УДК 004.65.3

В.А. Лужецький, Ю.В.Баришев, О.В.Оводенко
Вінницький національний технічний університет
E-mail: yuriy.baryshev@gmail.com
Донецький національний технічний університет
E-mail: ovoda@i.ua

МЕТОДИ ПІДВИЩЕННЯ СТІЙКОСТІ ХЕШ-ФУНКЦІЙ

Анотація
Лужецький В.А., Баришев Ю.В., Оводенко О.В. Методи підвищення стійкості хеш-функцій. *Проаналізовано сучасні атаки на хеш-функції та методи протидії ним. Синтезовано узагальнену підсилену конструкцію хешування. Запропоновано декілька хеш-функцій, як приклади реалізації конструкції.*
Ключові слова: *хешування, конструкція, колізія, підсилення.*

Вступ.

Останнім часом проблеми, пов'язані із забезпеченням швидкого та стійкого хешування, породили ряд задач, що необхідно якомога швидше розв'язувати, оскільки відоме хешування не задовольняє сучасним вимогам щодо цих двох характеристик. Оголошення організацією NIST конкурсу на новий стандарт хешування SHA-3 є ще одним підтвердженням цього. Однією із задач, що стоять перед фахівцями з хешування, є збільшення складності пошуку колізій для хеш-функцій, що розпаралелюються, та унеможливлення (утруднення) попередньої підготовки зловмисниками атаки.

Проблема, пов'язана зі стійкістю хеш-функцій до колізій гостро постала після відкриття атаки "дня народження", та її використання для побудови більш складних атак на основі мультиколізій [1]. У [2] Діном був запропонований підхід до здійснення атак на хеш-функції MD4, MD5 та SHA-1. Крім того, з'явився цілий клас атак, які базуються на попередній підготовці до атак, наприклад, атака Келсі-Кохно [3].

З метою покращення стійкості було запропоновано ряд математичних моделей хешування, які відповідно до традиції, що склалася в західній літературі, прийнято називати конструкціями. Однією з найбільш цікавих спроб покращити хешування для забезпечення його стійкості стала конструкція HAIFA (HAsh Iterative FrAmework), запропонована в [3], а також трансформована конструкція Меркля-Дамгаарда з подвійним перемішуванням, запропонована в [4]. Проте дані конструкції не змогли забезпечити необхідний для майбутнього стандарту SHA-3 рівень стійкості хешування, тому розробка нових конструкцій хешування є наразі актуальною.

Метою даного дослідження є визначення методів підвищення стійкості хешування, що ускладнюють зловмисникам пошук колізій при зламі хеш-значень.

Для досягнення мети необхідно розв'язати такі задачі:

– дослідити методи використання колізій для зламу хеш-значень;

– визначити методи утруднення знаходження колізій.

Аналіз відомих конструкцій хеш-функцій. Класичною математичною моделлю хешування є конструкція Меркля-Дамгаарда [5, 6]. Хешування відповідно до цієї конструкції передбачає розбиття масиву інформаційних даних **M** на блоки рівної довжини $\mathbf{M} = \{m_1, m_2, ..., m_l\}$. У випадку, коли інформаційні дані некратні довжині блоку даних хеш-функції, їх доповнюють певною послідовністю бітів. Для ускладнення атаки подовженням

повідомлення, при використанні даної конструкції передбачається доповнення даних, що хешуються, одним або декількома блоками, які містять довжину оригінального повідомлення [5], отриману таким чином конструкцію називають підсиленою конструкцією Меркля-Дамгаарда. Дана конструкція є найбільш широко використовуваною серед відомих хеш-функцій.

У подальшому будемо вважати, що блок даних m_{l+1} є блоком, що містить довжину оригінального повідомлення. Після того, як інформаційні дані були підготовлені відповідним чином, відбувається хешування за ітеративним правилом:

$$h_i = f\left(m_i, h_{i-1}\right), \tag{1}$$

де h_i – результат хешування *i*-го блоку інформаційних даних $\left(i \in [1; l+1]\right)$;

$f(\cdot)$ – функція ущільнення, яка забезпечує фіксовану розрядність результату.

Хеш-значенням h повідомлення **M** буде h_{l+1}, а h_0 є початковим заповненням або, як його ще прийнято називати, вектором ініціалізації і може бути використано як ключ при потребі ключового хешування. У даній конструкції, попри її широку поширеність, дослідження останніх часів виявили ряд недоліків.

У роботі [2] Дін розглянув декілька підходів до атак на хеш-функції. Фактично робота [2] стала суттєвим проривом в галузі криптоаналізу хеш-функцій. Дін сформулював такі методологічні підходи до пошуку колізій:

– пошук повторюваних ланцюжків у повідомленні;

– пошук фіксованих точок;

– дописування префіксів і пошук фіксованих точок.

Пошук повторюваних ланцюжків є інтуїтивно зрозумілою атакою. Він полягає у тому, що, коли при хешуванні функції трапляються ідентичні проміжні хеш-значення $h_i = h_j \left(i < j\right)$, то можна видалити всі блоки даних, які знаходяться між блоками m_i та m_j, включаючи один з цих блоків. Пошук фіксованих точок полягає в генеруванні випадкових блоків даних m_i^* та їх хешування із різними псевдопочатковими значеннями h_{i-1}^* доти, доки не буде знайдено проміжне значення хеш-функції таке, що:

$$\begin{cases} h_{i-1} = h_{i-1}^* \\ f\left(m_i^*, h_{i-1}^*\right) = \vec{0} \end{cases}, \tag{2}$$

де $\vec{0}$ – нейтральний елемент хеш-функції.

Під нейтральним елементом розуміється таке проміжне значення хеш-функції, наявність якого не впливає на наступні проміжні хеш-значення. У [2] також показано, як використовуючи фіксовані точки (2) можна змінити повідомлення певним чином, додавши до нього потрібний зловмиснику префікс. Дані методичні підходи були успішно використані ним для зламу MD4, MD5 та SHA-1, не зважаючи на те, що ці хеш-функцій використовують підсилену конструкцію Меркля-Дамгаарда. Завдяки обмеженню розрядності лічильника довжини повідомлення та відсутності теоретичних підвалин в обґрунтуванні стійкості цих хеш-функцій атаки Діна стали можливими для цих хеш-функцій.

Розвиток криптоаналізу, який відбувався втому числі і завдяки роботі [2], спричинив потребу у збільшенні довжини хеш-значення. У той же час, збільшення розрядності обумовлювало збільшення часу хешування, що не завжди є прийнятним для пересилання даних в системах реального часу. Саме тому з'явилися пропозиції обчислювати паралельно за допомогою різних хеш-функції декілька хеш-значень малої розрядності, а результуюче хеш-значення отримувати шляхом їх конкатенації. Такий підхід виявився менш стійким до атак, ніж від нього того очікували. Відома атака Жукса, запропонована в [1], використовує пошук мультиколізій, при розпаралеленні хешування шляхом каскадування, яке розглядалося в [6].

Розглянемо детальніше принцип цієї атаки.

Нехай є дві функції ущільнення $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$. Кожна з цих функцій використовує як вхідні данні блоки повідомлення, що хешується, сталої довжини $m_i\left(i=\overline{1,l+1}\right)$ та хеш-значення, отримане на попередній ітерації алгоритму h_{i-1}, тобто *i*-е значення буде розраховуватись за допомогою кожної окремої функції відповідно конструкції (1). Вихідне значення формується шляхом конкатенації хеш-значень $h_{l+1}^{(1)}$ та $h_{l+1}^{(2)}$, обчислених за допомогою $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$ відповідно. Таке паралельне обчислення дозволяє скоротити час хешування, проте, як довів Жукс, воно зменшує стійкість остаточного хеш-значення порівняно з хеш-значенням, визначеним за допомогою функції, яка має вдвічі більшу довжину вихідного хеш-значення, ніж $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$. Для своєї атаки Жукс запропонував знаходити мультиколізії в повідомленні для однієї функції, а потім серед них шукати такі, що викличуть колізію й в іншій функції. Мультиколізію Жукс отримав, знаходячи для кожного *i*-го блоку даних m_i, інший блок даних m_i^* такий, що виконується така умова:

$$f^{(1)}(m_i, h_{i-1}) = f^{(1)}(m_i^*, h_{i-1}). \quad (3)$$

На рис. 1 схематично зображено, як знаходить мультиколізії для хеш-функцій Жукс в своїй роботі [1], що використовують конструкцію Меркля-Дамгаарда або конструкції аналогічні до неї.

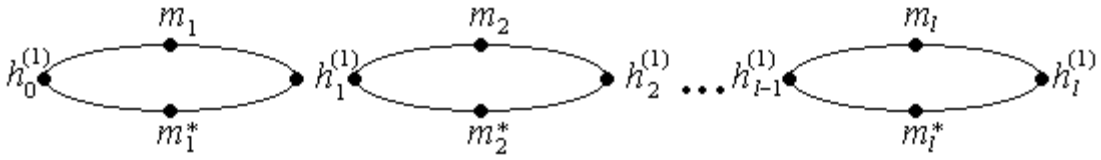

Рисунок 1 – Вигляд мультиколізії Жукса

Використовуючи мультиколізію, зображену на рис. 1, можна побудувати 2^l різних повідомлень, які викличуть колізію в першій функції, а потім серед них знайти таке повідомлення, яке призведе і до колізії в іншій. Для підсиленої конструкції Меркля-Дамгаарда можна застосувати такі ж самі дії, а для останнього блока даних, що хешуються, m_{l+1} не шукати колізії, оскільки цей блок даних містить довжину оригінального повідомлення, а тому його підміна одразу буде помічена.

Атака Жукса була узагальнена в подальших працях, однією з останніх серед таких є [4], в якій об'єктом атаки є модифікована конструкція Меркля-Дамгаарда. Авторами [4] було показано, що, оскільки дана конструкція зберігає властивості ітеративності, то можливо виконати атаку, аналогічну атаці Жукса.

Ще однією з атак на конструкцію Меркля-Дамгаарда є атака Келсі-Кохно, описана в [3], що базується на попередній підготовці зловмисником атаки, шляхом знаходження ланцюжків повідомлень, які згодом будуть використані для пошуку колізій. Дана атака особливо небезпечна в системах, де обмін даними виконують у режимі реального часу, оскільки основні обчислення зловмисник може виконати ще до початку обміну даними.

З метою ускладнення таких атак авторами [3] була запропонована конструкція HAIFA (HAsh Iterative FrAmework). Дана конструкція передбачає збільшення аргументів функції ущільнення за рахунок введення до аргументів довжини вже оброблених інформаційних даних та псевдовипадкового числа:

$$h_i = f(m_i, h_{i-1}, \#bits, salt), \quad (4)$$

де $\#bits$ – кількість вже захешованих бітів повідомлення;

$salt$ – псевдовипадкове число, яке може використовуватись, як ідентифікатор сесії або

автора (при цифровому підписі).

Відповідно до [3], лічильник захешованих бітів введено з метою утруднення пошуку фіксованих точок, що, в свою чергу, утруднює задачу підготовки до зламу такої функції. Розглянемо стійкість до атак, що базуються на попередній підготовці криптоаналітика, на прикладі атаки Келсі-Кохно. При знаходженні певного ланцюжка блоків інформаційних даних, зловмисник не може їх підставити в довільне місце повідомлення для утворення колізії, оскільки в такому випадку йому ще необхідно врахувати відносне місце цих блоків в повідомленні, що до передачі даних неможливо зробити (як в атаці Келсі-Кохно).

Псевдовипадкове число може використовуватись як частина ключа. Його використання пропонується для цифрового підпису або як ідентифікатор сесії обміну даними між сторонами, тобто воно або цілком невідоме зловмиснику, або розкривається безпосередньо перед сесією обміну даними, що також утруднює попередню підготовку зловмисником атаки.

Нестійкість конструкції HAIFA до атаки Жукса авторами [3] не заперечується, вони стверджують, що вони лише ускладнюють задачу зламу зловмиснику, оскільки без наперед відомого псевдовипадкового числа зловмисник не зможе почати шукати мультиколізії. Введення додаткових аргументів функції ущільнення дозволило утруднити криптоаналітику задачу заздалегідь підготовленої атаки, але в той же час, функція (4) не забезпечує нерозв'язуваність цієї задачі.

Крім того, з точки зору швидкості, конструкція HAIFA поступається конструкції Меркля-Дамгаарда, оскільки додаткові аргументи потребують додаткової обробки. Використання кількості вже захешованих біт для зловмисника є передбачуваним параметром, а тому не можна виключати розробку нових атак з попередньою підготовкою криптоаналітика, які будуть передбачати обчислення ланцюжків повідомлень, починаючи з першого блока даних. Крім того, дана конструкція орієнтується на вже розроблені хеш-функції, стійкість, яких не є теоретично доведеною, а тому розробка атак, аналогічних атакам Діна, не виключається.

Узагальнена конструкція хешування. Розглянемо узагальнену модель хешування. Зазначимо, що буде розглядатися лише ітеративне хешування, оскільки неітеративне є вузькоспеціалізованим і вкрай рідко використовуваним, тому що не дозволяє хешувати дані довільної довжини. Перед тим, як перейти до формалізації узагальненої моделі хешування, визначимо основні аргументи, які можуть використовуватися в хеш-функції.

Всі відомі конструкції ітеративного хешування як аргументи обов'язково мають блок повідомлення, що хешується, та проміжне хеш-значення, отримане на попередній ітерації, як це передбачено конструкцією Меркля-Дамгаарда (1). Очевидним узагальненням даної конструкції є збільшення кількості проміжних хеш-значень та блоків даних. Розширимо цю конструкцію, вважаючи, що проміжне хеш-значення, отримане на *i*-му кроці, може залежати від таких аргументів:

– усіх блоків інформаційних даних $\{m_i\}, i = \overline{1,l}$;

– $(i-1)$ попередніх проміжних хеш-значень;

– ключових даних, в тому числі і початкового заповнення, або вектора ініціалізації, яке позначимо через h_0.

Отже, проміжне хеш-значення на *i*-й ітерації буде обчислюватись за функцією такого виду:

$$h_i = f(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, m_{l+1}). \quad (5)$$

У загальному випадку можна виконувати підсилення конструкції, аналогічне до підсилення конструкції Меркля-Дамгаарда. Однак, оскільки ми намагаємось отримати узагальнену конструкцію, то вона буде відрізнятися від класичної підсиленої конструкції Меркля-Дамгаарда більш загальним характером. Тобто дана конструкція буде враховувати кількість

вже захешованих інформаційних бітів на кожній ітерації, замість дописування бітової довжини повідомлення до його кінця:

$$h_i = f\left(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, \#bits_i\right), \quad (6)$$

де $\#bits_i$ – бітова довжина вже захешованої частини повідомлення або її еквівалент.

Останній крок узагальнення, формалізований у (6), є близьким до того, що пропонується авторами [3], однак у роботі [3] розглядається виключно лічильник бітів. Однак замість даного лічильника можна використовувати лічильник, який визначає кількість вже захешованих байт, блоків даних тощо. Отже, аргумент $\#bits$, що використовується в конструкції (4) є частковим випадком аргументу $\#bits_i$.

Крім передбачених у конструкції (6) аргументів, при хешуванні можуть використовуватися й аргументи, що змінюються за певним псевдовипадковим законом. Такі аргументи часто застосовують для покращення характеристик випадковості вихідних значень хеш-функції [3]. Якщо розглядати загальний випадок, то числа псевдовипадкової послідовності можуть використовуватись декілька раз на різних ітераціях. Крім того, на одній ітерації може бути використано декілька псевдовипадкових чисел. З урахуванням цього, конструкція (6) набуде такого вигляду:

$$h_i = f(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, r_1, r_{2,}..., r_t, \#bits_i) \quad (7)$$

де $r_1, r_{2,}..., r_t$ – псевдовипадкові числа.

Псевдовипадкові числа можуть бути опубліковані відкрито, визначатись лише перед початком обміну даними або використовуватись як певний секретний параметр для цифрового підпису [3].

Опишемо відомі конструкції хешування, як окремі випадки узагальненої конструкції хешування (7). Відповідно конструкція Меркля-Дамгаарда залишиться у вигляді формули (1), а її підсилений варіант, що враховує на останній ітерації довжину оригінального інформаційного повідомлення, матиме такий вигляд:

$$h_i = f\left(h_{i-1}, m_i, \#bits_i\right). \quad (8)$$

Хоча в конструкції (8) кількість захешованих даних використовується лише на останній ітерації, коли оброблені всі інформаційні дані, але така її особливість дозволила утруднити зловмисникам задачу зламу, тому не можна спрощувати дану конструкцію, як це часто роблять, до вигляду формули (1).

Конструкція HAIFA з урахуванням (7) описується виразом:

$$h_i = f\left(m_i, h_{i-1}, r, \#bits_i\right). \quad (9)$$

Не зважаючи на те, що розробники деяких хеш-функцій, наприклад SHA-1, базувались на конструкції Меркля-Дамгаарда, але фактично хеш-функції розроблені ними, більше подібні до конструкції HAIFA, оскільки вони використовують псевдовипадкові числа та довжину повідомлення, хоча на цьому не акцентується увага з боку їх розробників.

Приклади хешування. Розглянемо хешування, задача зламу якого зводиться до задачі пошуку дискретного логарифма в полі простого числа [7], та запропонуємо для нього декілька варіантів реалізації узагальненої конструкції хешування (7). Не зважаючи на те, що задача пошуку дискретного логарифма в полі простого числа має декілька алгоритмів розв'язання, відмінних від повного перебору, зокрема Адлемана та Копперсіта-Одлижко-Шреппеля [8], така задача все одно є складною для розв'язання за допомогою сучасного апаратного забезпечення і потребує для цього значного проміжку часу. Крім того, дана задача має зрозумілу структуру для демонстрації прикладів застосування запропонованих ідей.

Даний тип хешування передбачає використання двох ключових параметрів: особистого ключа та великого секретного числа, яке застосовуються як вектор ініціалізації [7]. Особистий ключ домішується на кожній ітерації до проміжного хеш-значення, отриманого на

попередній ітерації:

$$h_{i-1}^{*} = h_{i-1} + k', \tag{10}$$

де k' – особистий ключ.

Використання особистого ключа, вочевидь, запропоноване для можливості застосування даної хеш-функції для створення цифрового підпису. Таким чином, дана хеш-функція зводиться до конструкції Меркля-Дамгаарда, описаної формулою (1). Оскільки операція піднесення до степеня виконується довго, в порівнянні з іншими операціями, то збільшення аргументів у хеш-функції може спричинити збільшення часу обчислення хеш-значення. Тому використовувати методи підсилення хешування необхідно, зважаючи на час виконання операцій над ними. Відповідно пропонується виконувати швидкі, тобто такі, що виконуються за один такт, операції над "допоміжними" операндами, а операцію піднесення до степеня за модулем виконувати лише один раз на ітерації. Найбільш зручними з цієї точки зору, на думку авторів, є операції додавання за модулем 2^n та порозрядне додавання за модулем два. Одним з прикладів реалізації методів підсилення хешування може бути хеш-функція запропонована в [9], що реалізує таку конструкцію:

$$h_i = f\left(m_{(i-a) \bmod l}, m_{(i-b) \bmod l}, h_{i-1}\right), \tag{11}$$

де a, b – деякі цілі константи, що належать проміжку $[0; l)$.

Конструкція (11) дозволяє зменшити ймовірність появи колізій, які виникають внаслідок нульового значення у вмісті блока даних, та ускладнити зловмиснику задачу заміни одного блока даних іншим, оскільки кожен блок даних використовується в хешуванні двічі та об'єднується з різними блоками даних. Зазначимо, що в конструкцію (11) можна ввести й еквівалент кількості захешованих біт, але введення цього операнда до обчислень хеш значення може виявитись необґрунтованим, оскільки він буде частково дублювати функції констант a, b, водночас потребуючи додаткових операцій для свого домішування. Натомість пропонується покращити конструкцію (11) так, як це запропоновано в [10]:

$$\begin{cases} h_i = f\left(m_i, m_{(i-r_i) \bmod l}, h_{i-1}\right) \\ r_i = rand(m_i) \end{cases}, \tag{12}$$

де $rand()$ – деяка функція генерації псевдовипадкових чисел.

У випадку, коли псевдовипадкові числа r_i мають рівномірний закон розподілу (що залежиться від особливостей реалізації функції $rand()$ та закону розподілу блоків інформаційного повідомлення), всі блоки інформаціних даних будуть мати однаковий влив на остаточне хеш-значення.

Висновки. У даній статті розглянуті відомі методи знаходження колізій, зокрема атаки з використанням фіксованих точок, мультиколізій та попередньої підготовки до атак. Оскільки такі атаки використовують недоліки конструкцій хешування, то і протидія ним повинна реалізовуватись саме на рівні конструкції, а не за рахунок використання специфічних особливостей реалізації хеш-функції.

Аналіз відомих методів підвищення стійкості в конструкціях показав, що кількість аргументів сучасних хеш-функцій повинна бути збільшена, порівняно з класичною конструкцією. Запропоновано узагальнену конструкцію хешування, яка охоплює відомі конструкції хешування, а крім того породжує низку методів захисту від відомих атак. Ці методи передбачають обчислення проміжного хеш-значення залежно від всіх попередніх проміжних хеш-значень, всіх блоків даних, лічильника обсягу вже захешованих даних та псевдовипадкових чисел. Наведено приклади хеш-функцій, які реалізують методи підвищення стійкості і базуються на операції піднесення до степеня за модулем.

Література

1. A. Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. K. Franklin, editor, CRYPTO'04, volume 3152 of Lecture Notes in Computer Science, pages 306-316. Springer, 2004.
2. R. D. Dean. Formal Aspects of Mobile Code Security. Ph.D Dissertation, Princeton University, 1999. / Режим доступу до ресурсу – http://cyphunk.files.wordpress.com/2006/02/ddean-thesis.pdf
3. E. Biham, O. Dunkelman. A Framework for Iterative Hash Functions: HAIFA. / Режим доступу до ресурсу - http://csrc.nist.gov/pki/HashWorkshop/2006/Papers/DUNKELMAN_NIST3.pdf
4. N. Fergusson, S. Lucks. Attacks on AURORA-512 and the Double-MIX Merkle-Damgard Transform // Режим доступу до ресурсу – http://eprint.iacr.org/2009/113.pdf
5. R. C. Merkle. Secrecy, Authentifications and Public Key Systems. A Dissertation Submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University in Particial Fulfillment of the Requirements. / Режим доступу до ресурсу – http://www.merkle.com/papers/Thesis1979.pdf
6. B. Preneel. Analysis and Design of Cryptographic Hash Functions. PhD thesis, Katholieke Universiteit Leuven, 1993 // Режим доступу до ресурсу – http://homes.esat.kuleuven.be/~preneel/phd_preneel_feb1993.pdf
7. Патент України на корисну модель № 18693 МПК G 09 C 1/00. Спосіб ключового хешування теоретично доведеної стійкості / Стасєв Ю.В., Кузнецов О.О., Євсеєв С.П., Чевардін В.Є., Малахов С.В., Гришко А.В.; заявник та патентовласник Харківський університет повітряних сил. – №u200605734 ; заявл. 25.05.06 ; опубл. 15.11.06, Бюл. №11.
8. ВасиленкоО.Н.Теоретикочисловые алгоритмы в криптографии.–М.:МЦНМО,2003.–328 с.
9. Патент України на корисну модель № 36582 МПК G 09 C 1/00. Спосіб ключового хешування теоретично доведеної стійкості / Лужецький В.А., Баришев Ю.В., Дмитришин О.В.; заявник та патентовласник Вінницький національний технічний університет. – №u200808802 ; заявл. 04.07.08 ; опубл. 27.10.08, Бюл. №20.
10. Патент України на корисну модель № 37465 МПК G 09 C 1/00. Спосіб ключового хешування теоретично доведеної стійкості / Лужецький В.А., Баришев Ю.В., Дмитришин О.В.; заявник та патентовласник Вінницький національний технічний університет. – №u200808805 ; заявл. 04.07.08 ; опубл. 25.11.08, Бюл. №22.

Abstract

Luzhetsky V., Baryshev Y., Ovodenko O. Methods of hash-functions infeasibility increasing. *The modern attacks on hash functions and their counteractions are analyzed. The generalized strengthening hash construction is synthezed. Several hash functions, that are examples of construction implementation, are offered.*

Key words: *hashing, construction, collision, strengthening.*

Аннотация

Лужецкий В.А., Барышев Ю.В., Оводенко А.В. *Методы повышения стойкости хеш-функций. Проанализированы современные атаки на хеш-функции и методы противодействия им. Синтезирована обобщенная усиленная конструкция хеширования. Предложено несколько хеш-функций, в качестве примеров реализации конструкции.*

Ключевые слова: *хеширование, конструкция, коллизия, усиление.*

Здано в редакцію: 01.04.10р.

Рекомендовано до друку: к.т.н, проф. Маренич К.М.