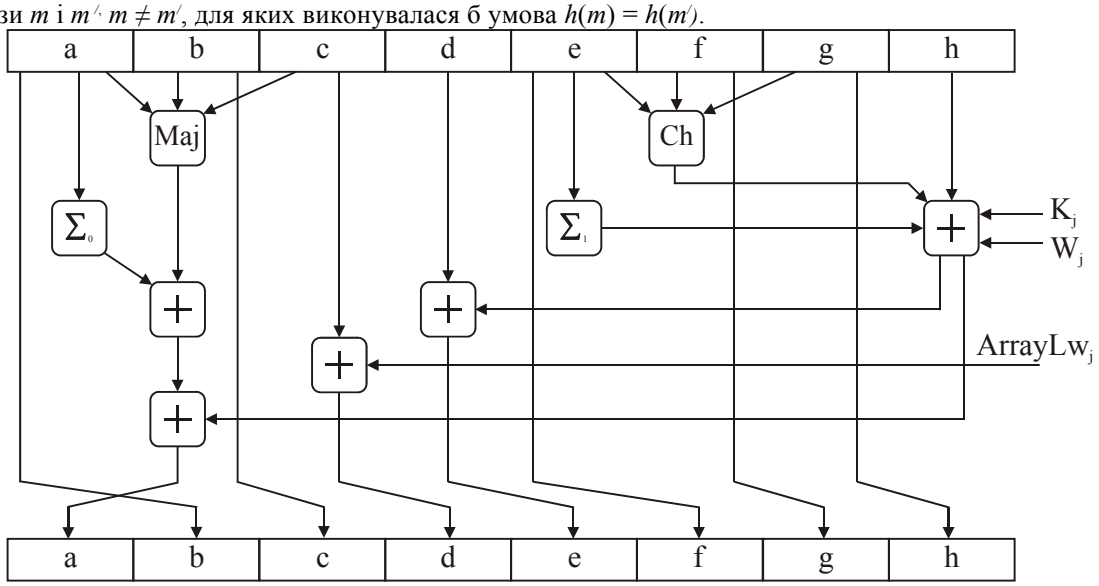

Рисунок 5 - Елементарна операція за модифікованим методом

Проаналізуємо виконання цих вимог у запропонованому методі. Запропонований метод передбачає визначення довжини повідомлення та перетворення останньої у відповідний формат для обробки в функції ущільнення. Цей процес відбувається паралельно і не впливає на безпосередню роботу алгоритму.

Щодо першої вимоги, то головним критерієм стійкості до обчислення прообразу є довжина профілю. Чим більша довжина хеш-результату, тим складніше знайти прообраз. При впровадженні запропонованого методу довжина результату функції ущільнення залишається незмінною, а саме 256 біт в функції хешування SHA-256. Тому складність знаходження прообразу залишається дорівнювати 2^{256} операцій обчислення хеш-значення і функція хешування за запропонованим методом задовольняє першій вимозі.

Щодо другої вимоги, яка є більш жорсткішою, складність знаходження колізії обчислюється в $2^{n/2}$ операцій, де n – довжина результату функції хешування. Значення стійкості $2^{n/2}$ пояснюється парадоксом задачі про дні народження. З впровадженням запропонованого методу довжина профілю функції хешування не змінюється і рівна 256 бітам, а тому стійкість функції хешування в 2^{128} залишається незмінною.

Отже, впроваджений метод задовольняє вище перерахованим вимогам, тобто функція хешування залишається криптографічно стійкою. При цьому перевагою запропонованого методу є неможливість проведення атаки на подовження/скорочення повідомлення.

Запропонований метод реалізовано програмно в середовищі Сі. Проведено експериментальні дослідження запропонованої функції хешування та базового алгоритму SHA-256. Дослідження показали, що швидкість виконання хешування за модифікованою функцією не значно поступається швидкості хешування за базовим алгоритмом і зі збільшенням потужності обчислювальної техніки різниця цих швидкостей прямує до нуля.

V Висновки

Запропоновано метод усунення проблеми подовження/скорочення повідомлення, що є в багатьох відомих функціях хешування. Суть методу полягає у використанні рандомізованого значення довжини повідомлення на кожному кроці роботи функції ущільнення хеш-функції. Стійкість модифікованої функції хешування не нижча, ніж базової функції SHA-256, але запобігає проведенню атаки на

подовження/скорочення повідомлення. Проведені експериментальні дослідження показали, що різниця між швидкістю роботи базової та модифікованої функції хешування практично відсутня і зі зростанням обчислювальної потужності прямує до нуля.

Таким чином було усунено можливість проведення атаки на подовження/скорочення повідомлення в схемах автентифікації, обумовлену слабкістю функцій хешування.

*Література: **1**. В. Н. Вервейко, А. И. Пушкарев, Т. В. Цепурит. Функции хэширования: классификация, характеристика и сравнительный аналіз // Радиотехника: Всеукр. межвед. науч.-техн. сб. - 2002 **2**. Столлингс В. Криптография и защита сетей: принципы и практика, 2-е издание, – М.: Издательский дом «Вильямс», 2001. – 452с. **3.** National Institute of Standards and Technology. Secure Hash Standart. – FIPS PUB 180-1, April 1995. **4**. Фергюсон Н., Шнайер Б., «Практическая криптография».: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 424 с.: ил. – Парал. тит. англ. **5**. Biham E., On the Applicability of Differential Cryptoanalysis to Hash Functions. In E.I.S.S Workshop on Cryptographic Hash Functions, pages 25-27, March 1992. **6**. Biham E. and Shamir A., Differential cryptoanalysis of FEAL and N-Hash. In Advances in Cryptology – Eurocrypt ’91, pages 1-16, 1991.1*