© Ostapenko N., Kinzeryavyy V., Kyrychenko K., Grycak A. Advanced hash function MD4 // Ukrainian Scientific Journal of Information Security, 2018, vol. 24, issue 2, p.118-123.

DOI: 10.18372/2225-5036.24.12955

УДОСКОНАЛЕНА ФУНКЦІЯ ГЕШУВАННЯ MD4

**Наталія Остапенко, Василь Кінзерявий,
Каріна Кириченко, Анатолій Грицак**

Національний авіаційний університет, Україна

ОСТАПЕНКО Наталія Вікторівна

Рік та місце народження: 1995 рік, Україна.
Освіта: Національний авіаційний університет, 2017 рік.
Посада: магістрант кафедри безпеки інформаційних технологій.
Наукові інтереси: криптографічний захист інформації, функції гешування.
Публікації: 1 тези доповідей на конференції.
E-mail: natali.ost5@gmail.com

КІНЗЕРЯВИЙ Василь Миколайович, к.т.н.

Рік та місце народження: 1985 рік, Україна.
Освіта: Національний авіаційний університет, 2007 рік.
Посада: доцент кафедри безпеки інформаційних технологій.
Наукові інтереси: крипторафічний захист інформації, криптоаналіз симетичних шифрів.
Публікації: більше 90 наукових публікацій, серед яких наукові статті у вітчизняних та міжнародних фахових виданнях, патенти та авторські свідоцтва.
E-mail: v.kinzeryavyy@gmail.com

КИРИЧЕНКО Каріна Сергіївна

Рік та місце народження: 1992 рік, Росія.
Освіта: Національний авіаційний університет, 2014 рік.
Посада: аспірант кафедри безпеки інформаційних технологій.
Наукові інтереси: інформаційна безпека, крипторафічний захист інформації, функції гешування.
Публікації: 12 наукових публікацій, серед яких наукові статті у вітчизняних та міжнародних фахових виданнях.
E-mail: karina0kira@ukr.net

ГРИЦАК Анатолій Васильович

Рік та місце народження: 1993 рік, Україна.
Освіта: Вінницький національний технічний університет, 2015 рік.
Посада: аспірант вінницького національного технічного університету.
Наукові інтереси: крипторафічний захист інформації, функції гешування.
Публікації: 8 наукових публікацій, серед яких наукові статті у вітчизняних та міжнародних фахових виданнях.
E-mail: grytsak.a.v@gmail.com

Анотація. *Сьогодні інформація розглядається як стратегічний ресурс. Модифікація інформації чи її незаконне розповсюдження може призвести до серйозних наслідків. Забезпечення захисту інформації ускладняються із розвитком обчислювальних систем. Цілісність важливих файлів операційної системи, програм чи*

даних контролюється функціями гешування. Гешування застосовується для побудови асоціативних масивів, пошуку дублікатів в серіях наборів даних, побудови унікальних ідентифікаторів для наборів даних, контрольного підсумовування з метою виявлення випадкових або навмисних помилок при зберіганні або передачі, для зберігання паролів в системах захисту, при виробленні електронного підпису. Не так давно галузь криптографії, що пов'язана з гешуванням зіткнулась із вагомою проблемою – забезпеченням стійкості до мультиколізій, які використовуються атакою Жукса. В своїй роботі Жукс показав, що стійкість геш-значення, яке обчислюють шляхом каскадування цих функцій не набагато більша ніж стійкість однієї з них. Крім того відомі алгоритми гешування не дозволяють у повній мірі вирішувати питання забезпечення криптографічної стійкості та високої швидкодії алгоритмів. Тому розробка нових та удосконалення існуючих функцій гешування з метою підвищення ефективності криптографічного захисту ніколи не втратить своєї актуальності. З огляду на це, у роботі запропоновано функцію гешування newMD4, що була розроблена на основі оригінальної функції гешування MD4. Розроблена функція гешування newMD4 має декілька нововведень у порівнянні із MD4: при стисненні замість чотирьох 32-бітних змінних A, B, C, D запропоновано використання п'яти 64-бітних змінних A, B, C, D, J; збільшено довжину геш-значення до 256-біт; замінені додаткові функції F, G, H; додані додаткові операції на кожному етапі. У роботі проведено експериментальні дослідження, щодо оцінки швидкісних та статистичних характеристик запропонованої функції гешування. За однакових умов, проведені експериментальні дослідження з оцінки швидкісних характеристик, які показали, що функція гешування newMD4 швидша за оригінальний MD4 в 1,43 разів. Для дослідження статистичних характеристик використано тести NIST STS, при цьому функції гешування використовувались для генерації послідовностей, статистичні характеристики яких перевірялися зазначеними тестами. Згідно результатів дослідження послідовності, що згенеровано за допомогою функції гешування newMD4 показали кращі статистичні характеристики порівняно із оригіналом.

Ключові слова: *криптографія, функція гешування, цілісність інформації, захист інформації, криптостійкість.*

Вступ і постановка проблеми

У наш час інформаційні технології займають важливе місце у нашому житті. Головними характеристиками інформації є конфіденційність, доступність та цілісність. Модифікація інформації чи її незаконне розповсюдження може призвести до серйозних наслідків. Тому для контролю цілісності файлів операційних систем, програм чи даних широко використовуються функції гешування [1]. Гешування застосовується для побудови асоціативних масивів, пошуку дублікатів в серіях наборів даних, побудови унікальних ідентифікаторів для наборів даних, контрольного підсумовування з метою виявлення випадкових або навмисних помилок при зберіганні або передачі, для зберігання паролів в системах захисту (в цьому випадку доступ до області пам'яті, де знаходяться паролі, не дозволяє відновити сам пароль), при виробленні електронного підпису (на практиці часто підписується не саме повідомлення, а його геш-значення). Дані функції здійснюють стискання даних змінної довжини в послідовність фіксованого розміру (геш-значення). Отримані геш-значення використовується для перевірки відсутності модифікації даних [2]. Тобто для знаходження функції гешування спочатку визначають, цілісність яких файлів потрібно контролювати. Для кожного файлу здійснюється обчислення значення його гешу за спеціальним алгоритмом зі збереженням результату. Через необхідний час проводиться аналогічний розрахунок і порівнюються результати. Якщо значення відрізняються, значить інформація міститься в файлі була змінена.

Останнім часом галузь криптографії, що пов'язана з гешуванням, зіткнулася із суттєвою проблемою – забезпеченням стійкості до мультиколізій, які використовує атака Жукса [3]. Знайшовши, так звані, мультиколізії в одній з функцій гешування, які обчислюються паралельно, Жукс показав, що стійкість геш-значення, отриманого шляхом каскадування цих функцій, не набагато більша за стійкість однієї з них (за умови однакової стійкості функцій). Отже, проблема, пов'язана зі стійкістю та тривалістю гешування, повернулася. Попри усунення виявлених недоліків, питання відповідності запропонованого методу вимогам, що висуваються до функції гешування (зокрема стійкість до виникнення колізій), залишається відкритим.

Незважаючи на велику кількість алгоритмів гешування, питання забезпечення криптографічної стійкості та високої швидкодії ніколи не втратять своєї актуальності. Тому розробка нових та удосконалення існуючих функцій гешування є *актуальною* задачею.

З огляду на це, *метою роботи* є підвищення ефективності захисту інформації за рахунок використання удосконаленої функції гешування. Під *ефективністю* будемо розуміти підвищення швидкодії та забезпечення стійкості. Для досягнення цієї мети необхідно *розв'язати такі задачі*: проаналізувати відомі функції гешування та вимоги до них, розробити функцію гешування newMD4, розробити програмне забезпечення та провести експериментальне дослідження запропонованої функції гешування.

Аналіз публікацій

Криптографічна функція гешування – це функція гешування, що приймає довільний блок даних і повертає рядок встановленого розміру, таке що випадкові або навмисні зміни даних з дуже високою ймовірністю змінять геш-значення [4].

Ідеальна криптографічна функція гешування має чотири основні або значимі властивості [5]:

– легкість обчислення геш-значення для будь-якого повідомлення;

– неможливо утворити повідомлення для заданого геш-значення;

– неможливо змінити повідомлення без зміни геш значення;

– неможливо знайти два різних повідомлення з тим самим геш значенням.

Нижче наведені відомі функції гешування.

MD4 (Message Digest 4) [6] – функція гешування, розроблена професором Массачусетського університету Рональдом Рівестом в 1990 році, і вперше описана в RFC 1186. Для довільного вхідного повідомлення функція генерує 128-розрядне геш-значення, зване дайджестом повідомлення.

Рівень безпеки, закладений у MD4, був розрахований на створення досить стійких гібридних систем електронного цифрового підпису, заснованих на MD4 і криптосистемі з відкритим ключем. Уразливості в MD4 були продемонстровані у статті Берта ден Бура та Антона Босселарса в 1991 році. Перша колізія була знайдена Гансом Доббертіном в 1996 році.

MD5 (Message Digest 5) [7] – 128-бітний алгоритм гешування, розроблений професором Рональдом Рівестом в 1991 році. Призначений для створення геш значення повідомлень довільної довжини [6]. Прийшов на зміну MD4, що був недосконалим. Описаний в RFC 1321.

SHA-1 (Secure Hash Algorithm 1) [8] – алгоритм криптографічного гешування. Описано в RFC 3174. Для вхідного повідомлення довільної довжини (максимум 2^{64} біт) алгоритм генерує 160-бітове геш-значення.

SHA-2 (Secure Hash Algorithm Version 2) [9] – збірна назва односторонніх функцій гешування SHA-224, SHA-256, SHA-384 і SHA-512. Функції гешування призначені для створення дайджестів повідомлень довільної бітової довжини. Застосовуються в різних додатках або компонентах, пов'язаних із захистом інформації. Функції гешування сімейства SHA-2 побудовані на основі структури Меркла-Демгарда.

ГОСТ Р 34.11-2012 [10] – чинний російській криптографічний стандарт, що визначає алгоритм і процедуру обчислення функції гешування. Розроблено Центром захисту інформації та спеціального зв'язку ФСБ Росії. Розмір гешу – 256 або 512 біт; розмір блоку вхідних даних – 512 біт.

RIPEMD [11] – це функція гешування була розроблена в 1992 р в рамках європейського проекту RIPE (RACE Integrity Primitives Evaluation) як альтернатива популярній на той час функції гешування MD4. До поточного моменту розроблені посилені варіанти функції гешування RIPEMD, наприклад RIPEMD-160, які рекомендуються до використання в багатьох міжнародних і національних стандартах, зокрема ISO / IEC 10118-3: 2004.

N-Hash [12] – криптографічна функція гешування на основі циклічної функції FEAL. Була розроблена в 1990 році телекомунікаційною компанією Nippon Teleg-raph and Telephone. Спочатку, функція N-Hash була призначена для того, щоб вирішити проблему підміни інформації на шляху між двома користувачами телефонного зв'язку і прискорити пошук даних.

Snefru [13] – криптографічна функція гешування, запропонована Ральфом Меркле. Дана функція перетворює повідомлення довільної довжини в геш довжини m (зазвичай $m=128$ або $m=256$). Використовуючи засоби диференційного аналізу, Елі Біхам і Аді Шамір показали, що двопрохідні функція Snefru 128 - розрядним гешем не є стійкою до колізій 1-го роду і 2-го роду.

Купина [14] – ітеративна криптографічна функція гешування. Прийнята як національний стандарт України ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування». Функція стиснення Купини складається з двох фіксованих 2n-бітних перестановок $T\oplus$ і T+, структура яких запозичена у шифра Калина. Зокрема, використовуються чотири таких самих S-блоків. Результат роботи функції гешування може мати довжину від 8 до 512 біт.

Проведено аналіз існуючих функцій гешування, отримано їх порівняльну характеристику. Помічена загальна тенденція втрати швидкості шифрування у криптостійких, на сьогоднішній день, функцій гешування. Тому є необхідність розробки нових та удосконаленні функцій гешування. Результати порівняльного аналізу відображені в табл. 1.

Порівняльний аналіз функцій гешування Таблиця 1

№ п/п	Функції гешування	Макс розмір повідомлення	Довжина геш-значення (біт)	Шв. шифрування (Мбіт/с)	Розмір блоку	К-сть раундів	Розмір слова	Стійкість до методів крипто-аналізу
1.	MD4	$<2^{64}$	128	2,36	512	48	32	-
2.	MD5	$<2^{64}$	128	1,74	512	64	32	-
3.	SHA-1	$<2^{64}$	160	0,75	512	80	32	-/+
4.	SHA-2/256	$<2^{64}$	256	1,85	512	64	32	+
5.	SHA-2/512	$<2^{128}$	1024	1,76	1024	80	64	+
6.	ГОСТ Р 34.11-2012	$<2^{64}$	256	0,11	512	12	32	+
7.	RIPEMD(160)	$<2^{64}$	128	3,60	512	8	32	-
8.	N-hash (12 етапів)	$<2^{64}$	128	1,66	512	8	128	-
9.	Snerfu	$<2^{64}$	128	1,70	512	64	128	-
10.	Купина–512	$<2^{96}$	512	3,25	1024	10	64	+

Викладення основного матеріалу

Розроблено функцію гешування newMD4, який частково зберіг основну структуру, але отримав декілька нововведень порівняно із оригінальним MD4:

1) при стисненні замість чотирьох 32-бітних змінних A, B, C, D запропоновано використання п'яти 64-бітних змінних A, B, C, D, J;

2) збільшено довжину геш-значення до 256-біт;

3) збільшено кількість етапів виконання стиснення;

4) замінені додаткові функції F, G, H;

5) додані додаткові операції на кожному етапі.

Опишемо алгоритм роботи запропонованого алгоритму newMD4.

Нехай маємо повідомлення M (довжиною T – біт) потрібно визначити геш-код H (довжиною 256 біт). Для цього потрібно виконати наступні етапи:

Етап 1. Додавання додаткових бітів заповнення до повідомлення M.

Даний етап виконується так: $M = M + XX$, де XX – додаткові біти повідомлення.

Тобто розмір повідомлення має бути таким, щоб після додавання 64 бітів розмір став кратним 512. Досягнення потрібного розміру здійснюється додавання одного біту із значенням «1» і всіх наступних із значенням «0», до потрібного розміру. При цьому варто врахувати діапазон кількості біт XX.

Етап 2. Додавання в кінець повідомлення M 64-бітного розміру повідомлення $M : M = M + YY$, де YY – 64-бітове представлення b (розмір вихідного повідомлення в бітах) додається в кінець результату попереднього етапу. При цьому якщо $b > 2^{64}$, використовуються тільки молодші 64 біти значення b. Біти додаються в кінці як два 32-бітних слова, в яких молодший байт розміщується на початку, як було обумовлено вище.

Етап 3. Ініціалізація буфера MD.

При розрахунку геш-коду буде використовуватись 5 слів A, B, C, D, J. Кожне із слів A, B, C, D, J представляє собою 64-бітовий регістр. Ці регістри ініціалізуються приведеними нижче значеннями (16-річне представлення, спочатку молодший байт):

$$A = 2341\,8673\ 2341\,8673,$$
$$B = \text{A674 28CF A674 28CF},$$
$$C = 6719\,\text{E563}\,6719\,\text{E563},$$
$$D = 4921\,\text{B73A}\ 4921\,\text{B73A},$$
$$J = 59\text{B2}\,47\text{A8}\,\text{D63F}\,\text{C34B}.$$

Етап 4. Обробка повідомлення M функціями стиснення.

На цьому етапі кожне із 64-бітних слів A, B, C, D, J змінюється функціями стиснення $Fun1$, $Fun2$, $Fun3$, $Fun4$ з використанням блоків вихідного повідомлення M. Повідомлення M розкладається на частини по 1024 біта, кожна з яких обробляється функціями стиснення. В результаті у векторах внутрішнього стану A, B, C, D, J буде акумулюватись тимчасове геш значення після кожної частини M. Результуюче геш значення повідомлення M отримується конкатенацією векторів внутрішнього стану A, B, C, D після обробки усіх 1024-бітних частин повідомлення M. Тобто, виписується 256 біт, починаючи з молодшого біта A, і закінчуючи старшим бітом D.

Процедура стиснення кожного блоку виконується у чотири кроки:

Крок 1. Спочатку виконується 16 операцій на основі функції стиснення $Fun1$, що дорівнюють $Fun1 = \left(A \oplus F(B,C,D,J) + X[k]\right) <<< s$, де k – індекс блоку даних, що обробляється, s – кількість разів виконання побітового циклічного зсуву вліво, $X(i)$ – i-блок повідомлення M, F – нелінійна функція, $F(X,Y,Z) = = (X \wedge \neg Y) \vee (\neg X \vee Z)$.

Крок 2. Далі виконується 16 операцій на основі функції стиснення $Fun2$, що дорівнює $Fun2 = \left(A \oplus G(B,C,D,J) \oplus X[k] + 5A827999\right) <<< s$, де k – індекс блоку даних, що обробляється, s – кількість разів виконання побітового циклічного зсуву вліво, $X(i)$ – i-блок повідомлення M, G – нелінійна функція, $G(X,Y,Z) = (X \wedge Y) \vee (Y \wedge Z) \wedge \wedge (X \oplus Z)$.

Крок 3. Потім виконується 16 операцій на основі функції стиснення $Fun3$, що дорівнює $Fun3 = \left(A \oplus H(B,C,D,J) \oplus X[k] + 6ED9EBA1\right) <<< s$, де k – індекс блоку даних, що обробляється, s – кількість разів виконання побітового циклічного зсуву вліво, $X(i)$ – i-блок повідомлення M, H – нелінійна функція, $H(X,Y,Z) = \neg(X \oplus Y \oplus Z)$.

Крок 4. Наприкінці виконується 16 операцій на основі функції стиснення $Fun4$, що дорівнює: $Fun4 = \left(A \oplus R(B,C,D,J) \oplus X[k] + 7952BAE1\right) <<< s$, де k – індекс блоку даних, що обробляється, s – кількість разів виконання побітового циклічного зсуву вліво, $X(i)$ – i-блок повідомлення M, R – нелінійна функція, $R(X,Y,Z) = (Y \oplus Z) \wedge X$.

Експериментальні дослідження запропонованої функції гешування newMD4

Для проведення експериментального дослідження було розроблено програмні реалізації функцій гешування MD4 та newMD4 у вигляді консольного додатку. Проведено експериментальні дослідження, щодо оцінки швидкісних та статистичних характеристик запропонованої функції гешування.

Статистичні властивості послідовностей, утворених за допомогою функцій гешування MD4 та newMD4 досліджено у середовищі статистичних тестів NIST STS. Діаграма проходження тестів NIST STS послідовністю, що згенерована функцією гешування newMD4 наведена на рис. 1.

У табл. 2 для порівняння наведено результати дослідження послідовностей, сформованих функціями гешування MD4 та newMD4. Як видно з результатів, функція гешування newMD4 пройшла комплексний контроль за методиками NIST STS та показала кращі результати ніж оригінальна функція гешування.

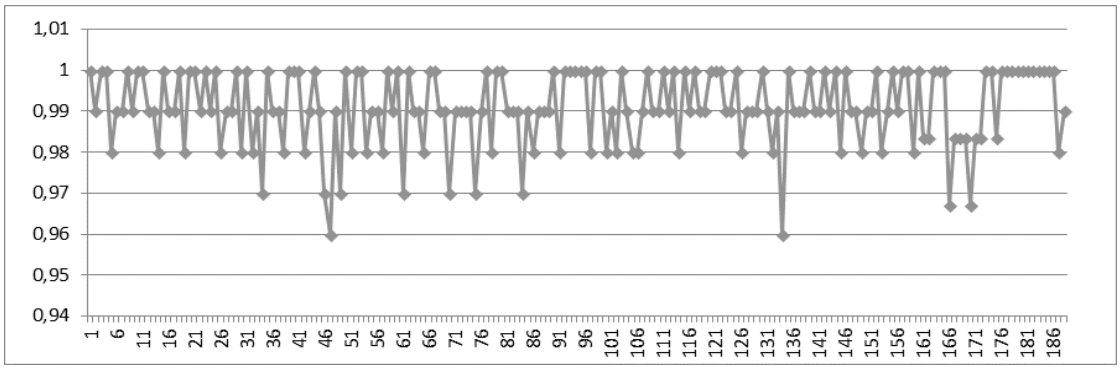

Рис.1. Діаграма проходження тестів NIST STS послідовності, що згенерована за допомогою функції гешування newMD4

Результати статистичного дослідження за методикою NIST STS Таблиця 2

Генератор	Кількість тестів, у яких тестування пройшли >=99% послідовностей	Кількість тестів, у яких тестування пройшли >=96% послідовностей	Кількість тестів, в яких значення P>0,01	Кількість тестів, в яких значення P>0,001
MD4	130 (69,1%)	188 (100%)	132 (70,2%)	188 (100%)
newMD4	146 (77,6%)	188 (100%)	140 (74,5%)	188 (100%)

Також досліджені *швидкісні характеристики* функції гешування. Тестування проводилося на трьох комп'ютерах (їхні характеристики приведені у табл. 3).

Експериментально показано, що розроблена функція гешування newMD4 має кращу швидкодію ніж функція гешування MD4 у 1,43 разів (табл. 4).

Характеристика комп'ютерів на яких проводилось експериментальне дослідження Таблиця 3

№	ОС	Процесор	Кількість ОП
1	Windows 7 (64 bit)	Intel Core i5-2410M 2,3 ГГц	4 Гб
2	Windows 7 (64 bit)	Intel Core i5-2410M 2,5 ГГц	6 Гб
3	Windows 8	Intel Core i5-2410M 2,3 ГГц	4 Гб

Результати перевірки швидкісних характеристик Таблиця 4

Розмір файлу	newMD4						MD4					
	1		2		3		1		2		3	
	t, с	v, МБ/с	t, с	v, МБ/с	t, с	v, МБ/с	t, с	v, МБ/с	t, с	v, МБ/с	t, с	v, МБ/с
1 МБ	0,047	21,27	0,038	26,31	0,044	22,72	0,068	14,70	0,061	16,39	0,067	14,92
10 МБ	0,43	23,25	0,33	30,30	0,41	24,39	0,64	15,62	0,58	17,24	0,63	15,87
100 МБ	3,93	25,44	3,23	30,95	3,39	29,49	5,62	17,79	4,87	20,53	5,24	19,08
1 ГБ	38,26	26,13	31,89	31,35	37,14	26,92	55,06	18,16	47,08	21,24	55,71	17,95

Висновки

У роботі запропоновано удосконалена функція гешування MD4 − newMD4. За однакових умов, проведені експериментальні дослідження з оцінки швидкісних характеристик даних функцій гешування, які показали, що запропонована функція гешування newMD4 дозволяє підвищити швидкодію у 1,43 разів порівняно із оригінальною MD4. Також, досліджено статистичні властивості послідовностей, що формувалися розглянутими функціями гешування. В результаті показано, що послідовності, які згенеровано за допомогою функції гешування newMD4 показали кращі статистичні характеристики порівняно із оригіналом.

Дослідження удосконаленої функції гешування варто продовжити і перевірити її стійкість до колізій першого і другого роду. Для виявлення колізій першого роду потрібно до обраного тексту спробувати підібрати інший текст у якого було б таке ж геш значення, як і у першого тексту. Для підтвердження стійкості до колізій другого роду, має бути обчислювально неможливо підібрати пару повідомлень, які мають однакове геш значення.

Література

[1] В. Мухачева, В. Хорошко. «Методы практической криптографии», К.: ООО «Полиграф Консалтинг», 2005. С.215.

[2] Хеш функції и цифрові підписи. URL: http://izi.vlsu.ru/teach/books/913/theory.html.

[3] Б. Шнайер, Н. Фергюсон «Практическая криптография», М: Диалектика, 2005. С. 105.

[4] A. Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. URL: https://www.iacr.org/archive/crypto2004/31520306/multicollisions.pdf.

[5] В. Бондаренко. «Введение в криптографию» : Учебное пособие. С: СГУ, 2000. С. 234.

[6] R. Rivest. «The MD4 Message-Digest Algorithm». URL: https://tools.ietf.org/html/rfc1320.

[7] R. Rivest. «The MD5 Message-Digest Algorithm». URL: https://www.ietf.org/rfc/rfc1321.

[8] D. Eastlake. «US Secure Hash Algorithm 1 (SHA1)». URL: https://tools.ietf.org/html/rfc3174.

[9] D. Bider. «SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol». URL: https://tools.ietf.org/html/rfc6668.

[10] V. Dolmatov. «GOST R 34.11-2012: Hash Function». URL: https://tools.ietf.org/html/rfc6986.

[11] A. Keromytis. «The Use of RIPEMD-160-96 within ESP and AH». URL: https://www.ietf.org/rfc/rfc2857.txt.

[12] S. Miyaguchi, K. Ohta, M. Iwata. «128-bit hash function (N-hash)». *NTT Review*. № 2 (6). 1996. P. 128–132.

[13] E. Biham, A. Shamir. «Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer» (Extended Abstract).

[14] C. Dobraunig. «Analysis of the Kupyna-256 Hash Function». URL: https://eprint.iacr.org/2015/956.pdf.

УДК 004.056.2 (045)

Остапенко Н.В., Кинзерявый В. Н., Кириченко К.С., Грицак А.В. Усовершенствована функция хеширования MD4

Аннотация. *Сегодня информация рассматривается как стратегический ресурс. Модификация информации или её незаконное распространение может привести к серьезным последствиям. Процесс обеспечения защиты информации усложняется с развитием вычислительных машин. Целостность важных файлов операционной системы, программ или данных контролируется функцией хеширования. Хеширование используется для построения ассоциативных массивов, поиска дубликатов в сериях наборов данных, построения уникальных идентификаторов для набора данных, контрольного суммирования с целью определения случайных или преднамеренных ошибок при сохранении или передаче, для сохранения паролей в системах защиты, при выработке электронной подписи. Совсем недавно отрасль криптографии, связанная с хешированием столкнулась с весомой проблемой - обеспечением устойчивости к мультиколизиям, которые используют атаку Жукс. В своей работе Жукс показал, что устойчивость хеш-значение, вычисляют путем каскадирования этих функций ненамного больше, чем устойчивость одной из них. Кроме того, известные алгоритмы хеширования не позволяют в полной мере решать вопросы обеспечения криптографической стойкости и высокого быстродействия алгоритмов. Поэтому разработка новых и усовершенствование существующих функций хеширования с целью повышения эффективности криптографической защиты не потеряет своей актуальности. Учитывая это, в работе предложено функцию хеширования newMD4, которая была разработана на основе оригинальной функции хеширования MD4. Разработана функция хеширования newMD4 имеет несколько нововведений по сравнению с MD4: при сжатии вместо четырех 32-битных переменных* A, B, C, D *предложено использование пяти 64-битных переменных* A, B, C, D, J; *увеличена длина хеш-значения до 256-бит; заменены дополнительные функции* F, G, H; *добавлены дополнительные операции на каждом этапе. В работе проведены экспериментальные исследования по оценке скоростных и статистических характеристик предложенной функции хеширования. При одинаковых условиях, проведенные экспериментальным исследования по оценке скоростных характеристик, которые показали, что функция хеширования newMD4 быстрее, чем оригинальный MD4 в 1,43 раз. Для исследования статистических характеристик были использованы тесты NIST STS, при этом функции хеширования использовались для генерации последовательностей, статистические характеристики которых проверялись указанными тестами. Согласно результатам исследования, последовательности, которые были сгенерированы с помощью функции хеширования newMD4 показали лучшие статистические характеристики по сравнению с оригиналом.*

Ключевые слова: *криптография, функция хеширования, целостность информации, защита информации, криптостойкость.*

Ostapenko N., Kinzeryavyy V., Kyrychenko K., Grycak A. Advanced hash function MD4

Abstract. *Today information is seen as a strategic resource. Modification of the information or it's illegal dissemination can lead to serious consequences. The process of ensuring information protection get's complicated with the development of computing machines. The integrity of important operating system files, programs or data controled of hash function. Hashing is used to construct associative arrays and duplicate in a series of datasets, build unique identifiers for a set of data, with a view to determining the Checksumming accidental or deliberate errors in you save or transfer, to save passwords on systems of protection, in the formulation of an electronic signature. More recently, industry related cryptographic hash faced a significant challenge-sustainability to multikolizijam that use attack Zhuks. In his work Zhuks has shown that sustainability, calculates the hash value by cascading these functions, not much more than the stability of one of them. In addition, the known hash algorithms do not allow to fully address the issues of ensuring the durability and performance of cryptographic algorithms. Therefore, developing new and improving existing hash functions with a view to enhancing the effectiveness of cryptographic protection will not lose its relevance. With this in mind, in the work of the proposed hash function newMD4, which was developed on the basis of the original hash function MD4. Designed hash function newMD4 has several innovations compared to MD4: if you are compressing instead of four 32-bit variables* A, B, C, D *proposed using five 64-bit variable* A, B, C, D, J; *increased the length of the hash value to 256-bits; replaced by additional functions* F, G, H; *added additional operations at each stage. In the work of the experimental research on the evaluation of Expressway and the statistical characteristics of the proposed hash function. Under the same conditions, conducted pilot studies to assess mental speed characteristics, which showed that the hash function newMD4 faster than the original MD4 in 1,43 times. For research of statistical characteristics of NIST tests were used in the STS, at the same hash function used to generate sequences, statistical characteristics of teristiki which tested the specified tests. According to the research results, consistency, using the hash function sgenerirovanye newMD4 showed better statistical characteristics compared to the original.*

Key words: *cryptography, hash function, information integrity, information protection, cryptestability.*

Отримано 01 *липня 2018 року*, затверджено редколегією 22 *липня 2018 року*