УДК 004.056

ОБОБЩЕННЫЙ ПАРАМЕТРИЗОВАННЫЙ АЛГОРИТМ ВОССТАНОВЛЕНИЯ ПРООБРАЗА ХЕШ-ФУНКЦИИ MD4 МЕТОДОМ ПОЛНОГО ОПРОБОВАНИЯ

Н.А. Коновалов

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский ядерный университет «МИФИ», 115409, Россия, Москва, Каширское шоссе, 31, nikitakonovalov2013@yandex.ru

Данная работа посвящена исследованию криптографической хеш-функции MD4 и некоторых особенностей её конструкции, позволяющих редуцировать функцию для задачи восстановления прообраза по известному образу при некоторых известных характеристиках прообраза методом полного опробования. На основе данных особенностей предложен редуцированный параметризированный алгоритм MD4, значительно сокращающий количество алгоритмических и логических операций. Частным случаем алгоритма является оптимизированный алгоритм поиска прообраза методом полного опробования, предложенный разработчиками специализированного программного обеспечения Hashcat [1]. Обобщенный алгоритм позволяет сократить количество шагов обновления состояния во внутреннем цикле алгоритма на 52% , а также сократить количество наиболее трудоемкой операции сложения по модулю 2^{32} на 66% в лучшем случае.

***Ключевые слова*:** хеш-функция MD4; восстановление прообраза; обратимые преобразования; метод полного опробования; обобщенный параметризованный алгоритм; оптимизация.

GENERALIZED PARAMETERIZED ALGORITHM FOR RECOVERING THE PREIMAGE OF THE MD4 HASH FUNCTION BY THE BRUTE FORCE METHOD

N.A. Konovalov

National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), 31 Kashirskoe Shosse, Moscow, 115409, Russian Federation, nikitakonovalov2013@yandex.ru

This work is devoted to the study of the cryptographic hash function MD4 and some features of its design, which allow reducing the function for the problem of restoring a preimage from a known image with some known characteristics of the preimage by the brute force method. Based on these features, a reduced parameterized MD4 algorithm is proposed, which significantly reduces the number of algorithmic and logical operations. A special case of the algorithm is the optimized preimage search algorithm by the brute-force method, proposed by the developers of the Hashcat software [1]. The generalized algorithm reduces

the number of state update steps in the inner loop of the algorithm by 52%, and also reduces the number of the most time-consuming modulo 2^{32} addition operation by 66% at best.

***Keywords*:** hash function MD4; restoring the preimage; reversible transforms; brute force method; generalized parameterized algorithm; optimization.

Введение

Криптографическая хеш-функция является одним из наиболее важных и применимых в прикладных системах криптографическим примитивом. В настоящее время наиболее используемыми в прикладных задачах хеш-функциями являются алгоритмы семейств MD и SHA. Такие алгоритмы строятся на базе парадигмы Меркла-Дамгарда [2] и повсеместно применяются в прикладных системах для безопасного хранения данных и проверки целостности.

Данная работа посвящена исследованию алгоритма MD4 из семейства алгоритмов MD. В работе исследуются некоторые особенности конструкции данного алгоритма, которые позволяют редуцировать алгоритм для задачи восстановления прообраза. Основная группа работ по исследованию поиска прообраза для хеш-функции MD4 [3, 4, 5] описывает алгоритмические подходы к решению данной задачи. В отличии от этих работ, настоящее исследование описывает обобщенный параметризированный алгоритм поиска прообраза опираясь на метод полного опробования кандидатов и предлагая технику редуцирования алгоритма для группы прообразов с известными свойствами. Один из частных случаев такой техники был описан в работе [1] разработчиками специализированного программного обеспечения для восстановления паролей Hashcat.

В настоящее время алгоритм MD4 используется в качестве криптографической хеш-функции для следующих прикладных систем:

- система хранения паролей в операционных системах типа Windows поколения NT и старше;
- система создания одноразовых паролей S/KEY [6];
- система синхронизации файлов и каталогов Rsync;
- система одноранговой сети eDonkey.

Цель настоящей работы – исследование криптографической хеш-функции MD4 и разработка алгоритма, дающего прирост скорости опробования кандидатов для метода полного опробования в задаче поиска прообраза. В ходе работы были поставлены и решены следующие задачи:

- исследование алгоритма MD4 и особенностей его конструкции;

– исследование актуальных методов и техник реализации метода полного опробования для восстановления прообраза алгоритма MD4.
– исследование возможности создания параметризированного редуцированного алгоритма для реализации метода полного опробования для восстановления прообраза алгоритма MD4.
– разработка и описание обобщенного параметризированного редуцированного алгоритма для реализации метода полного опробования для восстановления прообраза алгоритма MD4.

1. Теоретические основы

Алгоритм MD4 был предложен в качестве криптографической хеш-функции Р.Л. Ривестом и описан в работе [7] в 1990 г. Данная функция представляет из себя отображение

$$F: \{0,1\}^{(*)} \rightarrow \{0,1\}^{(128)}. \#(1)$$

Открытый текст (прообраз) M расширяется и разбивается на p блоков по k бит, а вычисление образа h происходит путем применения рекуррентной функции C (функция сжатия) над очередным промежуточным значением образа h_i и блоком открытого текста M_i:

$$C: \{0,1\}^{(128)} \times \{0,1\}^{(k)} \rightarrow \{0,1\}^{(128)}, k = 512;$$

$$h_{i+1} = C(h_i, M_i);$$

$$F(M_0, \dots, M_{p-1}) = h_p, i = \{0, \dots, p-1\}. \#(2)$$

В работе приводится описание вектора внутреннего состояния Q_j, вычисляемого на -м шаге функции C, где $Q_{-4}, Q_{-3}, Q_{-2}, Q_{-1}$ – начальные значения состояний, W_r – логическая функция над 32-х битными векторами, вектор m – 32-х битный вектор расширенного блока открытого текста и вектор t – раундовый константное 32-х битное значение. Шаги обновления состояния задаются следующими выражениями:

$$Q_0 = \left(Q_{j-4} \boxplus W_0(Q_{j-3}, Q_{j-2}, Q_{j-1}) \boxplus m_0 \boxplus t_0\right) \lll s_0, \#(3)$$

$$Q_j = \left(Q_{j-4} \boxplus W_r(Q_{j-1}, Q_{j-2}, Q_{j-3}) \boxplus m_{v_r(j)} \boxplus t_r\right) \lll s_j, j \in \{1, \dots, 47\}, \#(4)$$

Последними преобразованиями функции C являются четыре операции сложение по модулю 2^{32} векторов начального заполнения $Q_{-4}, Q_{-3}, Q_{-2}, Q_{-1}$ с векторами состояния $Q_{44}, Q_{45}, Q_{46}, Q_{47}$:

$$h_{i+1}^{(0)} = Q_{-4} \boxplus Q_{44},$$
$$h_{i+1}^{(1)} = Q_{-3} \boxplus Q_{45},$$
$$h_{i+1}^{(2)} = Q_{-2} \boxplus Q_{46},$$
$$h_{i+1}^{(3)} = Q_{-1} \boxplus Q_{47}. \#(5)$$

Для блока M_0 используются константные значения начального заполнения. Для последующих блоков M_{i+1} используются следующие начальные значения:

$$Q_{-4} = h_i^{(0)},$$
$$Q_{-3} = h_i^{(1)},$$
$$Q_{-2} = h_i^{(2)},$$
$$Q_{-1} = h_i^{(3)}. \#(6)$$

Результатом работы хеш-функции (образом) является конкатенация 32-х битных векторов:

$$h = h_p = h_{p-1}^{(0)} \,||\, h_{p-1}^{(1)} \,||\, h_{p-1}^{(2)} \,||\, h_{p-1}^{(3)}. \#(7)$$

Количество операций, используемых в стандартной реализации алгоритма MD4 приведено в Таблице 1.

Таблица 1 – Количество базовых операций в стандартной реализации алгоритма MD4

Операции над 32-х битными векторами	$\boxplus$	$\lll$	$\wedge$	$\vee$	$\oplus$	$\neg$
Количество операций	148	48	80	48	32	16

Известно, что равенство (4) является обратимым преобразованием. Обратное преобразование имеет следующий вид:

$$Q_{j-4} = (Q_j \ggg s_i) \boxminus W_r(Q_{j-1}, Q_{j-2}, Q_{j-3}) \boxminus m_{v_r(j)}^r, j \in \{0, \dots, 47\}. \#(8)$$

Кроме этого, операции сложения конечного состояния с исходным также обратимы:

$$Q_{44} = h^0 \boxminus Q_{-4},$$
$$Q_{45} = h^1 \boxminus Q_{-3},$$
$$Q_{46} = h^2 \boxminus Q_{-2},$$
$$Q_{47} = h^3 \boxminus Q_{-1}. \#(9)$$

Таким образом, зафиксировав блоки предполагаемого прообраза $m_1, ..., m_{15}$, можно получить промежуточные значения состояний из известного прообраза h:

$$Q_{47} = h^{(3)} \boxminus Q_{-1},$$
$$Q_{46} = h^{(2)} \boxminus Q_{-2},$$
$$Q_{45} = h^{(1)} \boxminus Q_{-3},$$
$$Q_{44} = h^{(0)} \boxminus Q_{-4},$$
$$Q_{43} = (Q_{47} \ggg 15) \boxminus W_2(Q_{46}, Q_{45}, Q_{44}) \boxminus m_{15}^{(2)},$$
$$\dots,$$
$$Q_{32} = (Q_{36} \ggg 3) \boxminus W_2(Q_{35}, Q_{34}, Q_{33}) \boxminus m_2^{(2)},$$
$$Q_{31} = (Q_{35} \ggg 15) \boxminus W_2(Q_{34}, Q_{33}, Q_{32}) \boxminus m_{12}^{(2)},$$
$$Q_{30} = (Q_{34} \ggg 11) \boxminus W_2(Q_{33}, Q_{32}, Q_{31}) \boxminus m_4^{(2)},$$
$$Q_{29} = (Q_{33} \ggg 9) \boxminus W_2(Q_{32}, Q_{31}, Q_{30}) \boxminus m_8^{(2)}. \#(10)$$

Обратное вычисление промежуточных значений состояний $Q_{29}, Q_{30}, Q_{31}, Q_{32}$ позволяет редуцировать алгоритм вычисления хеш-функции MD4 до вычисления только значений состояний $Q'_{29}, Q'_{30}, Q'_{31}, Q'_{32}$, сокращая количество базовых операций:

$$Q'_{-4} = 0x67452301,$$
$$Q'_{-3} = 0xefcdab89,$$
$$Q'_{-2} = 0x98badcfe,$$
$$Q'_{-1} = 0x10325476$$
$$Q'_0 = Q'_{-4} \boxplus W_0(Q'_{-3}, Q'_{-2}, Q'_{-1}) \boxplus m_0^{(0)}) \lll 3,$$
$$\dots,$$
$$Q'_{29} = (Q'_{25} \boxplus W_1(Q'_{28}, Q'_{27}, Q'_{26}) \boxplus m_7^{(1)}) \lll 5,$$
$$Q'_{30} = (Q'_{26} \boxplus W_1(Q'_{29}, Q'_{28}, Q'_{27}) \boxplus m_{11}^{(1)}) \lll 9,$$
$$Q'_{31} = (Q'_{27} \boxplus W_1(Q'_{30}, Q'_{29}, Q'_{28}) \boxplus m_{15}^{(1)}) \lll 13,$$
$$Q'_{32} = \left(Q'_{28} \boxplus W_2(Q'_{31}, Q'_{30}, Q'_{29}) \boxplus m_0^{(2)}\right) \lll 3. \#(11)$$

Используя технику обратных преобразований, разработчики специализированного программного обеспечения Hashcat, в работе [1], предлагают оптимизированный алгоритм поиска прообраза по известному образу для хеш-функции MD4 с использованием устройств ускорения вычислений. В ходе исследования было выяснено, что предложенный алгоритм можно обобщить и параметризировать для разных размеров предполагаемых прообразов.

Следует отметить, что для ускорения вычислений в задаче восстановления прообраза по известному образу используются специализированные устройства: графические ускорители, программируемые логические интегральные схемы (ПЛИС), интегральные схемы специального назначения. Алгоритм для восстановления прообраза по известному образу с использованием ускорителей вычислений реализует внешний цикл работы (на устройстве управления) и внутренний цикл работы (на устройстве ускорения вычислений). Внешний цикл работы обновляет некоторые фиксируемые значения из пространства предполагаемых векторов прообраза, а внутренний цикл производит параллельные вычисления для всего подпространства нефиксированных элементов.

2. Результаты исследования

В ходе исследования был разработан и предложен обобщенный параметризированный алгоритм восстановления прообраза по известному образу методом полного опробования кандидатов, частным случаем которого является оптимизированный алгоритм восстановления прообраза для хеш-функции MD4. Алгоритм использует особенности конструкции хеш-функций семейства MD и позволяет сократить количество операций для восстановления прообраза, тем самым значительно ускоряя процесс опробования кандидатов. Данный алгоритм является корректным за счет использования обратимых преобразований, допустимых в конструкции алгоритма хеш-функции MD4. Изменяемым входным параметром предложенного алгоритма восстановления прообраза является диапазон размера предполагаемого прообраза. Обобщенный алгоритм включает в себя 4 различных реализации в зависимости от размера l предполагаемого прообраза:

- $0 < l \leq 160$ бит;
- $160 < l \leq 288$ бит;
- $288 < l \leq 416$ бит;
- $416 < l \leq 440$ бит.

Для параметра $0 < l \leq 160$ предложенный алгоритм совпадает с алгоритмом, приведенным в работе [1].

Для описания обобщенного алгоритма для следующих параметров необходимо ввести дополнительные понятия. Пусть равенство

$$\bar{X} = \bar{X}^{(1)} \cup \ldots \cup \bar{X}^{(L)} \#(12)$$

задает объединение групп предполагаемых кандидатов-прообразов, сформированных по критерию размера кандидата, где L является максимальным размером, и выполняются условия

$$\bar{x}^{(l)} \in \bar{X}^{(l)}, l \in \{1, \ldots, L\}, \#(13)$$

где

$$\bar{x}^{(l)} = \left(x_0^{(l)}, \ldots, x_{l-1}^{(l)}\right), x_q^{(l)} \in \{0,1\}. \#(14)$$

Равенство

$$\bar{X}^{(l)} = \bar{Z}^{(n)} \,||\, \bar{V}^{(l-n)}, n < l \#(15)$$

определяет конкатенацию частей кандидатов-прообразов для внутреннего цикла $\bar{Z}^{(n)}$ и внешнего цикла $\bar{V}^{(l-n)}$ вычислений.

Для параметра $160 < l \leq 288$, зафиксировав во внешнем цикле все блоки сообщения, кроме блока m_4, появляется возможность предварительно вычислить состояния Q_0, Q_1, Q_2, Q_3, а также путем обратного преобразования вычислить состояния $Q_{31}, Q_{32}, Q_{33}, Q_{34}$. Кроме этого, для корректного описания алгоритма необходимо определить следующие равенства:

$$\bar{X}^{(l)} = \bar{V}_0^{(128)} \,||\, \bar{Z}^{(n)} \,||\, \bar{V}_1^{(l-n-128)}, \#(16)$$

$$\bar{V}^{(l-n)} = \bar{V}_0^{(128)} \,||\, \bar{V}_1^{(l-n-128)}. \#(17)$$

Алгоритм – Восстановление прообраза MD4 по известному образу с использованием ускорителей вычислений для параметра $160 < l \leq 288$

Вход: $h, i, l, \bar{X}$.
Выход: $\bar{x}$ или $\emptyset$.
Алгоритм:

1. Пока $l \leq L$:
   1.1 Для $\bar{X}^{(l)}$ зафиксировать $n, n \leq 32$.
   1.2 Для каждого $\bar{v}_q^{(l-n)} \in \bar{V}^{(l-n)}$ ***(внешний цикл)***:
      1.2.1 Зафиксировать $m_0, \ldots, m_{15}$; $m_4 = 0$.
      1.2.2 Вычислить $m^{(0)}, m^{(1)}, m^{(2)}$.
      1.2.3 Вычислить Q_0, Q_1, Q_2, Q_3.

1.2.4 Вычислить $Q_{31}, Q_{32}, Q_{33}, Q_{34}$.

1.2.5 Для каждого $\bar{z}_p^{(n)} \in \bar{Z}^{(n)}$ ***(внутренний цикл, параллельно)***:

1.2.5.1 Зафиксировать m_4.

1.2.5.2 Вычислить $m_4^{(0)}, m_4^{(1)}, m_4^{(2)}$.

1.2.5.3 Для состояний Q_0, Q_1, Q_2, Q_3 вычислить $Q_{31}^{'}, Q_{32}^{'}, Q_{33}^{'}, Q_{34}^{'}$.

1.2.5.4 Если выполняются равенства

$$Q'_{34} = Q_{34},$$
$$Q'_{33} = Q_{33},$$
$$Q'_{32} = Q_{32},$$
$$Q'_{31} = Q_{31},$$

вернуть $\bar{x}^{(l)}$ и закончить работу.

1.3 Увеличить l.

2. Вернуть $\emptyset$ и закончить работу.

Результатом использования данного алгоритма является сокращение количества шагов обновления состояния во внутреннем цикле до 30 – на 38% меньше в сравнении со стандартной реализацией алгоритма хеш-функции MD4.

Для параметра $288 < l \leq 416$ справедливы следующие равенства:

$$\bar{X}^{(l)} = \bar{V}_0^{(256)} \,||\, \bar{Z}^{(n)} \,||\, \bar{V}_1^{(l-n-256)}, \#(18)$$

$$\bar{V}^{(l-n)} = \bar{V}_0^{(256)} \,||\, \bar{V}_1^{(l-n-256)}. \#(19)$$

Зафиксировав во внешнем цикле все блоки сообщения, кроме блока m_8, появляется возможность предварительно вычислить состояния Q_4, Q_5, Q_6, Q_7, а также путем обратного преобразования вычислить состояния $Q_{30}, Q_{31}, Q_{32}, Q_{33}$.

Алгоритм – Восстановление прообраза MD4 по известному образу с использованием ускорителей вычислений для параметра $288 < l \leq 416$

Вход: $h, i, l, \bar{X}$.
Выход: $\bar{x}$ или $\emptyset$.
Алгоритм:

1. Пока $l \leq L$:

1.1 Для $\bar{X}^{(l)}$ зафиксировать $n, n \leq 32$.

1.2 Для каждого $\bar{v}_q^{(l-n)} \in \bar{V}^{(l-n)}$ ***(внешний цикл)***:

1.2.1 Зафиксировать $m_0, \ldots, m_{15}; m_8 = 0$.

1.2.2 Вычислить $m^{(0)}, m^{(1)}, m^{(2)}$.

1.2.3 Вычислить Q_4, Q_5, Q_6, Q_7.

1.2.4 Вычислить $Q_{30}, Q_{31}, Q_{32}, Q_{33}$.

1.2.5 Для каждого $\bar{z}_p^{(n)} \in \bar{Z}^{(n)}$ ***(внутренний цикл, параллельно)***:

1.2.5.1 Зафиксировать m_8.

1.2.5.2 Вычислить $m_8^{(0)}, m_8^{(1)}, m_8^{(2)}$.

1.2.5.3 Для состояний Q_4, Q_5, Q_6, Q_7 вычислить $Q_{30}^{'}, Q_{31}^{'}, Q_{32}^{'}, Q_{33}^{'}$.

1.2.5.4 Если выполняются равенства

$$Q'_{33} = Q_{33},$$
$$Q'_{32} = Q_{32},$$
$$Q'_{31} = Q_{31},$$
$$Q'_{30} = Q_{30},$$

вернуть $\bar{x}^{(l)}$ и закончить работу.

1.3 Увеличить l.

2. Вернуть $\emptyset$ и закончить работу.

Для параметра $288 < l \leq 416$ происходит сокращение количества шагов обновления состояния во внутреннем цикле до 26 – на 46% меньше в сравнении со стандартной реализацией алгоритма хеш-функции MD4.

Наиболее эффективно алгоритм работает для параметра $416 < l \leq 440$. В данном случае справедливы следующие равенства:

$$\bar{X}^{(l)} = \bar{V}_0^{(384)} \,||\, \bar{Z}^{(n)} \,||\, \bar{V}_1^{(l-n-384)}, \#(20)$$

$$\bar{V}^{(l-n)} = \bar{V}_0^{(384)} \,||\, \bar{V}_1^{(l-n-384)}. \#(21)$$

Зафиксировав во внешнем цикле все блоки сообщения, кроме блока m_{12}, появляется возможность предварительно вычислить состояния Q_8, Q_9, Q_{10}, Q_{11}, а также путем обратного преобразования вычислить состояния $Q_{32}, Q_{33}, Q_{34}, Q_{35}$.

Алгоритм – Восстановление прообраза MD4 по известному образу с использованием ускорителей вычислений для параметра $416 < l \leq 440$

Вход: $h, i, l, \bar{X}$.
Выход: $\bar{x}$ или $\emptyset$.
Алгоритм:

1. Пока $l \leq L$:

1.1 Для $\bar{X}^{(l)}$ зафиксировать $n, n \leq 32$.

1.2 Для каждого $\bar{v}_q^{(l-n)} \in \bar{V}^{(l-n)}$ ***(внешний цикл)***:

1.2.1 Зафиксировать $m_0, ..., m_{15}$; $m_{12} = 0$.

1.2.2 Вычислить $m^{(0)}, m^{(1)}, m^{(2)}$.

1.2.3 Вычислить Q_8, Q_9, Q_{10}, Q_{11}.

1.2.4 Вычислить $Q_{32}, Q_{33}, Q_{34}, Q_{35}$.

1.2.5 Для каждого $\bar{z}_p^{(n)} \in \bar{Z}^{(n)}$ ***(внутренний цикл, параллельно)***:

1.2.5.1 Зафиксировать m_{12}.

1.2.5.2 Вычислить $m_{12}^{(0)}, m_{12}^{(1)}, m_{12}^{(2)}$.

1.2.5.3 Для состояний Q_8, Q_9, Q_{10}, Q_{11} вычислить $Q_{32}^{'}, Q_{33}^{'}, Q_{34}^{'}, Q_{35}^{'}$.

1.2.5.4 Если выполняются равенства

$$Q'_{35} = Q_{35},$$
$$Q'_{34} = Q_{34},$$
$$Q'_{33} = Q_{33},$$
$$Q'_{32} = Q_{32},$$

вернуть $\bar{x}^{(l)}$ и закончить работу.

1.3 Увеличить l.

2. Вернуть $\emptyset$ и закончить работу.

Для параметра $416 < l \leq 440$ происходит сокращение количества шагов обновления состояния во внутреннем цикле до 23, что на 52% меньше в сравнении со стандартной реализацией.

Корректность алгоритмов данного типа определяется свойством обратимости преобразований алгоритма хеш-функции MD4. Оценка трудоемкости алгоритмов зависит от выбранных параметров и опирается на оценку трудоемкости стандартного алгоритма поиска прообраза по известному образу с использованием ускорителей вычислений $O(\sum_{l=1}^{m} | \bar{V}_{l-n} |)$.

Заключение

В Таблице 2 приводится сравнение наиболее значимых характеристик обобщенного параметризированного алгоритма с различными параметрами.

Предложенный алгоритм позволяет сократить количество шагов обновления состояния во внутреннем цикле до 52% в сравнении со стандартной реализацией, тем самым сокращая количество базовых операций, в том числе наиболее трудоемкой операции сложения по модулю 2^{32} до 66%.

Таблица 2 – Некоторые характеристики обобщенного алгоритма для различных параметров

Параметр	Количество шагов обновления состояния во внутреннем цикле	Сокращение количества шагов обновления состояния во внутреннем цикле в сравнении со стандартной реализацией	Количество операций сложения по модулю 2^{32} ($\boxplus$) во внутреннем цикле
$0 < l \leq 160$	33	31%	68
$160 < l \leq 288$	30	38%	64
$288 < l \leq 416$	26	46%	54
$416 < l \leq 440$	23	52%	50

Построение алгоритмов подобного типа возможно для некоторых других хеш-функций семейства MD и SHA в связи с обратимостью преобразования обновления состояния и схожести конструкции данных хеш-функций с исследуемой хеш-функцией MD4.

Библиографические ссылки

1. Steube J. Optimizing computation of Hash-Algorithms as an attacker. 2013. URL: https://hashcat.net/events/p13/js-ocohaaaa.pdf.
2. Lai X., Massey J.L.: Hash Function Based on Block Ciphers // EUROCRYPT. 1992. С. 55–70.
3. Dobbertin H.: The first two rounds of MD4 are not one-way. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 284–292. Springer, Heidelberg (1998).
4. De D., Kumarasubramanian A., Venkatesan R.: Inversion attacks on secure hash functions using sat solvers. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 377–382. Springer, Heidelberg (2007).
5. Kuwakado H., Tanaka H.: New algorithm for finding preimages in a reduced version of the MD4 compression function // IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences (Japan). 2000. № E83-A(1). P. 97–100.
6. Haller N. The S/KEY One-Time Password System // RFC 1760 (Informational) (February 1995).
7. Rivest R.L. The MD4 Message Digest Algorithm. In Menezes A., Vanstone S.A., eds.: CRYPTO. Vol. 537 of Lecture Notes in Computer Science. Springer. 1990. P. 303–311.