УДК 003.26

АЛГОРИТМІЧНІ АСПЕКТИ ПОШУКУ ПРООБРАЗІВ ГЕШ-ФУНКЦІЙ НА ПРИКЛАДІ MD5

Антон Кудін, Богдан Коваленко

У даній статті розглядаються аспекти застосування метода Аокі побудови прообразу функцій хешування, що базуються на схемі Меркла-Дамгарда (на прикладі функції MD5). На відміну від оригінального методу, де використовується декілька диференційних шляхів для побудови теоретичної атаки, у даній роботі увага приділяється побудові потужних множини диференційних шляхів високої ймовірності, без яких принципово неможливо будувати практичні атаки такого типу. Пропонується одна з можливих схем побудови множини диференційних шляхів високої ймовірності для даної атаки та наводиться оцінка складності побудови прообразу для скорочених версій функції хешування MD5. Отримані результати доводять теоретичну слабкість до атаки скороченого алгоритму хешування MD5, а також накладають додаткові обмеження на конструкції нових функцій хешування.

Ключові слова: *захист інформації, функція хешування, прообраз, колізія, диференційний шлях, «зустріч посередині», MD5, схема Меркла-Дамгарда.*

Вступ. Функція хешування – це відображення, що перетворює повідомлення довільної довжини у хеш-код фіксованої довжини n, тобто $F:\{0,1\}^* \to \{0,1\}^n$. Криптографічна хеш-функція повинна мати додаткові властивості, основні з яких: стійкість до слабкої колізії, стійкість до сильної колізії та стійкість до побудови прообразу.

Більшість сучасних криптографічних протоколів використовують функції хешування. Як правило, надійність таких функцій є критичною для надійності усього протоколу. Оскільки сам хеш-код певною мірою ідентифікує деяку інформацію, то хеш функція повинна забезпечувати конфіденційність цієї інформації (бути стійкою до пошуку прообразу) та цілісність інформації (слабка та сильна стійкість до колізії). Варто зазначити, що стійкість до пошуку прообразу є найбільш критичною властивістю криптографічної функції хешування, оскільки усі інші атаки можна легко звести до пошуку прообразу.

У ході роботи розглядалися методи побудови прообразу на функції хешування родини MD4. Функції цієї родини використовують схему Меркла-Дамгарда [1] та мають вигляд:

$$F:\{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n. \qquad (1)$$

На основі ідеї Кнельвольфа [3] побудови прообразу за допомогою множин диференціальних шляхів, що використовуються для побудови слабкої колізії, запропоновано алгоритм побудови множин диференціальних шляхів, а також атаки пошуку прообразу на алгоритм хешування MD5 [2] з їх використанням. Також було запропоновано та реалізовано алгоритм пошуку оцінки ефективності даної атаки на прикладі алгоритму хешування MD5. Атака легко може бути перенесена на інші функції хешування родини MD4.

Побудова прообразу на основі диференціальних шляхів

У даному пункті розглядається можливість побудови одноблокового прообразу з використанням множини спеціальних диференціальних шляхів [3] з високою ймовірністю

Прообразом функції хешування називається таке повідомлення $M \in \{0,1\}^*$, що для заданого хеш-коду $H \in \{0,1\}^n$ виконується рівність:

$$F(IV, M) = H, \qquad (2)$$

де IV – стандартний стартовий вектор.

Розглянемо метод зустрічі посередині для знаходження одноблокового прообразу.

Ідея Кнельвольфа полягає у використанні множини диференціальних шляхів слабкої колізії, що мають високі ймовірності.

Припустимо, що ми можемо провести декомпозицію:

$$F(IV, M) = F_2 \circ F_1(IV, M) \equiv F_2(F_1(IV, M), M). \qquad (3)$$

Таким чином, що функції F_1 та F_2^{-1} легко обчислюються (з відомих IV та M).

Якщо ми маємо дві множини повідомлень S^1 S^2 і при цьому були знайдені такі $i \in S^1, j \in S^2$, що $F_1(IV, M + i + j) = F_2^{-1}(H, M + i + j)$, то значення $M + i + j$ буде прообразом для хеш-коду H. Проте очевидно, що складність такого перебору буде порядку 2^{128} викликів функцій F_1 та F_2^{-1}. Основний крок атаки полягає у тому, що з великою ймовірністю може виконуватися рівність $F_1(IV, M + j) + i = F_2^{-1}(H, M + i) + j$, якщо i та j – диференціальні шляхи слабкої колізії для функцій F_1 та F_2^{-1} з високими ймовірностями. Тобто

рівняння набуває вигляду $F_1(IV, M+j)-j = F_2^{-1}(H, M+i)-i$ і в цьому випадку, внаслідок парадоксу днів народжень, складність може зменшитись до 2^{64} викликів функцій F_1 та F_2^{-1}.

Припустимо, що у нас для функцій F_1 та F_2^{-1} існують набори $\Delta^1, \Delta^2 \subset \{0,1\}^m$, $D^1, D^2 \subset \{0,1\}^n$ та відомі ймовірності:

$$p_d^1 = P\{F_1(IV,M) - F_1(IV,M+d) = \Delta_d^1\}, d \in D^1$$
$$p_d^2 = P\{F_2^{-1}(IV,M) - F_2^{-1}(IV,M+d) = \Delta_d^2\}, d \in D^2, \quad (4)$$

де ймовірності $\{p_d^1\}$, $\{p_d^2\}$ мають бути достатньо високими.

Опишемо алгоритм пошуку прообразу.

Вхід: Набори, $D_1, D_2, \Delta^1, \Delta^2$, стартовий вектор *IV*, хеш-код *H*.

Результат: прообраз.

1. $M \leftarrow Rand()$.
2. Для усіх $d_2 \in D^2$ сформувати список:
$$L_1[d_2] \leftarrow F_1(IV, M+d_2) - \Delta_{d_2}^2.$$
3. Для усіх $d_1 \in D^1$ сформувати список:
$$L_2[d_1] \leftarrow F_2^{-1}(H - IV, M+d_1) - \Delta_{d_1}^1.$$
4. Знайти такі $(d_1, d_2) \in D^1 \times D^2$, що $L_1[d_2] == L_2[d_1]$. Якщо при цьому $F(IV, M+d_1+d_2) == H$, то повернути $M+d_1+d_2$ інакше повернутися до кроку 1.

Покажемо коректність роботи алгоритму. З ймовірністю $p_{d_2}^2$ буде справджуватися $\Delta_{d_2}^2 = F_2^{-1}(H-IV, M+d_1+d_2) - F_2^{-1}(H-IV, M+d_1)$ і з ймовірністю $p_{d_1}^1$ буде справджуватися $\Delta_{d_1}^1 = F_1(IV, M+d_1+d_2) - F_1(IV, M+d_2)$. Тепер, якщо ми дійсно знайшли такі $d_1 \in D^1$ та $d_2 \in D^2$, що $F_1(IV, M+d_2) - \Delta_{d_2}^2 = F_2^{-1}(H-IV, M+d_1) - \Delta_{d_1}^1$, то це означатиме, що з ймовірністю $p_{d_2}^2 \cdot p_{d_1}^1$ отримаємо $F_1(IV, M+d_1+d_2) = F_2^{-1}(H-IV, M+d_1+d_2)$, а отже $F(IV, M+d_1+d_2) = H$, $M+d_1+d_2$ – прообраз.

Наведемо приблизну оцінку складності алгоритму. Оцінимо складність алгоритму. Нехай потужності множин диференціалів повідомлень $|D^1| = 2^{r_1}$, $|D^2| = 2^{r_2}$, а ймовірності диференціалів нижньої та верхньої частин шляхів можна оцінити як $\tilde{p}_1 = 2^{-r_1}\sum_{d \in D^1} p_d^1$, $\tilde{p}_2 = 2^{-r_2}\sum_{d \in D^2} p_d^2$. Тоді одна ітерація алгоритму покриватиме приблизно $p_1 p_2 2^{r_1+r_2}$ варіантів. Складність ітерації буде приблизно $\frac{2^{r_1}+2^{r_2}}{2}$ викликів функції стискання (вважаючи, що складність обчислення F_1 приблизно така ж, як F_2^{-1} і вдвічі менша за F). Необхідна кількість ітерацій складає приблизно $\frac{2^{n-r_1-r_2}}{p_1 p_2}$, де n – довжина виходу хеш-функції. В алгоритмі наявні помилки 2-го роду, тобто може повертатися повідомлення, що не є прообразом. На одній ітерації таких помилок буде приблизно $(1-p_1p_2)2^{r_1+r_2-n}$, відповідно на повній кількості ітерацій складатиме $\frac{1}{p_1 p_2} - 1$. Проте оскільки значення $p_1 >> 0, p_2 >> 0$, то помилкою 2-го роду можна знехтувати. Отже отримуємо оцінку складності

$$L^* \approx \frac{2^{n-1}}{p_1 p_2} \cdot \left(\frac{1}{2^{r_1}} + \frac{1}{2^{r_2}}\right)\Gamma_{max}, \quad (5)$$

де *n* – довжина виходу хеш-функції, $\Gamma_{max} = max\{\Gamma_\uparrow, \Gamma_\downarrow\}$, $\Gamma_\uparrow$ та $\Gamma_\downarrow$ – складність обчислення нижньої та верхньої частини функції стискання відповідно.

З рівняння (5) та виходячи з того, що $L^* < 2^n$ отримуємо критерій доцільності побудови атаки:

$$J = \frac{1}{2 p_1 p_2} \cdot \left(\frac{1}{2^{r_1}} + \frac{1}{2^{r_2}}\right) \cdot \frac{\Gamma_{max}}{\Gamma_{full}} < 1. \quad (6)$$

Далі, позначатимемо кортежем $\langle S_\uparrow, S_{meet}, S_\downarrow \rangle$ схему атаки, де $(S_\uparrow, S_{meet})$ – кроки, яким відповідає функція F_1, $(S_{meet}-3, S_\downarrow)$ – F_2, що введені в (3). При чому, якщо $S_\uparrow > S_{meet}$, то це означає, що розглядаються кроки $S_\uparrow - S_{max}$ та $0 - S_{meet}$, де S_{max} – максимальний крок. Для випадку $S_{meet} - 3 > S_\downarrow$ – аналогічно, розглядаються кроки $S_{meet} - 3 - S_{max}$ та $0 - S_\downarrow$.

Пошук прообразу через псевдопрообраз

Іноді для знаходження прообразу ефективніше є пошук псевдопрообразу, а потім його конвертація до прообразу.

Для функції (1) псевдопрообразом для хеш-коду H називатимемо таку пару $\langle V, M \rangle$,

$M \in \{0,1\}^n$ та $V \in \{0,1\}^m$, який не обов'язково дорівнює IV, що виконуватиметься:

$$\mathrm{F}(V, M) = H. \quad (7)$$

Якщо є можливість ефективно, значно швидше за повний перебір, знаходити псевдопрообраз, то можна також, в загальному випадку, ефективно будувати двоблоковий прообраз, використовуючи парадокс днів народжень. У такому випадку складність пошуку двоблокового прообразу буде:

$$L = \sqrt{2^m \times \Gamma_{pseudo}}, \qquad n^* = \sqrt{\frac{2^m}{\Gamma_{pseudo}}}, \quad (8)$$

де Γ_{pseudo} – складність пошуку псевдопрообразу(кількість викликів функції F), $\Gamma_{pseudo} < 2^{m-1}$.

Дослідження способів генерації диференціальних шляхів на прикладі MD5

Алгоритм MD5 працює, як нелінійний регістр зсуву з зовнішнім керуванням. Початкове заповнення регістру — фіксований 128-бітний стартовий вектор. Після виконання 64-х кроків до отриманого значення додається початкове заповнення. У випадку, якщо блоки повідомлення (зовнішнє керування) ще не закінчилися, отримане значення стає початковим заповненням для наступного блоку, інакше це заповнення стає хеш-кодом.

Нижче описані функції MD5Compress та MD5. Перша виконує раунди хешування для одного 512-бітного блоку повідомлення при заданому початковому векторі, в друга виконує хешування усього повідомлення з необхідною попередньою обробкою даних, використовуючи функцію MD5Compress. В обох функціях використовуються такі операції над 32-бітними словами, якими представленні усі дані, як «$\wedge$»(побітове логічне І), «$\vee$» (побітове логічне АБО), «$\oplus$» (побітове логічне сумування за модулем 2), «$\neg$» (інверсія бітів), «+» (сумування слів за модулем 2^{32}).

Робота алгоритму MD5Compress:

Вхід алгоритму: блок повідомлення m – масив з 16 4-байтних слів, IHV_i – стартовий вектор-масив з 4-х 4-байтних слів.

Вихід алгоритму: IHV_{i+1} – результуючий вектор.

1. Ініціалізація констант Addition Constant:

$$AC_t = \lfloor 2^{32} \,|\sin(t+1)|\, \rfloor, t = 0..63. \quad (9)$$

2. Rotation Constants:

$$(RC_t, RC_{t+1}, RC_{t+2}, RC_{t+3}) = \begin{cases} (7,12,17,22), & t=0,4,8,12, \\ (5,9,14,20), & t=16,20,24,28, \\ (4,11,16,23), & t=32,36,40,44, \\ (6,10,15,21), & t=48,52,56,60. \end{cases} \quad (10)$$

3. Визначення раундової функції:

$$f(X,Y,Z)_t = \begin{cases} F(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z), & t=0..15, \\ G(X,Y,Z) = (Z \wedge X) \vee (Z \wedge Y), & t=16..31, \\ H(X,Y,Z) = X \oplus Y \oplus Z, & t=32..47, \\ I(X,Y,Z) = Y \oplus (\neg Z \vee X), & t=48..63. \end{cases} \quad (11)$$

4. Визначення розкладу повідомлення:

$$W_t = \begin{cases} m_t, & t=0..15, \\ m_{(1+5t) \bmod 16}, & t=16..31, \\ m_{(5+3t) \bmod 16}, & t=32..47, \\ m_{7t \bmod 16}, & t=48..63. \end{cases} \quad (12)$$

5. Ініціалізація регістра зсуву:

$$IV = (Q_{-3}, Q_0, Q_{-1}, Q_{-2}) = \\ = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476) \quad (13)$$

6. Власне ітерації алгоритму, $t = 1..64$:

$$\begin{cases} F_t = f_t(Q_t, Q_{t-1}, Q_{t-2}); \\ T_t = F_t + Q_{t-3} + AC_t + W_t; \\ R_t = RL(T_t, RC_t); \\ Q_{t+1} = Q_t + R_t, \end{cases} \quad (14)$$

де $(Q_0, Q_{-1}, Q_{-2}, Q_{-3}) = (IHV_i[1], IHV_i[2], IHV_i[3], IHV_i[0])$.

6. Результуючим вектором буде значення:

$$MD5Compress(IHV, B) = \\ = (IHV_i[0] + Q_{61}, IHV_i[1] + Q_{64}, IHV_i[2] + Q_{63}, IHV_i[3] + Q_{62}) = \\ = (IHV_{i+1}[0], IHV_{i+1}[1], IHV_{i+1}[2], IHV_{i+1}[3]). \quad (15)$$

Робота алгоритму MD5:

Вхід алгоритму: M – довільне повідомлення, довжиною не більшше як $2^{64}-1$ байтів, масив 4-х байтних слів. Вихід алгоритму: H – масив з 4-х 4-х байтних слів – хеш-код.

1. Паддінг. Додається 1 до кінця повідомлення, а потім доповнюється нулями до довжини 512n+448, в решту 64 біта записується кількість бітів повідомлення. Отримаємо масив $\{M_i\}_{i=0..n}, M_i = \{X_j^i\}_{j=0..15}$.

2. Виконуються обчислення $IHV_{i+1} = MD5Compress(IHV_i, M_i), i = 0..n$,

$IHV_0 = \{0x67452301, 0xefcdab89, 0x98adcfe, 0x10325476\}$.

3. Значення IHV_{n+1} і буде хеш-значенням H.

Алгоритми побудови диференціальних шляхів

Для атаки, що розглядається, під диференціальним шляхом розумітимемо набір бітових умов або диференціальну характеристику та ймовірність його проходження. Власне нас зараз цікавитимуть лише умови на крайні 4 слова (різниця на виході) та ймовірність цієї різниці.

Варто зазначити, що побудова оптимальних шляхів та знаходження оптимальних диференціалів є складною задачею і в загальному випадку тут не розглядається.

Варто також зазначити, що дві раундові функції алгоритму MD5 мають властивість поглинання («absorbtion property») для бітових умов. Ця властивість для булевої функції F означає, що:

$$\forall \pi \in \{0,1,2\}, \forall x_0 \in \{0,1\}\ \exists x_1, x_2 \in \{0,1\}: \begin{cases} x'_0 = x_0 \oplus 1 \\ x'_1 = x_1 \\ x'_2 = x_2 \\ F(x_\pi, x_{\pi+1}, x_{\pi+2}) = F(x'_\pi, x'_{\pi+1}, x'_{\pi+2}) \end{cases}, \quad (16)$$

де під «+» розуміється додавання за модулем 3, а під знаком $\oplus$ – додавання за модулем 2.

Для побудови множини нижніх диференціальних шляхів використовуються диференціали виду $\delta W = \langle 0,0,...,\delta w_i, \delta w_{i+1}, 0,0, \delta w_{i+4}, 0,... \rangle$ де $\delta w_i = 1$, $\delta w_{i+1} = -2^{RC_i - RC_{i+1} \bmod 32}$, $\delta w_{i+4} = -2^{RC_i \bmod 32}$, де значення $\{RC_i\}$ визначені в (10), а W визначені в (12). Саму різницю повідомлень позначатимемо як $\mathsf{D}^i_{\uparrow}$. Диференціал δw_i називатимемо далі збуджуючим, а інші – компенсуючими. Властивість (16) дозволяє для такого шляху отримувати ймовірність $p \approx 2^{-4}$ до чергової появи у розкладі повідомлення ненульового диференціалу з цього шляху.

Для побудови множини верхніх диференціальних шляхів використовуються диференціали виду $\delta W = \langle 0,0,...,\delta w_i, \delta w_{i-1}, \delta w_{i-2}, \delta w_{i-3}, \delta w_{i-4}, 0,... \rangle$ де $\delta w_i = 1$, $\delta w_{i-1} = 2^{-RC_{i-1} \bmod 32}$, $\delta w_{i-2} = 2^{-RC_{i-2} \bmod 32}$, $\delta w_{i-3} = 2^{-RC_{i-3} \bmod 32}$, $\delta w_{i-4} = -2^{-RC_{i-4} \bmod 32}$. Саму різницю повідомлень позначатимемо як $\mathsf{D}^i_{\downarrow}$. Найбільша ймовірність диференціального шляху $p \approx 2^{-4}$ до чергової появи у розкладі повідомлення ненульового диференціалу з цього шляху.

Зрозуміло, що диференціальних шляхів заданих таким чином та їх лінійних комбінацій є дуже багато. Тепер виникає задача пошуку такої підмножини з усіх можливих диференціальних шляхів, що диференціали матимуть високі ймовірності, а також виконувався критерій та мінімізувався функціонал (6).

Введемо штраф або вартість для диференціала в реалізації атаки. Покажемо це для множини нижніх диференціальних шляхів, для множини верхніх усі міркування аналогічні. Для диференціалу δW він виглядатиме як:

$$\varepsilon(\delta W, S_{start}, S_{finish}) = \\ = \varepsilon_{head}(\delta W, S_{start}, S_{finish}) + \varepsilon_{tail}(\delta W, S_{start}, S_{finish}). \quad (17)$$

В формулі (17) повний штраф диференціального шляху поданий у вигляді суми двох штрафів: ε_{head} – вартість на виході локальної колізії, та ε_{tail} – штраф при проходженні кроків, де $\delta w_i \neq 0$, при тому, що δw_i уже використовувався у локальній колізії (тобто кількість бітових умов починає не спадати).

Припустимо, що диференціал поданий у вигляді: $\delta W = \sum_{i=0}^{N} a_i \cdot \mathsf{D}^{k_i}_{\uparrow}$, $k_i \in \{0,...,63\}$. Тоді оцінити штраф ε_{head} можна як:

$$\varepsilon_{head}(\delta W, S_{start}, S_{finish}) = \sum_{i=0}^{N} \varepsilon'(i, S_{start}, S_{finish})$$

$$\varepsilon'(i, S_{start}, S_{finish}) = \begin{cases} 0, & i \geq S_{finish} - 1 \\ (S_{finish} - i) \cdot |a_i|, & S_{finish} - 4 \leq i < S_{finish} - 1, \\ |a_i|, & i > S_{finish} - 4 \end{cases} \quad (18)$$

де операція $|\cdot|$ повертає вагу Хеммінга аргументу.

Штраф ε_{tail} можна подати у рекурсивному вигляді:

$$\varepsilon_{tail}(\delta W, S_{start}, S_{finish}) = \sum_{i=S_{start}}^{S_{finish}-1} \varepsilon''(i)$$

$$\varepsilon''(i) = \begin{cases} 0, & i \leq \max\{k_i\} \\ \min(16, \varepsilon''(i-1) + \varepsilon''(i-2) + \varepsilon''(i-3) + \varepsilon''(i-4) + |W_i|), & i > \max\{k_i\}. \end{cases} \quad (19)$$

Тепер необхідно сформулювати алгоритм вибору підмножини диференціальних шляхів.

Розглянемо довільний диференціал, що поданий у вигляді $\delta W = \sum_{i=0}^{N} a_i \cdot \mathrm{D}_{\uparrow}^{k_i}$, $k_i \in \{0,\ldots,63\}$. Оцінкою його повного штрафу може бути:

$$\varepsilon(\delta W, S_{start}, S_{finish}) = \sum_{i=0}^{N} |a_i| \cdot \varepsilon(\mathrm{D}_{\uparrow}^{k_i}). \qquad (20)$$

Наближення (20) може являтися оцінкою знизу для штрафу. Також вважатимемо, що штраф - логарифм ймовірності диференціального шляху зі знаком «-».

Для оцінки складності атаки пошуку прообразу на скорочений S_{red}-раундовий алгоритм необхідно встановити $S_{\uparrow} = 0$, $S_{\downarrow} = S_{red}$, побудувати оцінку для кожної схеми $\langle S_{\uparrow}, S_{meet}, S_{\downarrow} \rangle$, де M_{meet} перебирається від $S_{\uparrow}$ та $S_{\downarrow}$ (функціонал (6) оптимізується чисельно, наприклад, за допомогою методу субградієнтного спуску), залишається схема з найкращою оцінкою.

Оцінки складності атаки

Горизонтальна лінія позначає межу складності у 2^{128} операцій, нижче якої теоретична атака доцільна (тобто виконується критерій (6)). З діаграми залежності видно, що є сенс знаходити прообраз щонайбільше для 48-крокової функції, складність інвертування якої складає приблизно $2^{121.6}$ операцій хешування. Найменшу складність $\approx 2^{117.4}$ отримуємо для 17-крокової функції (хоча зрозуміло, що для такої кількості раундів є значно швидші підходи).

На рис. 1 показані найкращі значення S_{meet} – кроків, на яких відбувається зустріч.

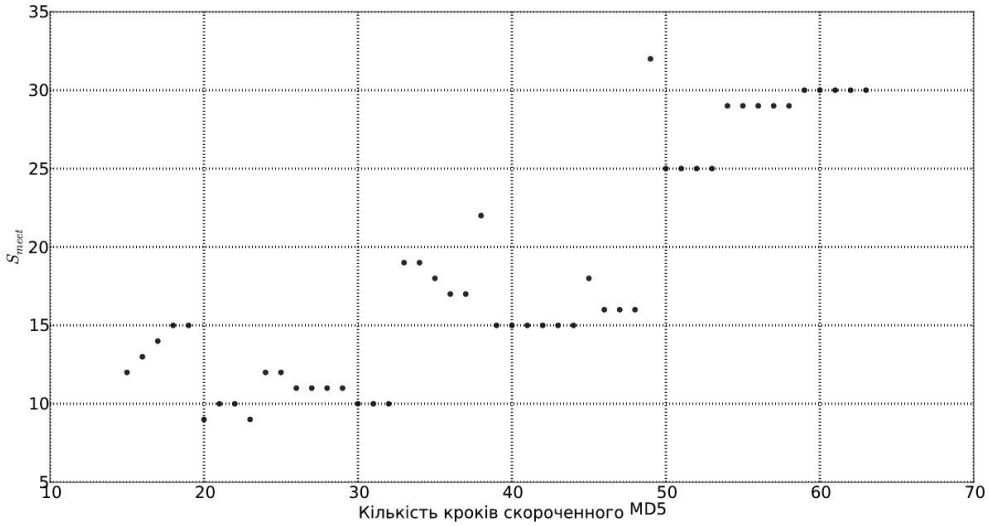

Рис. 1. Залежність номеру кроку зустрічі від кількості кроків функції

В ряді випадків атаку можна покращити через генерацію псевдопрообразів.

На рис. 2 показані оцінки складності 1-блокового прообразу, псевдопрообразу та 2-блокового прообразу на основі псевдопрообразу.

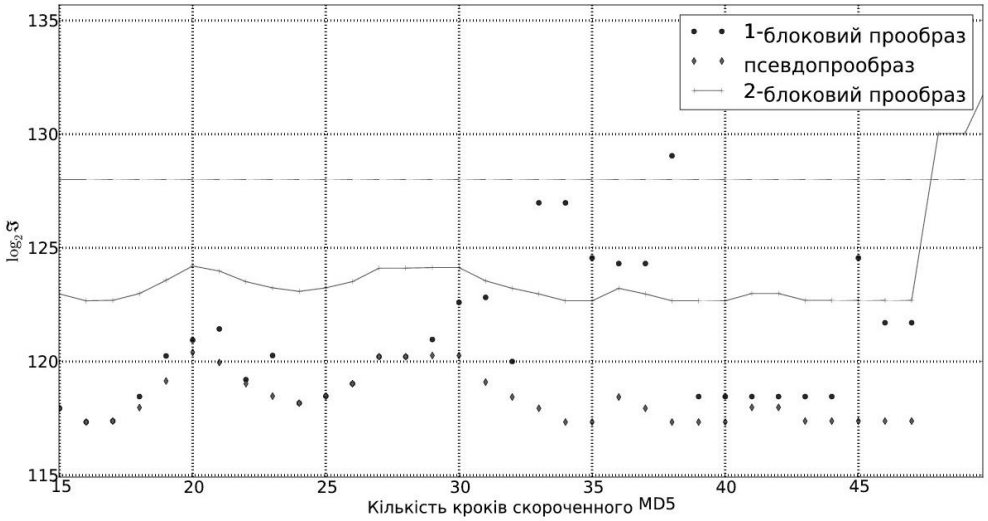

Рис. 2. Складність 1-, 2-блокового прообразів та псевдопрообразу від кількості кроків функції

З графіків видно, що в деяких випадках, а саме для 34-, 35-, 36-, 37-, 38- та 46-крокової функцій пошук прообразу через псевдопрообраз дійсно ефективніший за пошук одноблокового прообразу.

Висновки. У результаті даної роботи була розроблена теоретична атака пошуку прообразу на алгоритм хешування MD5 з використанням ідеї Кнельвольфа, був запропонований можливий метод побудови диференціальних для атаки, наведені верхні оцінки складності атаки.

З експериментальних даних видно, що складність атаки нелінійно залежить від кількості кроків усіченої функції хешування. В той же час, вона не сильно різниться для різної кількості кроків, оскільки основний внесок у штрафи для диференціальних шляхів роблять мінімально необхідні бітові умови локальної колізії. Проте після 46 кроків, штрафи формуються переважно за рахунок кроків з некомпенсованими бітовими умовами, це призводить до різкого зростання складності атаки.

Подальші перспективи досліджень вбачаються у дослідженні методів побудови кращих диференціальних шляхів, у контексті атак побудови прообразу, та перенесенні методик з диференційного аналізу блокових шифрів, таких як усічені диференціали та часткові узгодження (partial matching).

ЛІТЕРАТУРА

[1]. Ralph Charles Merkle. Secrecy, authentication, and public key systems. PhD thesis, Stanford, CA, USA, 1979. AAI8001972.

[2]. R. Rivest. The MD5 Message-Digest Algorithm, 1992.

[3]. Simon Knellwolf and Dmitry Khovratovich. New preimage attacks against reduced sha-1. Cryptology ePrint Archive, Report 2012/440, 2012. http://eprint.iacr.org/.

REFERENCES

[1]. Ralph Charles Merkle. Secrecy, authentication, and public key systems. PhD thesis, Stanford, CA, USA, 1979. AAI8001972.

[2]. R. Rivest. The MD5 Message-Digest Algorithm, 1992.

[3]. Simon Knellwolf and Dmitry Khovratovich. New preimage attacks against reduced sha-1. Cryptology ePrint Archive, Report 2012/440, 2012. http://eprint.iacr.org/.

АЛГОРИТМИЧЕСКИЕ АСПЕКТИ ПОИСКА ПРООБРАЗОВ ХЕШ-ФУНКЦИЙ НА ПРИМЕРЕ MD5

В данной статье рассматриваются аспекты применения метода Аоки построения прообраза функций хеширования, основанных на схеме Меркла-Дамгарда (на примере функции MD5). В отличии от оригинального метода, где используется несколько дифференциальных путей для построения теоретической атаки, в данной работе внимание уделяется построению мощных множеств дифференциальных путей высокой вероятности, без которых принципиально нельзя провести практическую атаку такого типа. Предлагается схема построения множества дифференциальных путей высокой вероятности для данной атаки и приводится оценка сложности построения прообраза для сокращенных версий алгоритма хеширования MD5. Полученные результаты показывают теоретическую слабость к атаке сокращенного алгоритма хеширования MD5, а также предоставляют дополнительные ограничения на конструкции новых функций хеширования.

Ключевые слова: защита информации, функция хеширования, прообраз, коллизия, дифференциальный путь, «встреча посередине», MD5, схема Меркла-Дамгарда.

ALGORITHMIC ASPECTS OF PREIMAGE SEARCH FOR HASH FUNCTIONS ON THE EXAMPLE OF MD5

In this paper some issues of Aoki method for Merkle-Damgard hash functions preimage was considered. Instead of the original method, which requires several differential paths for theoretical attack, we are targeted on generation of large set of high-probability differential paths. Methods of set of high-probability differential paths generating was suggested, complexity evaluation for reduced MD5 preimage searching was obtained. These results demonstrate weakness of reduced MD5 algorithm, also they provide additional constrictions for new hash function constructions.

Index terms: information security, hash function, preimage, collision, differential path, «meet-in-the-middle», MD5, Merkle-Damgard scheme.

Кудін Антон Михайлович, доктор технічних наук, професор кафедри математичних методів захисту інформації, Фізико-Технічний інститут Національного технічного університету України «КПІ».
e-mail: pplayshner@gmail.com.

Кудин Антон Михайлович, доктор технических наук, профессор кафедры математических методов защиты информации, Физико-Технического института Национального технического университета Украины «КПИ».

Anton Kudin, Doctor of Science, Institute of Physics and Technologies, National technical university of Ukraine «KPI».

Коваленко Богдан Анатолійович, аспірант кафедри математичних методів захисту інформації, Фізико-Технічний інститут Національного технічного університету України «КПІ».
e-mail: animantbk@gmail.com.

Коваленко Богдан Анатолиевич, аспирант кафедры математических методов защиты информации, Физико-Технического института Национального технического университета Украины «КПИ».

Bogdan Kovalenko, postgraduate student, Institute of Physics and Technologies, National technical university of Ukraine «KPI».